Sensing and real-time expert system for a masonry building robot

by
Shereen Ghanim Akrawi (BSc.)

City University

This thesis is submitted for the Degree of Doctor of Philosophy

Department of Civil Engineering

March 1998

I dedicate this thesis to my
Father.

Contents

Contents III
List of figures XII
List of tables XX
Acknowledgements XXIII
List of symbols XXIV
Declaration XXVI
Abstract XXVII

Chapter 1: Introduction
1.1 Aim of research 01
1.2 Research objectives 01
1.3 The vision of automated masonry construction 02
1.4 Building materials and methods 08
1.4.1 Brick types 08
1.4.2 Block types 09
1.4.3 Mortar 10
1.4.3.1 Pointing and jointing 11
1.5 Building brick and block walls 12
1.5.1 Types of bond 12
1.5.2 Cavity walls 14
1.6 Thesis structure 15

Chapter 2: Construction robotics
2.1 Background to construction robotics 17
2.1.1 Automating masonry tasks 19
2.2 Development in masonry automation 20

Chapter 3: Materials and methods
3.1 Introduction 31
3.2 Scope of the research 33
3.2.1 Background 33
3.2.2 Scope and limitations 34
3.2.3 Assumptions 37
3.3 Methodology 38
3.3.1 Experimental robot cell 38
3.4 Cartesian robot 42

3.4.1 Inverse kinematics 43
3.4.2 Robot performance 45
3.4.3 Robot functional library........ 45
3.5 Conveyor........ 46
3.5.1 Conveyor design and operation 46
3.5.1.1 Conveyor sensing and logic 48
3.5.1.2 Block measurements 49
3.5.1.3 Block pick-up position (BPP) 52
3.5.2 Conveyor target 55
3.6 Robot block gripper........ 57
3.6.1 Block gripper sensors 58
3.6.1.1 Ultrasonic sensors 58
3.6.1.2 Displacement transducers 59
3.6.2 Block gripping 60
3.6.2.1 Gripper to block alignment 61
3.6.2.2 Gripper to block centre alignment 63
3.7 Profile laser........ 63
3.7.1 Design of the system........ 65
3.7.2 Processing laser profile data 66
3.7.2.1 Block to gripper geometry 68
3.8 Mortar pump and dispenser........ 73
3.8.1 Peristaltic pump 73
3.8.2 Nozzle........ 74
3.8.3 Mortar dispensing 75
3.8.3.1 Single pass move for dispensing mortar 76
3.9 Laser beacon 83
3.10 Accelerometer 84
3.10.1 Accelerometer positioning........ 86
3.10.2 Accelerometer data capturing 86
3.10.2.1 Power spectrum of signal........ 87
3.10.3 Accelerometer signal processing 87
3.11 Robot path planning 88
3.11.1 Multi-axis and multi-card programming 90
3.11.2 Result of investigation 91
3.12 Conclusion........ 92

Chapter 4: Robot End Effector Vibration

4.1 Introduction 95
4.2 Review 95
4.3 Determining robot vibration 96
4.3.1 Linear moves 97
4.3.2 Parabolic moves 98
4.3.2.1 The parabolic 'U' moves 98
4.3.2.2 Segmented parabolic moves 99
4.3.3 S-curve moves 102
4.3.3.1 Single part S-curve move 102
4.3.3.2 Three part S-curve move 103
4.4 Results of robot vibration experiments 105
4.4.1 Performance results of the moves 105
4.4.1.1 Long moves 105
4.4.1.2 Medium length moves 108
4.4.1.3 Very short moves 112
4.4.2 Natural frequency vibration 114
4.4.3 Comparing the performance of parabolic moves 115
4.5 Conclusion 117

Chapter 5: Conveyor Location

5.1 Introduction 121
5.2 Review 122
5.3 Conveyor Location using Target 123
5.3.1 Scanning target shape and centre (stage 1) 124
5. 3.1.1 Implementation of method 125
5. 3.1.2 Verification of method 126
5. 3.1.2.1 Target detection simulation 126
5.3.1.2.2 Target detection by robot 128
5.3.1.2.2.1 Accuracy of angle and centroid 130
5.3.2 Final location of the target centre (stages 2 & 3) 135
5.3.2.1 Correction of detected target angle (stage 2) 136
5.3.2.1.1 Sensor readings related to direction angle 136
5.3.2.1.2 Method of angle correction 137
5.3.2.2 Target centre location (stage 3) 139
5.3.2.2.1 Sensor readings for edges 140
5.3.2.2.2 Procedure for locating target centre 141

5.3.3 Block pick point location (stage 4) 142
5.4 Target location experiments 142
5.4.1 Accuracy of target location 142
5.4.2 Search algorithm failure experiment 143
5.4.2.1 Effect of error in stage 1 detected angle 144
5.4.2.2 Effect of error in estimated centroid 145
5.4.2.3 Effect of error on stage 1 detected angle and centroid 146
5.4.3 Results of block pick point detection experiments 147
5.5 Conclusion 147

Chapter 6: Experiments In Block Sensing
6.1 Introduction 149
6.2 Verification of block sensing method 150
6.2.1 Accuracy of sensor measurement of block 150
6.2.1.1 Verification of block length 'L' sensing 150
6.2.1.2 Accuracy of block measurements 151
6.2.2 Gripper to block alignment 154
6.2.3 Accuracy of block pick-up 155
6.3 Conclusion 156

Chapter 7: Block Manipulation Experiments
7.1 Introduction 159
7.2 Experiments in laser profiler measurement 160
7.3 Block placing experiments 160
7.4 Results of block manipulation experiments 163
7.4.1 Accuracy of laser profiler measurements 163
7.4.2 Results of block placing experiment 166
7.4.2.1 Tests using calibration block 167
7.4.2.2 Tests using Celcon blocks 169
7.5 Conclusion 170

Chapter 8: Experiments in Mortar Dispensing
8.1 Introduction 173
8.2 Review 173
8.3 Background work 178
8.3.1 Mortar qualities 178
8.3.2 Pump and nozzle settings 180

8.3.3 Application of mortar 183
8.4 Robotic mortar dispensing 183
8.4.1 Mortar preparation 183
8.4.1.1 Flow-Table test 184
8.4.2 Dispensing moves 185
8.5 Wall constructions and dispensing results 187
8.6 Conclusion 192

Chapter 9: Robot Intelligence
9.1 Introduction 195
9.2 Construction robot intelligence 197
9.3 Expert system shell 204
9.4 Expert systems developed 208
9.4.1 Expert system: Assembly order 208
9.4.1.1 CAD design generation: step1 212
9.4.1.2 Project description: step 2 213
9.4.1.2.1 Generating 'NEW project description' file 215
9.4.1.2.1.1 Determining item *block* 216
9.4.1.2.1.1.1 Block level description attributes 217
9.4.1.2.1.1.2 Attribute *block* rules 219
9.4.1.2.1.2 Determining item *block_direction* 220
9.4.1.2.1.3 Determining item *block_to_wall_pos* 220
9.4.1.3 Corner building order: step 3 221
9.4.1.3.1 General building direction 221
9.4.1.3.2 Corner building rules 221
9.4.1.4 Masonry project symbolic representation 226
9.4.1.4.1 Project shape identification: step 4 227
9.4.1.4.2 Order of building shapes: step 5 230
9.4.1.5 Order of block building: step 6 233
9.4.1.6 Re-order block building list: step 7 237
9.4.1.7 Yaw axis building direction: step 8 238
9.4.1.8 Generating the theoretical task: step 9 243
9.4.1.9 Testing the assembly rules 243
9.4.2 Expert system: block picking 244
9.4.2.1 Process of building the expert system 245
9.4.2.1 Picking up a block 246
9.4.2.2 Determining the various attributes 249

9.4.2.2.1 Determining sensor readings 249
9.4.2.2.1.1 V_sensor attributes 249
9.4.2.2.1.2 H_sensor attributes 252
9.4.2.2.1.3 LVDT attributes 252
9.4.2.2.2 Checking the robot position 253
9.4.2.2.3 Checking conveyor status 254
9.4.2.2.4 Aligning gripper to block 254
9.4.2.2.5 Determining the block location 255
9.4.2.3 Running the expert system 258
9.4.2.4 Testing the expert system 261
9.4.3 Expert system: move optimisation 262
9.4.3.1 Optimum move rules 263
9.4.3.2 Move generation rules 264
9.5 Conclusion 265

Chapter 10: Conclusion
10.1 Effectiveness of robot cell 267
10.2 Material choice 269
10.3 Mortar dispensing 270
10.4 System performance and end-effector vibration 272
10.5 Automatic location 275
10.6 Laying masonry units 277
10.7 Rule-base intelligence for construction 279
10.8 Rule-base intelligence for work planning 280
10.9 Rule-base intelligence for optimum move 281
10.10 Integration 281
10.11 Outline system cost 283
10.12 Future developments 283

Appendices
Appendix A: Experimental Cell Equipment
A.1 Robot inverse kinematics (Chamberlain, 1989) A-1
A.2 Conveyor logic overview (Reilly, 1992) A-4
A.3 Ultrasonic and photoelectric raw data file A-4
A.4 Survey of 'Celcon' blocks used A-6
A.5 Laser analog displacement sensor (LADS) A-7
A.6 PICO A/D converter A-7

A.7 Accelerometer A-8
A.7.1 Calibrating the accelerometer A-9
A.7.2 Accelerometer signal filter A-10
A.7.3 Accelerometer data capturing A-10
A.7.3.1 *PC-30* board A-10
A.7.3.2 Status-30 software A-11
A.7.3.3 Sampling theorem A-11
A.7.4 Results of end-effector manual rocking experiment A-12
A.7.5 Results of robot motion experiments A-12
A.7.5.1 Strong vibration move A-12
A.7.5.2 Medium vibration move A-13
A.7.5.3 Low vibration move A-14

Appendix B: Robot Move Profile
B.1 Profile of a Parabolic 'U' move B-1
B.2 Profile of a S-curve move B-2

Appendix C: Robot Motion Experiments
C.1 Linear move experiments C-1
C.2 Parabolic move experiments C-4
C.2.1 Parabolic 'U' move C-4
C.2.2 Segmented parabolic moves tested C-5
C.3 S-Curve move C-9
C.3.1 Minimum time three part S-curve move C-9
C.3.2 Minimum time single part move C-10
C.3.3 Three part S-curve move C-11
C.3.4 Single part S-curve C-13

Appendix D: Target Location
D.1 Theory of moments D-1
D.2 Target location experiment results D-3
D.2.1 Target scanning co-ordinate file (stage 1) D-3
D.2.2 Target allocation data file D-3

Appendix E: Block Manipulation Experiments
E.1 Results of block manipulation experiments E-1
E.2 Results of block placing experiment E-4

E.2.1 Results of experiments using calibration block E-4
E.2.2 Tests using Celcon blocks E-5

Appendix F: Mortar Dispensing Experiments

F.1 Results of Tilcon mix dispensing experiments F-1
F.2 Results of Pozament mix dispensing experiments F-3

Appendix G: Rule Base Expert Systems

G.1 Expert system: Assembly order G-1
G.1.2 Example of ordering a building project G-19
G.2 Expert system: block picking G-21
G.2.1 Rule set BP G-21
G.2.2 Results of testing the block_pick expert system G-37
G.3 Expert system: move optimisation G-43

Appendix H: Accelerometer signal processing

H.1 Verification of software H-1
H.2 Frequency selective filtering H-3
H.2.1 Setting the sampling rate H-5
H.3 Accelerometer signal processing H-8
H.3.1 Numerical integration method H-9
H.3.2 Treatment of constants in numerical integration H-9
H.3.3 Verification of integration method H-10
H.3.3.1 Results of verification H-11
H.3.4 Procedure and trials with accelerometer H-12
H.3.4.1 Accuracy analysis H-16
H.3.5 Verification of displacements H-18
H.3.5.1 Experiment set-up H-18
H.3.5.2 Processing the laser signal H-18
H.3.5.3 Analysing and comparing the signals H-19
H.3.5.4 Manual rocking motion experiment H-20
H.3.5.5 Move experiments H-21
H.3.5.5.1 Strong vibration move H-21
H.3.5.5.2 Medium vibration move H-23
H.3.5.5.3 Low vibration move H-24

Appendix I: Robot Path Planning

I.1 Move description theory I-1
I.2 Linear velocity moves I-1
I.3 Parabolic velocity moves I-2
I.3.1 Parabolic 'U' move I-3
I.3.2 Segmented parabolic move I-6
I.3.2.1 Parabolic velocity with max. negative inflection I-6
I.3.2.2 Linear velocity I-7
I.3.2.3 Parabolic velocity with max. positive inflection I-8
I.3.2.4 Blending the segments I-8
I.3.2.5 Velocity check during segments I-9
I.4 The S-curve move I-10
I.4.1 Minimum time S-curve moves I-14
I.5 Determining system limits I-16
I.5.1 Determining system maximum acceleration I-16
I.5.1.1 Peak acceleration results I-17
I.5.2 Determining system maximum velocity I-18
I.5.2.1 Peak velocity results I-19
I.5.3 Setting maximum velocity and acceleration I-19
I.6 Verification of move I-20
I.6.1 Profile of a parabolic 'U' move I-21
I.6.2 Profile of a single Part S-curve move I-21
I.6.3 Accuracy of the profile I-22

References R-1

List of figures

Figure 1.1: Stages of the building process with a key parameter list 05
Figure 1.2: Struck flush 11
Figure 1.3: Curved recessed 11
Figure 1.4: Struck or weathered 11
Figure 1.5: Overhung struck 12
Figure 1.6: Square recessed 12
Figure 1.7: Flemish bond. Elevation of wall with corner and stopped end 13
Figure 1.8: English bond. Elevation of wall with corner and stopped end 13
Figure 1.9: English graden wall bond 13
Figure 1.10: Stretcher bond 14
Figure 1.11: Rat-trap bond 14
Figure 2.1: Schematic side-view of Blockbot (Slocum *el at.*, 1988) 20
Figure 2.2: Robot ensemble for wall construction (Muspratt, 1988) 21
Figure 2.3: General concept for robotic masonry system (Altobelli *et al.*, 1993) 25
Figure 2.4: Mobile robot for masonry construction at the University of Stuttgart.26
Figure 2.5: Concept robotic masonry system (Rihani and Bernold, 1994) 29
Figure 3.1: Picture of the experimental robot cell 32
Figure 3.2: Design of experimental robot cell 39
Figure 3.3: Automating the building process 40
Figure 3.4: Symbolic representation of the robot's configuration 44
Figure 3.5: (a_1=0, a_2,a_3) Values for Gripper_TCP 44
Figure 3.6: (a_1=0, a_2,a_3) Values for Block_pick_TCP 44
Figure 3.7: (a_1=0, a_2,a_3) Values for Sensor_scan_TCP 45
Figure 3.8: (a_1=0, a_2,a_3) Values for Mortar_dispensing_TCP 45
Figure 3.5.1: Conveyor belt with target 47
Figure 3.5.2: Conveyor layout 48
Figure 3.5.3: Photoelectric sensors on the conveyor 49
Figure 3.5.4: Ultrasonic sensors on the conveyor 49
Figure 3.5.5: Ultrasonic sensor array on the conveyor 51
Figure 3.5.6: Block length and offset measures of the block relative to the sensors 51
Figure 3.5.7(a): Block profile using original ultrasonic sensor data 52
Figure 3.5.7(b): Block profile using generated data from the line fitting to ultrasonic sensor data 52
Figure 3.5.8: Adjusting block position on the conveyor at the pick-up stage 54

Figure 3.5.9: Conveyor, block and robot co-ordinate systems 55
Figure 3.5.10: Conveyor target dimensions 56
Figure 3.5.11: Conveyor target and end-effector sensor used to locate target 56
Figure 3.5.12: Target position relative to block pick position 57
Figure 3.6.1: Clamp type block/brick gripper at robot end-effector 57
Figure 3.6.2: Suction type block/panel gripper 58
Figure 3.6.3: Block gripper 59
Figure 3.6.4: Block gripped showing the use of the two displacement transducers 59
Figure 3.6.5: Gripper to block rotation 60
Figure 3.6.6: Gripper to block position before adjustment 62
Figure 3.7.1: Laser profile 64
Figure 3.7.2: Example of a block gripped at an angle 64
Figure 3.7.3: Laser profiler scanning a block 64
Figure 3.7.4: Reference point ‘P’ relative to robot centre of rotation 65
Figure 3.7.5: Raw signal of the laser profiler 67
Figure 3.7.6: Processed signal of the laser profile 67
Figure 3.7.7: Determining edges in the laser profile and their corresponding position readings 67
Figure 3.7.8: Example of the gripper at an angle to the laser profiler 69
Figure 3.7.9: Reference point ‘P’ relative to centre of rotation 70
Figure 3.7.10: Gripper side 73
Figure 3.7.11: Gripper front 73
Figure 3.8.1: Peristaltic pump, power supply and control unit 73
Figure 3.8.2: Showing the front rollers of the peristaltic pump mechanism, and the hose element 74
Figure 3.8.3: Nozzle attached to the delivery pipe of the pump 74
Figure 3.8.4: Schematic diagram showing dimensions of the nozzle, with an internal opening 61 mm × 6 mm 75
Figure 3.8.5: Individual blocks ‘buttered’ using the robot to rotate the block 76
Figure 3.8.6: Picture showing the robot being used to ‘butter’ an individual block 76
Figure 3.8.7: Mortar laid on pre-positioned blocks 76
Figure 3.8.8: Geometry of the mortar application move during the second stage 81
Figure 3.8.9: Geometry of the mortar application move during the third stage .. 81

Figure 3.8.10: Mortar thickness to be dispensed on the block using the dispensing move in the x,z plane........81
Figure 3.8.11a: Start of the dispensing move showing the nozzle's fixed position........82
Figure 3.8.11b: During stage(1) of the dispensing move, when applying mortar on the bedding face of the block showing reference point 'P', with distance travelled 'd' and gripper moved angle 'φ'........82
Figure 3.8.11c: During stage(2) of the dispensing move when applying mortar around the corner block moved an angle 'φ_c'........82
Figure 3.8.11d: Start of stage(3) of the dispensing move when applying mortar in the side of the block........82
Figure 3.8.11e: During stage(3) of the dispensing move when applying mortar on the side of the block. Gripper moved an angle of 'φ_c'........83
Figure 3.8.11f: End of the dispensing move........83
Figure 3.9.1: Laser beacon........84
Figure 3.10.1: Miniature heavy duty accelerometer........84
Figure 3.10.2: Power supplies........84
Figure 3.10.3: Accelerometer cutaway........85
Figure 3.10.4: End effector of robot........86
Figure 3.11.1: Summary of minimum time parabolic 'U' move........91
Figure 3.11.2: Summary of minimum time S-curve 'U' move........92
Figure 4.1: Different linear moves with the equal time and move length........97
Figure 4.2: Parabolic U moves of the same increment Δp........99
Figure 4.3: Different parabolic moves with the same time and move length........101
Figure 4.4: Single part S-curve moves of the same increment........103
Figure 4.5: Different 3-part S-curve moves with the same time and move length........104
Figure 4.6: Settling time of 2 m moves with respect to move time........107
Figure 4.7: The quality of 2 m moves with respect to move time........108
Figure 4.8: Settling time of 1 m moves with respect to move time........111
Figure 4.9: The quality of 1 m moves with respect to move time........111
Figure 4.10: Settling time of 4 mm moves with respect to move time........113
Figure 4.11: The quality of 4 mm moves with respect to move time........114
Figure 4.12: Complex frequency vibration of end-effector settling........115
Figure 4.13: Natural frequency vibration of end-effector settling........115
Figure 4.14: End-effector lateral acceleration result of parabolic U move........116

Figure 4.15: End-effector lateral acceleration result of segmented parabolic move 117
Figure 5.1: Top view of target scanning area within the robot workspace.......... 124
Figure 5.2a: The radius of gyration indicates shape 125
Figure 5.2b: The second moments indicates direction 125
Figure 5.3: AutoCAD drawing of the scanned area and targets at different location 127
Figure 5.4: Target set-up on portable table 130
Figure 5.5: Block pick position relative to target centre 135
Figure 5.6: Searching for the centre of the target using the horizontally pointing sensor 136
Figure 5.7: Determining sensor reading at varying angle to the target........ 137
Figure 5.8: Determining target edge using horizontal sensor........ 141
Figure 6.1: Block rotated at different angles 155
Figure 7.1: Example of a block placing experiment set-up........ 162
Figure 7.2: Target positioning for block placing experiment set-up 163
Figure 8.1: Dipping blocks into mortar before setting them onto the wall (Bohm *et al.*, 1991)........ 175
Figure 8.2: Computer controlled mortar supply (Atobelli *et al.*, 1993) 176
Figure 8.3: Multi-functional technology unit (Pritschow *et al.*, 1995)........ 177
Figure 8.4: Calibration of pump at delivery height of 400 mm (Ahmed 1993) .. 181
Figure 8.5: Rippled effect on mortar bed due to nature of peristaltic pump used 182
Figure 8.6: Continuous application of mortar on pre-laid bricks........ 183
Figure 8.7: The mix components are first weighed 184
Figure 8.8: The Flow-test........ 185
Figure 8.9: Mortar application on the bedding side of the block........ 186
Figure 8.10: Three block assembly with the middle block offset by 20 mm (to be punched out 186
Figure 8.11: When laying the block, a sideways and down movement is used after lowering the block 186
Figure 8.12: Showing the robot laying a block after mortar has been dispensed on it........ 188
Figure 8.13: Buttered blocks used in building small walls 189
Figure 8.14: Example of a laid block with mortar on the bedding side only 189
Figure 8.15: Three blocks assemblies with the middle block offset 189

Figure 8.16: Two views of the starting position of the mortar dispensing move (The nozzle is touching the block) 190
Figure 8.17: During stage 1 of the dispensing move 191
Figure 8.18: Near the end of stage 1, of the dispensing move 191
Figure 8.19: End of stage 1, start of stage 2 dispensing mortar round the corner of the block 191
Figure 8.20: During stage 3 of the dispensing move, applying mortar on the side of the block 191
Figure 8.21: At the end of stage 3 of the dispensing move 191
Figure 9.1: Creating the production programme 197
Figure 9.2: Knowledge Shop main screen 205
Figure 9.3: Expert systems creation process 207
Figure 9.4: Knowledge Shop system structure 208
Figure 9.5: Steps for determining the order of building masonry project 210
Figure 9.6: Example of robot laying a corner block 211
Figure 9.7: CAD output of a masonry project 213
Figure 9.8: First course of the masonry project shown in figure 9.7 214
Figure 9.9: Second course of the masonry project shown in figure 9.7 215
Figure 9.10: Third course of the masonry project shown in figure 9.7 215
Figure 9.11: The dependencies of attributes bloc, block_direction, and block_in_wall_pos 216
Figure 9.12: Attribute list used to determine blocks in corner placing order 223
Figure 9.13: Example of corner building order 224
Figure 9.14: Collision zone to be avoided 226
Figure 9.15: Illustrations of the different shapes expected in the project designs and their names 227
Figure 9.16: Shape identification, and order of building the shapes attributes ... 228
Figure 9.17: Block building order attribute network 233
Figure 9.18 Robot placing brick using the wrong yaw-axis angle, which can impact the robot motor on the already laid block 239
Figure 9.19: Robot yaw-axis orientations 239
Figure 9.20: Yaw-axis building direction network of attributes 240
Figure 9.21a: Illustration of (Rule_AO_44) showing the correct and wrong way to lay a block 241
Figure 9.21b: Illustration of (Rule_AO_45) showing the correct and wrong way to lay a block 241

Figure 9.22: Network of attributes for the block picking expert system.............246
Figure 9.23: Reading of sensors when the gripper is in position to check for the block pick up 250
Figure 9.24: Reading of V_sesnsor when the gripper is in position to pick up a block 250
Figure 9.25: The pick block 'C' code calling programme, integrated with the pick_block module and the gripper sensing and conveyor information........259
Figure 9.26: Network of attributes for the move optimisation rules..................263
Figure A.1: Robot end-effector A-3
Figure A.2: Top level conveyor logic (|Reilly, 1992)..A-4
Figure A.3: Identification of LADS parts..A-7
Figure A.4: PICO A/D converter..A-8
Figure A.5: Accelerometer A-8
Figure A.6: Accelerometer reading '-g' A-9
Figure A.7: Accelerometer reading '+g' A-9
Figure A.8: Low-pass active filter for the accelerometer signal A-10
Figure F.1: Typical bead geometry for increasing offsets of 5 mm, 10 mm, 15 mm to 20 mm (top to bottom) using Tilcon mix. Dispensing at nozzle angel 45°; pumpsetting = 35 rpm; conveyor belt speed = 90 mm/s F-1
Figure F.2: Typical Pozament mortar bead geometry for increasing offsets of 10 mm, 15 mm to 20 mm (top to bottom). Dispensing at nozzle angel 30°; pump setting = 40 rpm; conveyor belt speed = 45 mm/s......................................F-3
Figure H.1: Sin wave of amplitude 2 ....H-2
Figure H.2: Power spectrum of the sin wave H-2
Figure H.3: Amplitude spectrum of the sin wave..H-3
Figure H.4: Accelerometer raw signal...H-4
Figure H.5: Accelerometer filtered signal H-4
Figure H.6: Amplitude spectrum of un-filtered signal H-5
Figure H.7: Amplitude spectrum of filtered signal H-5
Figure H.8: Signal sampled at 1000 Hz H-6
Figure H.9: Amplitude spectrum (FFT) of signal sampled at 1000 Hz...............H-6
Figure H.10: Signal sampled at 500 Hz H-6
Figure H.11: Amplitude spectrum (FFT) of signal sampled at 500 Hz...............H-7
Figure H.12: Signal sampled at 200 Hz H-7
Figure H.13: Amplitude spectrum (FFT) of signal sampled at 200 Hz...............H-7
Figure H.14: Normalised amplitude spectrum (FFT) of signal sampled at 1000 Hz H-8

Figure H.15: Normalised amplitude spectrum (FFT) of signal sampled at 500 Hz H-8
Figure H.16: Normalised amplitude spectrum (FFT) of signal sampled at 200 Hz H-8
Figure H.17: Sin wave for *x* H-11
Figure H.18: Integral of the sin wave H-11
Figure H.19: Double integral for sin wave H-12
Figure H.20: Corrected double integral of sin wave H-12
Figure H.21: Transit acceleration of robot end-effector H-14
Figure H.22: Integral of acceleration H-14
Figure H.23: Integral of velocity H-15
Figure H.24: The derived displacement of the end-effector H-15
Figure H.25: Integration process examined H-17
Figure H.26: Analog displacement experiment set-up H-19
Figure H.27: Raw signal : accelerometer and laser beam H-20
Figure H.28: Laser and accelerometer displacements H-20
Figure H.29: Difference in average displacement amplitude of accelerometer and laser results H-21
Figure H.30: Raw signal : accelerometer and laser beam H-22
Figure H.31: Laser and accelerometer displacements H-22
Figure H.32: Difference in average displacement amplitude of accelerometer and laser results H-22
Figure H.33: Raw signal : accelerometer and laser beam H-23
Figure H.34: Laser and accelerometer displacements H-23
Figure H.35: Difference in average displacement amplitude of accelerometer and laser results H-24
Figure H.36: Raw signal : accelerometer and laser beam H-24
Figure H.37: Laser and accelerometer displacements H-25
Figure H.38: Difference in average displacement amplitude of accelerometer and laser results H-25
Figure I.1: Linear move I-1
Figure I.2: Parabolic move profiles I-2
Figure I.3: V_{peak} and A_{peak} parabolic moves I-4
Figure I.4: Parabolic 'U' move I-5
Figure I.5: Linear acceleration: maximum negative inflection I-7
Figure I.6: Constant acceleration I-7

Figure I.7: Parabolic velocity: maximum positive inflection at start, with two blended moves I-9
Figure I.8: General parabolic move segment....................I-10
Figure I.9: Three part S-curve move I-11
Figure I.10: Single part S-curve move....................I-13
Figure I.11: S-curve move acceleration....................I-13
Figure I.12: Signal part S-curve move....................I-16
Figure I.13: Determining maximum acceleration....................I-17
Figure I.14: Determining maximum velocity....................I-18
Figure I.15: Position Vs time of a parabolic 'U' move I-21
Figure I.16: Velocity Vs time of a parabolic 'U' move....................I-21
Figure I.17: Position Vs time of a S-curve move I-22
Figure I.18: Velocity Vs time of a S-curve move....................I-22

List of tables

Table 1.1: Key to figure 1.1 04
Table 1.2: Cost analysis of a block building project 08
Table 3.7.1: Processing the laser profile scan 66
Table 3.10.1: Comparison of direct laser measurements with estimate from accelerometer 88
Table 4.1: Linear moves made for t_{tot} of 3.32 seconds and ΔP of 1981 mm 97
Table 4.2: Parabolic U type moves of $\Delta p = 991$ mm 98
Table 4.3: Segmented parabolic moves of $\Delta p = 991$ mm and $\Delta t = 5$ seconds 101
Table 4.4: Single part S-curve move with Δp of 991 mm 102
Table 4.5: Three part S-curve moves of $\Delta p = 1981$ mm and Δt 3.32 seconds 104
Table 4.6: Results of 2 m moves 106
Table 4.7: Results of minimum time moves of 2 m 107
Table 4.8: Result of 1 m moves 110
Table 4.9: Results of minimum time moves of 1 m 110
Table 4.10: Results of 4 mm moves 113
Table 4.11: Parabolic U moves investigated 115
Table 4.12: Moves of 1981 mm made in 5 seconds 116
Table 5.1: Reference to targets in figure 5.2 127
Table 5.2: Results of target direction estimation using moments 128
Table 5.3: First stage detection of target angle: worst case scan 132
Table 5.4: First stage detection of target angle: best case scan 132
Table 5.5: First stage detection of target centroid: worst case scan 133
Table 5.6: First stage detection of target centroid: best case scan 134
Table 5.7: Second stage detection of target angle 135
Table 5.8: Results of sensor reading when end-effector is at 90° and aligned to target 137
Table 5.9: Results of target edge offset 141
Table 5.10: Target angle experiments: testing at different positions 143
Table 5.11: Target angle experiments: repeatability test 143
Table 5.12: Testing the effectiveness of the search algorithm (target at 0° angle to the x-axis) 144
Table 5.13: Testing the effectiveness of the search algorithm (target at 90° angle to the x-axis) 145

Table 5.14: Testing the effectiveness of the search algorithm (target at 90° angle to the x-axis) 146
Table 5.15: Testing the effectiveness of the search algorithm (target at 90° angle to the x-axis) 146
Table 6.1: Calibration block used to measure the 'Length_Correct' for Senr's reading 152
Table 6.2: Experiments using conveyor sensing on the calibration blocks 153
Table 6.3: Results of experiments on block dimension measurements 153
Table 6.4: Adjusting gripper to block experiment results 155
Table 7.1: Results of each experiment calculation from mean of 6 block scan results 168
Table 7.2: A and A_2 direct and sensed values 165
Table 7.3: B and B_2 direct and sensed values 165
Table 7.4: Block length 'l' direct and sensed values 166
Table 7.5: Gripper width 'W_2' and 'α' angle's direct and sensed values 166
Table 7.6: Block to gripper 'γ' angle's direct and sensed values 166
Table 7.7: Error in (A,A_2) values 168
Table 7.8: Error in the block to gripper angle measurements 168
Table 7.9: Error in block laying 168
Table 7.10: Error in (A,A_2) direct and sensed values 169
Table 7.11: Error in the block to gripper angle measurements 169
Table 7.12: Error in block laying 169
Table 8.1: Mortar mixes suitable for pumping 180
Table 8.2: Results of dispensing mortar on blocks using single pass moves using robot speed of 43 mm/sec 190
Table 8.3: Results of dispensing mortar on bricks using single pass moves using robot speed of 43 mm/sec 190
Table 9.1: List of attributes that are immediately linked to attribute 'block' 217
Table 9.2: List of attributes that are immediately linked to attribute 'Pick_block' 247
Table 9.3: List of attributes that are immediately linked to attribute 'block_location' 258
Table 10.1 Automating task activities 282
Table A.1: Survey of blocks A-6
Table A.2: Specification of laser displacement sensor A-7
Table A.3: Specification of A/D converter A-8
Table A.4: Accelerometer specification A-9

Table A.5: Example of data captured from accelerometer signal A-11
Table A.6: Data for figure 2.8.33 .... A-12
Table A.7: Data for figure 2.8.36 .... A-13
Table A.8: Data for figure 2.8.39 .... A-13
Table A.9: Data for figure 2.8.42 .... A-14
Table D.1: Data file of stage 1 target scanning D-03
Table E.1: Result of six scans from a laser profile of a block picked horizontally E-1
Table E.2: Result of six scans from a laser profile of a block picked horizontally E-1
Table E.3: Result of six scans from a laser profile of a block picked horizontally E-2
Table E.4: Result of six scans from a laser profile of a block picked at an angle E-2
Table E.5: Result of six scans from a laser profile of a block picked at an angle E-3
Table E.6: Result of six scans from a laser profile of a block picked at an angle E-3
Table F.1: Result of experiments using the recommended Tilcon mix F-2
Table F.2: Result of experiments using the recommended Pozament mix F-4
Table I.1: Maximum acceleration using V_{peak} = 1000 mm/sec I-17
Table I.2: Maximum acceleration using V_{peak} = 914 mm/sec I-18
Table I.3: Maximum acceleration using V_{peak} = 759 mm/sec I-18
Table I.4: Maximum velocity using A_{max} = 1524 mm/sec^2 .... I-19
Table I.5: Maximum velocity using A_{max} = 3506 mm/sec^2 .... I-19
Table I.6: Maximum velocity and acceleration I-19

Acknowledgements

Denis Chamberlain	City University	Supervising my work. His help, advice and inspiration throughout my PhD, and through the many difficult times that I went through.
EPSRC	Formerly SERC	Funding and grant
John Rielly	City University (CRU)	For building parts of the experimental robot cell, and his help throughout my PhD.
Julien Chua and Rushna Begum,	City University (CRU)	For their friendship, support and encouragement during my PhD.
Dr Mathew Coop, Alastair Paterson, Graham Blekely, and Andy Bates	City University	For their friendship, support and encouragement during my PhD.
My sister May Akrawi, Ali Mahmood, Behzad Emadi, and Nilesh Mehta		For their advice, encouragement and proofreading my thesis. Their help was valuable.
My family		For their love and support during these years, and for their patience and understanding during the hard times I was going through. Without them it would not have been possible.
My friends Saba Beden, Yousra Shanshal, Lana Shanshal, Rima Abu-Rahme, and Rula Al-Chorbachi		For their emotional support, their friendship, encouragement. They were there for me in times of need, and without them it would not have been possible.
Zayed Mahfooth		For helping me print my thesis. His help was valuable.

List of Symbols

Notation	
RBES	Rule Base Expert System
A,B,C,D	constants
$P_{m,}$, S	distance
ΔP	distance covered
P_{tot} ,P	total distance covered
P_s	time for rising portion of an 'S' curve move
P_l	half the distance travelled during a 'U' move
P_s	distance covered in the rising portion of the 'S' curve move.
P_{accel}	distance covered when accelerating
P_{slew}	distance covered when moving at constant speed
P_{decel}	distance covered decelerating
V	velocity
V_{peak}	peak velocity (system capabilities)
V_{max}, , $V_{extreme}$	maximum velocity reached during a move
V_s	starting velocity
V_f	final velocity
V_m	mean velocity
V_l	end velocity of a parabolic move
a	acceleration
A_{max}	maximum acceleration reached during a move
t	time
t_{tot} , T	total time of move
T_k	time for rising portion of an 'S' curve move
Δt	elapsed time
t_{accel}	time to accelerate
d_{ave}	average displacements amplitudes
LADS	laser analogue displacement sensor
FFT	Fast fourier transform
DOF	Degrees of freedom
SMCC	Smart Motion Control Card
BCP	Block Centre Point
BPP	Block Pick-up Position on the conveyor
NBPP	New Block Pick-up Position on the conveyor
SBPP	Safe Block Pick-up Position on the conveyor
θ	Angle of the target and the conveyor relative to the robot co-ordinates system.
ϕ	Angle of the block to the conveyor
β	Angle of the robot gripper to the block
σ	Angle of the robot's yaw-axis.

α	Angle, about the roll axis, between the laser strike and the gripper
γ	Angle of the block to the gripper when gripped
λ	Angle of the triangles attached to the gripper
φ	Angle moved by block in the mortar dispensing move
χ	Angle moved by block when applying mortar to the corner of the block during the mortar dispensing move
$(x^{/}, y^{/})$	Robot and gripper co-ordinate system
$(x^{//}, y^{//})$	Block co-ordinate system
$x^{///}, y^{///}$	Conveyor co-ordinate system
H_sensor	Horizontally pointing ultrasonic sensor on the gripper
V_sensor	Vertically pointing ultrasonic sensor on the gripper
LVDT	Displacement transducers
TCP	Tool Centre Point
AutoCAD	Computer aided design software package
t_b	Thickness of mortar on bedding side of block
t_p	Thickness of mortar on side of block
l	Ideal length of a experimental blocks
h	Ideal height of a experimental block
TL	Actual length of block measured using conveyor sensor data
H	Actual height of block measured using conveyor sensor data
W	Actual width of block measured using conveyor sensor data
MW	The width of the applied mortar
d_1	Length of the block $+ t_p$
d_2	Height of the block $+ t_b$
φ	Block angular displacement during mortar dispensing move
Q	Dispensing rate of mortar pump
V_1	Relative velocity of move when applying mortar on bedding face of the block
V_2	Relative velocity of move when applying mortar on side of the block
V_c	Relative velocity of move when applying mortar round the corner of the block

Declaration

Abstract

The construction industry is striving to eliminate dangerous and repetitive work, as well as increase quality and productivity in the various tasks. For this reason, there is growing interest in the use of automation and robotics. However, the requirements of robots for construction are different from those of industrial robots, due to the characteristics of the construction tasks and the relatively unstructured working environment.

The main objective of this research is to investigate the enabling technology for a masonry tasking robot, utilising an experimental robot cell. To truly automate the masonry construction task, there is need to utilise the advancement in robotic technology, especially to deal with the unstructured environment. This view is in-line with this research, which attempts to solve part of the complex problems of automating the building task, by using forms of sensing and intelligence. Concentration on this is the main distinguishing difference between this work and the few other attempts at physical realisation and experimentation in masonry automation. In terms of research and development of masonry tasking machines and robots, there is much activity on an international scale. Concerning the provisions for machine intelligence in this, it appears that the work reported has the most advanced provisions for computer intelligence. This work is of general relevance to construction robots because imprecision, dynamic performance, unplanned events and cell component relocation are considered.

The experimental robot cell, built at City University, is used in the research. Standard construction materials have been adopted with imprecise dimensions. Using a CAD/CAM facility, building project designs are translated into robot's 'theoretical task'. However, because the masonry material is unpredictable, this can not be directly implemented without real-time adjustments derived from sensing. Not withstanding, advantage is taken of pre-processing, with real-time accommodation of discrepancies, obstacle avoidance and un-planned events. In this, the robot cell performs intelligent processing using rule-based expert systems to carry out the masonry building tasks. Such a process contributes to the fundamental basis for the automation of all stages of building, from architectural planning through to execution of the construction work.

A further complication is that the form of a move influences the dynamic response of the robot structure. Therefore, the structural dynamics response of the robot is taken into account in order to optimise performance. Rule based expert systems are investigated to enable a goal driven, intelligent planning approach to be implemented, that can provide an effective dynamic plan for the building task, as well as real-time adjustments to the automatically generated 'theoretical task'.

Chapter: 1 Introduction

1.1 Aim of research

The purpose of this research is to investigate the enabling technology for an advanced masonry tasking robot.

1.2 Research objectives

The main objective of this research is to investigate the enabling technology for a masonry tasking robot, utilising an experimental robot cell. In this, the robot cell performs intelligent processing to carry out the masonry building tasks. Such a process contributes to the fundamental basis for the automation of all stages of building, from architectural planning through to execution of the construction work.

The sub-objectives of the research can be identified as follows:

(i) To investigate the effectiveness of an experimental robot cell comprising a Cartesian robot, material conveyor, mortar dispensing pump, laser leveller and sensing system designed for masonry automation.

(ii) To investigate whether or not conventional, standard production masonry material, with significant dimensional tolerance, can be utilised in automated handling processes.

(iii) To automate the mortar dispensing operation and investigate whether conventional or modified mortar can be used for bonding. Also, to determine the effectiveness of the robot in forming strong bonds, when used to lay the buttered blocks. This is compared to manual work.

(iv) To investigate the performance of the robot, as affected by the vibration of the end-effector, for various move types.

(v) To investigate the means for automatic location of the peripheral equipment of the robot cell. This is necessary in enabling the robot to deal with the unstructured environment of the construction site.

(vi) To determine the accuracy and efficiency in the laying of masonry units in a predetermined position, while incorporating the necessary run time adjustments.

(vii) To investigate the effectiveness of applying intelligence, integrated with sensing, to the performance of the robot in the actualisation of the building process in the unstructured environment of the construction site.

(viii) To investigate the use of machine intelligence in determining the order of building the construction task.

(ix) To investigate the use of machine intelligence for motion control, and determining optimum moves.

(x) To investigate the integration of the design stage of construction task with the process of realisation of the robot building task, using sensing and intelligent processing in the different construction tasks.

(xi) To investigate an outline of the cost of the system.

1.3 The vision of automated masonry construction

Masonry automation and robotics have been studied at research institutes in a number of countries, mostly using simulation tools which enable top level examination of robot concept performing complete projects. Very little hardware research was apparent with notable gaps in the sensing, intelligence and material supply provisions. A number of masonry unit designs were apparent, these tend to reflect the different national preference and traditions in architecture and the construction process. There was little consensus on the matter of size, format and bonding or interlocking system. However, it was commonly agreed that the best prospects lay with dual purpose units i.e. those which would be suitable for both manual and robotic construction. Until robotic construction is proven to the satisfaction of the UK industry, this is a significant hurdle on the account of Health and Safety handling of weight restrictions for site operatives (20 kg limit). Furthermore, enquiries with both contractors and material suppliers in the UK indicated a clear preference for automation using existing standard production masonry units.

To truly automate the masonry construction task, all stages of the building task needed to be considered, from the architectural planning through to the execution of the construction process. The need to consider the unstructured environment of the construction site is a vital issue to be tackled thus ensuring the success of the automation process. In this research the enabling technology for a masonry tasking robot will be investigated using sensing and intelligent processing, to carry out the masonry building task. The overall vision of the research is to enable the masonry robot to build a variety of masonry constructions, using standard materials. To tackle the complex problem of automating the building task first, the various requirements of a robot that enables it to function in the unstructured environments had to be defined and tackled. An example of that is the automatic location of equipment (figure 1.1). The second step was to define the various stages of the building tasks, automate each, and integrate the whole process. An industrial conveyor used for material supply, as part of the experimental robot cell used for this research. This has a target and sensors mounted on it. Also vertical axis laser beacon, a mix dispensing station, a laser profiler and a gantry type robot are included. The robot has a clamp type gripper with ultrasonic sensors and displacement transducers attached to it. The gripper is used to manipulate the building materials. A detailed description of the experimental robot cell used for the research is covered in chapter 3.

Figure 1.1 shows the six stages of the building process that are defined and tackled in this research, using the experimental robot cell. The first stage involves the automatic location of the conveyor, which leads to the location of the block pick position (BPP). This is needed to enable the robot to work in the unstructured environment of the construction industry. The location determined is passed to the second stage. The second stage involves a closer search for the BPP, which is needed to enable the gripper operation of the building material. The conveyor sensing information is passed to the controlling PC for processing, and the dimensions and orientation of the blocks on the conveyor are determined and used to calculate the new block pick position (NBPP). This is required as a result of using standard blocks which have tolerances. The third stage of the building process ensure the safe gripping of the block, using the NBPP position, determined form stage 2, and the various gripper sensing information. Using the conveyor sensing information which determined the dimensions of the block, and gripper sensing, both from stage 3, the fourth stage determines the way the block is picked. This is needed to ensure the accuracy of the mortar dispensing process and the block laying operation. To enable the accurate and efficient realisation of the predetermined building project, real time adjustment need to be made to compensate for the dimensional tolerances in the blocks used. The mortar thickness applied is adjusted to compensate for the variations between the actual dimensions of the block and expected in the generated theoretical task. The exact location of the block in the gripper and the thickness of the mortar is passed to the fifth stage of the building process which is to apply the mortar on the block. The final stage of the building process is used to lay the block using the theoretical location of the block laying position from the designed building project. The theoretical task is designed using a special CAD programme. Real time adjustments are made to that position to incorporate the way the block was picked and the dimensions of the block which are taken form stage 4 and 5.

The six stages of the building process are described above. These show the various tasks that are addressed in this research and the method of integrating the various information that help to realise a theoretical building task into an actual construction. As stated elsewhere in the thesis the issue of overall mobility for the robot cell is beyond the planned scope. This research concentrates on these parts mentioned above.

Table 1.1 *Key to figure 1.1*

Parameter	Key
ϕ	Angle of the block to the conveyor
β	Angle of the robot gripper to the block
γ	Angle of the block to the gripper when gripped
(x', y')	Robot and gripper co-ordinate system
H_sensor	Horizontally pointing ultrasonic sensor on the gripper
V_sensor	Vertically pointing ultrasonic sensor on the gripper
t_b	Thickness of mortar on bedding side of block
t_p	Thickness of mortar on side of block
l	Ideal length of a experimental blocks
h	Ideal height of a experimental block
TL	Actual length of block measured using conveyor sensor data
H	Actual height of block measured using conveyor sensor data
W	Actual width of block measured using conveyor sensor data
BPP	Block Pick-up Position on the conveyor
SBPP	Safe Block Pick-up Position on the conveyor
NBPP	New Block Pick-up Position on the conveyor

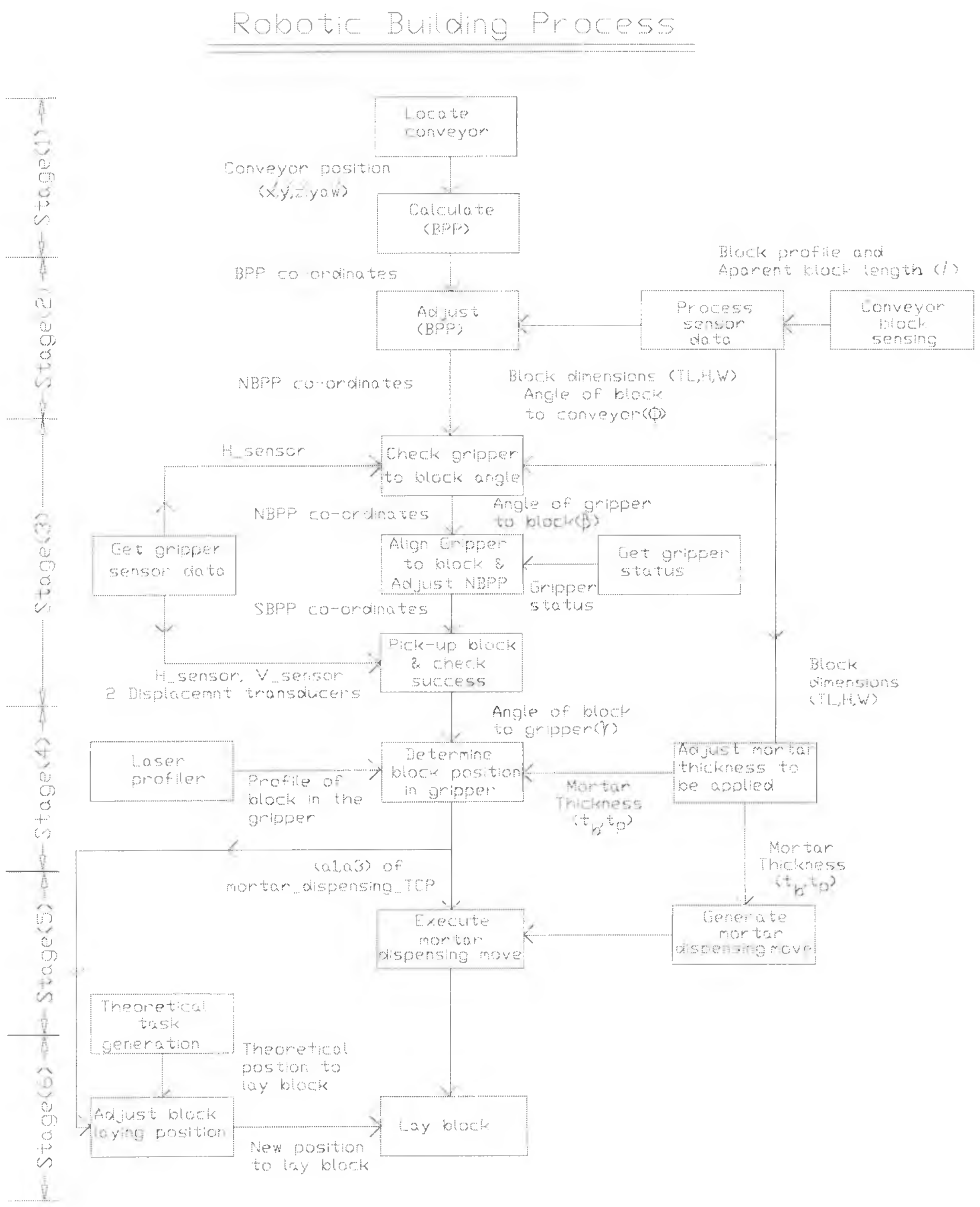

Robotic Building Process
Stage(1)
Stage(2)
Stage(3)
Stage(4)
Stage(5)
Stage(6)
Locate conveyor
Conveyor position (x,y,z,yaw)
Calculate (BPP)
BPP co-ordinates
Block profile and Aparent block length (l)
Adjust (BPP)
Process sensor data
Conveyor block sensing
NBPP co-ordinates
Block dimensions (TL,H,W) Angle of block to conveyor(Φ)
H_sensor
Check gripper to block angle
NBPP co-ordinates
Angle of gripper to block(β)
Get gripper sensor data
Align Gripper to block & Adjust NBPP
Get gripper status
Gripper status
SBPP co-ordinates
Pick-up block & check success
H_sensor, V_sensor 2 Displacemnt transducers
Block dimensions (TL,H,W)
Angle of block to gripper(γ)
Laser profiler
Profile of block in the gripper
Determine block position in gripper
Mortar Thickness (t_b,t_p)
Adjust mortar thickness to be applied
(a1,a3) of mortar_dispensing_TCP
Mortar Thickness (t_b,t_p)
Execute mortar dispensing move
Generate mortar dispensing move
Theoretical task generation
Theoretical postion to lay block
Adjust block laying position
Lay block
New position to lay block

Figure 1.1 *Stages of the building process with a key parameter list*

Progress today in the adoption of robot technology in the construction industry is largely dependent on a culture change in the industry. Piecemeal adoption of individual automation robot solutions such as masonry handling are not likely. However, with the growing use of IT in both the supply industries and the construction/building industry, a future opportunity is apparent. This research addresses elements of the solution.

The productivity expectancy in masonry automation varies significantly between researchers, figure quoted are in the range of a few seconds to 60 seconds per masonry unit. However, the fastest rates are completely unrealistic with current technology, operating intelligence and sensing systems. Also, high speeds with high payload robots are currently short of this high speed performance. To date, achievements in research indicate cycle time in excess of 1 minute. In the whole calculation of economics, it needs to be emphasised that limited aspects of productivity i.e. block laying rates is one of many considerations, the size of the work to be done at a single location, the complexity of the architectural design and requirements for special blocks and arrangements of blocks and of course the commissioning and decommissioning times are all important.

An outline cost/benefit analysis is undertaken. The main consideration of this analysis is to compare the cost of masonry construction done manually to that done using the robotic solution proposed, for on-site constructions. The cost categories include:

(i) Personnel costs, which are wages and general employment costs, such as health insurance. For the work done manually, it is estimated that the productivity rate of one person is 6 hr/m^3. Assuming the wages for a masonry worker is 12.5/hr the personnel cost for manual work is £75/m^3. For the robotic solution, we estimate productivity at 40 sec/block. Using the adopted blocks, results in a rate of 1.11 hr/m^3 for a single worker. This results in wages cost of £13.8/m^3.
(ii) Materials costs i.e. masonry blocks and mortar, which are common in both cases.
(iii) Cost of machinery used in all phases of the work. For manual work, this will mean only a mixing unit. Here, two types of costs need to be considered, (i) the cost of masonry machinery per m^3, related to productivity. For manual work this is taken as £0.5/m^3, and £30/m^3 using the robot. These figures are estimated by taking into consideration the depreciation values for the equipment over five years and the volume of work which can be constructed in this time. (ii) cost of technology, which is used for

the robotic solution. This is estimated for the CAD work satiation and other computer software used, to be around £4.5/m^3.

(iv) Transportation costs for machinery and workers need to be considered for the robotic solution. This clearly depends on distances, but we assume a cost of £200/job. No transportation cost will be needed for the work done manually.

(v) Cost of setting up equipment on-site and calibration of the automation machinery.

(vi) Maintenance cost of machinery, for robotic work, is estimated to be £9000/year. This will be equivalent to £36/day for the robotic work, assuming there are 50 working weeks in the year. There is no maintenance cost for manual work.

(vii) The equipment capital cost for the manual work is estimated to be £1000 and £150,000 for the robotic solution. Interest cost, as a result of using credit payment for the equipment, is £40/year for equipment used for manual work, and £6000/year for equipment used for robotic work. This will be equivalent to £0.1/day, for manual equipment, and £16.5/day, for robotic equipment, as interest payment.

When a full cost analysis is made the prerequisite would be to define the type, size and price of blocks to be used, which effects the number of bricks/blocks per m^3. Taking into consideration, that the robot can handle heavier and larger block than humans. From the production rates mentioned above, an estimate of the overall volume of block that can be laid per year is carried out. This will be 505 m^3/year for work done manually and 1369 m^3/year using the robot.

A cost analysis of a typical block building project is now calculated. Assuming the project is to build 5 accommodation units each of dimensions 10m $\times$10m wide, and 7m in height, using 0.1m width blocks, we calculated the project as follows:

Total wall surface area per unit = (10m+10m+10m+10m) $\times$7m $\times$10m = 280m^2.

and,

Total masonry volume of 5 units = 5 $\times$ 280m^2 $\times$ 0.1m = 140m^3 of blockwork.

Using the cost values stated above, the time and cost values for this project are computed for manual and robotic work. The time to complete the project will be, Manual work = ($6hr/m^3 \times 140m^3$)/ 8hr/day = 105 days

and,

Robotic work = (1.11 hr/$m^3 \times 140m^3$)/8hr/day = 19.5 days

Using the times calculated to complete the project other costs are calculated as shown in table 1.2.

Table 1.2 Cost analysis of a block building project

Cost considerations	*Manual work*	*Robotic work*
Days needed to complete the work	105 days	19.5 days
Personnel cost	£10,500	£1932
Machinery cost	£70	£4200
CAD technology cost	NA	£630
Machine maintenance cost	NA	£702
Interest on credit value	£10.5	£322
Total cost of project	**£10,685.5**	**£7805.5**

In this example (table 1.2), it can be seen that it is more cost effective to use the robotic solution. Where smaller elements of work occur, however, the robotic solution would tend to loose its advantage.

1.4 Building materials and methods

1.4.1 Brick types

In this section we discuss the various bricks and blocks used in masonry work. Bricks are still the most popular form of walling unit for domestic construction. Their size and variety of colours and textures make them attractive architecturally. Bricks varying in the materials used, method of manufacturing and form. These are considered below.

BS 3921[22] classifies clay bricks and blocks in three different ways,

(i)Variety and functions:-

(i) Common: Suitable for general building work but generally poor appearance.

(ii) Facing: specially made or selected to give an attractive appearance.

(iii) Engineering: dense and strong semi-vitreous to defined limits for absorption and strength.

(ii) Quality:-

(i) Internal: suitable for internal use only.

(ii) Ordinary: normally sufficiently durable for external use

(iii) Special: durable in situations of extreme exposure.

(iii) Types:-

(i) Solid: not more than 25% small holes.

(ii) Perforated: small holes exceeding 25%.

(iii) Hollow: large holes exceeding 25%.

(iv) Cellular: holes closed at one end exceeding 20%.

The standard brick size is 215 ×102.5 ×65mm. Modular bricks are also produced, ranging from 288 to 190mm long ×90mm wide ×90 or 65mm high.

The clay bricks are made from clay composed mainly of silica and alumina, with small amounts of lime, iron, manganese and other substances. The different types of clay produce a wide variety of colour and textures. Other types of bricks apart from clay are Calcium silica bricks which are made form sand and lime. There are also fire bricks which are made from refractory clay with a high fusing point. The most commonly used bricks are 'Flettons' These are made from oxford clay and are produced in large quantities. They have a hard exterior skin but are relatively soft inside, and are usually used for facing. 'Stocks' are mainly yellow in colour and made from London clay, and are good all-purpose bricks specially facing work. 'Staffordshire blues' are bluish-grey engineering bricks made form clays containing iron oxide and are burnt at high temperature. These are suitable for positions requiring great strength. 'Southwater reds' These come from Sussex and have similar properties to Staffordshire blues but red in colour. 'Facing brick' they may be machine-made or hand-made to a variety of colour and textures.

1.4.2 Block types

There are two types of blocks, concrete and clay. The British Standard recognises three types of block: solid, hollow, and cellular. They are made to various size, a commonly used size being 440 ×215mm with a wide range of thickness. The Standard give a minimum strength for all block types. Where sever exposure or pollution is likely, blocks should have an average compressive strength of not less than 7 N/mm^2, but blocks lower strength can be protected by rendering. As for Clay blocks, these are hollow with keyed surfaces and have a high thermal insulation value.

1.4.3 Mortar

Brick and blocks are bedded in and jointed with mortar. A good mortar spreads readily, remains plastic while bricks are being laid, to provide a good bond between bricks and mortar, resist frost and acquire early strength. Mortar should not be stronger than necessary, as an excessively strong mortar concentrates the effects of any differential movement in fewer and wider cracks. In extreme cases, the bricks and blocks may be fractured rather then the mortar joint giving movement.

There are a number of mortars in use (i) Lime mortar: this type of mortar develops strength slowly, and it is widely used, (ii) Cement mortar: this type of mortar is workable but not suitable for everyday use, apart from heavily loaded brickwork, (iii) Cement-lime mortar: this is the most useful general purpose mortar. The best properties of both cement and lime are utilised to produce a mortar which has good workability, water-retention and bonding qualities, and also develop strength without an excessively high mature strength, (iv) Air-entrained mortar: this has a mortar plasticiser which entrains air in the mix, providing an alternative to lime for improving the working qualities of lean cement-sand mixes, (v) Masonry cement mortar: this consists of a mixture of Portalnd cement, very fine mineral filler and an air-entraining agent and (vi) Mortar containing special cements: this has a high alumina cement that may be used where high early strength or resistance to chemical attack is required.

Weak mortars should always be used with bricks or blocks of high-drying shrinkage. The mortar should not contain more cement than necessary to give adequate strength in the brickwork.

1.4.3.1 Pointing and jointing

Mortar joints may be finished in a number of ways after the brickwork is completed. Where the work is carried out while the mortar is still fresh, the process is called 'jointing'. When the mortar is allowed to harden, and then some removed and replaced with fresh mortar, the process is termed 'pointing'. The five main types of finish are illustrated in figures 1.2-1.6.

'Struck flush', illustrated in figure 1.2, gives maximum strength and weather resistance to brickwork, but may appear irregular with uneven bricks. 'Curved recessed', illustrated in figure 1.3, gives a superior finish to struck flush, but with little difference in strength or weather resistance. 'Struck or weathered', illustrated in figure 1.4, produces interplay of light and shadow on brickwork, but less strength and weather resistance than the previous two finishes. 'Overhung struck', illustrated in figure 1.5, produces variation of light and shade, but weakens brickwork and provides a surface on which rainwater may lodge, and this may result in discoloration or frost damage. The last type is 'Square recessed', illustrated in figure 1.6, which should only be used with very durable bricks, as it offers less strength and weather resistance, in spite of good appearance.

Jointing is quicker and cheaper than pointing and, as the surface finish is part of the bedding mortar, there is less risk of the face joints failing through frost action or insufficient adhesion. On the other hand, it may not look as attractive as pointing, and it is difficult to keep work clean and of uniform colour.

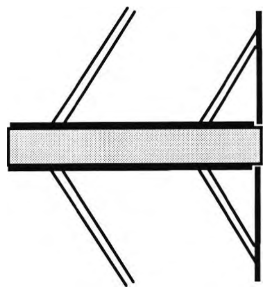


Figure 1.2 *Struck flush*

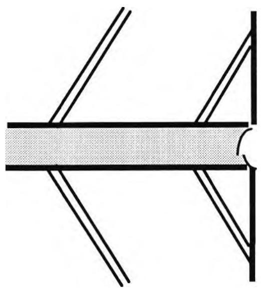


Figure 1.3 *Curved recessed*

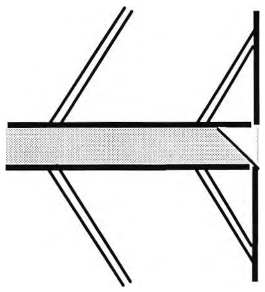


Figure 1.4 *Struck or weathered*

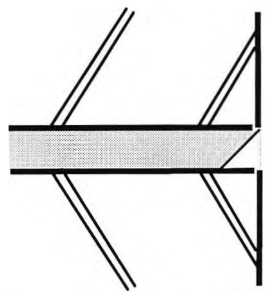


Figure 1.5 *Overhung struck*

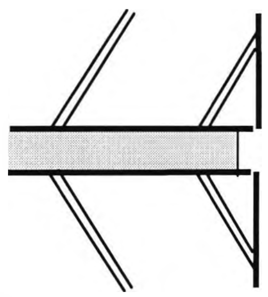


Figure 1.6 *Square recessed*

In the research, only concrete blocks have been used in small bonded assemblies. The use of brick, whilst feasible with the gripper design adopted, is not considered. In respect to jointing/pointing, this too is not within the scope of the research and, therefore not addressed. Regarding mortar strength, CP110 indicated a shear strength of 0.21 N/mm^2 for the mortar/unit interface.

1.5 Building brick and block walls

Bricks and blocks are bonded to give maximum strength and adequate distribution of loads over the wall. Bonded walls provide lateral stability and resistance to side thrust, and the bond can be selected to give an attractive appearance to the wall face.

1.5.1 Types of bond

The choice of bond is influenced by the situation, function and thickness of wall. For instance with external walls to buildings appearance is often important, whilst with manhole walls strength would be the main consideration. Below is a list of the various types of bonds.

(i) Flemish bond: This is the most commonly used bond for solid brick walls as it combines an attractive appearance with reasonable strength (figure 1.7). The term ‘double Flemish’ is used where the bond is used on both faces. Bricks are laid as alternate header and stretchers in the same course above and below.

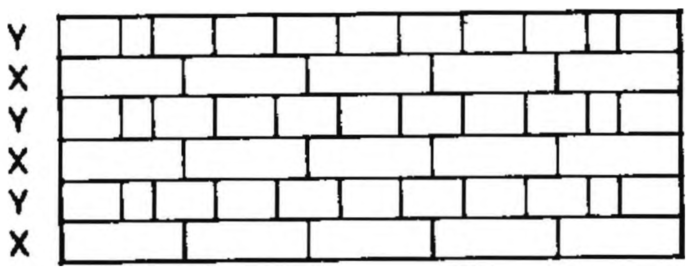


Figure 1.7 *Flemish bond. Elevation of wall with corner and stopped end.*

(ii) English bond: This consists of stretchers throughout the length of one course and headers throughout the next course (figure 1.8). It is rather stronger than Flemish bond because of the absence of internal straight joints, and is particularly well suited for use in manhole and retaining walls.

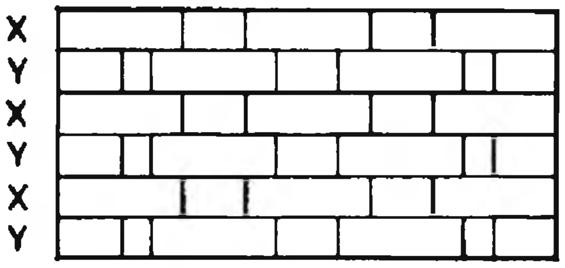


Figure 1.8 *English bond. Elevation of wall with corner and stopped end.*

(iii) Garden wall bonds: These are used to reduce the number of headers and the cost of facing bricks, yet at the same time to produce a wall of reasonable appearance and strength. English garden wall bond is made up of three courses of stretchers to each course of headers (figure 1.9) and Flemish garden wall bond consists of three stretchers to the header in each course, and these are well suited for one-brick boundary walls.

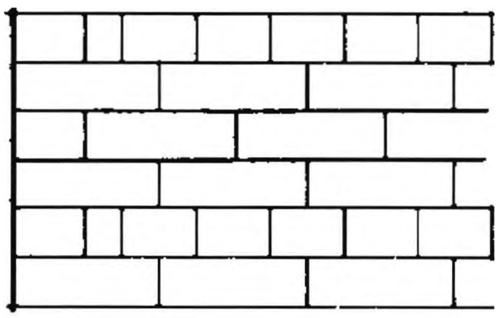


Figure 1.9 *English garden wall bond*

(iv) Stretcher bond: This consists of stretchers throughout and the centre line of each stretcher is directly over the centre line of the cross joint in the course below (figure 1.10). This bond is used for half-brick walls, including the leaves of cavity walls.

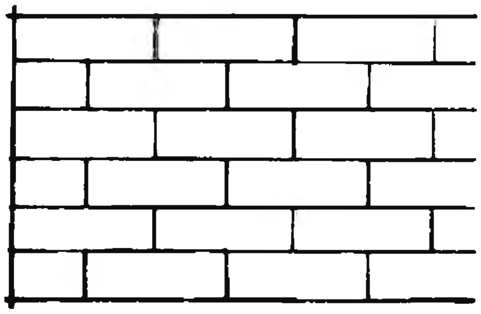


Figure 1.10 *Stretcher bond*

(v) Rat-trap bond: This is used on one-brick walls to reduce the weight and cost of the wall (figure 1.11). The bricks are laid on edge and a series of cavities are formed within the wall. If used externally, the outer face would normally be rendered.

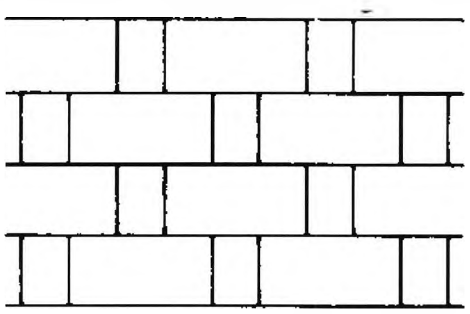


Figure 1.11 *Rat-trap bond*

1.5.2 Cavity walls

Solid brick walls of the thickness generally used in domestic construction are not always damp-proof. Moisture may penetrate the walls through the bricks, joints or through minute cracks which form between the bricks and joints as the mortar dries out. In addition the thermal insulation value of solid walls is limited. These disadvantages can be overcome by using cavity or hollow walls, consisting of an air space between two leaves with upper floor loads supported by the inner leaf. The building regulations in Approved Document A1/2 require the two leaves to each be not less than 90mm thick and to be provided with adequate lateral support by roofs and floors. The cavity should not be less than 50mm in width.

Whilst modern brickwork, with brick/block cavity wall construction, necessitates use of the stretcher bond, a number of classical bonds, some ancient, do exists. With further research, beyond that proposed, a 'flexible' robot cell, any bond design could be achieved.

1.6 Thesis structure

Chapter 2 covers a review of construction robotics. In chapter 3, a detailed description of the experimental equipment is given and the experimentation methodology. The supporting theory and its software implementation is also explained. A method investigated for block picking is covered in chapter 3, along with further details of the equipment used. Further details of the thesis structure is given in chapter 3.1.

The theory and programming of the different move types of the robot is covered in appendix H. The system's limits and capabilities are discussed, and trial results given. These moves and system limits are applied to examine the vibration of the robot's end-effector, using the outcomes of chapter 2. These experiments are covered in chapter 4.

The problem of equipment location is dealt with in chapter 5. The search algorithm developed for this is described and tested. The accuracy and efficiency of the method, the algorithm tested, and the software implementation are covered.

Experiments on methods for sensing block dimensions as well as picking-up a block safely from the conveyor are covered in chapter 6. Experiments were carried out to determine the accuracy of block picking method, and these are described in chapter 7.

Chapter 8 gives detailed analysis of the work done in the area of masonry bonding, together with the experiments and results achieved.

The three different rule-based expert systems, developed within this research, are described in chapter 9. This includes a description of the move optimising rule-base, which utilises the results from chapter 4. The expert system that produces the order of building of the designed project, produced using a special CAD programme, is described. Also, the run time expert system that is integrated with the robot sensing and functionality, which is used to pick up a block, is described along with the experiments that demonstrate its functionality.

Chapter 10, the conclusion, covers the achievements of this research, discussing the findings of the project and making recommendations for future work.

Chapter: 2 Construction robotics

2.1 Background to construction robotics

It is claimed that automation in the construction industry has been delayed because of the various characteristics of different buildings (Wen and Rovetta, 1991). The industry is striving to eliminate dangerous and repetitive work, as well as increase quality and productivity in the various tasks. The implementation of robots in factory industrialisation has brought many advantages, such as increased productivity and quality, relieving workers from tedious, hard work and dangerous environments, as well as making savings in material waste. The requirements of robots for construction are different from those of industrial robots, due to the characteristics of the construction tasks and the unstructured environment of construction site. There is also a need for integrating the design and implementation phases to streamline the entire construction process. The application of CAD-CAM would play an important role in the mechanisation of construction projects.

Many prototype robots have been developed over the past decade (Everett and Slocum, 1994), however, only a few examples are found working on the construction site today. One example of this, is the fireproofing Spraying Robot (SSR-3) of Shimizu Corp., Tokyo, the first construction robot to be built and used on the construction site.

Even though machines are superior in terms of physical strength in carrying out intensive tasks that require speed and repetitive motion, skilled human workers often remain more productive and cost-effective than machines for the skill-intensive tasks. At present, the best prospects are for man-machine systems incorporating robotic subsystems (Chamberlain, 1997). For construction robotics to become feasible progress needs to be made in the application of sensors and artificial intelligence to the practical issue of construction tasks.

In a recently published survey, carried out for the last 15 years of building robots that have been developed and tried on site, (Warszawski and Navon, 1996) it is shown that the technology remains at a very preliminary stage. From the various applications of building robots, 6% were involved in brick/block masonry work. Of the robots surveyed, only 4% were pre-programmed with sensors and 3% with intelligent planning. An encouraging figure was noted that 29% of the robot surveyed were actual robots employed on site, of which 14% were actually commercially available, and the rest at initial stages of development, or prototypes. From those used, around 33% of them relied on human operation by remote control. This lack in progress towards take up of advanced robotics can be explained by the difficulty in the financial justification, as the benefits of improved quality are not easily apparent to the construction industry. Companies tend to look for easily predictable increases in productivity, difficult to quantify when justifying the high levels of capital expenditure needed to develop the end product, with no guarantees of the success.

Robotics technology has not yet been suitably adapted to the environment of building sites. These tend to be very dusty environments, potentially reducing the life of the robot, as well as having a rough and undulated terrain. In some cases, the same robot will need to conduct more than one task, therefore it must travel over a wide area. This leads to unproductive time as the robot travels to new locations, and is set-up for the new task, further reducing the potential cost savings. To facilitate this mobility, robots may be put on tracks, or be given long reach (ROCCO Andres *et al.*, 1994). Such a long arm robot will give rise to new challenges, such as reduced stiffness which leads to increased levels of vibration and the possibility of inaccuracies of positioning, unless the robot is given sufficient time to settle at the end of a movement or active damping control employed. This, in turn, will also reduce productivity, as the time for productive tasks is lost. The robot used in the authors research has this type of problem. Vibration about the vertical jaw axis, which is elasticity compliant, tends to spoil the potential productivity. In this research we try to find the optimal speed for no significant vibration at the effector end i.e. minimum end of move settling time. This way, the robot effectiveness can be maximised. Whilst some form of active damping might be feasible through control, this was thought unrealistic and not possible within the resource restraints of the work. This is valid as the research outcome are applicable to existing robots, few of which have the capability for such control.

2.1.1 Automating masonry tasks

An important area for automation. that has been identified by the construction industry, is masonry building. There are several reasons for this, including, labour shortage and required improvements in consistency in quality (Bernold *et al.*, 1992). Masonry work is also considered to be physically damaging to workers. This is apparent in a study on back injuries in construction (Rihani and Bernold, 1994), which showed that permanent back injuries can be caused by continuous lifting and carrying of masonry units. This leaves employers open to law suits in the future for damage done to employees. In the EU, this problem is being tackled through laws, limiting the weight workers are allowed to carry at various heights from the floor. The construction industry in the UK is noted to be undergoing an increasing shortage of skilled workers, as well as not appealing to the younger generation (Garas, 1991). In the UK, the labour content in masonry construction has a high value (Chamberlain, 1991b). Other factors, are that it is a highly repetitive and labour intensive task (Chamberlain *et al.*, 1992), as well as an important building element with high value and architectural significance (Chamberlain *et al.*, 1993). All these points contribute to it being a natural target for automation.

In Russia, masonry work is both the most common and laborious process in civil engineering (Malinovsky *et al.*, 1990). In Germany, there is a noted decrease in skilled workers, with no new young people joining the construction industry (Andres *et al.*, 1994), mainly due to the poor working conditions of construction sites. Addressing the need for increasing production in building brick work (Bock *et al.*, 1993) has prompted the introduction of advanced technology into the field of masonry construction in Germany.

2.2 Development in masonry automation

At the Massachusetts Institute of Technology, a design has been developed for a robot to dry-stack precision cement blocks in building walls, which are then surface bonded (Slocum and Schena, 1988). This robot, Blockbot, shown in figure 2.1, is mounted on a mobile scissors lifts, which moves along the wall as it builds. The materials would either be placed on the robot vehicle, or be delivered by an extendible conveyor assisted by human workers who feed it with blocks from pallets. The study estimated a target speed of 8 blocks per minute for building walls with no openings. This system has not been built, it remains a concept.

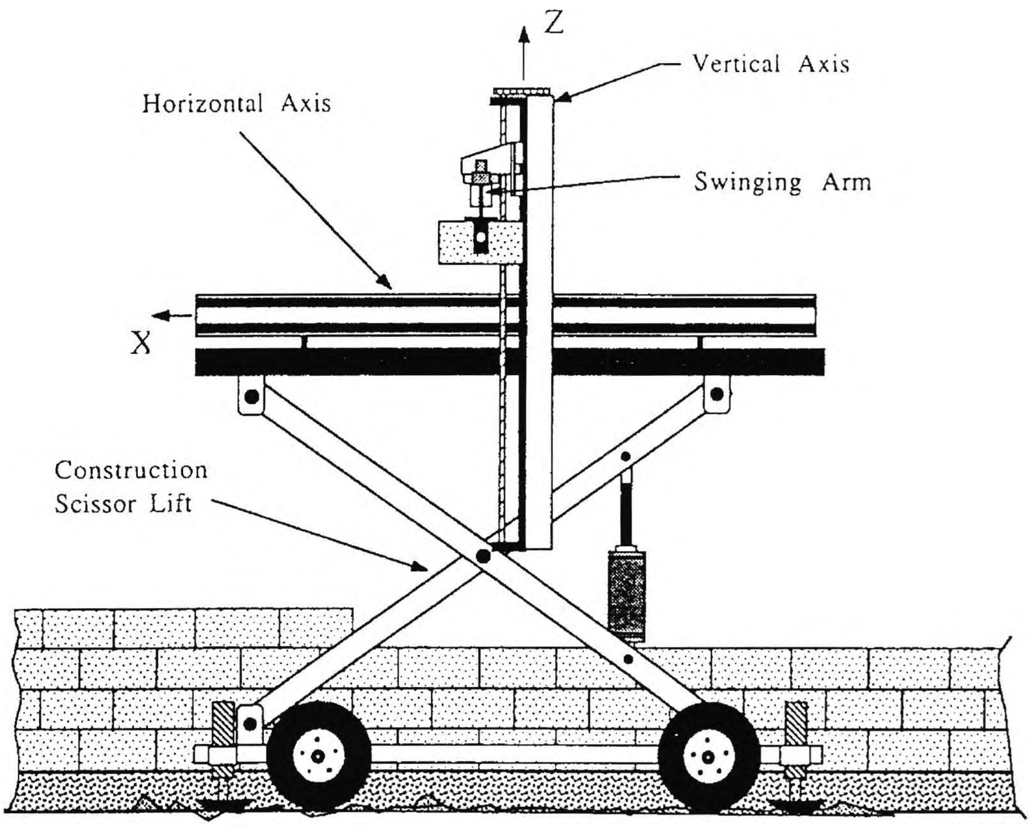


Figure 2.1 *Schematic side-view of Blockbot (Slocum and Schena, 1988)*

Another design concept was worked out at the University of Illinois (Muspratt, 1988), for the brick/block wall building robot, brickbot, illustrated in figure 2.2. The idea is to achieve an automated, on-site mobile factory for masonry building. The main feature of this robot is a X-Z plane mobile robot rack, that is relocated as each wall section is built.

The rack has four different manipulators mounted on it, each carrying out a different function. The first manipulator delivers a X-bead of grout on the top surface of a completed course. The second is fitted with a U-head suction gripper to lift the blocks from the conveyor and place them into position. The third is fitted with a grout nozzle to fill the vertical joints. Finally, the fourth is fitted with a rotary brush to clean the excess grout from the exposed surface of the completed course. Standard, whole and half bricks are used, with the suggestion of placing bar codes on each, to facilitate identification and orientation, as well as providing an aid for inventory control. Again this system has not been built, it remains a concept.

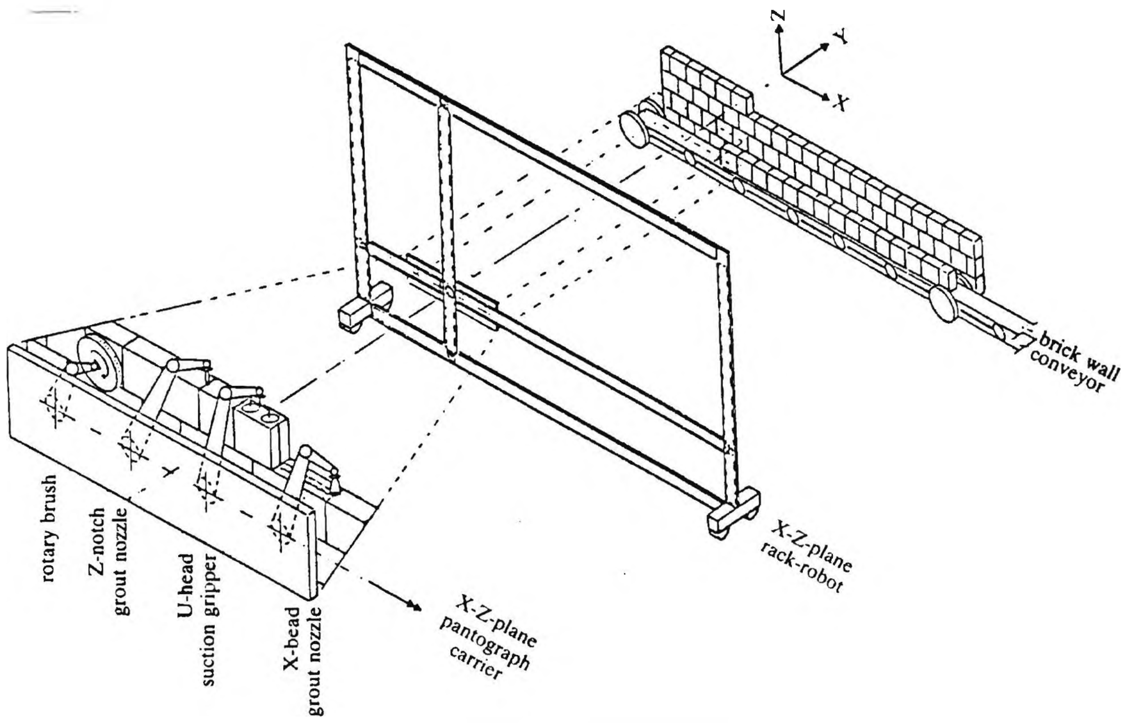


Figure2.2 *Robot ensemble for wall construction (Muspratt, 1988)*

In Finland, two concepts were outlined (Lehtinen *et al.*, 1989), to determine the functional requirements, and technical specifications of masonry robotics. The first was based on a SCARA type robot with 6 Degrees Of Freedom (DOF), for building partitioning walls using lime sand bricks with restricted dimensional tolerance. A glue-like material was used for bonding, in which the bricks are dipped. The system is composed of a de-palletising gantry robot, a conveyor, a brick lift and a moving masonry unit. Similar to Blockbot, a carriage with positioning devices is used, which is the base of scissors lift. The mortar tray, cutter station and SCARA robot are all mounted on the carriage. The aim is to run a few tasks in parallel, allowing an estimated cycle time of 10 seconds for each block. External positioning measurement systems and sensors connected to the wheels are used to position the carriage. The second system is based on a standard industrial robot with 6 DOF. Two pallet bricks are mounted on it as well as a cutting unit and a mortar unit. These two systems have not been built.

A study of the tasks that mobile robots can be utilised for (Spee, 1989), included a mobile robot used for masonry construction. This is a semi-automated system, developed and built by a small German firm, and comprises a flexible handling tool that assists the human worker in the physically demanding part of the building task. Using hollowed concrete blocks, with dimensions of 100 cm x 65 cm, the robot is controlled using a wireless communication device to carry out the picking and placing of blocks. Another German project (Bohm, 1991) also developed a semi-automated machine that focuses on the human operator. This machine, called the Mason's Elevator Handling Machine (MEHM), comprises a platform running on wheels, that is controlled from the bricklayers workplace. The machine's platform holds the masonry blocks and mortar, and is moved along the wall being built. The machine has a handling unit, controlled by the bricklayer, which grips the masonry blocks. Using this, the bricklayer manipulates the blocks with no physical effort when dipping the bricks into mortar and laying them. This system has no robotic element.

In Israel, a project has been pursued that adapted readily available industrial robots for interior finishing works (Rosenfeld *et al.*, 1990). The mobile robot has 6 DOF and is mounted on a three-wheeled semi-guided platform, with a vacuum gripper attached to the end of the robot's arm. By dry stacking lightweight gypsum blocks with interlocking edges, with dimensions of 70 cm x 50 cm and 27 kg each, the robot builds interior partitions that are later strengthen by the application of a special plaster. The robot's tasks are pre-programmed, and uses touch sensors (Rosenfeld *et al.*, 1991) attached to the robot's arm to calibrate the robot's position relative to it's workspace.

A Russian investigation (Malinovsky *et al.*, 1990) proposed a concept for a brick laying complex. The complex comprised of a brick layer machine and a mortar extruding machine with feeder. This is placed on a revolving portable cantilever robot. A crane loader would be used to supply the brick pallets, and a mortar generator to supply the mortar continuously. The blocks are placed on the feeder and moved under the mortar extruding machine, which applies a thin layer of mortar on their top surface. The brick layer then grips and rotates the bricks 180° ready to be laid. The study estimated an increase in labour productivity of 7 to 15 times. This project tried to present a complete solution, while neglecting the details and the complexity of the building task. Part of the designed concept was realised, which was the application of a layer of mortar, extruded on a continuous line of bricks delivered by a conveyor.

In the UK, efforts to develop a gantry type robot at City University started in 1989 (Chamberlain *et al.*, 1990). The study carried out by the author is part of this effort. The main aim is to provide a technologically advanced robotic system for masonry construction, using standard materials. Working with imprecise bricks/blocks requires the use of thick bonding materials to compensate for the dimensional variations. For sensing, a vision system was developed to inspect materials to detect defective bricks/blocks. A CAD model of the masonry units was used to guide the inspection process.

Another German project (Anliker F., and Anliker M, 1991) has developed a partially automated system for the production of pre-fabricated wall panels using standard mortar. The system, which has been built, relies on the operator to lay the blocks, following drawings of wall plans. After three days of hardening, the assembled walls are transported to the construction site to be put together. The system, which is aided by two workers, has proven to be effective in increasing the speed of construction (60-70 m^3 wall per day). The research (Bley and Anliker, 1994) has the aim of integrating the design phase with the building phase, utilising a CAD-CAM system. This system is intended to allow the design of walls that are then broken down into detailed sub-units of the production process. A programme is used to calculate the number and size of blocks required, which is fed into the production computer. The production plant process begins with an external gripping device, that picks the block from pallets and places them on a conveyor. They are then put on to a positioning belt, after the bricks are cut to size, according to the CAD generated data. Once a complete layer is finished, it is clamped and turned over onto a turn over table, to be integrated into the wall being built.

At the North Carolina State University, a prototype bricklaying robot was developed and used for testing the feasibility of the automation of the bricklaying process (Bernold *et al.*, 1992). The frame work of the study was based on the concept of a flexible manufacturing machine (FMS), which consists of a computer controlled material handling machine. The study attempted to automate the basic operation of masonry work, from the preparation of units and application of a thick layer of mortar to the laying of the bricks, using standard materials. A pilot system was developed and built which consisted of a mortar spreader, a conveyor belt and a Cartesian manipulator for the block manipulation. Further work on this was carried out at the University of Maryland (Altobelli *et al.*, 1993), where a scaled prototype was used for handling actual materials. Experiments were carried out using these prototypes to determine the bond strength achieved when automating the process of the mortar application, as well as the laying of the block. The results of these tests are discussed in chapter 9. A concept system for the building of pre-fabricated brick panels was also presented as part of this. The system, illustrated in figure 2.3, comprises a semi-fixed production unit with a gantry robot that moves on rails. At one end of the work centre the masonry units are depalletisied, cut and placed on a conveyor belt that supplies the building centre. The building centre comprises a mortar mixer, spreader and the placing robot. This system remains a concept.

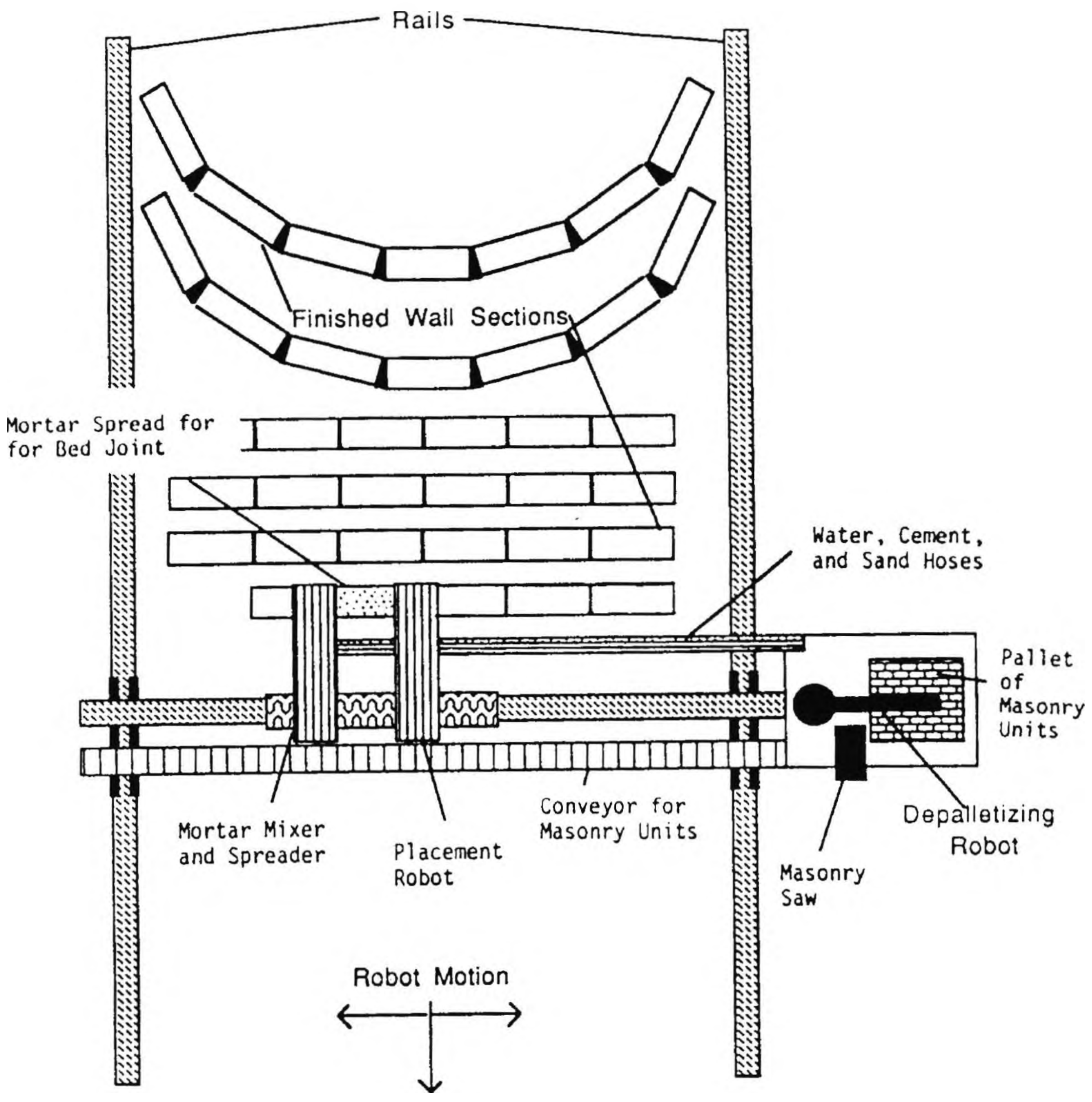

Rails
Finished Wall Sections
Mortar Spread for for Bed Joint
Water, Cement, and Sand Hoses
Pallet of Masonry Units
Depalletizing Robot
Mortar Mixer and Spreader
Placement Robot
Conveyor for Masonry Units
Masonry Saw
Robot Motion

Figure 2.3 *General concept for robotic masonry system (Altobelli et al., 1993)*

In Luxembourg, the FAMOS bricklaying system for lining steel converters with bricks has been developed within a EUREKA project (Wurth, 1992). It comprises a SCARA type robot operating on a working platform inside the converter. Additional components include a depalletising robot, an elevator with a brick feeding mechanism, and a brick centring device. The system assists the worker by transporting the masonry units and by extruding mortar on the brick which he lays manually.

A German system, at the University of Stuttgart, has been developed in the light of the national characteristics of masonry construction in Germany. It is a mobile robot for on-site construction of masonry and takes into considerations the need for adaptation to the different design requirements of clients, using materials with small dimensional tolerances. Their initial work concentrated on the requirements of the control system of the robot (Pritschow *et al.*, 1993), which was designed to enable it to interface with the CAD generated data of the building projects, as well as provide adequate sensor data processing capabilities. The robot being developed is based on a commercially available construction machine that is light weight and compact. In a later paper (Pritschow *et al.*, 1994) the requirements of the robot were discussed. It results in a 7 DOF robot with external sensors that provide the robot control system with information about the position and orientation of the mobile platform and the TCP. An update on the progress of this suggested system was covered in a later paper (Pritschow *et al.*, 1995). It proved possible to use thin-bed mortar as a bonding material after selecting blocks manufactured to tolerances of ±1 mm. The blocks are picked up from pre-positioned pallets using a vacuum gripper, and then positioned using a multi-functional technology unit, mounted on the robot carriage. This unit helps to correct the way the blocks are gripped, it measures the block and accommodates the application of a thin layer of mortar. The developed robot, shown in figure 2.4, has been used to build a few assemblies.

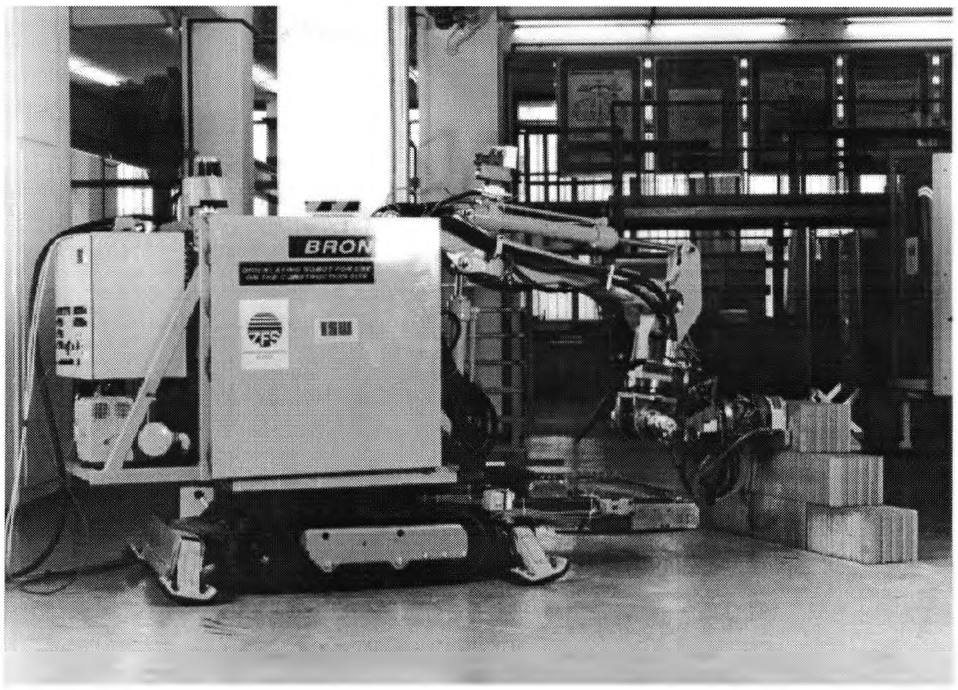


Figure 2.4 *Mobile robot for masonry construction at the University of Stuttgart*

A robot assembly system for computer integration construction, ROCCO, has been developed as part of a European ESPRIT III project. The concept of the system (Bock *et al.*, 1993), is for the prefabrication of small building components, based on the idea of a computer integrated construction system. The aim is to integrate the different tasks of the building process, from the design and planning to the execution of the construction work. Quality control of masonry units is carried out using sensors. To level out and compensate for the inaccuracies of the floor, the first layer is built manually. The conceptual structure of the robot has been presented (Andres *et al.*, 1994) with a description of the need for assembly tools with positioning devices for fault compensation. Blocks of dimensions 50 cm x 25 cm x 25 cm are used. Emphasis is given to the work preparation phase, which includes the generation of the data based on a CAD-representation of the building, the construction site layout, and the robot programme (Bock and Leyh., 1995). The use of commercially available, cavity blocks, with ±0.5 mm dimensional tolerances, enabled the application of a thin layer of mortar. The hydraulically driven robot, which is mounted on a mobile platform, is a 6 DOF articulated manipulator with a reach of 8.5 m and payload of up to 500 kg. Their first on-site, trial experiments are reported to have confirmed the robot's ability to erect walls. To verify the position of each axis, digital positioning sensors are used (Gambao *et al.*, 1996). A laser based position sensor is also used to provide information about the robot TCP. A Man Machine Interface has been developed to automatically generate the systems commands and sequence from the CAD building design (Balagure *et al.*, 1996). This also permits graphical simulation of the robot's tasks. Expert knowledge rules are used for the planning of the robot's work.

A study of the critical issues of automating the masonry building process was carried out at North Carolina State University (Rihani and Bernold, 1994). The basic design concept is of an Experimental Robotic Masonry System (ERMaS), illustrated in figure 2.5. The main components of the system are the gantry frame and a brick placement arm with a force/torque sensor attached to it. A mortar pump, pneumatic brick gripper and brick supply conveyor are included in the system. A sensor controller is connected to the PC that is used for commanding the brick manipulation. This concept system was not built, but instead a prototype sub-system were built to test the technologies of mortar extrusion and the effectiveness of automation of brick placement. The mortar application tests showed that, by varying the pump speed, mortar thickness can be controlled with sufficient precision. As the project assumes to be working with conventional materials, this factor was considered critical when adjusting the varying block lengths in the mortar bed. The results of these experiments are discussed in further detail in chapter 9.

None of the reported robotic systems are fully functional, as yet, but progress has been made in the last five years in solving the many complex masonry building tasks. The only commercially functional project reported, is the pre-fabrication of walls in Germany.

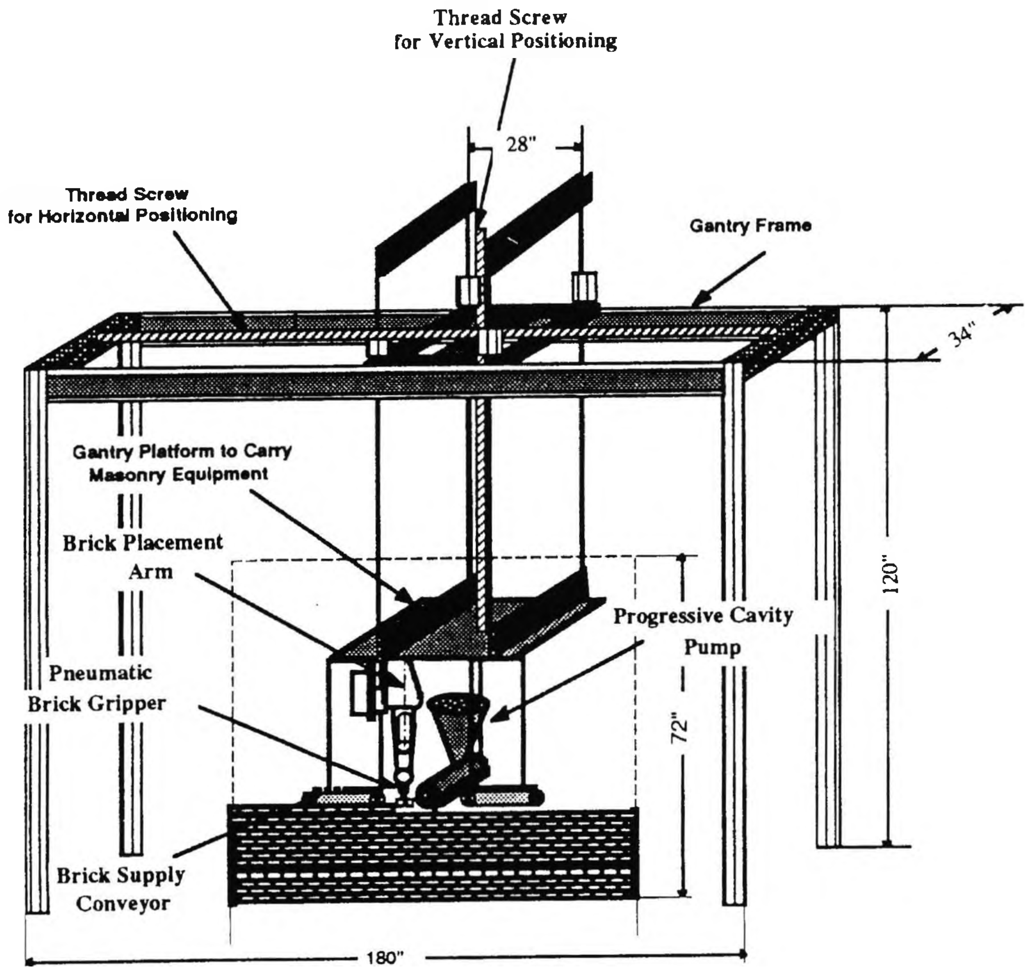

Thread Screw for Vertical Positioning
28"
Thread Screw for Horizontal Positioning
Gantry Frame
34"
120"
Gantry Platform to Carry Masonry Equipment
Brick Placement Arm
Progressive Cavity Pump
Pneumatic Brick Gripper
72"
Brick Supply Conveyor
180"

Figure 2.5 *Concept robotic masonry system (Rihani and Bernold, 1994)*

Chapter 3: Materials and Methods

3.1 Introduction

The background work of this research is covered in this chapter. The scope and limitations of the research as well as any assumptions taken are also discussed. In this chapter the equipment making up the experimental robot cell, that was designed, built, and tested (figures 3.2 and 3.1) is described in detail. The methods of use, along with the theory developed and reasons for the equipment's choice and design are also covered. In the introduction the theory of the experimental methods developed and a brief description of the experimental cell design was introduced, explaining the overall method of integrating the various hardware and software.

The experimental robot cell is shown in figure 3.2. A 5 m long industrial conveyor, with sensing provisions on it, was used for the delivery of building material, as well as checking for defective units. The conveyor has a target mounted on it, which enables automatic location of the conveyor. The robot has a clamp type gripper, used for the picking-up and placement of masonry units. Two ultrasonic sensors and two displacement transducers are mounted on it, to ensure the safe manipulation of the units, as well as the location of the target on the conveyer. A laser profiler was used to provide an independent check of the position of a picked masonry unit in the gripper. This was carried out to calculate the necessary adjustments needed for applying mortar and laying the blocks in its theoretical position. The mix dispensing station comprised a peristaltic pump, a mix feed line, an extruding nozzle and a control box which communicates with the main PC via a parallel port. Mortar dispensing is intended to be fully automated. The robot was used to provide the motion and manipulation necessary for the application of mortar on both sides of a block. A single pass move was designed for the dispensing operation, its theory covered in this chapter. A vertical axis laser beacon was designed and built to provide horizontal referencing. It was not used in this study. The vibration of the robot's end effector was measured using a miniature, heavy duty accelerometer mounted on the end effector. The method developed to process the accelerometer data from the acceleration-time domain to the displacement-time domain is described and verified in this chapter.

Experiments are carried out to test the accuracy and effectiveness of the equipment design and the method of use in this project. In chapter 4, experiments carried out to test the vibration of the end-effector when performing different move types, using an accelerometer, are covered. The experiments carried out to test the effectiveness of the conveyor target design, and a description of the search algorithm developed to locate the target, are covered in chapter 5. Chapter 6 covers the experiments carried out to test the accuracy of the methods developed to measure block dimensions, pick-up a block safely. In chapter 7, various tests are carried out to check the efficiency of the laser profiler's determination of the block position in the gripper. Also, the effectiveness of the use of that information, when making the necessary adjustments for the block laying operation. The dispensing move is tested by experiments done on the construction of small constructions of blocks. This is covered in chapter 8. A rule based expert system, developed to pick-up a block safely, is described and tested in chapter 9. This rule base integrates the rules used with the various 'C' functions developed in this chapter, and uses the various sensing data.

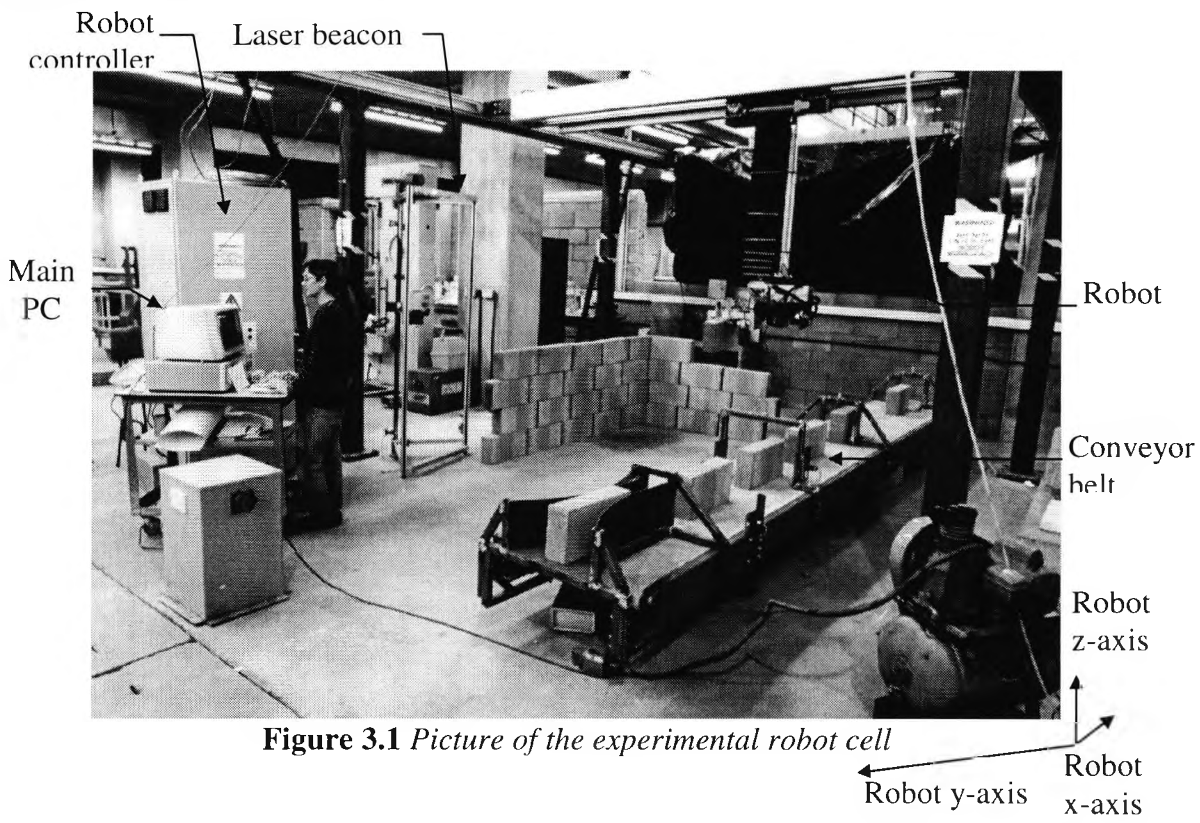


Figure 3.1 *Picture of the experimental robot cell*

3.2 Scope of the research

3.2.1 Background

An experimental robot cell, built at City University (Chamberlain *et al.*, 1990) (figure 3.2), is used in the research. Standard construction materials have been adopted with imprecise dimensions. An expert systems is investigated to enable a goal driven, intelligent planning approach to be implemented, that can provide an effective dynamic plan for the building task. At the start of the project, the supply of materials was intended to be picked from a pallet, but, at a later stage (Chamberlain *et al.*, 1991a), it was decided that a more appropriate solution was a microprocessor driven conveyor that could survey and deliver quality checked material to the gripper presentation point. The conveyor's detailed functions and components are covered in a later paper (Chamberlain *et al.*, 1992), which shows the positions of the ultrasonic and photoelectric sensors mounted on it, as well as the vision station, which is triggered by the interruption of a high resolution fibre optical beam.

The GRASP robot simulation software was used to investigate the robot cell design and simulate constructions. CAD-CAM integration was also developed, where a masonry project is interactively designed and coded (Chamberlain *et al.*, 1992). This facility automatically generates project definitions of the designs, containing the order list and details of the masonry components and their corresponding locations and orientations, making up the 'theoretical task'. The real time inspection of the masonry units was achieved by the use of image processing techniques such as edge and line detection algorithms (Ala *et al.*, 1992).

Using the robot cell, experiments were conducted to build small assemblies. The robot gripper was developed to manipulate the blocks and also a peristaltic mortar pump controlled by the main PC (Chamberlain *et al.*, 1993). End-effector sensing, integrated with a rule-base expert system, is used to pick and place the masonry units accurately, avoiding potentially damaging contact forces. The need for the location of equipment on the construction site was also considered a challenge to the automation tasks in the construction sites (Chamberlain, 1994). The ability of the robot to automatically redefine it's workspace is considered a realistic requirement in the automation process. This was dealt with by applying intelligent sensing to locate targets mounted on equipment. To date, the question of overall mobility has not been investigated.

Comparing other attempts with this research effort, they were mainly either theoretical concepts that solve the complete process, or machines that assist bricklayers. The latter were found to be effective in speeding up production rates. To truly automate the masonry construction task, there is a need to utilise the advancement in robotic technology, especially to deal with the unstructured environments of construction sites. This view is in-line with the research undertaken by the author, which attempts to solve part of the complex problems of automating the building task, by using forms of sensing and intelligence. Concentration on this is the main distinguishing difference between the author's work and the few other attempts at physical realisation and experimentation of masonry automation.

3.2.2 Scope and limitations

The author attempts to solve some of the problems in automating the masonry building task. This is achieved by defining the building tasks and simplifying them in order to demonstrate methods of tackling them. For example, to investigate the robot vibration, which is a highly complex dynamic problem that is extremely difficult to accurately model, a practical experimental approach is adopted. This is based on investigation of the actual robot system using an accelerometer and data processing. The experiments covered for that are limited to the investigation of the uni-directional movement in the x-axis only. This demonstrates the principles of the method that has been developed. These experiments provide data for the expert system which automatically optimises and chooses the best types of move.

Collision freeways for the robot movement are utilised. This is a complication arising from the nature of construction sites, where the workspace layout may change. The problem has been simplified to deal with the location of the conveyor only, for the purpose of demonstrating the manner in which it is tackled. The location of the conveyor, in-turn, helps establish the block pick-up position. Through experimentation, the accuracy and success of the method developed to locate the conveyor will be determined. For this, the search area is limited to the x,y positive quadrant of the robot work space. This reduces the search, though demonstrates the effectiveness of the search algorithm. The method of using a target would be equally applicable for similar, though distinguishable, targets mounted on other peripherals in the robot cell.

The gantry robot's capabilities are investigated at the start as it is important to check how the system delivers the command trajectories. This is necessary to avoid generating a command trajectory that the robot is incapable of following. This covers investigation of the maximum velocity and acceleration, and the successful execution of the parabolic and s-curve moves.

The building tasks addressed in this research are:

(i)-Designing the masonry project, and determining the position, orientation, and dimension of each individual block.

(ii)-Determining the order of building the designed masonry project.

(iii)-Determining the optimised robot moves to carry out the building process.

(iv)-Locating the block pick-up position on the conveyor.

(v)-Determining the dimensions of the blocks.

(vi)-Picking-up a block from the conveyor.

(vii)-Determining the way the block was picked, and adjusting for that in the operation of the application of mortar, and the position the block is to be laid.

(viii)-Application of mortar on individual blocks.

(ix)-Laying the blocks in a pre-determined position.

For the second task, intelligence has been investigated to consider collision in construction and its effect on order of building blocks in a design project. This clearly affects the order of building the corners. However the addition intelligence to decide which end of the block is buttered in the construction of corners has not been considered.

The use of ultrasonic sensors will be investigated to carry out tasks 4, 5, and 6. The reasons for this choice of sensors was because they provided an inexpensive, accurate and highly effective way of providing the robot with sensing information. However, these sensors are not suitable for use outdoors, due to readings being effected by wind, humidity, and temperature variations.

A flexible approach is to be taken on the extent to which the system can be fully integrated. This recognises the anticipated complexity of the problem and expected time constraint.

In tackling the automation of the mortar dispensing process, the main factors to be considered are the choice of pump type, mortar mix material, the dispensing nozzle shape and design, the dispensing rate, the angle and offset of the nozzle, and the masonry unit preparation. These factors have been examined in various studies at City University (Kimble, 1991), (Ahmed, 1993), and (Charles, 1993), forming the background to this research. Trials of dispensing and building small assemblies are to be carried out. Investigation of the mortar dispensing task will not be integrated with the other tasks, due to restricted availability of the mortar pump. The method of mortar application that will be investigated is the application of mortar on one, two or three faces of the block, and possibly filling a few closer joints. This method was chosen due to equipment limitations, which did not allow for the dispensing of mortar in a continuous fashion over the plan of the design. Instead the robot will be used to present each block to the dispensing unit for mortar application.

The issue of mobility of the robot is not within the scope of this research. The work carried out in this research is done to prove that a fully functional system is achievable, within the reach of the technology used, rather than demonstrate a complete working solution to automation of masonry construction.

3.2.3 Assumptions

The building materials used in this project are standard masonry blocks. Also the use of mortar type bonding is to be employed. Mortar will be used to compensate for variations in block height.

Cell peripherals will be detected using a specially designed target mounted on the conveyor, that is sought with the aid of an ultrasonic sensor, mounted on the end-effector, and software logic.

For the provisions of intelligence, an expert system rule base is to be used. This will be developed using an expert system shell.

The designs and shapes of the masonry project the system would be capable of processing are limited. The number of wall legs in each masonry project are limited to eight, which automatically limits the designs allowed. These condition may seem overly simplistic, but their addition would only include more rules to be added to the system developed.

Blocks are to be delivered to a pick-up position using a conveyor belt that is manually loaded.

Assemblies are built with the assumption that the blocks are picked and laid using pre-determined theoretical positions i.e. the block location in the gripper is known.

The disruptive vibration of the robot system is to be overcome by move design rather than active control, which is impossible with the available robot.

3.3 Methodology

An experimental robot cell (figure 3.2), described in the following section, has been built to investigate automation of the various building tasks.

The experimental method adopted is to investigate the technology and methods with particular attention to accuracy and effectiveness. The various building tasks take advantage of the pre-processing with real-time accommodation of discrepancies and unplanned events, such as cell components relocation. The picking and placing tasks, as well as the application of bonding material, are tested, using sensing combined with artificial intelligence. For the provisions of intelligence a rule based expert system shell was used. This shell has the facility of translating the developed expert system into C language coding, enabling it to be integrated with the sensor information and the robot functionality. The robot functional library is coded in C programming language.

3.3.1 Experimental robot cell

Figure 3.2 is a diagram representation of the experimental robot. The robot cell has a 5 m long industrial conveyor, which is used to supply the masonry material. This has a target and sensors mounted on it. A vertical axis laser beacon is included to provide horizontal referencing. A mix dispensing station, a laser profiler system, and a gantry type robot are the remaining elements. The robot has a clamp type gripper with ultrasonic sensors and displacement transducers attached to it. A robot working envelope of 5 m x 2.5 m plan and 2 m elevation is available. Masonry units of up to 50 kg can be manipulated on the robot's three prismatic and two rotational axes. An accelerometer, attached on the robot's end-effector during experimentation, is used for determining its vibration. The robot's motion is controlled by three parallel SMCC (Smart Motion Control Cards), which communicate with a master 80386 processor via an RS232 communication link. The master processor communicates with the conveyor's microprocessor via the serial port. The laser profiler, the ultrasonic sensors, and the displacement transducers communicate with the main processor via an analogue to digital (A/D) converter connected to the parallel port.

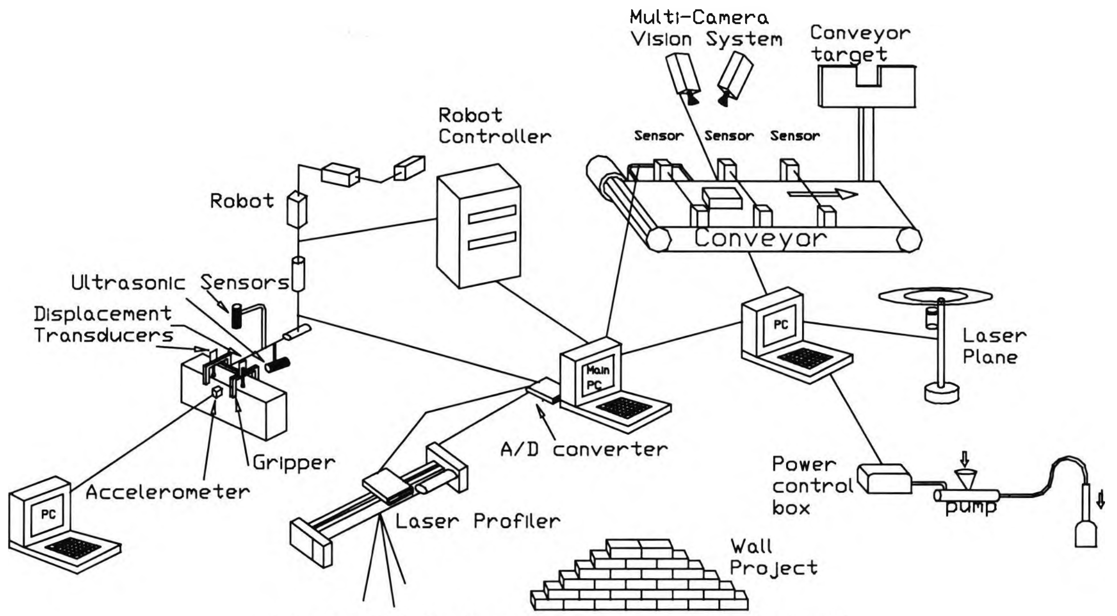


Figure 3.2 *Design of experimental robot cell*

Rule-based expert systems, integrated with the robot sensors, is used to provide intelligence for the robot. This allows the robot to function safely, making run time decisions, relevant to working within the unstructured environments of the construction site. This problem is solved by making minimum assumptions about the positions of the building materials, the location of equipment, or following the strict pre-programmed task, as normally associated with passive robot systems.

A system capable of automating the building process, accommodating a wide variety of project designs, will involve little repetition of the robot work, as each assembly task is different from the previous. This requires a highly flexible programming system to be developed in order to generate both the building task and the robot programs. To achieve such a system the process is divided into three steps, which are described in the following and illustrated in figure 3.3.

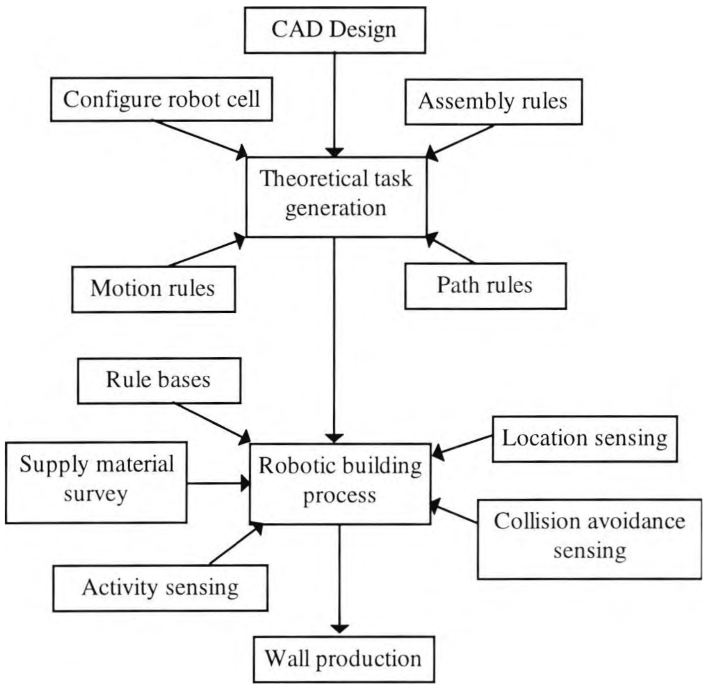


Figure 3.3 *Automating the building process*

(i)- The first step is to generate the building task. For this, a CAD programme developed at City University was used (Chamberlain *et al.*, 1992 and 1993). This converts each building project into a surface model and, on conclusion, produces a project description file. This process is covered in detail in chapter 9, together with an example of a generated project. The project description file includes information on each block type, it's centroid XYZ co-ordinates and orientation.

(ii)- The second step is to translate the project description file into a theoretical building task for the robot. The configuration of the robot cell is accounted for in the calculation of optimum moves. Also, the order of assembly has to be determined. A rule-based expert system is the proposed means of translating into the 'theoretical task'.

The limits and capabilities of the robot will be examined through experimentation. Using the robot's, different move types are executed with their move profiles captured and compared to the theoretical ones. A complication in the research is the vibration of robot's end-effector, which is, due to partial compliance about the robot's Yaw axis. To understand this, and achieve optimum moves, the effective move time (Effective Move Time = Move Time + Settling Time) is proposed. An accelerometer mounted at the furthest radial point of the end-effector will be used to investigate this. The various move types the robot is capable of (linear, parabolic, and 'S' curve) will be investigated, and the best types of moves determined. This information will be used for the knowledge part of the rule based expert system, that selects optimum moves for the robot.

A rule-base expert system will determine the building order of the construction task, taking into consideration the shape of the building to be realised and accommodating the robot end-effector design. The building order takes into consideration the possible collision zones arising from the movement of the end-effector during construction. Using a facility of that expert system shell, the rules will be verified and missing rules shown. The completed expert system for the building will be tested on designs of masonry projects.

(iii)-The third stage is to carry out the robotic building process. For this, various sensors will be used to support the run time adjustments needed for practical realisation of the theoretical task. This is necessary because standard masonry units, with substantial tolerances, are to be used. Conveyor sensing will provide the actual dimensions of bricks/blocks, as well as rough location on the conveyor. Experiments, to test the methods developed to achieve this, using standard concrete blocks and a wooden calibration block, will be carried out.

The mortar thickness applied is to be adjusted to compensate for the discrepancies between the expected dimensions of the masonry units and the dimensions measured on the conveyor. Using the peristaltic pumps and the robot end-effector to manipulate the block, controlled mortar dispensing experiments will be conducted to test automation of the mortar dispensing operation, using standard or modified masonry bonding mortar.

Ultrasonic sensors and displacement transducers mounted on the end-effector will be used to assist the location and picking of bricks/blocks at the conveyor pick-up point and avoid obstacle collision during the block picking operation. A rule-based expert system is developed that will be integrated with the robot's functional library and gripper sensors to provide run-time intelligence for the robot, to carry out the safe block pick-up operation. Also, the use of independent sensing for determination of the location of the masonry units relative to the gripper is to be studied. Each stage of the process will be tested individually, using standard blocks and a wooden calibration block, to test efficiency and accuracy.

3.4 Cartesian Robot

A gantry type robot and gripper is the central component of the experimental cell shown in figure 3.1. The robot, which was designed and built at City University, using standard components, comprises 5 DOF; three prismatic (x,y,z Cartesian), and two rotational (roll and yaw) axes. Its handling capacity is 50 Kg with full manipulation, and 500 kg with the yaw axis locked in position. It has a working envelop of 5 m x 2.5 m plan and 2 m elevation. For the purpose of our study, a clamp type gripper was attached to the robot, for block manipulation and sensing as described in section 2.4. The robot is controlled using Smart Motion Controller Cards (SMCC), which are a third-generation, state-of-the-art device for high speed precision closed-loop digital servo-control. The cards are programmed by means of high level instructions from the host computer, via a serial link.

The reason for the choice of this robot was to achieve a working envelop large enough to build masonry projects of up to 4 m x 2 m plan and 1.5 m elevation. Also, it is designed to test the automation of parts of the building process, including masonry unit manipulation, mortar dispensing and sensing. The issue of overall mobility, needed in a construction site realisation, was not investigated in this project.

3.4.1 Inverse kinematics

For manipulation using the adopted gripper, the inverse kinematics equations of the robot had to be worked out. Figure 3.4 shows the symbolic representation of the robot's degrees of freedom (DOF). Given the required tool centre point (TCP) co-ordinates x,y,z and the orientation of the 'α' (roll-axis) and 'σ' (yaw-axis), the corresponding absolute motor count values needed to be found. For that, the robot's axes were each calibrated and the resulting calibration constants, symbolised as $k_1 \rightarrow k_5$ used along with the transformation matrix in equation 3.1. The derivation of the overall transformation matrix, equation 3.1, is given in appendix A.1.

$$\begin{bmatrix} \cos\sigma & -\sin\sigma\cos\alpha & \sin\sigma\sin\alpha & u \\ \sin\sigma & \cos\sigma\cos\alpha & -\cos\sigma\sin\alpha & v \\ 0 & \sin\alpha & \cos\alpha & w \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_p \\ y_p \\ z_p \\ \sigma_p \\ \alpha_p \end{bmatrix} = \begin{bmatrix} k_1c_1 \\ k_2c_2 \\ k_3c_3 \\ k_4c_4 \\ k_5c_5 \end{bmatrix} \qquad \text{eqn 3.1}$$

The offsets, (a_1,a_2,a_3) from the robot's intersection of rotational axes were measured for the different TCP options required. Figure 3.4 shows these offsets in relation to the complete robot representation. Figures 3.5-3.8 show the different offset measurements for the various TCP used. Here, a_1 is the sideways offset of the TCP in the y_5 direction, a_2 is the forward offset of the TCP in the x_5 direction and a_3 is the vertical offset of the TCP in the z_5 direction. Incorporating these offset values into equation 3.1, results in equations 3.2-3.6, used when programming the robot's absolute moves in terms of the TCP required. Tests were carried out to verify the offset measurements, the equations used and the 'C' programmes developed to compute and execute moves. These tests were done using a thedolite and various targets placed on the ground.

$$c_1 = (1/k_1)\times(x_p - a_2\cos\sigma + a_1\sin\sigma\cos\alpha - a_3\sin\sigma\sin\alpha) \qquad \text{eqn 3.2}$$

$$c_2 = (1/k_2)\times(y_p - a_2\sin\sigma - a_1\cos\sigma\cos\alpha + a_3\cos\sigma\sin\alpha) \qquad \text{eqn 3.3}$$

$$c_3 = (1/k_3)\times(z_p - a_1\sin\alpha - a_3\cos\alpha) \qquad \text{eqn 3.4}$$

$$c_4 = (1/k_4)\times\sigma_p \qquad \text{eqn 3.5}$$

$$c_5 = (1/k_5)\times\alpha_p \qquad \text{eqn 3.6}$$

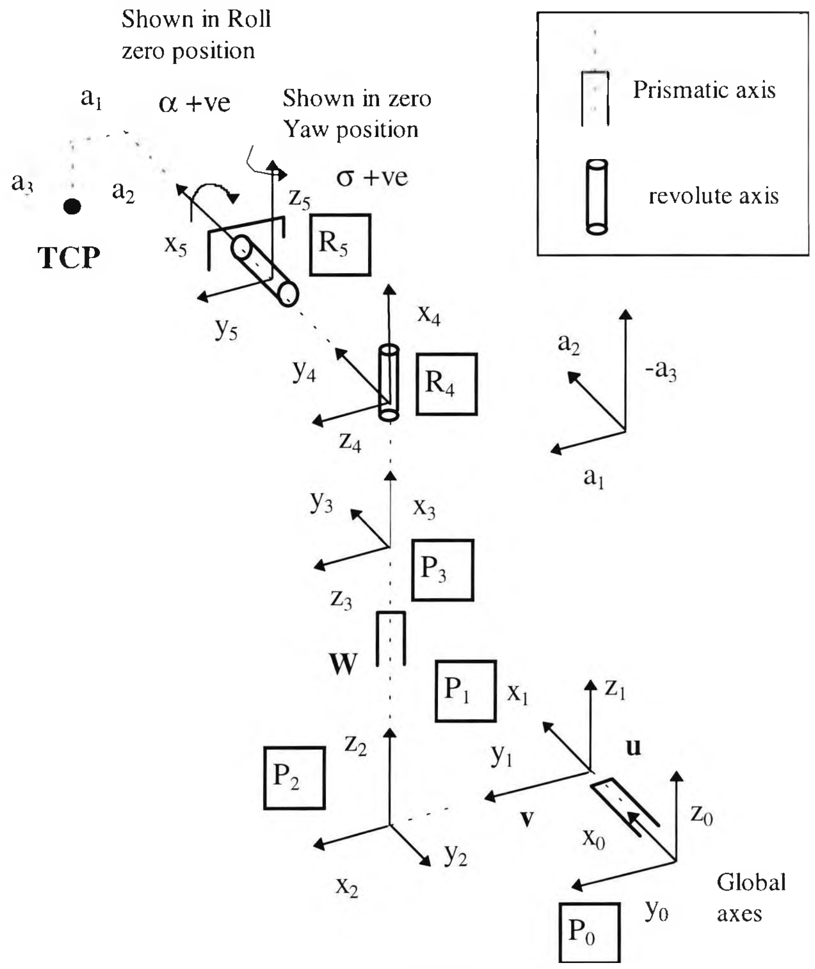
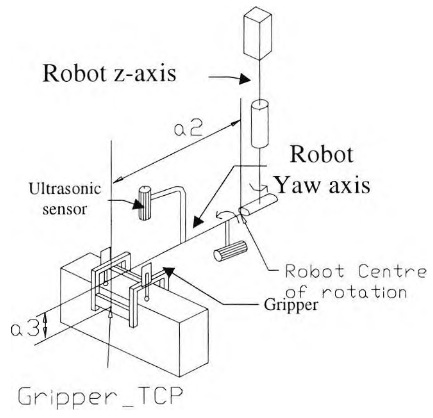


Figure 3.4 *Symbolic representation of the robot's configuration*



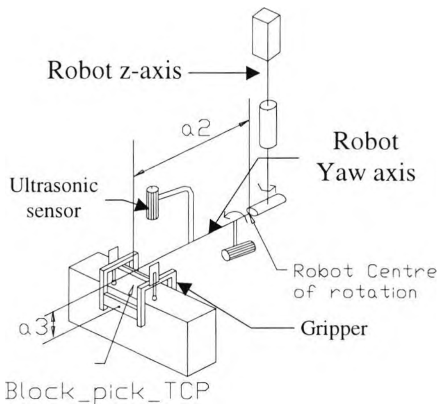


Figure 3.5 *(a_1=0,a_2,a_3) Values for Gripper_TCP*

Figure 3.6 *(a_1=0,a_2,a_3) Values for Block_pick_TCP*

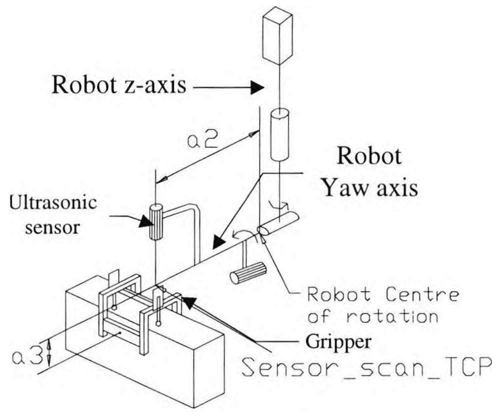


Figure 3.7 *(a_1=0,a_2,a_3) Values for Sensor_scan_TCP*

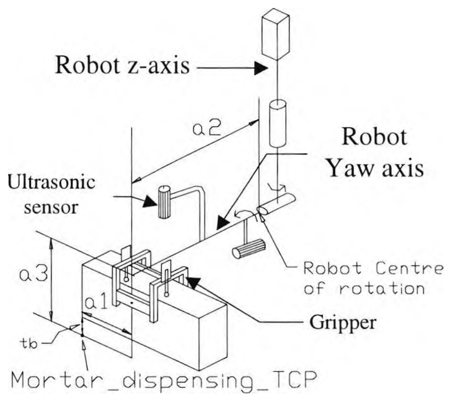


Figure 3.8 *(a_1,a_2,a_3) Values for Mortar_dispensing_TCP*

3.4.2 Robot performance

The robot has a theoretical absolute position accuracy of ±0.5 mm for light load manipulation. Each SMCC's high speed architecture allows servo update times of the order of 150 μs for simultaneous motion on two axes. It also provides special functions, such as the parabolic path generator, where all point are defined by position, velocity and the time to make the move. Third order algorithms are used to provide smooth, continuous paths and interpolated motion control, with zero acceleration at move segments boundaries, providing so called 'S-Curve moves'. A detailed description and theory of the various moves the robot is capable of performing are covered in appendix H. The steps in programming the different types of moves are covered in chapter 4. The robot speed limits are different for each axis. For the x-axis, they are 1 m/sec, the y-axis 1 m/sec, the z-axis 0.5 m/sec, and the yaw axis 300 deg/sec (BALDOR, 1989). The system limits for the x-axis were determined through the experiments covered in chapter 3.7. An investigation was also carried out to verify the systems capabilities of performing the parabolic and S-curve moves, as described in chapter 3.9, appendix H.

3.4.3 Robot functional library

An extensive library of 'C' functions was developed to operate the robot. As the SMCC provides direct operator controlled input switches, such as RUN, ABORT, RESET and HOME, they are all incorporated in the lower level 'C' functions. These 'C' functions generate groups of high level commands, which are then communicated from the main PC to the SMCC, via a serial link. Examples of these 'C' functions are operation of the gripper, gripper status, and interrogation of robot for position, velocity and status of the limit switches. They also cover functions to set the speed and accelerations for the various move types, for absolute or relative moves on each axis. Other, more complex functions initialise and home the robot, and check the various moves requested to establish if they are within the system limits of the robot. Functions to carry out moves using the various TCP choices were also developed. Further functions enable picking of a block, processing conveyor sensor data, data capture for generation of velocity and position profiles. These are described in the appropriate sections in the thesis.

The robot 'C' functional library was successfully integrated with the rule-base expert system, the mortar pump control functions, the conveyor control and communications functional library, as well as sensor data capture by the A/D converter. Use of the 'C' language facilitated the successful integration of the various software.

3.5 Conveyor

3.5.1 Conveyor design and operation

For the supply of material to the building robot, a continuous, belt-type conveyor driven by microprocessors was used (figure 3.5.1). The conveyor presented the masonry material ready for pick-up in a predictable and consistent, yet approximate, configuration. The block and brick units are fed manually onto the conveyor. A loading bay is provided to guide the masonry material onto the belt. This ensures that the material remains upright and also constrains the range of lateral position and orientation of the material to within limits wherein the correct operation of the onboard sensors may be guaranteed. Masonry unit size tolerances ranges from ±3 mm in length to ±2 mm in height and width, making it necessary to measure units at the supply stage. The conveyor is controlled using a 8 Mhz single-chip based microprocessor which operates assembly logic, multiple sensing and a companion vision system (figure 3.5.2). Communication between the host processor, conveyor and vision processors ensure the continuous delivery of suitable units. The controller is capable of operating the conveyor autonomously, monitoring its sensors to ensure a high quality supply of masonry units for the robot. It has networking facilities and can communicate with a local PC or directly with the main robot controller. Since the passage of masonry unit along the belt triggers off various events, the method of control is positional in nature. Details of the conveyor design and logic are given elsewhere (Chamberlain *et al.*, 1991a and 1992) .

The main concept is to use the conveyor for delivery, inspection and presentation of units for picking, this to be accomplished with minimum operator intervention. This is done using a conveyor drive position encoder, photoelectric beam switches, highly-focused ultrasonic analogue range sensors and a vision processing system. To assess the quality of the units delivered, they are checked dimensionally and superficially for gross defects such as edge and corner damage. The first two stages are provided by sensors mounted on the conveyor and the last the companion vision system. Details of the vision system are given elsewhere (Chamberlain *et al.*, 1991a and 1992), and (Ala, 1994). Units can be rejected at any of these stages, becoming discarded to a bin at the end of the conveyor. For the purpose of this study the first two stages were investigated. The vision system was not available during the reported study.

The conveyor belt is to be driven by a DC motor with fine speed control (±0.25% accuracy). Stop, ramp-up and ramp-down control is enabled to allow objects to be moved set distances along the belt. A 0.0-5.0 VDC analogue input signal is used to control the conveyor motion.

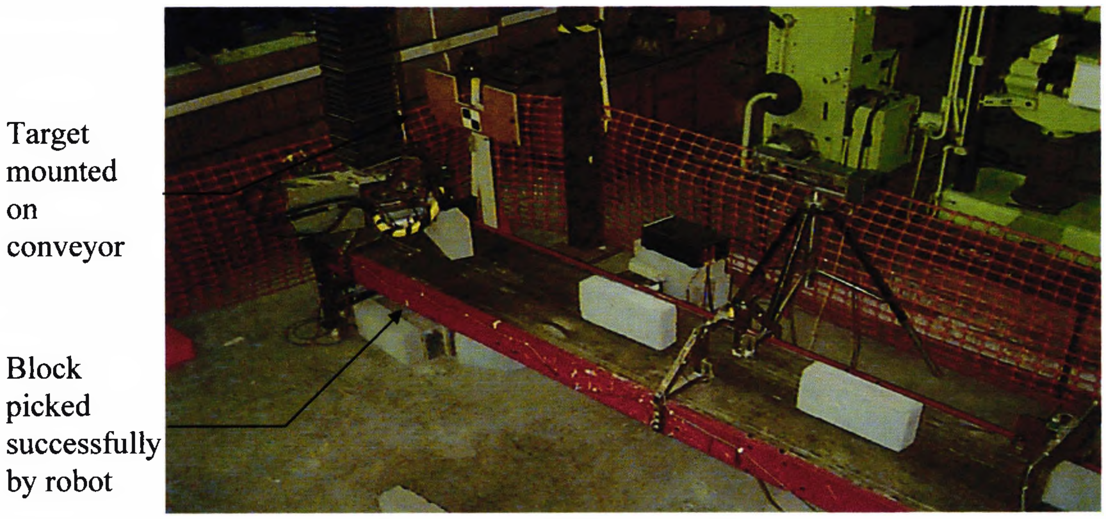

Target mounted on conveyor
Block picked successfully by robot

Figure 3.5.1 *Conveyor belt with target*

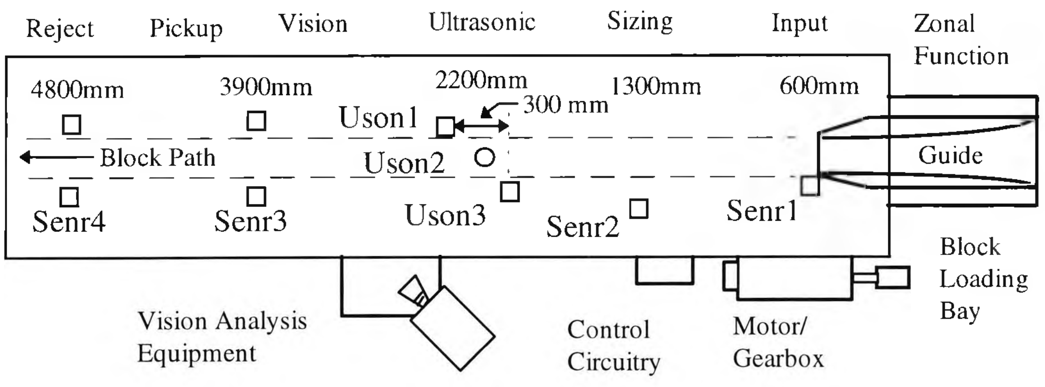


Figure 3.5.2 *Conveyor layout*

3.5.1.1 Conveyor sensing and logic

Figure 3.5.2 shows the layout of the conveyor, its sensors and the vision system. Figure A.2 in appendix A.2 shows the top-level operational logic. The various sensor information, defining the different states of the supply material, is sent to the main PC, where it is processed. On power up, after system initialisation, the conveyor waits until Senr1 (figure 3.5.3), a retro-reflective beam, is interrupted by the arrival of a masonry unit. After a short delay for placement of further masonry units, the unit moves on to Senr2 (figure 3.5.3), a high repeatability, through beam device with an operational range of 400 mm. Using the encoder counts, this enables determination of the apparent length and prompts further supply. The term apparent is used as units need not necessarily be aligned with the belt axis. Grossly incomplete units are detectable at this stage. Moving on, the unit arrives at the staggered array of precise analogue ultrasonic sensors (±0.25 mm accuracy) denoted as Uson1-Uson3 (figure 3.5.4), which perform a 3-D profile of the masonry unit. The readings from the sensors are stored in a block file using the conveyor processor, and then downloaded to the main PC for processing. This information allows the block dimensions, shape, accurate pick-up position, and orientation to be determined, as described in chapter 3.5.1.2. In the full implementation the ultimate assessment of masonry units is provided by the vision station that is triggered by the high-resolution fibre optic beam Senr3. An identical Senr4, is provided to define the robot pick-up point. Once this is triggered, the conveyor stops waiting for the robot to pick-up the block. These two sensors provide a precision stop position with the conveyor to ±0.6 mm along the belt.

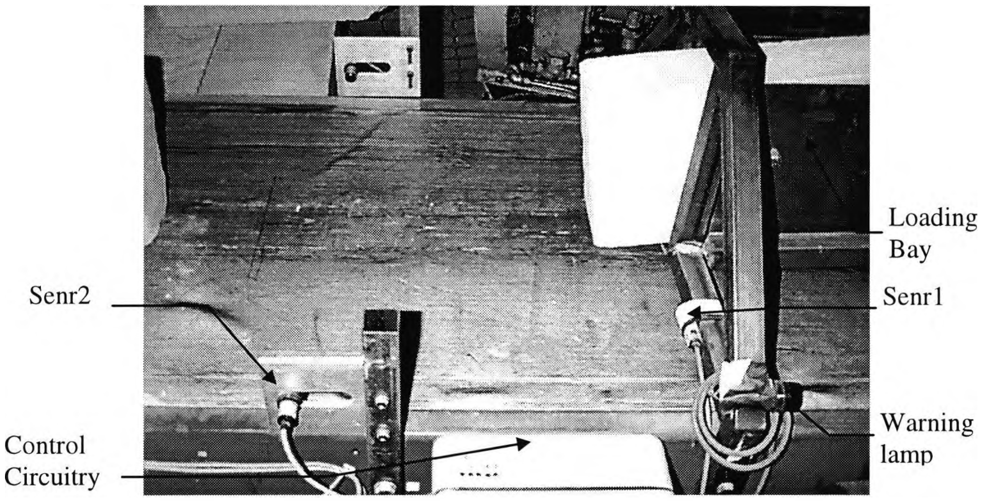


Figure 3.5.3 *Photoelectric sensors on the conveyor*

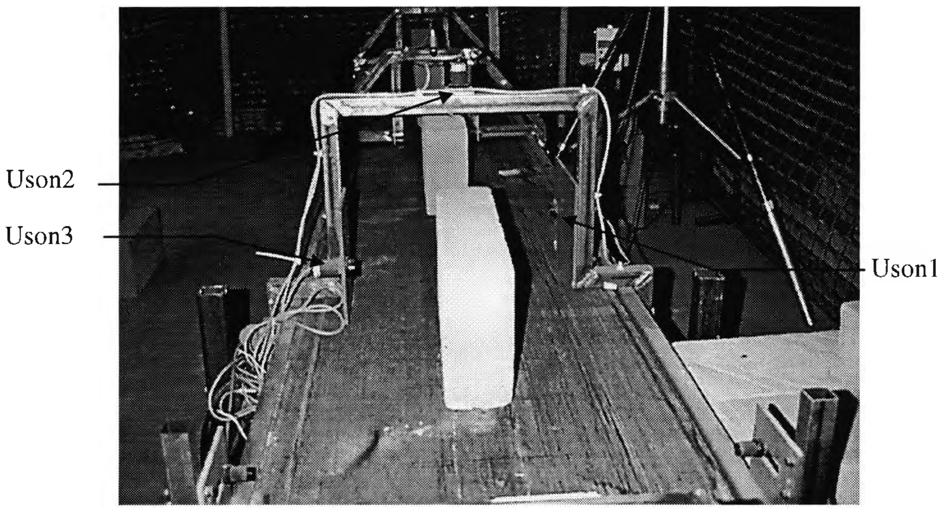


Figure 3.5.4 *Ultrasonic sensors on the conveyor*

3.5.1.2 Block measurements

Using the block file downloaded from the conveyor to the main PC, the block dimension and orientation was determined, which is part of stage 2 of the building process illustrated in figure 1.1. This ultimately facilitates real-time adjustments for safe picking of a block, as well as the accommodation of any discrepancies between the actual block dimensions and those of the theoretical task (refer to figures 3.3 and 1.1). An example of a block file is included in appendix A.3. The apparent length of the block, 'L' measured by Senr2 (figure 3.5.3), does not take into account the inclination of the block to the conveyor axis. The rest of the file contains the block profile captured by the ultrasonic sensors, Uson1-3. For this, movement of the conveyor was set at constant speed and the distance between each sensor set to 150 mm ±1 mm (figure 3.5.2), making it easy to synchronise the start of the ultrasonic sensor readings. The 150 mm spacing between these sensors was found to be the minimum in order to prevent auto detection i.e. one sensor detecting another rather than a block. This made it possible to calculate the block width 'W' as shown in figure 3.5.6. A 'C' programme '*Convesens.c*', based on the theory described below, was developed to read and convert the block file data and fit lines to the block profile, using a method of regression. A method developed to calculate the true width 'W', height 'H', true length of the block 'TL', as well as the angle of the block to the conveyor 'ϕ' is described below. A LISP programme (Chamberlain *et al.*, 1992 and 1993) was used to plot the block profiles in AutoCAD. An example of this is shown in figures 3.5.7(a) and 3.5.7(b). The following are the steps in the measurement algorithm:

Steps to calculated block dimensions (TL,H,W) and angle to the conveyor (ϕ):

1- Cut the first and last 25 readings of the each of the ultrasonic sensor data, as they represent the readings of the edges of the block, and convert the sensor reading to millimetres using sensor calibrations data.

2- Fit lines to the data of each ultrasonic sensor reading, using a method of line regression. Using the resulting equations of the lines, generate the new data points for each side (figure 3.5.7(b)).

3- Calculate 'ϕ', the angle of the block to the conveyor, from the arctangent of the average gradients of the lines fitted to the Uson1 and Uson3 sensor's readings.

4- The distances 'Offset1' and 'Offset3' (figure 2.3.6), being the distance from sensors Uson1 to the block, and Uson3 to the block, are taken from the constant values of the equations of the lines, fitted to Uson1 and Uson3 sensor readings.

5- The width of the block 'W' is calculated using equation 3.5.1.

$$W = (HD - offset1 - offset3) \times \cos(\phi) \qquad \text{eqn 3.5.1}$$

, where 'HD' is the horizontal distance between Uson1 and Uson3 as shown in figure 3.5.6.

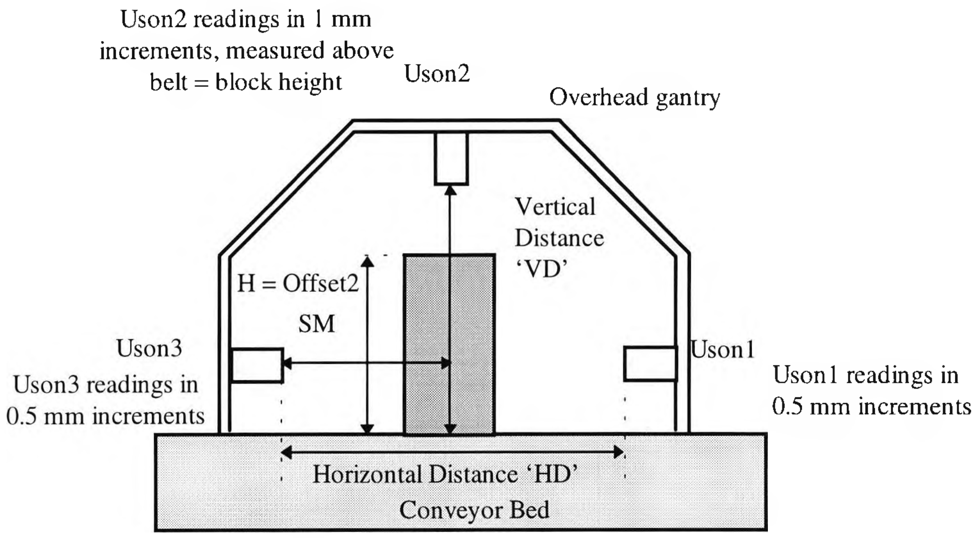
6- The height of the block 'H', which equals the Offset2, was calculated from the line fitted to the data of Uson2 (figure 3.5.5).

7-The true length of the block 'TL' (figure 3.5.6), was calculated using equations 3.5.3-3.5.4, which depends on the block orientation to the conveyor axis. If the block is inclined in the direction shown in figure 3.5.6 (i.e. ϕ was +ve) then equation 3.5.3 is used. If the angle is inclined in the opposite direction to the conveyor (i.e. ϕ was -ve), then equation 3.5.4 is used:

$$TL = ((L - (W \times \sin(\phi)) / \cos(\phi) \qquad \text{eqn 3.5.3}$$

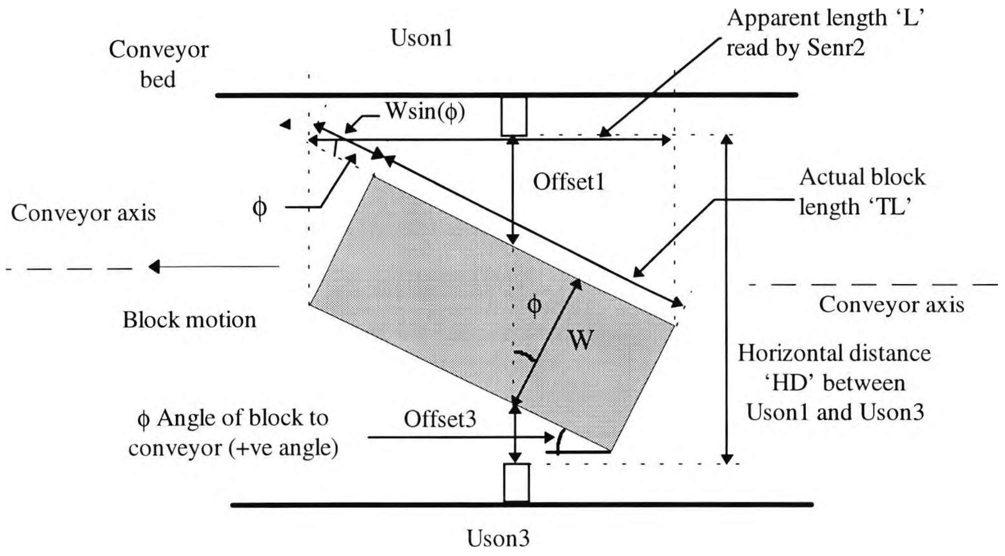
$$TL = ((L + (W \times \sin(\phi)) / \cos(\phi) \qquad \text{eqn 3.5.4}$$

Experiments determining the accuracy of this method are covered in chapter 6.



Uson2 readings in 1 mm increments, measured above belt = block height
Uson2
Overhead gantry
Vertical Distance 'VD'
H = Offset2
SM
Uson3
Uson3 readings in 0.5 mm increments
Uson1
Uson1 readings in 0.5 mm increments
Horizontal Distance 'HD'
Conveyor Bed

Figure 3.5.5 *Ultrasonic sensor array on conveyor*



Conveyor bed
Uson1
Apparent length 'L' read by Senr2
Wsin(ϕ)
Offset1
Actual block length 'TL'
Conveyor axis
ϕ
Block motion
ϕ
W
Conveyor axis
Horizontal distance 'HD' between Uson1 and Uson3
ϕ Angle of block to conveyor (+ve angle)
Offset3
Uson3

Figure 3.5.6 *Block length and offset measures of the block relative to the sensors*

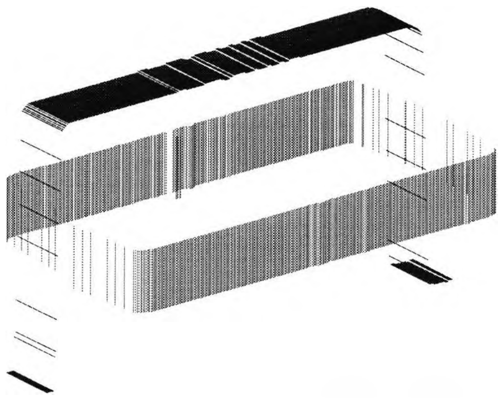


Figure 3.5.7(a) *Block profile using original ultrasonic sensor data*

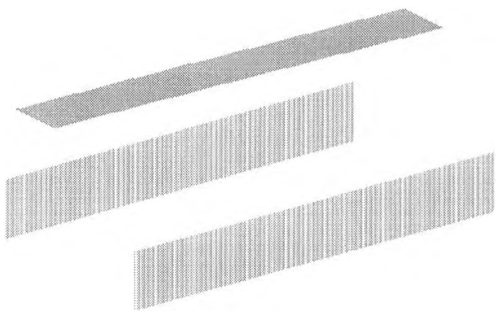


Figure 3.5.7(b) *Block profile using generated data from the line fitting to ultrasonic sensor data*

3.5.1.3 Block pick-up position (BPP)

The block is at a 'ready for picked-up' state when any part of it first interrupts Senr4 (figures 3.5.1 and 3.5.8). Figure 3.5.8 shows two examples of blocks at the pick-up stage on the conveyor. The first example is the calibration block, aligned to the conveyor, and at exactly the middle of the conveyor. This block has length l = 440 mm, width 100 mm, and height 213 mm. Such a block was specially manufactured in timber, and was used to confirm the block pick-up position (BPP) co-ordinates in the robot co-ordinate system. The (x,y) BPP co-ordinates are the mid point of the reference block at the position 'ready for pick-up'. The plan angle θ, of the conveyor relative to the robot x-axis, used as the BPP_yaw angle and the (x,y) co-ordinates of the BPP are accurately determined prior to the start of the building process. This is part of stage 1 of the building process as shown in figure 1.1 using the robot '*block_pick_TCP*' co-ordinate system. This was done using a target aligned to the conveyor, searched for with the ultrasonic sensors mounted on the robot end-effector (refer to section 3.5.2 and chapter 5). The z co-ordinates of the BPP was measured to be at the point when there was a clearance of 12 mm between the block and the gripper, which is the position where the gripper is ready for gripping. The second example in figure 3.5.8 is a block inclined at a -ve angle 'ϕ' to the conveyor, with it's centre, BCP, at an offset from the middle of the conveyor. The New Block Pick-up Position (NBPP) is calculated from the adjustments made to the BPP using the BCP, as part of stage 2 of the building process (figure 1.1).

This enabled the robot to pick-up a block from the conveyor safely with no risk of collisions. Figure 3.5.9 shows the different co-ordinate systems used, which are of the robot's $(x^{/}, y^{/})$, the block's $(x^{//}, y^{//})$, and the conveyor's $(x^{///}, y^{///})$ co-ordinate systems.

The adjustments to the BPP were calculated using sensor derived measurements. Block dimensions TL, H, and W, calculated from the conveyor sensor data. Distance 'SM', which is the distance from Uson3, to the middle of the conveyor (figure 3.5.8), is also used, as well as 'SR', the reading of Uson3 at the middle of the block side. A procedure, described in this section, to calculate these adjustments was included in the 'C' programme, '*Block_pick.mak*'. An assumption was made for this calculation, which demonstrates the steps taken to adjust the BPP. The assumption was that the conveyor's location was in the direction of the x-axis, as shown in figure 1.1. The possibility of the conveyor being otherwise is covered by target based location, as presented in chapter 3.5.2. The stages of the adjustment algorithm as given in the following:

Application procedure (adjust_BPP):

(1)-Adjust the robot yaw angle making the gripper aligned to the block, by subtracting the angle 'ϕ' from the conveyor angle, using equation 3.5.6.

$$NBPP\_yaw = BPP\_yaw - \phi \qquad \text{eqn 3.5.6}$$

(2)-Adjust the robot's (x,y) position, such that the gripper's position would be at the top of the block (figure 3.5.8). This is done by calculating the offsets of the BCP from the BPP in x and y directions using equations 3.5.7-3.5.11. The value for X_adjust, the offset needed in the x-axis direction, is calculated using equation 3.5.7. If the block is inclined at a positive angle to the conveyor, then equation 3.5.8 is used to adjust the BPP_X co-ordinate, otherwise equation 3.5.9.

$$X\_adjust = |((W/2)\sin(\phi)) + ((TL/2)\cos(\phi)) - (l/2)| \qquad \text{eqn 3.5.7}$$

$$NBPP\_X = BPP\_X + X\_adjust \qquad \text{eqn 3.5.8}$$

$NBPP\_X = BPP\_X - X\_adjust$ eqn 2.3.9

The value for Y_adjust, the offset needed in the y-axis direction, is calculated using equation 3.5.10, which is subtracted from the BPP_Y co-ordinates as shown in equation 3.5.11, to produce the NBPP_Y co-ordinates.

$Y\_adjust = ((W/2)\cos(\phi)) - SM + SR$ eqn 3.5.10

$NBPP\_Y = BPP\_Y - Y\_adjust$ eqn 3.5.11

(3)-Adjust the BPP_Z position using the difference in the height measurements of the ideal block and the actual height H.

$NBPP\_Z = BPP\_Z + (H - Ideal\_Height)$ eqn 3.5.12

These adjustments to the BPP are the first stage in determining the accurate position of the block at the pick-up stage. Further checks and adjustments are carried out to the NBPP, using the sensors on the end-effector of the robot. These are part of stage 3 of the building process (figure 1.1), which is described in section 3.6.3.

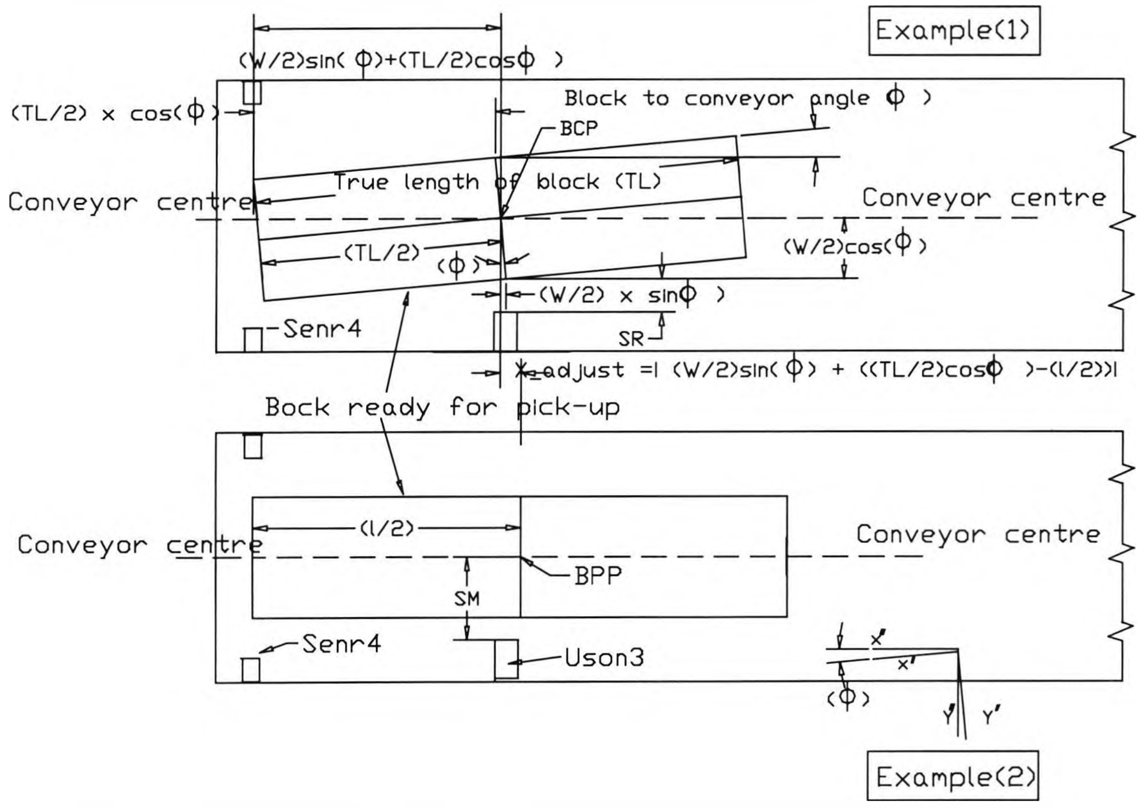


Figure 3.5.8 *Adjusting block position on the conveyor at the pick-up stage*

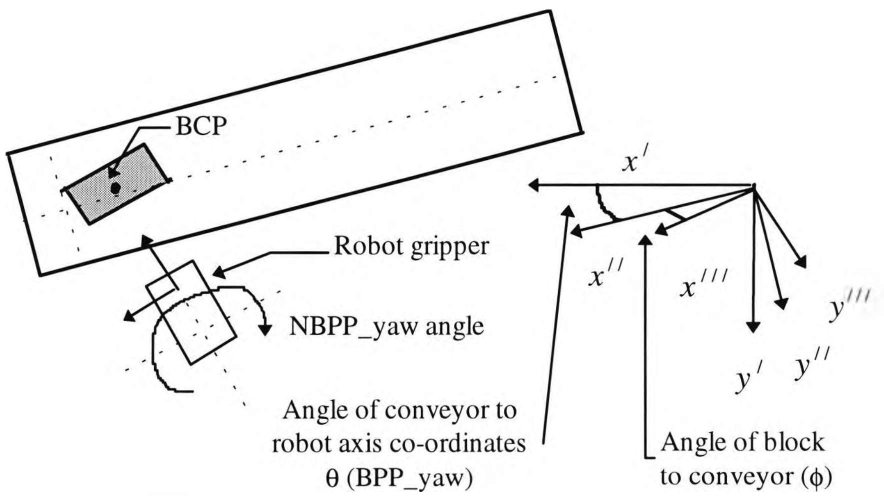

BCP
Robot gripper
NBPP_yaw angle
Angle of conveyor to robot axis co-ordinates θ (BPP_yaw)
Angle of block to conveyor (ϕ)
$x^{/}$
$x^{//}$
$x^{///}$
$y^{/}$
$y^{//}$
$y^{///}$

Figure 3.5.9 *Conveyor, block and robot co-ordinate systems*

3.5.2 Conveyor Target

Unlike in manufacturing plants, where robots are pre-programmed to carry out, tasks that benefit from fixed positioning of equipment and exact presentation of materials, construction site robots will inevitably have to accommodate frequent rearrangement of equipment and material presentation. Furthermore, normal production masonry materials in the UK have significant dimensional tolerances and it is intended that their use should be accommodated in the robot technology. Therefore, measures need to be taken to ensure the robot is able to recognise and locate objects, accommodating changes and discrepancies, as failure to do so could have devastating consequences for the robot and it's workspace. In the case of the experimental cell used in this project, we need to locate the conveyor and the BPP on it. The conveyor is shown in figure 3.5.12. A target, designed for this purpose, was built and mounted on it, aligned as shown in figure 3.5.12. The reason for the design of the target was because it was an inexpensive, yet extremely effective method of locating the conveyor, whilst utilising the ultrasonic sensors already mounted on the robot end-effector (see figure 3.5.11).

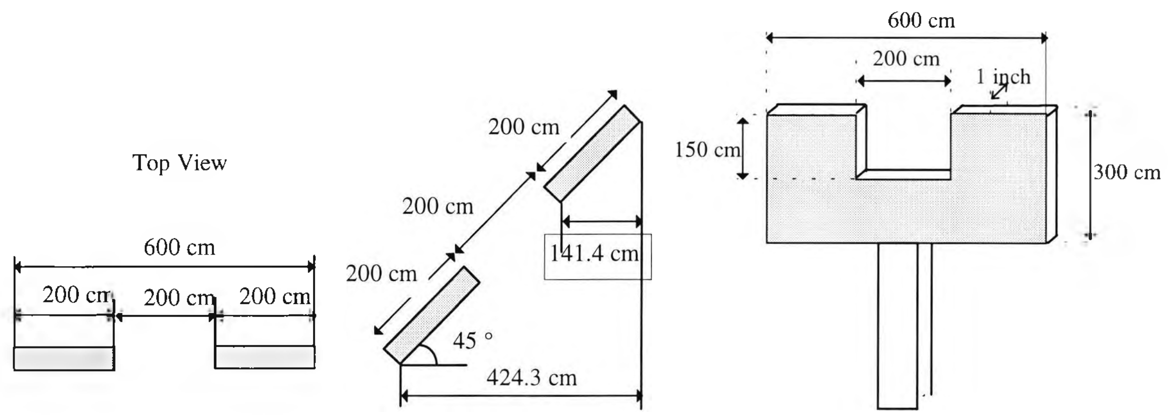


Figure 3.5.10 *Conveyor target dimensions*

Taking into consideration the range, accuracy and positioning of the sensors, the target dimensions and shape were designed as shown in figures 3.5.10 and 3.5.11. The design was worked to ensure the target was large enough and wide enough not to be missed, and unique in shape so not to be confused with other objects. In use, it serves for location in both the horizontal and vertical planes. Chapter 5 describes the target location logic, which has been developed through experimentation.

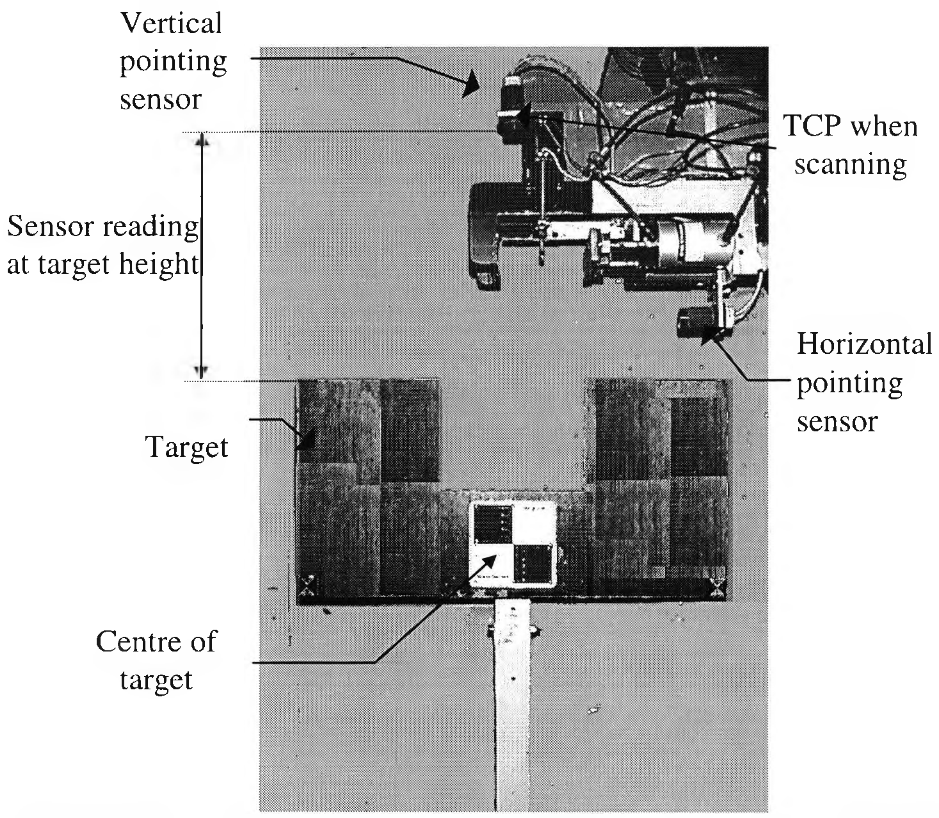


Figure 3.5.11 *Conveyor target and end-effector sensors used to locate target*

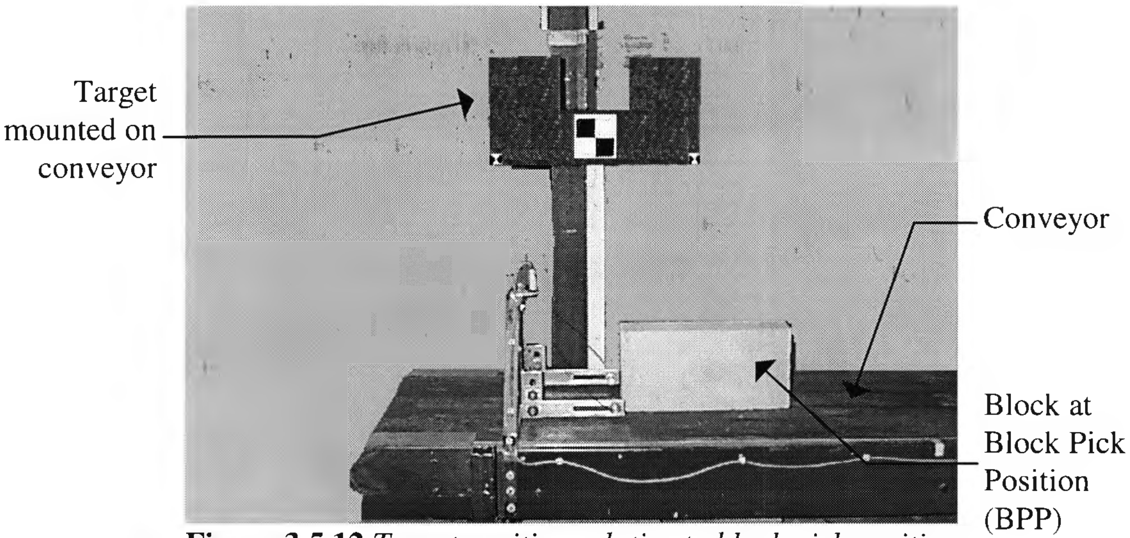

Target mounted on conveyor
Conveyor
Block at Block Pick Position (BPP)

Figure 3.5.12 *Target position relative to block pick position*

3.6 Robot block gripper

Two types of masonry unit grippers were designed to be used as the robot's end-effector. The first is a clamp type gripper with a 90-112 mm jaw opening range, suitable for brick and block assembly, with sensor provisions (figure 3.6.1). The second is a triple cup absorption device which provides suction gripping, suitable for block and large panel assembly (figure 3.6.2). Grip sensing is also provided for this tool, which has the ability to provide local controlled lateral motion for use when placing masonry units.

The clamp type gripper was used in this research. It was fitted to the base of the robot arm, enabling pick-up of masonry units from the conveyor (figure 3.5.1), manipulation of them for mortar application (figure 3.5.11), and location of them in the wall project (figure 9.6).

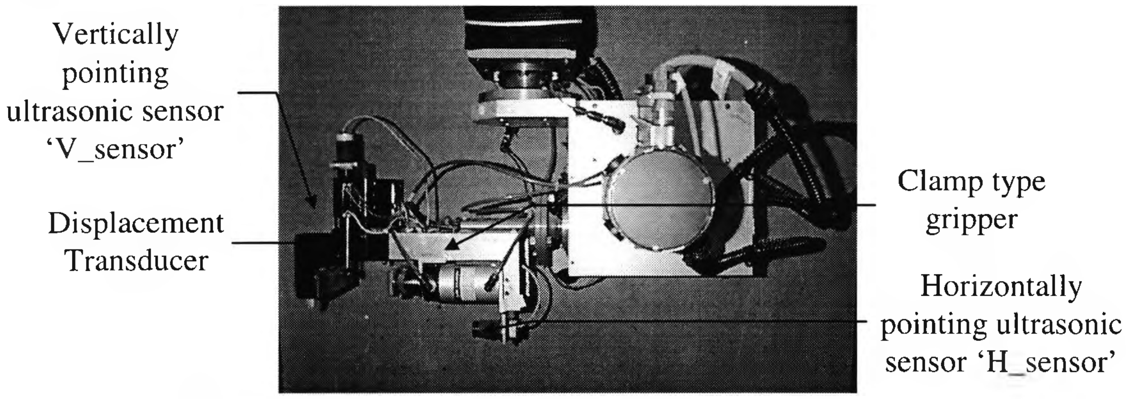

Vertically pointing ultrasonic sensor 'V_sensor'
Displacement Transducer
Clamp type gripper
Horizontally pointing ultrasonic sensor 'H_sensor'

Figure 3.6.1 *Clamp type block/brick gripper at robot end-effector*

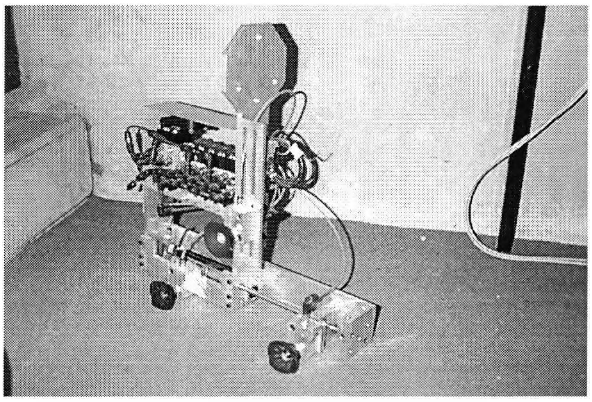


Figure 3.6.2 *Suction type block/panel gripper*

3.6.1 Block gripper sensors

Two types of sensor were mounted on the robot end-effector, these being ultrasonic and displacement transducers. They support the location, gripping and placement of masonry units. Real-time sensing, integrated with a rule-based expert system (described in chapter 9), was used to determine the high level run-time instructions for the robot. This provided a safe and reliable execution of the block pick-up operation from the conveyor. The sensor signals were converted from analog to digital using the A/D converter (PICO) described in appendix A.6.

3.6.1.1 Ultrasonic sensors

Two, highly focused, near range ultrasonic sensors of ± 0.25 mm accuracy and 400 mm working range were mounted on the clamp type gripper (figures 3.6.1 and 3.6.3). These were used to locate the masonry units prior to pick-up. Once gripped, they confirm the action's success or failure. They were also used to search for, and locate the conveyor target (figures 3.5.11 and 5.6). For a description of that method refer to section 3.5.2 and chapter 5. One ultrasonic sensor was mounted to measure vertical distance (V_sensor) with a deadband of 100 mm. The other was mounted to measure horizontal distances, with no deadband value to it. Sensor positioning and deadband values where set to allow sensing of masonry units before and after gripping them (figure 3.6.3).

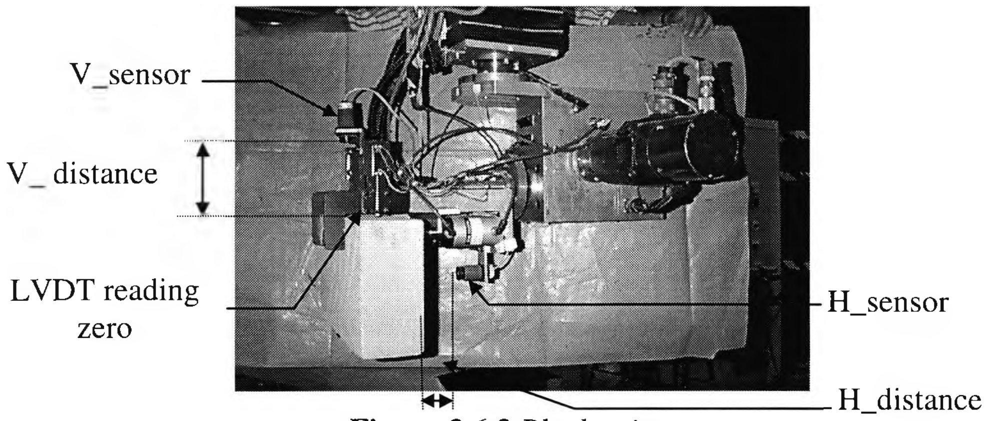


Figure 3.6.3 *Block gripper*

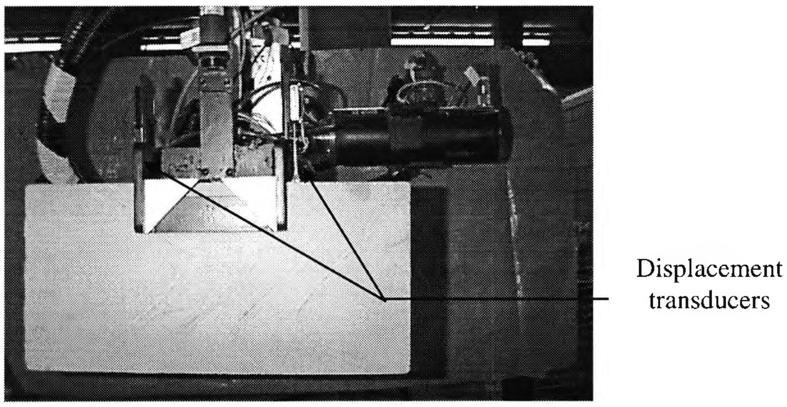


Figure 3.6.4 *Block gripped showing the use of the two displacement transducers*

3.6.1.2 Displacement Transducers

A pair of linear voltage displacement transducers (LVDT), having roller contacts fitted at their ends, were located on either side of the gripper fingers (figures 3.6.1 and 3.6.4). They have an accuracy measure of ±0.05 mm for a range of 25 mm. Their use is to help determine the position of picked blocks in relation to the gripper, giving both angle and offset. Experiments were carried to check their effectiveness in the block laying operation in chapter 7. This is done by first reading their offsets to the block, then using the values to calculate the angle of the picked block relative to the gripper. The LVDT is a centre device which means it's linear measuring range is expressed in positive and negative displacements. However, this was changed to suite our application, by offsetting the transducer's readings. The zero displacement position was re-defined (figure 3.6.3) to be at the point when its contact distance is 5 mm away from the gripper top, this position being the closest safe position for a block to be when gripped. From this its effective range was 15 mm.

3.6.2 Block gripping

Whilst the ultrasonic sensor array on the conveyor gives the position and angle of each block, there is a possibility of belt drift and disturbance of masonry material between this measurement and arrival at the pick-up position. To avoid damage to the robot gripper, it is necessary for the gripper to confirm the position of material at the pick-up position. This is important because the gripper jaw opening is limited to GL = 112 mm, the spacing between its fingers being GW = 164 mm. The width of the blocks used is 100 mm ±1.6 mm (table A.1 in appendix A.4), which gives a clearance distance of 5.2 mm each side of the gripper, assuming the mid point of the gripper is exactly on top of the middle of the block and aligned to it. For every 1° skew of the gripper to the block, its edge will be closer to the block's side by 1.45 mm. This displacement 'x' (figure 3.6.5) was calculated using:

$$x = \sin(\beta) \times ((GW/2) + ((GL/2) \times \tan(\beta/2))) \qquad \text{eqn 3.6.1}$$

, where β is the skew angle. For safety, a maximum error of 3.26° should be guaranteed, giving a minimum clearance of 0.5 mm from the block edge, as can be seen in figure 3.6.5. Again, this was calculated on the assumption that the mid-point of the gripper is exactly at middle of block, and the maximum expected width of blocks at 101.6 mm. In the case where the gripper centre of at 1.5 mm offset from the middle of the block, then the maximum safe skew is 2°.

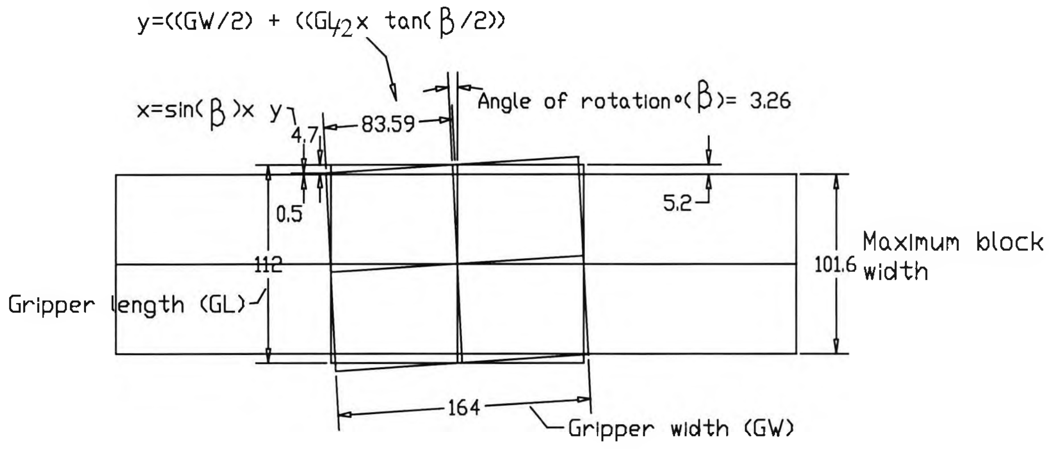


Figure 3.6.5 *Gripper to block rotation*

The procedure developed to implement the sensor measurement is given in the following:
First, the gripper is aligned with the block by measuring the angle of the gripper to the block, and adjusted for it, according to stage 3 of the block building process (figure 1.1). The position of the centre of the gripper relative to the centre of the block is then adjusted. These adjustments result in the safe block pick-up position co-ordinates (SBPP). The 'C' function '*Check_pick_pos*(ϕ)', developed to deliver these adjustments, is included in the file '*block_pick.mak*'. The starting point of these procedures are the adjusted co-ordinates of BPP (i.e. NBPP) calculated using the conveyor sensing information, as described in section 3.5.1.3. Using the NBPP makes it reliable to assume that the gripper is located roughly central to the block, and almost aligned to it. NBPP is in the robot co-ordinate system.

3.6.2.1 Gripper to block alignment

To align the gripper with a block, the angle 'β' has to be determined (figure 3.6.5). This is the error in the estimation of ϕ, the angle of the block to the conveyor (section 3.5.1.2 and figure 3.5.6). Once β is established, a move to correct for that angle is made. Using the horizontally pointing ultrasonic sensor on the gripper, 'H_sensor', in the procedure described below, the adjustments are made to the NBPP_yaw co-ordinates. At the start of the procedure, the gripper was positioned at the NBPP position, with it's z-axis raised. To perform the aligning procedure, the gripper needs to move laterally and forward in the $(x^{/}, y^{/})$ co-ordinate system (the robot's co-ordinate system), which align with the gripper's co-ordinates shown in (figure 3.6.6). The following procedures were developed to deal with this. The final x,y position is in the robot's, '*block_pick_TCP*' (illustrated in figure 3.6), co-ordinate system. The procedures are as follows:

(i) To move_forward a distance d in the $(x^{/}, y^{/})$ co-ordinate system:

If $-90 \le \phi \ge 90$

$$x^{/} = x^{/} + (d \times \cos(\phi)), \quad y^{/} = y^{/} + (d \times \sin(\phi))$$

If $-180 \le \phi < -90$

$$x^{/} = x^{/} - (d \times \cos(180 + \phi)), \quad y^{/} = y^{/} - (d \times \sin(180 + \phi))$$

If $90 < \phi \le 180$ or $\phi < -180$

$$x' = x' - (d \times \cos(180 - \phi)), \quad y' = y' + (d \times \sin(180 - \phi))$$

(ii) To move laterally a distance d in the (x', y') co-ordinate system:

If $0 < \phi \leq 90$

$$x' = x' - (d \times \sin(\phi)), \quad y' = y' + (d \times \cos(\phi))$$

If $-90 \leq \phi \leq 0$

$$x' = x' + (d \times \sin(-\phi)), \quad y' = y' + (d \times \cos(-\phi))$$

If $-180 < \phi < -90$

$$x' = x' + (d \times \sin(180 + \phi)), \quad y' = y' - (d \times \cos(180 + \phi))$$

If $(90 < \phi \leq 180)$ or $(x \leq -180)$

$$x' = x' + (d \times \sin(180 - \phi)), \quad y' = y' + (d \times \cos(180 - \phi))$$

(iii) To align gripper to block:

1. Move the gripper laterally 150 mm in the x', y' co-ordinate system, and read the H_sensor value (S1).
2. Move the gripper laterally -300 mm in the x', y' co-ordinate system, and read the H_sensor value (S2).
3. Calculate the angle of the gripper to the block β,
   $\beta = \tan^{-1}((S1 - S2)/300)$
4. Correct gripper angle for pick-up
   If $(-180 \leq NBPP\_yaw < 0)$
   $SBPP\_yaw = SBPP\_yaw - \beta$
   If $(0 \leq NBPP\_yaw \leq 180)$ or $(NBPP\_yaw > -180)$
   $SBPP\_yaw = SBPP\_yaw + \beta$

An experiment was carried out to check the accuracy of this procedure. A description of it and the results are in chapter 6.3.3.

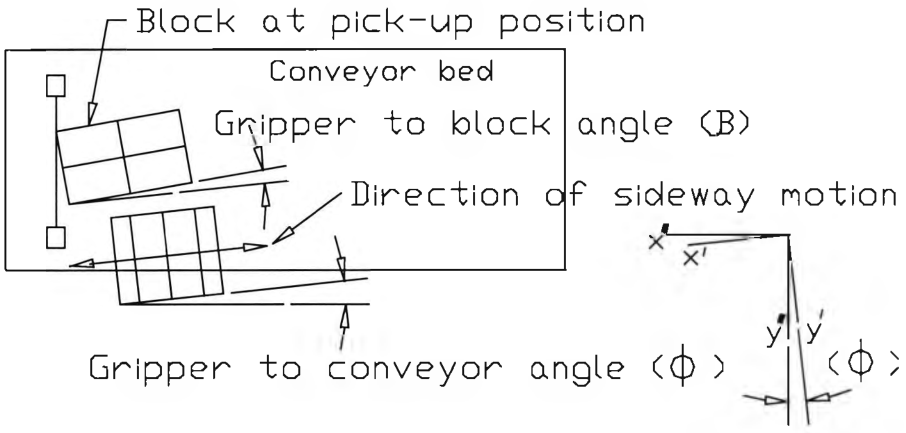

Block at pick-up position
Conveyor bed
Gripper to block angle (B)
Direction of sideway motion
Gripper to conveyor angle (ϕ)
x
x'
y
y'
(ϕ)

Figure 3.6.6 *Gripper to block position before adjustment*

3.6.2.2 Gripper to block centre alignment

After successfully aligning the gripper with the block, the H_sensor is used to adjust the gripper centre to coincide with centre of the block. This final step, also part of stage 3 of the building process illustrated in figure 1.1, ensures that the gripper opening has ample clearance to both sides of the block, safe enough to lower the gripper without the risk of collision with the block. This was achieved by adjusting H_distance (figure 3.6.3). The ideal H_distance to pick-up a block is 124 mm, measured using the calibration block (width of 214.49 mm). Using the actual block width 'W', calculated from the conveyor sensing information (section 3.5.1.2), the discrepancy in the width measurements were accounted for in the adjustment to H_distance using equation 3.6.2. Using the H_sensor reading, the robot position was then moved forward a '*hd*', in the direction of the current robot yaw position (i.e. SBPP_yaw), using the x', y' robot's co-ordinate system, according to the procedure described in section 3.6.2.1 and equation 3.6.3:

$$H\_distance = H\_distance + (ideal\_width - W)/2 \quad \text{eqn 3.6.2}$$

$$hd = H\_distance - H\_sensor \quad \text{eqn 3.6.3}$$

After the move is made, the new co-ordinates are SBPP_x, and SBPP_y in the '*Block_pick_tcp*' co-ordinate system.

3.7 Profile laser

For accuracy in applying mortar and laying blocks, there was a need to know the exact positioning of the picked block in the gripper. In section 3.6.1.2 the use of gripper mounted LVDT's is described as a means of determining the angle and offset of a gripped block. This does not, however, give information on the position of the gripper along the block. For this purpose, and confirmation of the block position, a laser profiler has been applied. This was designed and built at City University.

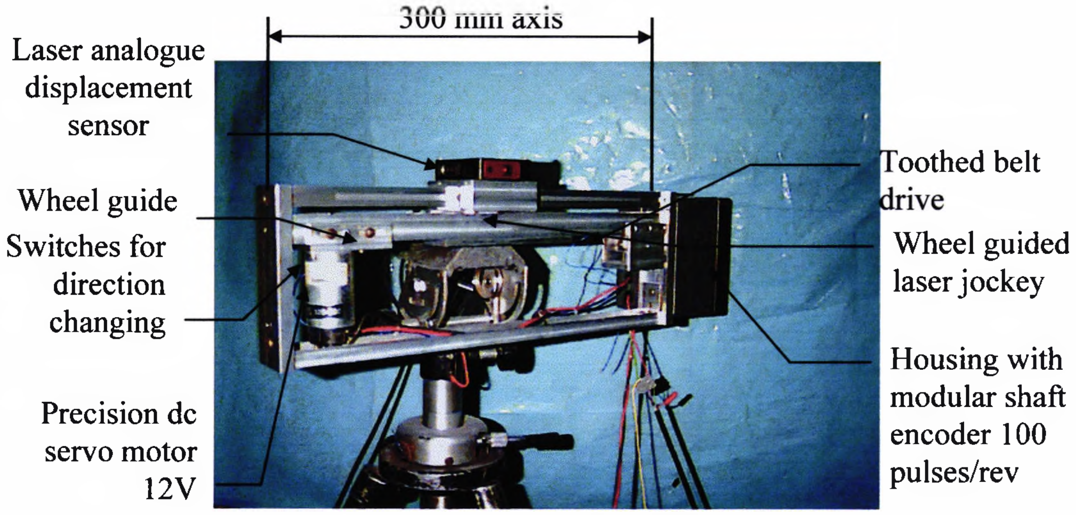

300 mm axis
Laser analogue displacement sensor
Wheel guide
Switches for direction changing
Precision dc servo motor 12V
Toothed belt drive
Wheel guided laser jockey
Housing with modular shaft encoder 100 pulses/rev

Figure 3.7.1 *Laser profiler*

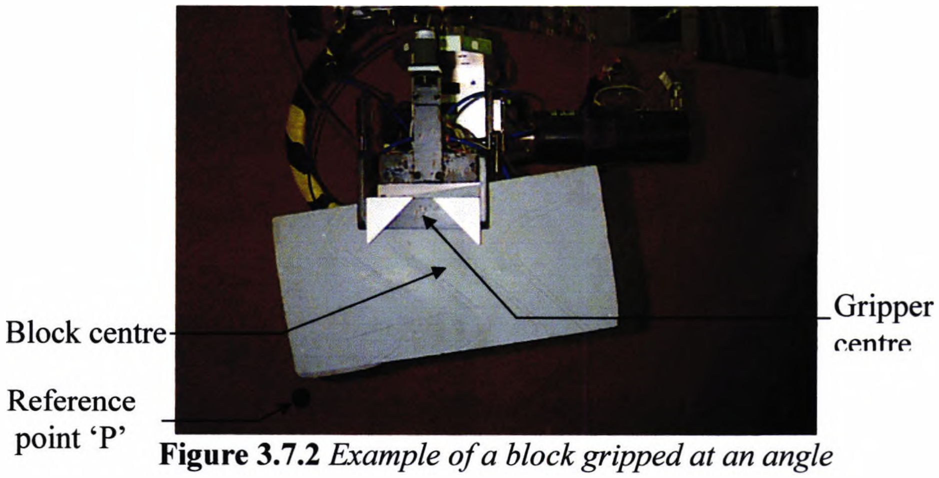


Figure 3.7.2 *Example of a block gripped at an angle*

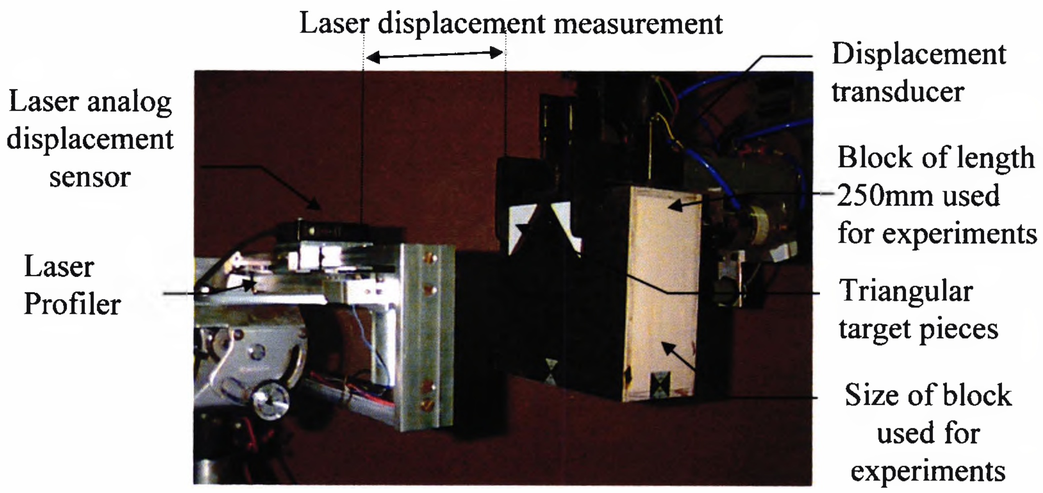

Laser displacement measurement
Laser analog displacement sensor
Laser Profiler
Displacement transducer
Block of length 250mm used for experiments
Triangular target pieces
Size of block used for experiments

Figure 3.7.3 *Laser profiler scanning a block*

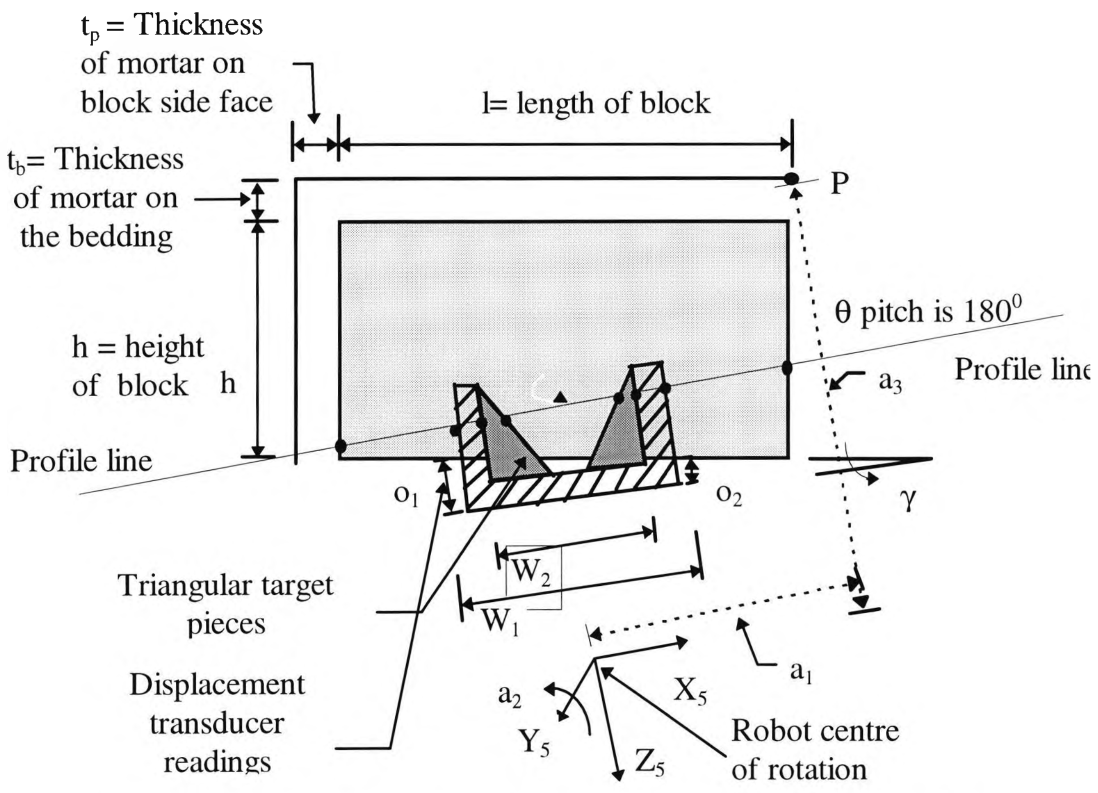


Figure 3.7.4 *Reference point 'P' relative to robot centre of rotation*

3.7.1 Design of the system

The laser device used is an analogue displacement sensor (LADS) having an accuracy of ±0.01 mm (figure A.3). A description of the device and its specifications are given in appendix A.5. It is mounted on a linear motion axis. The system's motion is provided by a 12V DC servo motor, which drives the laser unit by means of a toothed belt. Switches on either side of the profiler change the direction of the motion. The position of the laser device is measured using a modular shaft encoder. The translational range of the laser profiler is 300mm, which is unfortunately was too small to profile the length of block adopted. A calibration block of length 250 mm, therefore, was used for the purpose of experimentation (figure 3.7.3). The signal from the laser device and the encoder were converted from analog to digital, using an A/D converter (PICO) described in appendix A.6. The encoder measurements were calibrated to give readings in millimetres. A description of the laser signal calibration are given in section 3.10.3.5.2.

The dimensions of the triangular targets are 60 x 60 mm. These are shown in figure 3.7.8. They are mounted as shown in figures 3.7.2-3.7.3. Their use is explained in section 3.7.2. The laser profiler is positioned close to the block pick-up from the conveyor. With the block gripped, the robot presents the gripper to the laser profiler, aligned to it (figure 3.7.3). The mid-point of the gripper is set to be at the mid-point of the profiler's range, making sure it is at an offset within the range of the LADS.

3.7.2 Processing laser profile data

The profile readings of the laser device are analysed to extract the edges of the gripper fingers, the triangular targets and the block. An example of the raw signal is shown in figure 3.7.5. Some processing is needed to extract the edges. This was done by using a signal window that is passed through the data and the median of this found. The size of this window is an odd number. The processed data depends on the median of the data points in the window. Different size windows were tried, the larger the window the less accurate the edges starting point. A window of size 7 was the smallest one giving a satisfactory result. An example of the processed data is shown in figure 3.7.6, in which the block profile can be seen very clearly. The edges of the block and gripper are apparent, these detected using the method of 'gradient edge detection' Gonzalez *et al.*, 1987). This was done by taking the gradient of a window of three neighbouring values. Naturally, an edge has a steep gradient and, in our case, the gradient was taken as >5 for an edge (figure 3.7.7). The corresponding position readings of the edge points were noted. From this the various measurements needed for calculating the block to gripper orientation were computed by taking the difference in the positions, as shown in table 3.7.1. Refer to figure 3.7.8 for an illustration showing the various variables used in table 3.7.1. The measurements required are explained in the next section.

Table 3.7.1 *Processing the laser profile scan*

Variable name	Position reading of edges (mm)
A_2	$\lvert pos(edge1)-pos(edge2) \rvert$
A	$\lvert pos(edge7)-pos(edge8) \rvert$
B_2	$\lvert pos(edge2)-pos(edge3) \rvert$
B	$\lvert pos(edge7)-pos(edge6) \rvert$
C_2	$\lvert pos(edge3)-pos(edge4) \rvert$
C	$\lvert pos(edge6)-pos(edge5) \rvert$
D	$\lvert pos(edge4)-pos(edge5) \rvert$

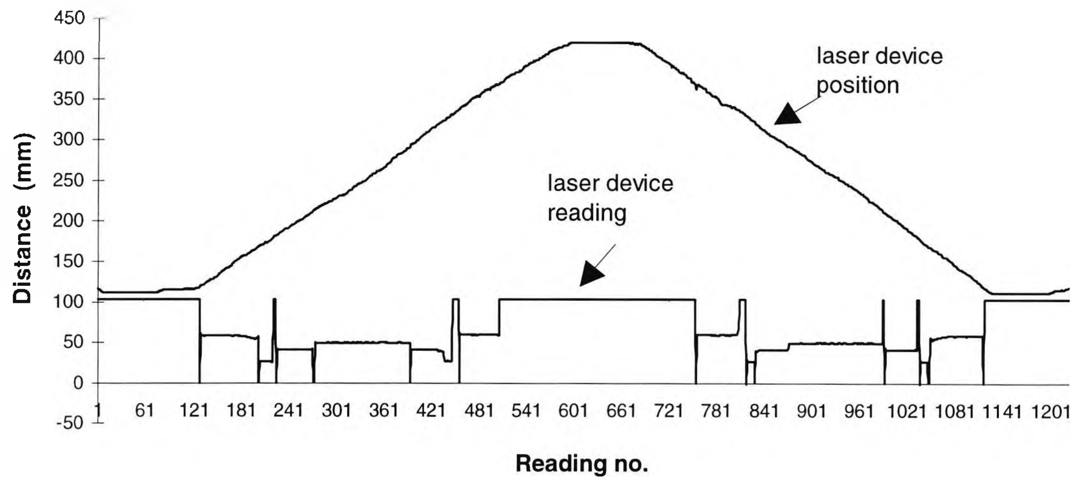


Figure 3.7.5 *Raw signal of the laser profiler*

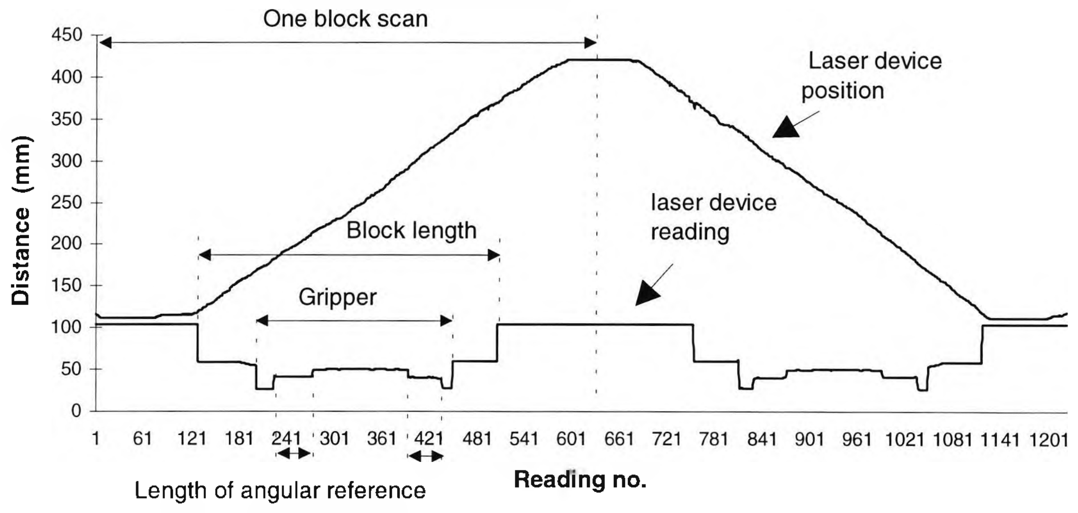


Figure 3.7.6 *Processed signal of the laser profiler*

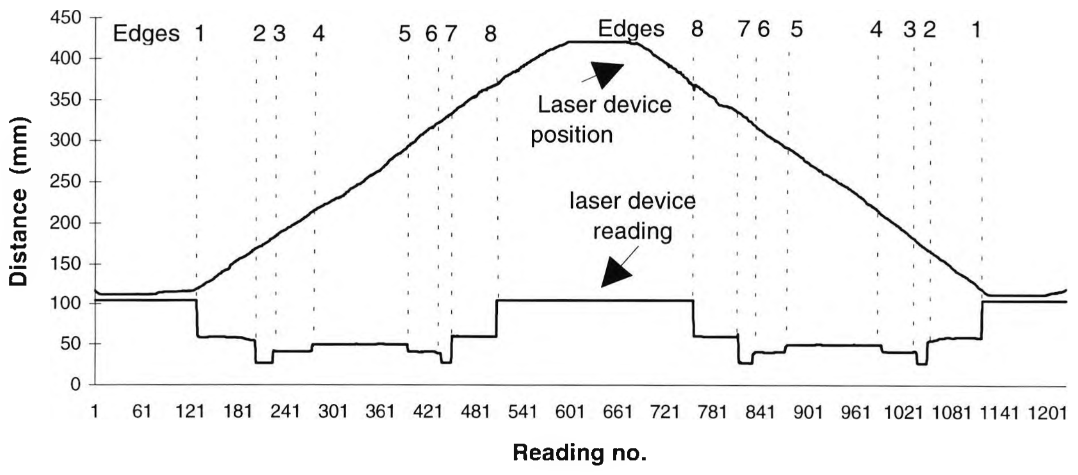


Figure 3.7.7 *Determining edges in the laser profile and their corresponding position readings*

3.7.2.1 Block to gripper geometry

Determining the exact position of the picked block in the gripper is vital for the dispensing of mortar onto the block, and the laying operation. Real-time adjustments needed to be made, which take into consideration the actual block dimensions and the location of the block in the gripper. This was done by determining the position of the reference point ‘P’, relative to the robot’s centre of rotation (figure 3.7.4). ‘P’ being the point at which the dispensing nozzle will be set, for the mortar application move described in section 3.8. The location of the point ‘P’, relative to the block, is determined from the thickness of the mortar t_b and t_p that is to be applied, as well as the length and height of the block (figure 3.7.4). There was a need to determine the position of that point ‘P’, independently of the gripper’s orientation (i.e. regardless of the roll_axis position). This was investigated using the two triangles mounted on the gripper. The data from the processed laser profile of the block is used to obtain the measurements for the variable A, A_2, B, B_2, C, and C_2, needed for the calculations, as shown in table 3.7.1. An example of this is illustrated in figure 3.7.8, where the angle about the roll axis, between the laser strike and the gripper is α, and the angle of the block to the gripper γ (figure 3.7.4 & 3.7.8). This makes the angle of the block to the laser profile to be (α-γ), as shown in figure 3.7.9. The length of the block ‘l’ measured using the laser profile is calculated using equation 3.7.1.

$$l = (A_2 + B_2 + C_2 + D + C + B + A) \times \cos(\alpha - \gamma) \qquad \text{eqn 3.7.1}$$

The goal, in locating the gripped block relative to the gripper, is to determine the point ‘P’ shown in figure 2.5.4, according to the following procedure:

(i)-Adjustments are made to the thickness of the mortar to be applied on the block, where t_b is the thickness of the mortar on the bedding face, and t_p the thickness on the vertical joint face. This is needed because of the expected inconsistencies between the block dimensions (from the theoretical task) and the actual dimensions measured using the conveyor sensors (described in section 3.5.1.2). The adjustment for t_b is made using equation 3.7.2, ‘H’ being the measured height from the conveyor sensor data. The adjustments for t_p is made using equation 3.7.3, where 'TL' is the measured length from the conveyor sensor data.

$$t_b = t_b + (ideal\_width - H) \qquad \text{eqn 3.7.2}$$

$$t_p = t_p + (ideal\_length - TL) \qquad \text{eqn 3.7.3}$$

(ii)-The angle of the block to the gripper, γ, illustrated in figures 3.7.4 and 3.7.8-3.7.11, is calculated using equation 3.7.4. The distances O_1 and O_2 are measured using the LVDT transducers mounted on both sides of the gripper. 5mm is added to the readings, this being the zero offset of transducer readings (figure 2.5.10). W_1 is the distance between the two LVDT transducers.

$$\gamma = (O_1 - O_2)/W_1 \qquad \text{eqn 3.7.4}$$

(iii)- The angle, about the roll axis, between the laser strike and the gripper α, is calculated using equation 3.7.5, where C_1 , D, and C_2 are the distances between the detected edges of the triangular targets as shown in figure 3.7.8. W_2 is the actual inner width of the gripper.

$$\alpha = \cos^{-1}(W_2/(C_1 + D + C_2)) \qquad \text{eqn 3.7.5}$$

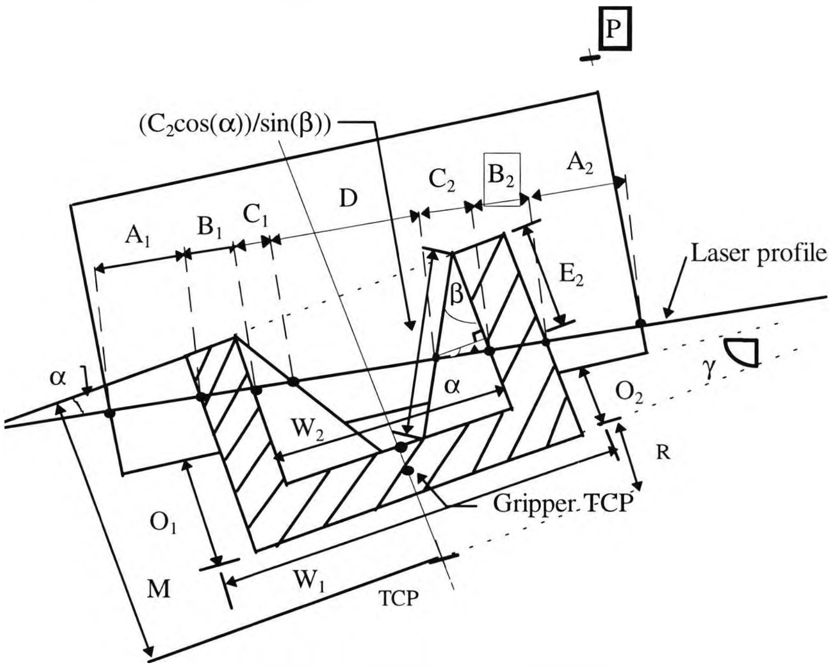


Figure 3.7.8 *Example of the gripper at an angle to the laser profiler*

(iv)-Finally, the offset of the reference point 'P', from the centre of rotation of the robot is calculated. This was done in steps, using the information calculated above, and the known measurements of the gripper from the centre of rotation. The first step is to calculate the transformation matrix from block to gripper co-ordinates and vice versa using equations 3.7.6-3.7.7. $x^{/}, z^{/}$ are the robot co-ordinates, and $x^{//}, z^{//}$ are the block co-ordinates.

$$\begin{Bmatrix} x^{//} \\ z^{//} \end{Bmatrix} = \begin{bmatrix} \cos(\gamma) & \sin(\gamma) \\ -\sin(\gamma) & \cos(\gamma) \end{bmatrix} \begin{Bmatrix} x^{/} \\ z^{/} \end{Bmatrix} \qquad \text{eqn 3.7.6}$$

$$\begin{Bmatrix} x^{/} \\ z^{/} \end{Bmatrix} = \begin{bmatrix} \cos(\gamma) & -\sin(\gamma) \\ \sin(\gamma) & \cos(\gamma) \end{bmatrix} \begin{Bmatrix} x^{//} \\ z^{//} \end{Bmatrix} \qquad \text{eqn 3.7.7}$$

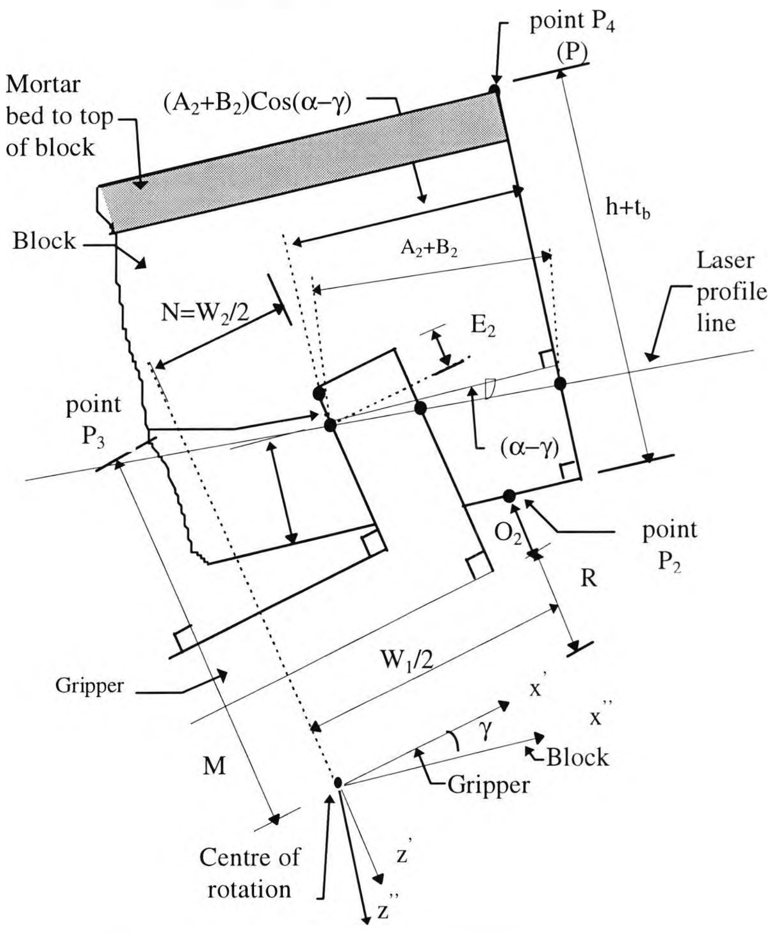


Figure 3.7.9 *Reference point 'P' relative to centre of rotation*

The second step is to calculate the co-ordinates at points P$_2$ $(x_2^{/}, z_2^{/})$, shown in figure 3.7.9, in the x,z plane as following:

$$(x_2^{/}, z_2^{/}) = ((W_1/2), -(O_2 + R)) \qquad \text{eqn 3.7.8}$$

, where R is the distance from the centre of rotation to the bottom side of the gripper, and O$_2$ the result of the transducer reading. The co-ordinates of point P$_3$ $(x_3^{/}, z_3^{/})$ are calculated in the x,z plane as follows:

$$(x_3^{/}, z_3^{/}) = ((W_2/2), -(M - E_2)) \qquad \text{eqn 3.7.9}$$

, where M is the distance from the centre of rotation to the top line of the gripper and E$_2$ calculated using:

$$E_2 = C_2(\cos(\alpha)/\tan(\beta)) + C_2 \sin(\alpha) \qquad \text{eqn 3.7.10}$$

Points P$_2$ & P$_3$ are then expressed in the block co-ordinate system, using the transformation matrix equation 2.5.6:

$$x_2^{//} = \cos(\gamma)x_2^{/} + \sin(\gamma)z_2^{/} \qquad \text{eqn 3.7.11}$$

$$z_2^{//} = -\sin(\gamma)x_2^{/} + \cos(\gamma)z_2^{/} \qquad \text{eqn 3.7.12}$$

and

$$x_3^{//} = \cos(\gamma)x_3^{/} + \sin(\gamma)z_3^{/} \qquad \text{eqn 3.7.13}$$

$$z_3^{//} = -\sin(\gamma)x_3^{/} + \cos(\gamma)z_3^{/} \qquad \text{eqn 3.7.14}$$

The co-ordinates of point ‘P’, denoted as point P$_4$, are then expressed in the block co-ordinate system. $x_4^{//}$ is calculated as follows:

$$x_4^{//} = x_3^{//} + ((A_2 + B_2)\cos(\alpha - \gamma)) \qquad \text{eqn 3.7.15}$$

, where A$_2$, and B$_2$ are the distances between the detected edges of the gripper and the block edge (figure 3.7.9). Substituting the value of $x_3^{//}$ from equation 3.7.13 into equation 3.7.15 results in:

$$x_4^{//} = \cos(\gamma)x_3^{/} + \sin(\gamma)z_3^{/} + ((A_2 + B_2)\cos(\alpha - \gamma)) \qquad \text{eqn 3.7.16}$$

$z_4^{//}$ is calculated as follows:

$$z_4^{//} = z_2^{//} - (H + t_b) \qquad \text{eqn 3.7.17}$$

Substituting the value of $z_2^{//}$ from equation 3.7.12 into equation 3.7.17, therefore:

$$z_4^{//} = -\sin(\gamma)x_2^{/} + \cos(\gamma)z_2^{/} - (H + t_b) \qquad \text{eqn 3.7.18}$$

Finally, using the transformation matrix equation 3.7.7, the co-ordinates of point P$_4$ $(x_4^{//}, z_4^{//})$ are expressed in the robot co-ordinate system $(x_4^{/}, z_4^{/})$ as follows:

$$x_4^{/} = \cos(\gamma)x_4^{//} - \sin(\gamma)z_4^{//} \qquad \text{eqn 3.7.19}$$

and,

$$z_4^{/} = \sin(\gamma)x_4^{//} + \cos(\gamma)z_4^{//} \qquad \text{eqn 3.7.20}$$

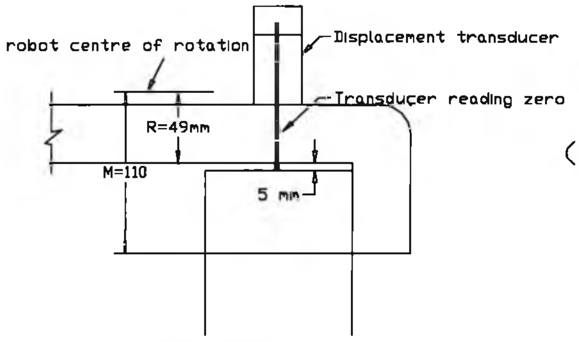
Substituting the values of $x_4^{//}, z_4^{//}$ from equations 3.7.16, and 3.7.18 into equations 3.7.19 and 3.7.20 result in the expressions:

$$\begin{aligned} a_1 = x_4^{/} = \cos^2(\gamma)x_3^{/} + \sin(\gamma)\cos(\gamma)z_3^{/} + (A_2 + B_2)\cos(\alpha - \gamma)\cos(\gamma) + \\ \sin^2(\gamma)x_2^{/} - \sin(\gamma)\cos(\gamma)z_2^{/} + (H + t_b)\sin(\gamma) \end{aligned} \qquad \text{eqn 3.7.21}$$

and

$$\begin{aligned} a_3 = z_4^{/} = \sin(\gamma)\cos(\gamma)x_3^{/} + \sin^2(\gamma)z_3^{/} + (A_2 + B_2)\cos(\alpha - \gamma)\sin(\gamma) - \\ \cos(\gamma)\sin(\gamma)x_2^{/} + \cos^2(\gamma)z_2^{/} - (H + t_b)\cos(\gamma) \end{aligned} \qquad \text{eqn 3.7.22}$$

These values are the offsets a$_1$, from the centre of rotation in the x-axis direction, and a$_3$ from the centre of rotation in the z-axis direction. These are used when calculating moves with the point ‘P’ as the TCP. These types of moves are needed when executing the mortar dispensing operation described in section 3.8.



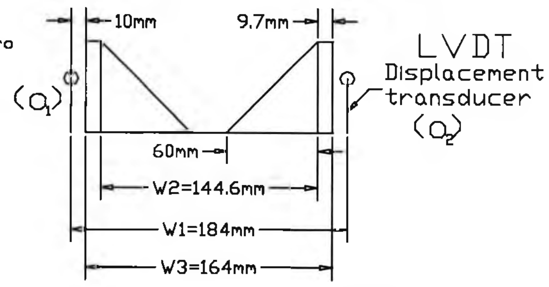


Figure 3.7.10 *Gripper side* **Figure 3.7.11** *Gripper front*

3.8 Mortar pump and dispenser

3.8.1 Peristaltic pump

A peristaltic type pump was used for mortar dispensing (figure 3.8.1). This type of pump relies on rotating rollers to squeeze a hose element (figure 3.8.2), which in-turn progressively pushes the mortar from the inlet pipe to the delivery pipe. Over the inlet is mounted a hopper which feeds mortar by gravity. Even compression of the mortar is provided by compression rollers, with minimum friction, thus providing minimum wear of the pump mechanics. The pump can rotate at between 0-100 rpm with a hose diameter of 25 mm. Its control box comprises a speed control card for the DC motor and a microcontroller card for interfacing with a PC via a parallel port. This allows starting, stopping, and the speed of the pump to be controlled by the robot cell logic.

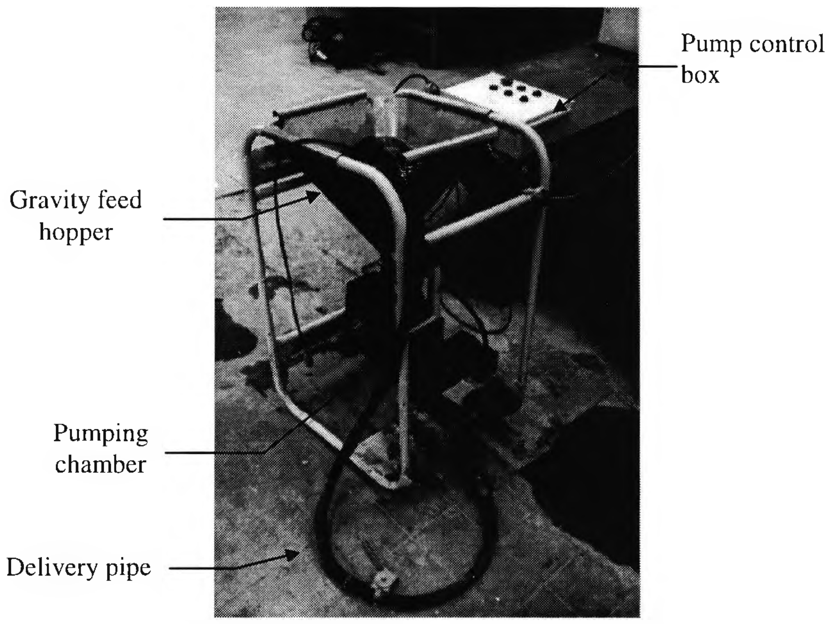


Figure 3.8.1 *Peristaltic pump, power supply and control units*

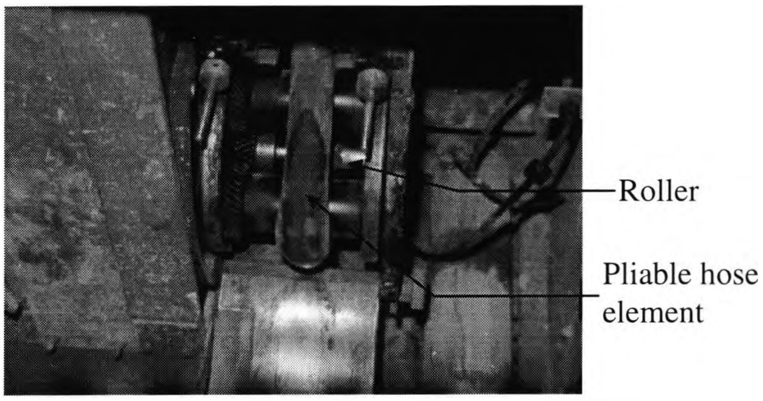


Figure 3.8.2 *Showing the four rollers of the peristaltic pump mechanism, and the hose element*

3.8.2 Nozzle

A nozzle, developed at City university (Kimble, 1991), was used for the dispensing of the mortar (figure 3.8.3). The dimensions and design of the nozzle (figure 3.8.4) were derived empirically through trials with wet sand and mortars. The thickness of the bead delivered is typically 10-11 mm this relating to the thickness of mortar joints. To enable its use in vertical joint injection, the external thickness of the nozzle was designed to be less than 10 mm. For the mortar width to be about 70 mm, the necessary internal width was determined as 61 mm. A primary factor in the design of this nozzle was that it's cross section area was the same as that of the connecting delivery pipe, thus assisting continuity of flow.

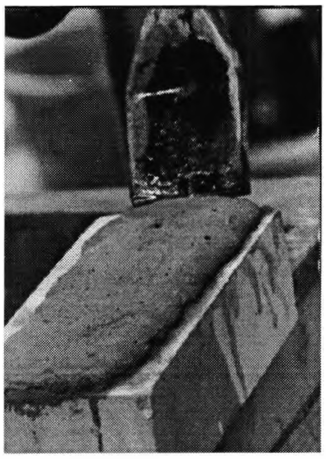


Figure 3.8.3 *Nozzle attached to the delivery pipe of the pump*

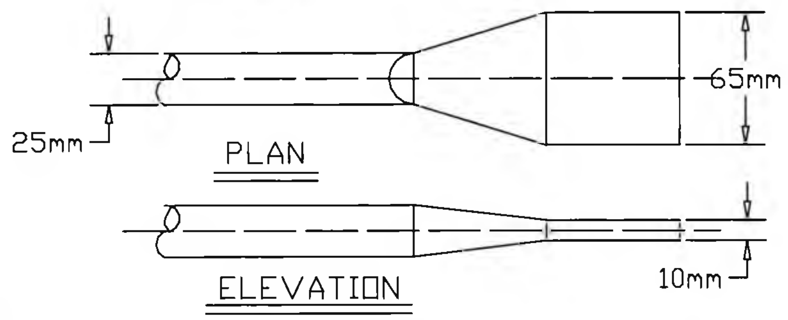


Figure 3.8.4 *Schematic diagram showing dimensions of the nozzle, with an internal opening of 61 mm × 6 mm*

3.8.3 Mortar dispensing

Two types of mortar application method were investigated. The first method required each individual block to be "pre-buttered" on two faces, and then placed into position. In commercial practice, this would be done on the conveyor, using a combined rotating and dispensing device mounted on it. Such operations require the pump to have intermediate start and stop facilities, as any time lagging would cause excess mortar to contaminate the conveyor. This method would increase productivity, by allowing the robot to simultaneously concentrate on the tasks of picking and laying masonry units. As this system would have been costly to make, the robot was used instead, to present the masonry units to the dispenser and rotate them under the nozzle during the application of mortar (figures 3.8.5 and 3.8.6). A special move, carried out by the robot, was designed for that purpose. This is described in section 3.8.3.1. The second method is to apply the mortar on pre-positioned blocks (figure 3.8.7), after each layer of the project is complete and then inject mortar into the vertical joints.

For conformity to the project dimensions and the need for alignment, two options are considered. The first is to apply uniform thickness mortar and bed down to compensate for the variation in the block length and height, and the second to compensate for each block by adjusting the thickness of the mortar application. The thickness of the mortar can be adjusted automatically by varying the dispensing rate of the pump or varying the speed of the robot during the dispensing move.

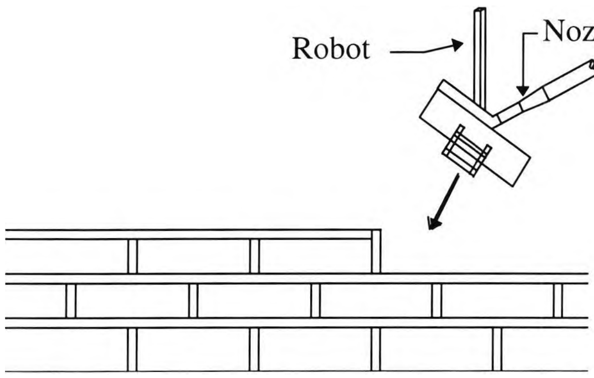


Figure 3.8.5 *Individual blocks 'buttered' using the robot to rotate the block*

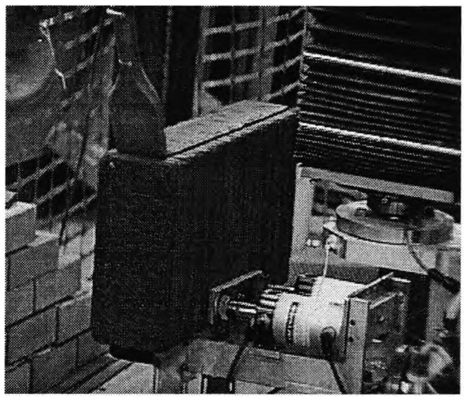


Figure 3.8.6 *Picture showing the robot being used to 'butter' an individual block*

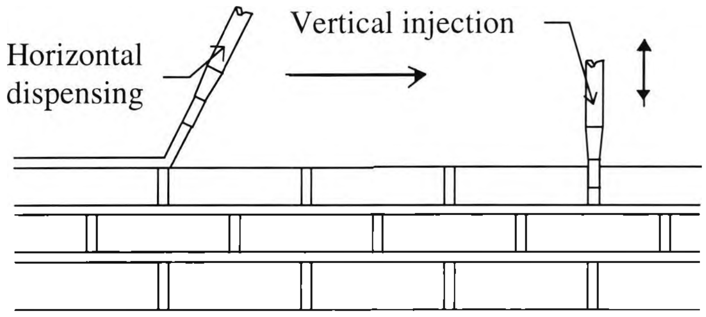


Figure 3.8.7 *Mortar laid on pre-positioned blocks*

3.8.3.1 Single pass move for dispensing mortar

A single pass move was designed for mortar dispensing on both sides of a block. This is part of stage 5 of the building process. This was done by fixing the nozzle (x_0,z_0) and rotating and translating the block in the x,z plane relative to the nozzle (figures 3.8.10-3.8.11f). The move was designed to be in three stages. During the first stage the bedding face of the block is buttered (figures 3.8.11a and 3.8.11b), during the second stage the corner of the block is buttered (figures 3.8.10 and 3.8.11c), and in the third stage, the joint end of the block is buttered (figures 3.8.11d and 3.8.11e). In section 3.7.3 it was shown how the mortar thickness, 't_b' for the bedding face, and 't_p' for the side of the masonry unit were adjusted using equations 3.7.2-3.7.3, to compensate for the discrepancies in the block dimensions. The width of the mortar bed 'MW' is set to be around 70 mm. The dispensing move uses the reference point 'P' as it's TCP (figure 3.8 in section 3.4.1). This which, described in section 3.7.3, takes into account the adjusted mortar thickness and the block grip according to equations 3.7.21-3.7.22. The dispensing move is made up of 'N' move increments, in the x,z plane. The total angular rotation of the block 'φ_t' is 90°. Taking the length and height of the block to be 'TL' and 'H' the length 'd_1' is the block length plus the mortar thickness on the side, and the length 'd_2' is the block height plus the mortar thickness on the bedding face. The dispensing rate of the pump, 'Q' (litres/min), which can be set between 0-3 litres/min, is held constant throughout the move. The relative velocities V_1 and V_2, for the dispensing moves of the bedding face and the adjacent face are calculated respectively:

$$V_1 = Q/(t_b \times MW) \qquad \text{eqn 3.8.1}$$

$$V_2 = Q/(t_p \times MW) \qquad \text{eqn 3.8.2}$$

The velocity during stage 2 of the dispensing move V_c (i.e. when going round the corner of the block), is taken as the average of the two velocities V_1 and V_2. The radius 'r' of the circumference of the corner, which the nozzle moves along (figures 3.8.8-3.8.10), is chosen to equal the smallest of the mortar thickness. The distance travelled for the first stage is (d_1-r), the second stage is (πr/2), and the third stage is (d_2-r), as shown in figures 3.8.8-3.8.10. The total time of the move T_{total}, is equal to the total time taken for each stage of the process (T_1, T_2, T_3), where

$$T_1 = (d_1 - r) / V_1 \qquad \text{eqn 3.8.3}$$

and $$T_2 = (\pi r) / (V_1 + V_2) \qquad \text{eqn 3.8.4}$$

and $$T_3 = (d_2 - r) / V_2 \qquad \text{eqn 3.8.5}$$

A continuous roll-axis function is employed using equation 3.8.6 to represent the position of angle φ at time 't', and equation 3.8.7 to represent the angular velocity at time 't':

$$\varphi = At + Bt^2 + Ct^3 \qquad \text{eqn 3.8.6}$$

$$(d\varphi/dt) = \dot{\varphi} = At + 2Bt + 3Ct^2 \qquad \text{eqn 3.8.7}$$

Time t_c, represented in equation 3.8.8, is the time when the nozzle is at the corner of the block (figure 3.8.11c):

$$t_c = T_1 + (T_2 / 2) \qquad \text{eqn 3.8.8}$$

The angle the block is rotated, up to the time t_c, is φ_c. Substituting these values in equation 3.8.6 gives:

$$\varphi_c = At_c + Bt_c^2 + Ct_c^3 \qquad \text{eqn 3.8.9}$$

, with the distance 'd' moved in that time being:

$$d = (d_1 - r) + (\pi r / 4) \qquad \text{eqn 3.8.10}$$

At the end of the move, the time taken to complete the move is T_{total}. At this time, the block would have rotated an angle φ_t . Substituting these values in equation 3.8.6, we get:

$$\varphi_t = (\pi/2) = AT_{total} + BT_{total}^2 + CT_{total}^3 \qquad \text{eqn 3.8.11}$$

Differentiating equation 3.8.6 twice gives the angular acceleration to be:

$$(d^2\varphi/dt^2) = 2B + 6Ct \qquad \text{eqn 3.8.12}$$

At the corner, the move is at constant velocity, i.e. with no acceleration, thus resulting in the expression:

$$0 = 2B + 6Ct_c^3 \qquad \text{eqn 3.8.13}$$

Solving for the constant B in equation 3.8.13, results in:

$$B = -3Ct_c \qquad \text{eqn 3.8.14}$$

Substituting the value of constant B in equation 3.8.9 gives:

$$\varphi_c = At_c - 2Ct_c^3 \qquad \text{eqn 3.8.15}$$

Solving for the constant A in equation 3.8.15, we get:

$$A = (\varphi_c + 2Ct_c^3)/t_c \qquad \text{eqn 3.8.16}$$

Substituting the values of constants A and B into equation 3.8.11, and solving the equation for the constant C results in:

$$C = ((\pi/2) - (\varphi_c T_{total}/t_c))/(T_c^3 - 3T_{total}^2 t_c + 2T_{total} t_c^2) \qquad \text{eqn 3.8.17}$$

, where the total time of the move is:

$$T_{total} = T_1 + T_2 + T_3 \qquad \text{eqn 3.8.18}$$

Assuming the starting point of the move is where the nozzle is at x_0,z_0 (figure 3.8.11a), the position of reference point 'P' during the move is calculated in the x,z plane, with reference to starting point. This is be calculated for the three stages of the move using the procedure described below:

Procedure to calculate the mortar application move increments:

(i) Calculate the values of d_1, d_2 using the block dimensions (TL and H), and the thickness of the mortar bed (t_p and t_b).

(ii) Set the value of the radius r

- if ($t_b < t_p$) , then $r = t_b$, otherwise $r = t_p$

(iii)Calculate the values of V_1 and V_2, using the width 'MW' and thickness of the mortar bed, and the fixed dispensing rate Q, in equations 3.8.1-3.8.2 respectively.

(iv)Calculate the values of T_1, T_2, and T_3 using equations 3.8.3-3.8.5 respectively. From those values calculate the value of t_c and T_{total} , in equations 3.8.8 and 3.8.18.

(v) Calculate constant C using t_c and T_{total} in equation 3.8.17. Using that, calculate the values of constants A and B from equations 3.8.15 and 3.8.14 respectively.

(vi)Using the value T_{total}, and the number of move increments the process will be carried out in 'N', the value of the time increments Δt is

- $\Delta t = T_{total} / N$

(vii)The displacement 'd' and the angle 'φ' covered at time 't' for 'N' increments is calculated. For each increment, depending on which stage of the move we are at, the following is computed (figure 3.8.8):

- $t = t + \Delta t$
- Calculate the angular displacement φ , and velocity $\dot{\varphi}$, at time 't' and, using equations 3.8.6 and 3.8.7 respectively.
- <u>For ($t = 0 \rightarrow t = T_1$) i.e. during the first stage</u>
    - Distance travelled at time 't' is:

      $d = V_1 \times t$
  - The x ,z positions of point 'P' at distance 't' are:

    $(x_{pos} = x_0 - d\cos\varphi)$,

$(z_{pos} = z_0 - d\sin\varphi)$.

Integrating the above expressions with respect to time gives the velocities at that point as,

$(x_{vel} = -V_1\cos\varphi + (\dot{\varphi}d)\sin\varphi)$, $(z_{vel} = -V_1\sin\varphi - (\dot{\varphi}d)\cos\varphi)$

- For (t >T$_1$ → t =(T$_1$ + T$_2$)) i.e. during the second stage (figure 3.8.9)
  - Distance travelled at time 't' is, $d = (d_1 - r) + V_c(t - T_1)$

The x ,z positions of point 'P' at distance 't' and angular displacement around the corner 'χ ' are

$(x_{pos} = x_0 - r\sin\varphi - (d_1 - r)\cos\varphi + r\sin(\varphi - \chi))$,

$(z_{pos} = z_0 - r\cos\varphi - (d_1 - r)\sin\varphi + r\cos(\varphi - \chi))$ Integrating the above expressions with respect to time gives the velocities at that point as: $(x_{vel} = -r\dot{\varphi}\cos\varphi + (d_1 - r)\dot{\varphi}\sin\varphi + r(\dot{\varphi} - \dot{\chi})\cos(\dot{\varphi} - \dot{\chi}))$,

$(z_{vel} = -r\dot{\varphi}\sin\varphi - (d_1 - r)\dot{\varphi}\cos\varphi + r(\dot{\varphi} - \dot{\chi})\sin(\dot{\varphi} - \dot{\chi}))$,

where the corner angle covered 'χ' at time 't' is:

$\chi = (d - (d_1 - r))/r$, with the radial velocity for that angle being:

$\dot{\chi} = V_c/r$

- For (t >(T$_1$+T$_2$) → t = T$_{total}$) i.e. during the third stage (figure 3.8.9)
  - Distance travelled at time 't' is:

    $d = V_2(t - (T_1 + T_2)) + (d_1 - r) + (\pi r/2)$
  - The x ,z positions of point 'P' at distance 't' are:

    $(x_{pos} = x_0 - d_1\cos\varphi - (d - (\pi r/2) - (d_1 - r) + r)\sin\varphi)$,

    $(z_{pos} = z_0 - d_1\sin\varphi - (d - (\pi r/2) - (d_1 - r) + r)\cos\varphi)$

  Integrating the above expressions with respect to time gives the velocities at this point as:

  $(x_{vel} = \dot{\varphi}d_1\sin\varphi - \dot{\varphi}(d - (\pi r/2) - (d_1 - r) + r)\cos\varphi - V_2\sin\varphi)$,

  $(z_{vel} = \dot{\varphi}d_1\cos\varphi - \dot{\varphi}(d - (\pi r/2) - (d_1 - r) + r)\sin\varphi - V_2\cos\varphi)$

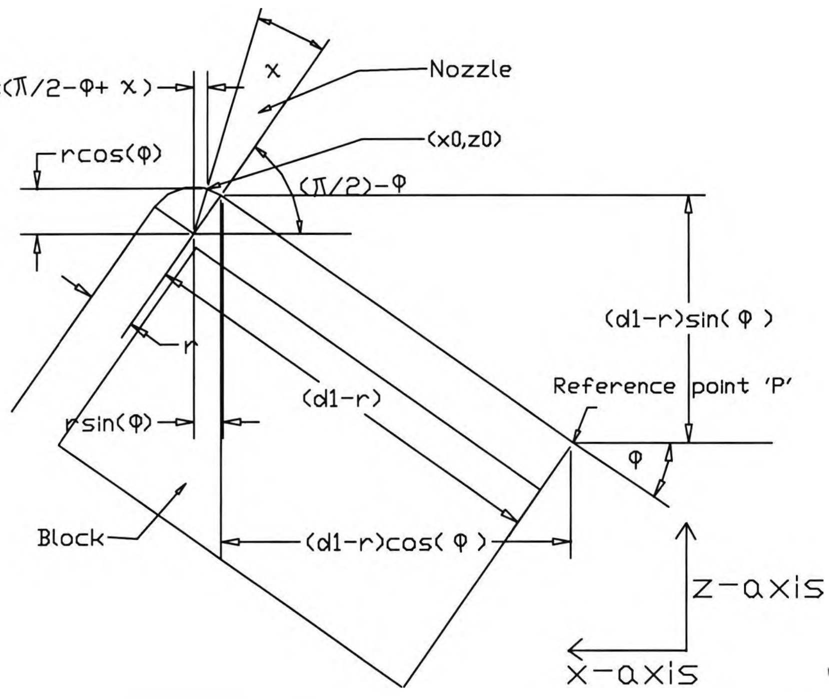


Figure 3.8.8 *Geometry of the mortar application move during the second stage*

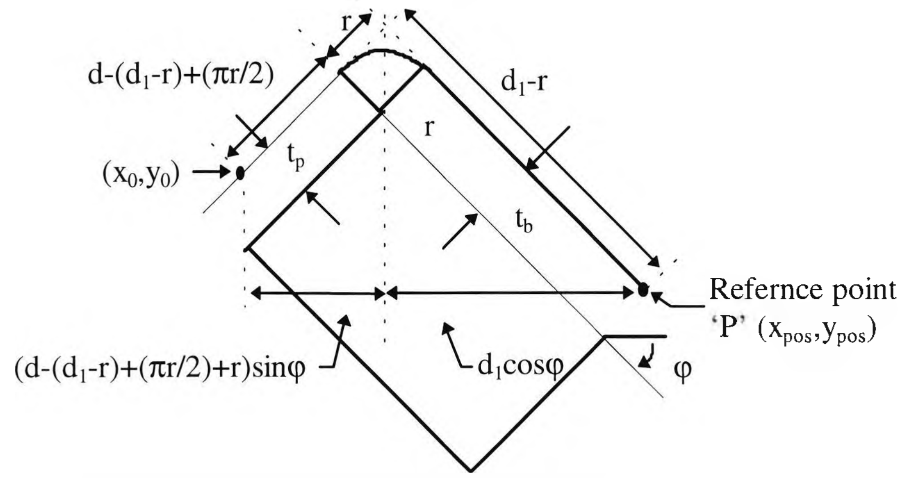


Figure 3.8.9 *Geometry of the mortar application move during the third stage*

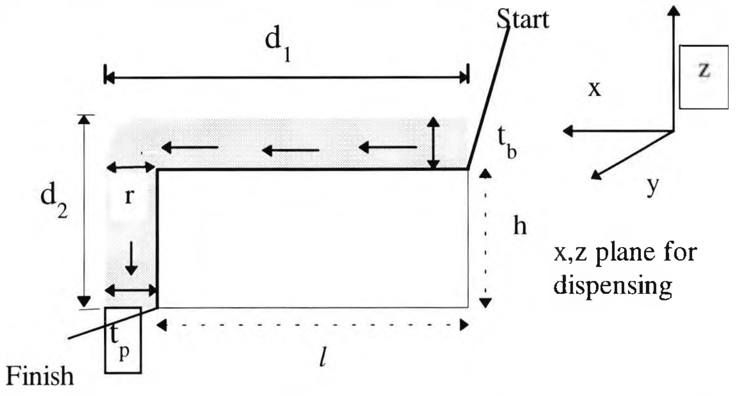
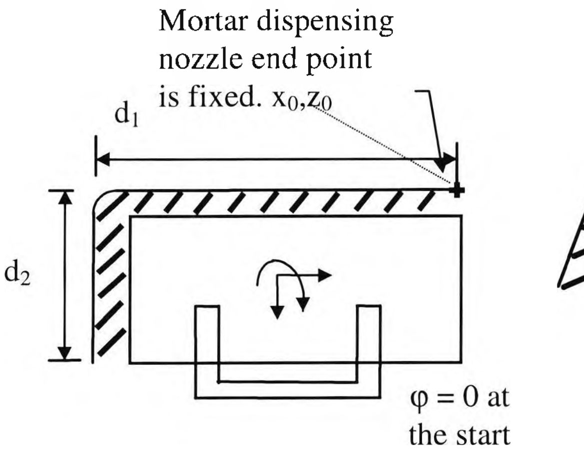


Figure 3.8.10 *Mortar thickness to be dispensed on the block using the dispensing move in the x,z plane*



Mortar dispensing nozzle end point is fixed. x_0,z_0
d_1
d_2
$\varphi = 0$ at the start

Figure 3.8.11a *Start of the dispensing move showing the nozzle's fixed position*

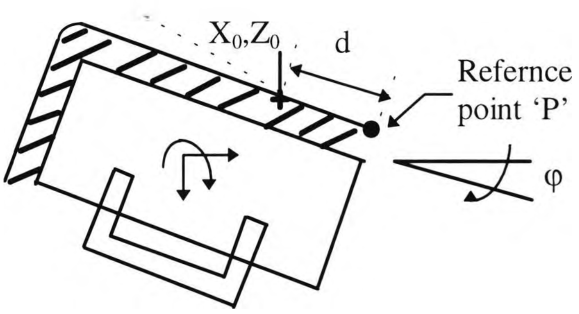


Figure 3.8.11b *During stage*(1) *of the dispensing move, when applying mortar on the bedding face of the block showing reference point 'P', with distance travelled 'd' and gripper moved an angle 'φ'*

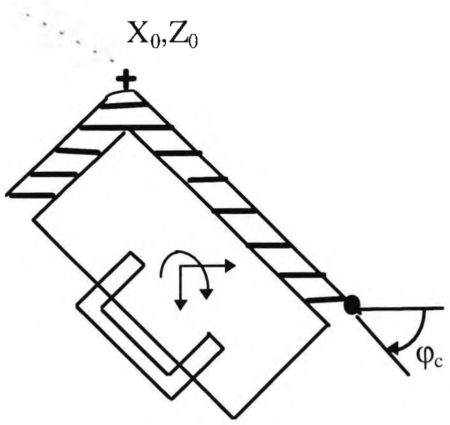


Figure 3.8.11c *During stage*(2) *of the dispensing move when applying mortar around the corner. Block moved an angle 'φ_c'*

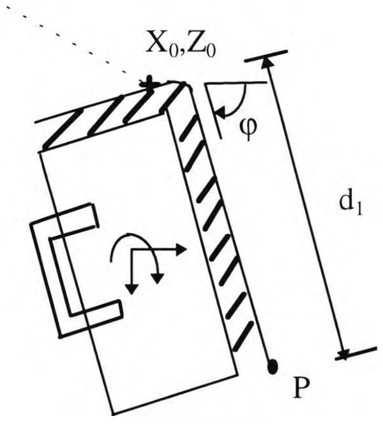


Figure 3.8.11d *Start of stage*(3) *of the dispensing move when applying mortar on the side of the block.*

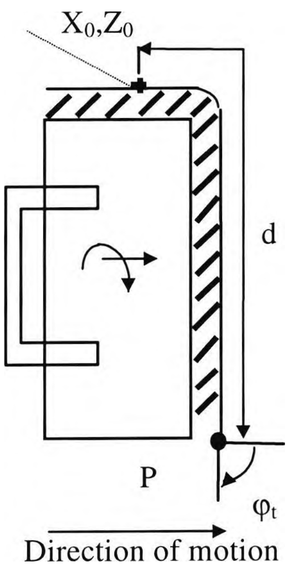


Figure 3.8.11e *During stage*(3) *of the dispensing move when applying mortar on the side of the block. Gripper moved an angle of 'φ_t'.*

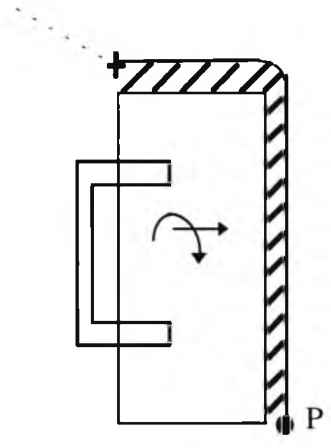


Figure 3.8.11f *End of the dispensing move.*

3.9 Laser beacon

A rotating planer laser was developed for level reference in the operation of the robot. A 'companion level eye' unit is supplied with the laser and this detects the presence of the laser beam strike. This unit has a voltage output which can be used for synchronisation and thus integration into the robot cell. In spite of the absolute positional accuracy (±0.5 mm) of the gantry robot used, a laser navigation beacon (figures 3.1 & 3.9.1) is included in the operation. This was done to make provisions for a future cell, which would be a mobile robot. The device comprises a laser beacon which is a Spectra Physics rotating laser level mounted on a microprocessor driven a vertical positioning axis (±0.3 mm). A 400 watt DC shunt motor is used to drive a rack and pinion arrangement, with an optical encoder providing the positioning information. The laser, which is electronically self levelling from a rough inclination of up to 4°, provides a horizontal planar reference to within ±1.5 mm at a radius of 30.5 m. The companion level eye detector can detect the laser strike over a 20 mm vertical position window and the output from this is fed to the robot controller. The beacon's microprocessor controller supports three modes of operation, the first being position searching, which is used to determine the robot's current vertical position, the second for datum maintenance, which is used to find the datum plane, and the third as level setting for use in block laying to a set level. Resource did not, however, permit integration of this device in the robot cell.

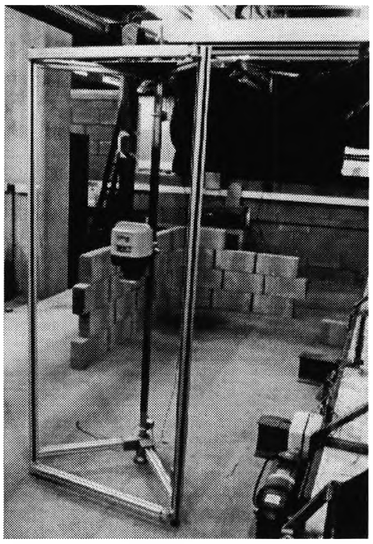


Figure 3.9.1 *Laser beacon*

3.10 Accelerometer

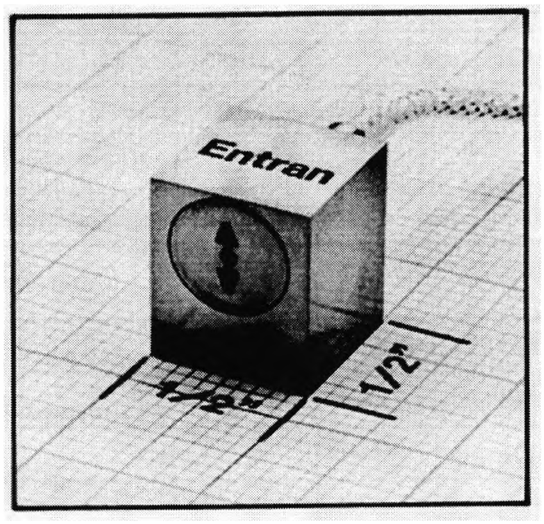


Figure 3.10.1 *Miniature heavy duty accelerometer*

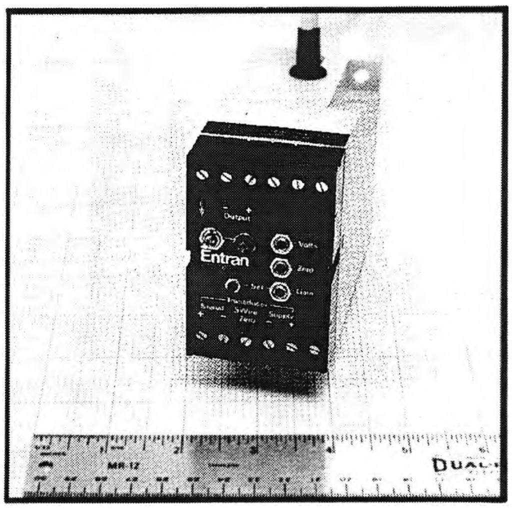


Figure 3.10.2 *Power supply*

The vibration of the end effector of the robot was measured using the miniature, heavy duty accelerometer shown in figure 3.10.1. This is needed to try and optimise robot moves by determining the settling time of a move, i.e. the time taken for the oscillations of the robot end-effector to die down after a move for it to be ready for use. The accelerometer is damped, and has a ‘g’ range of ±5 g output, weight of 10 grams and is built to withstand high overloads in operation and installation. The accelerometer operates in both static (steady state) and dynamic measurement of acceleration, vibration and shock. Its full scale output can be used without amplification and directly interface with most readout instrumentation. The power supply used for the accelerometer is shown in figure 3.10.2. A fuller specification of the device is given in appendix A.7.

The accelerometer employs either a full or half active Wheatstone Bridge consisting of semiconductor strain gages. The strain gages are bonded to a simple cantilever beam which is end loaded with a mass as shown in figure 3.10.3. Under acceleration a force on the cantilever is created by the g effect on the mass $F = m \times a$. The accelerated mass creates a force which in turn provides a bending moment to the beam. This moment creates a strain (proportional to the acceleration) which results in a bridge unbalance. With an applied voltage, this unbalance produces a mV deviation at the bridge output, which is proportional to the acceleration vector.

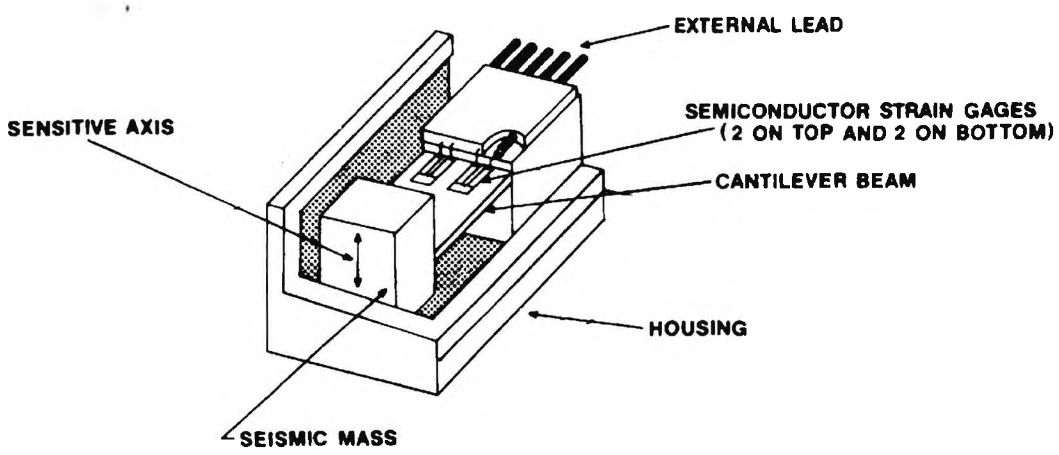


Figure 3.10.3 *Accelerometer cutaway*

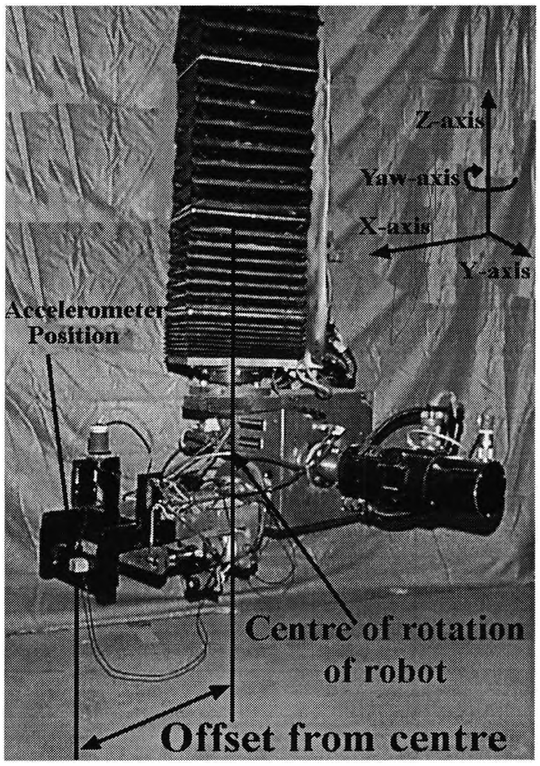


Figure 3.10.4 *End effector of robot*

3.10.1 Accelerometer positioning

The accelerometer was mounted at the furthest radial point of the end effector, which is offset from the centre of rotation of the robot's yaw axis by 105 mm, as indicated in figure 3.10.4. The acceleration vector is mounted to detect horizontal vibration. This arrangement enables measurement of the combined transitional and rotational accelerations, which, in turn, enables measurement of the sideways displacement of the end effector. Due to the part transitional and part rotational movement of the end-effector, the sideways displacement will be not strictly linear. However, small amount of rotation occurring means that the disturbance can be treaded as purely translational. This translational motion is considered to be a sufficient indicator of the robot response.

3.10.2 Accelerometer data capturing

The accelerometer reading is in analog form. The *PC-30* card was used as an analog to digital converter and an input interface to the computer. This has an A/D resolution of 12 Bits, set with a full scale input range of -5V to +5V. The card is plugged into an *XT* computer, which processes the data using an application program called *Status-30.* This combines the features of an oscilloscope, and a data logger. This application does, however, not provide any facilities to eliminate error in the captured data. It has a graphical interface and pull-down menus that enable the setting of the sampling frequency, the number of samples and the A/D Range. It can records 4096 samples, and allows sampling of up to 16 channels simultaneously. Captured data can be stored in a variety of formats, and can be readily made compatible with most spreadsheet and graphing programs. A sample of the *Status-30* application, an output file generated and details of the *PC-30* card, can be seen in appendix A.7.3.2 table A.5.

3.10.2.1 Power spectrum of signal

To analyse the frequencies of the signal captured, for the purpose of reducing noise and unwanted high frequencies, while preserving the low ones, a 'C' program was developed. This employed functions for Fast Fourier Transform (FFT) (Ramirez, 1985), prepared as C program. Verification of the algorithm and the C program was carried out and is covered in appendix H.1 and H.2.

The sampling rate of the signal, which was found to be 200Hz was found to be a reasonable sampling rate of our signal. The sampling rate captured all the frequencies that were of interest. This is verified by comparing signals sampled at various frequencies through experiments covered in appendix H.2.1.

3.10.3 Accelerometer signal processing

Apart from the acceleration of the end-effector, the displacement amplitude is of interest. To obtain approximate displacement values, a double integration process was applied to the accelerometer data. This is achieved by single and double numerical integration of the acceleration-time domain data using Simpson's first rule. This approach has been adopted on account of its simplicity and accuracy (Philipson *et al.*, 1974). A set of 'C' code programs have been prepared and verified for this.

To verify the results the accuracy of the displacement result an independent check was conducted. This was accomplished by comparing results of the displacement of the end-effector achieved from processing the accelerometer data with that using a laser analogue displacement sensor (LADS). This is covered in details in appendix H.

The experiments showed that the worst error in the integration method, based on the accelerometer signal was 0.8 mm. Table 2.8.1 summarises the results of the experiments carried out in this section. The maximum difference in the readings of the accelerometer and the LADS is 0.7 mm, and on average, there is a difference of 0.18 mm. This difference will not effect our end result significantly, as our main interest is to estimate the settling time of the end effector of the robot after a move (i.e. when the amplitude has decayed to ± 0.5 mm).

Table 3.10.1*Comparison of direct laser measurement with estimate from accelerometer*

Experiments	**Maximum displacement amplitude difference (mm)**	**Average displacement amplitude difference (mm)**
Manually rocking end-effector	0.41	0.18
Strong vibration move	0.83	0.19
Medium vibration move	0.77	0.21
Low vibration move	0.80	0.13
Average	**Maximum value 0.7 mm**	**Average difference 0.18 mm**

In this section it has been demonstrated that the method developed to measure the robot end-effector vibrations, using an accelerometer, is reliable and accurate. The experiments carried out to study the behaviour of the different moves the robot is capable of, and the optimum moves concluded from the results, are covered in chapter 4.

3.11 Robot path planning

The theory and programming of the different move types the robot used (figure 3.1) is covered in appendix I. The system limits are determined, along with investigations into the performance of different move types, to prove the system capabilities described appendix I. These are considered necessary steps to the understanding of the capabilities of the robotic system chosen for this research.

Motion Control for the robot, described in section 3.4, is achieved by parallel processing on three Smart Motion Control Cards (SMCCs). Path requirements are determined by the host computer and communicated to the SMCCs via a serial link (parallel option also available). This is achieved at a high level, using sequences of ASCII characters. These command sets can be handled on a buffered or non-buffered basis, the later being executed immediately and the former executed as a program stored on the SMCCs. The host computer is programmed using the Microsoft 'C' programming language. This version of 'C' has been adopted as common to all logical units within the robot cell.

Whilst most industrial robots are operated on pre-determined point to point cycles under constant acceleration conditions, the project requirements are for continuous motion under variable acceleration conditions. Continuous path control is necessary for the so called 'on the fly' path control needed for collision avoidance and other closed loop sensor driven activity. Variable acceleration is desirable for the following reasons, firstly, motors need not be subjected to shock loading and associated wear, secondly, motor power consumption is reduced and, finally, motion induced vibration is minimised. The latter is important, as excessive vibration leads to the necessity for considerable settling time at the end of moves and thus reduced productivity. This is particularly significant with the experimental robot because the Yaw axis is partially compliant.

The consideration of productivity inevitably leads to the requirement for least time moves, these constrained by the need to achieve smooth, minimal vibration inducing paths. The SMCCs selected for this project allows linear variations in acceleration and thus parabolic velocity variation. This is operated simultaneously on all axes according to given move requirements. To determine multi-axis moves on a least time basis it is necessary to determine the critical time axis i.e. the axis with the longest least time move of all axes. The move time determined for the critical time axis is then taken as the time for the move, the remaining axes operating on sub-critical paths.

In what follows, the general forms of two types of parabolic moves are determined, these denoted as 'U' and 'S' type moves. The 'U' move uses a single parabolic function and the 'S' move uses two blended parabolic functions.

It is important to note that the trajectories described in this chapter are all command trajectories and not necessary the actual trajectories that the system delivers. It is easy to generate a command trajectory that the robot is incapable of following, and this must be guarded against in writing any motion program. In this chapter, the system's capabilities are checked through experiments. The system's production of some of the command trajectories is described. The theory and results of this chapter support programming of the moves in chapter 4 and the development of the rule base for the theoretical task generation in chapter 9.

3.11.1 Multi-axis and multi-card programming

Even though the SMCC is a two axis controller, some restrictions apply to the use of the two axes because they are not independent of each other. The SMCC cards are daisy chained which automatically causes multiple cards to be time synchronised for the purposes of linear and circular interpolation. No other measure has to be taken to insure proper synchronisation and interpolation. Interpolation can be done between all the cards. There is, however, one limitation that must be observed, and that is that the X and Y axis of any card must be used simultaneously to perform the same type of operation (linear or circular).

For programming the minimum time, multi-axis move, the following steps have to be followed.

i. Determine for each card the type of move best suited to each axis, considering the distance for the move on each axis.
ii. Decide on the type of move for each card (i.e. linear, parabolic or an S-curve move).
iii. Find the best class of move from the category of moves chosen (e.g. single part parabolic or 3-part parabolic).
iv. Determine the critical axis for each card, by finding the greatest shortest move time of the axes having movement, and fixing that to be the move time for each card.

v. Recalculated moves to be executed in that fixed time.

3.11.2 Result of investigation

To programme the robot efficiently the capabilities of the robot had to be determined. This was achieved through a detailed description of the theory of the different move types the robot is capable of performing, and through experimentation (appendix I). The robot is capable of performing moves with constant acceleration (linear move) as well as variable acceleration (parabolic and S-curve moves). Programming the robot to perform minimum time moves is decided on the bases of the distance a move has to cover and the type of move we wish to make. Figures 3.11.1 and 3.11.2 show a summary of the minimum time parabolic and S-curve moves. These points enable avoidance of impossible moves, such as the ones which require the robot to reach system limits without having enough distance to achieve them.

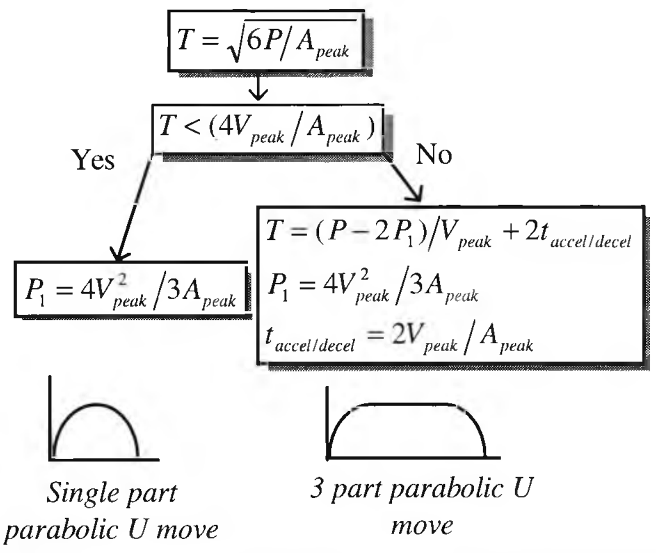
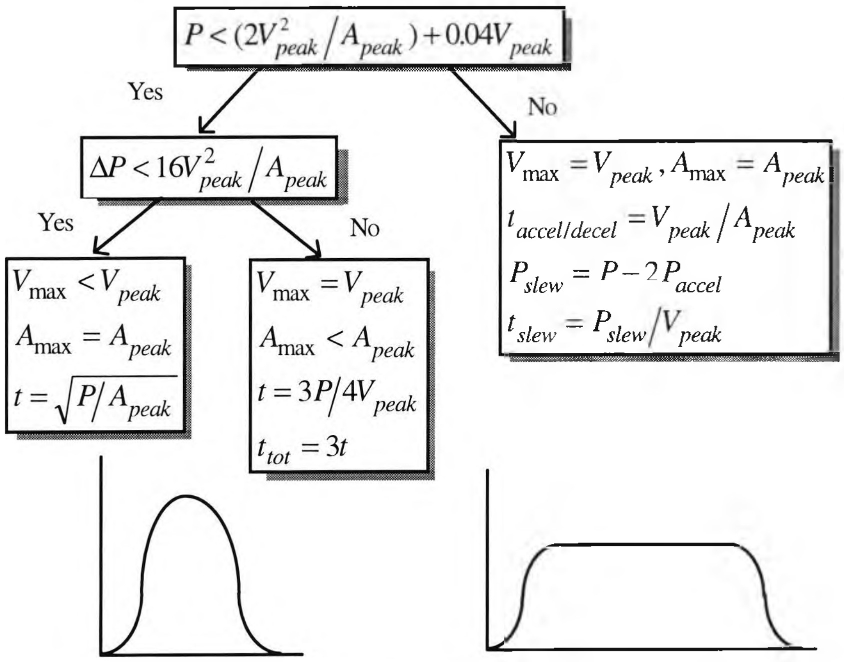

$T = \sqrt{6P/A_{peak}}$
$T < (4V_{peak}/A_{peak})$
Yes
No
$P_1 = 4V_{peak}^2/3A_{peak}$
$T = (P - 2P_1)/V_{peak} + 2t_{accel/decel}$
$P_1 = 4V_{peak}^2/3A_{peak}$
$t_{accel/decel} = 2V_{peak}/A_{peak}$
Single part parabolic U move
3 part parabolic U move

Figure 3.11.1 *Summary of minimum time parabolic 'U' move*

Formulae have been presented to check if parts of moves exceed system limits as well as the capabilities of the robot to make minimum time multi axis moves. Determining the system limits through experiments it was found that both V_{max} and A_{max} are as shown in table I.6. The maximum velocity of the system varies with the maximum acceleration committed, and vice-versa. The use of extreme velocities and acceleration were found to be unreliable and resulted in violent moves. The most reliable value for V_{max} was found to be 1000 mm/sec which had the corresponding A_{max} of 3232 mm/sec^2. Moves made using these limits induces high levels of vibration on the robot, but did not cause the system to hang-up.



$P < (2V_{peak}^2 / A_{peak}) + 0.04V_{peak}$
Yes
No
$\Delta P < 16V_{peak}^2 / A_{peak}$
Yes
No
$V_{\max} < V_{peak}$
$A_{\max} = A_{peak}$
$t = \sqrt{P/A_{peak}}$
$V_{\max} = V_{peak}$
$A_{\max} < A_{peak}$
$t = 3P/4V_{peak}$
$t_{tot} = 3t$
$V_{\max} = V_{peak}, A_{\max} = A_{peak}$
$t_{accel/decel} = V_{peak} / A_{peak}$
$P_{slew} = P - 2P_{accel}$
$t_{slew} = P_{slew}/V_{peak}$
Single Part S-Curve Move
3 part S-Curve Move

Figure 3.11.2 *Summary of minimum time S-curve move*

Data on some of the moves described were captured through a C program *'plot.c'*. This was done to compare the theoretical command move to the move delivered by the system. Comparing some of the results with the expected theoretical ones, a maximum of 0.4% error on velocity data was found. The inability to interrogate the motion control card at a fast enough rate contributes to the discrepancies of the results.

3.12 Conclusion

The enabling technology for the automation of the building process has been described in this chapter. The scope of this research as well as any assumptions taken were discussed. The methods developed for its use have been covered and appropriately verified, including the software functions developed to carry out the different building tasks. A detailed description of the various stages of the building process was set out, as well as how they integrate and relate with each other, to produce a solution for automating the building process.

The gantry robot, used for the experiments was described, along with its performance and the driving software. The type of gripper used as the robot end-effector was described and the inverse kinematics presented for manipulating with it. A method of quantify the robot end-effector vibration, using an accelerometer, was covered and verified. The experiments showed that the worst error based on the accelerometer signal was 0.8 mm. On average the readings were accurate to 0.18 mm, with a maximum error of 0.7 mm. Experiments carried out using this method are covered in chapter 4. The conveyor and sensors mounted on it were also described together with a method of processing the sensor information to measure the block and locate it on the conveyor. The conveyer target's design and shape were described as well as the sensing method to locate it.

Ultrasonic and displacement transducers, both mounted on the gripper were described together with a method of using them to pick-up a block safely from the conveyor. Experiments carried out to test this method are covered in chapter 6. A method for confirming the exact position of the picked block in the gripper was covered, which uses the displacement transducer and a laser profiler. Experiments carried out to test this arrangement are covered in chapter 7.

A peristaltic pump, used for the mortar dispensing operation, was described. The software used to control it was developed and integrated with the robot functionality. A special dispensing move, using the robot end-effector to manipulate the block, was developed and the software for it prepared. These were used in building a few assemblies, which are described and discussed in chapter 8.

Chapter 4: Robot End Effector Vibration

4.1 Introduction

This chapter examines the vibration of the robot end-effector when performing different moves, under various conditions. For the z yaw axis, the robot has lower torsional stiffness than originally anticipated and is thus prone to oscillation. This work aims to determine the optimum type of move for a given set of conditions, enabling the robot to be used in an efficient and reliable manner. The theory behind the different moves was described in chapter 3. The method, described and verified in section 2.8 was adopted. It involves using an accelerometer, placed on the robot end-effector, to quantify the settling time and amplitude of deflection of the end-effector, enabling the overall 'quality' of the move to be determined.

The robot vibration is a form of instability, effected by the type of move, its duration, and direction it takes. The orientation of the gripper relative to the motion, and the load it carries are also considerations. The robot vibration is a highly complex dynamic system, that is extremely difficult to mathematically model. As such, it would be necessary to model the elastic system of the structure as well as the play within components such as bearings. Such an investigation of the problem is beyond the scope of this research. Rather, a practical approach is adapted based on investigation of the actual robot system. The experiments covered in this chapter have been limited to the investigation of movement on the x-axis. This demonstrates the principles behind the method that has been developed. Utilising the observations and experimental data from this section, a RBES (described in chapter 9) is later developed. The ultimate aim is to find the optimal moves, with minimum settling time and overall vibration, allowing the use of the robot in the most effective way.

4.2 Review

Current industrial robot programming methods typically involve using low velocities and accelerations, well within capabilities. This sub-optimal setting leads to little or no vibrations but reduced productivity. Therefore, it is extremely useful in practical terms to develop a process in which robots can be optimally used with substantial time and cost savings, without undue mechanical wear.

4.3 Determining robot vibration

Experiments were conducted to determine the influence of the move type on settling time. Various types of moves, under various conditions, e.g. using a variety of acceleration and velocities of the robot , were tested. For each move, the effective move time (Effective Move Time = Move Time + Settling Time) of the robot was measured accounting for the residual vibration induced by that move. The latter was determined using an accelerometer. From the information gathered on each move, the time at which the robot is in a useable state was determined, where the useable state is defined as lateral displacements with amplitude $\leq$ 0.5 mm at the most distant point of the end-effector. The accelerometer was mounted on the end effector of the robot, on the yaw-axis to allow the recording of the vibration induced horizontally. Refer to section 2.8 for details on the accelerometer and the theory of the method used to analyse the accelerometer data.

The linear, parabolic, and S-curve move types, described in chapter 3, were tested consecutively. For each move type, long moves of 2 m and 1 m, as well as short moves of 15 mm and 4 mm were carried out, using various acceleration and velocity values. This was done by varying the total time of the moves (t_{tot}), or the acceleration/deceleration time $t_{accel/decel}$, for each particular move distance (Δp) to be performed. The data resulting from each move was recorded and analysed. All the moves tested were confined to the x-axis alone. The y-axis was set at zero, the z-axis was raised, and the yaw-axis was aligned to the direction of motion. Refer to figure 2.1 for a plan of the robot showing the different axes. The starting velocity and the ending velocities of the experimental moves made were programmed as equal to zero.

4.3.1 Linear moves

The linear moves tested were calculated using the equations described in section 3.3. The move time (t_{tot}) is first set for each move length (Δp) tested. The acceleration/deceleration ($t_{accel/decel}$) of each move is then varied. An example of this is shown in figure 4.1 and table 4.1. Figure 4.1 is the velocity time graph of linear moves having equal move times and move increments, with varying acceleration/deceleration times. Table 4.1 shows an example of the way moves were planned for a move increment Δp of 2 meters, and move time t_{tot} of 3.32 seconds. For a listing of all the linear moves tested refer to appendix C.1.

Table 4.1 *Linear moves made for t_{tot} of 3.32 seconds and ΔP of 1981 mm*

Move Time =(T+ $t_{accel/decel}$) (secs)	$t_{accel/decel}$ (secs)	T (secs)	t_{slew} =T-$t_{accel/decel}$ (secs)	V_{max} (mm/s)	A_{max} (mm/s^2)
3.32	1.66	-----	-----	1194	719
3.32	1.56	1.76	0.2	1127	722
3.32	0.98	2.34	1.36	846	866
3.32	0.39	2.93	2.54	676	1732
3.32	0.15	3.17	3.02	624	4263

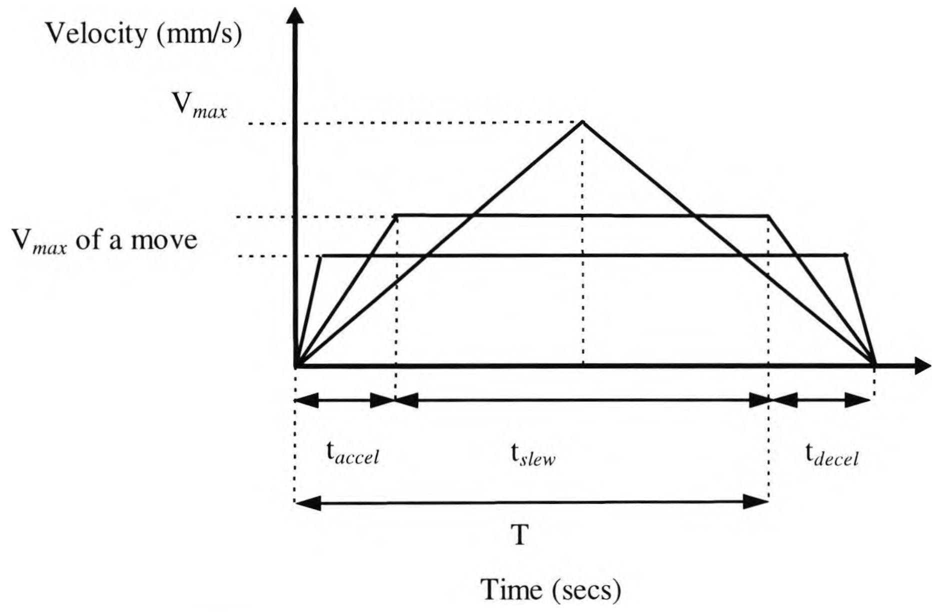


Figure 4.1 *Different linear moves with the equal time and move length*

The moves without a slew element will have a total move time of 2 $\times$ $t_{acel/decel}$. In the case of a small t_{tot}, these types of moves will reach the systems maximum velocity V_{peak}. The first move in table 4.1 is an example of that.

4.3.2 Parabolic moves

4.3.2.1 The parabolic 'U' moves

The parabolic U types moves tested were calculated using the equations described in section 3.4.1. The length of the move (Δp) is first set, and the time for that move increment (Δt) is varied. Starting with a high value, it is decreased, in stages, to a minimum time possible for that specific move increment. The maximum acceleration and maximum velocity reached in each move will subsequently increase as there is less time to cover the same distance. Figure 4.2 is the velocity time graph of a parabolic move having the same move increment, with various acceleration/deceleration times to complete the move. Table 4.2 shows an example of the way the moves were investigated. This is for a move increment Δp of 991 mm. For a listing of all the parabolic U type moves tested, refer to appendix C.2.1.

Table 4.2 *Parabolic U type moves of Δp = 991 mm*

Δt (secs)	**Move increment Δp (mm)**	**V_{max} (mm/s)**	**A_{max} (mm/s^2)**
7.03	991	211	120
3.52	991	423	481
1.95	991	761	1558
1.66	991	895	2157
1.46	991	1015	2771

To calculate the minimum time parabolic U type move, for a given move increment (Δp), the system limits of V_{peak} = 1000 mm/s is used in equation 3.11. The move will never reach A_{peak}, therefore, it will not be accounted for. For the example shown in table 4.2, the minimum time move of this type will be 1.46 seconds, which uses a V_{peak} of 1015 mm/s. This value was used for experimentation purpose only. It was found to be unreliable and was subsequently avoided (refer to section 3.7 on system limits).

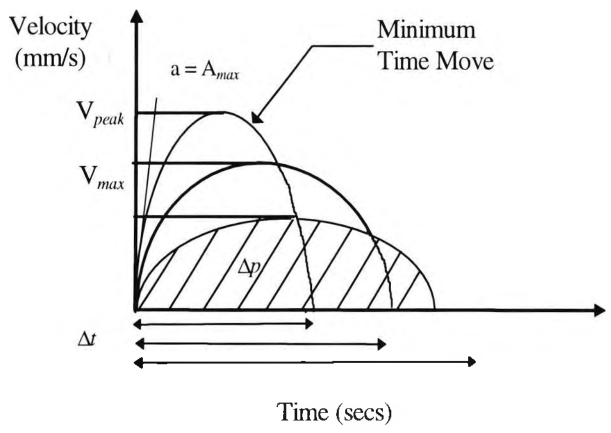


Figure 4.2 *Parabolic U moves of the same increment Δp*

4.3.2.2 Segmented parabolic moves

The type of acceleration/deceleration for this type of parabolic move (described in section 3.4.2) can vary. There is a choice of a constant acceleration, a linear acceleration of positive maximum inflection, or a linear acceleration of negative maximum inflection. Due to lack of time, only the parabolic moves accelerating/decelerating with a linear acceleration of maximum positive inflection were tested (refer to section 3.4.2.3 for more details). The moves tested all had an equal acceleration/deceleration time (i.e. $t_{accel} = t_{decel}$). For each move increment P_{tot} and move time t_{tot}, a range of moves were computed. An example of these moves is illustrated in figure 4.3. They differ in their maximum velocity (V_{max}) and maximum acceleration (A_{max}). Using equations. 3.23 and 3.27, the total distance travelled and the time of the move are calculated as:

$$P_{tot} = 2 \times P_{accel/decel} + P_{slew}$$

$$= 2(2/3 \times t_{accle/decel} \times V_{max}) + (V_{max} \times t_{slew}) \qquad \text{eqn. 4.1}$$

and,

$$t_{tot} = 2 \times t_{accel/decel} + t_{slew} \qquad \text{eqn. 4.2}$$

Substituting the value of $t_{aceel/decel}$ from equation 4.2 into equation 4.1, the value of t_{slew} is calculated to be,

$$t_{slew} = 3 \times ((P_{tot}/V_{max}) - (2/3 \times t_{tot})) \qquad \text{eqn. 4.3}$$

To make sure the move will have a time t_{slew} of at least 0.04 seconds (this is the minimum time allowed for each segment, refer to chapter 3.4.2 for more details) the following condition derived from equation 4.3, will have to be accounted for:

$$V_{\max} \leq 3P_{tot} / (0.04 + 2t_{tot}) \qquad \text{eqn. 4.4}$$

To avoid $A_{\max}$ exceeding system limit (A_{peak}), the following has to hold in equation 3.25,

$$A_{peak} \geq 2V_{\max} / t_{accel/decel} \qquad \text{eqn. 4.5}$$

Solving equation 4.2 for $t_{accel/decel}$ and substituting that in equation 4.5 give the expression

$$A_{peak} \geq 4V_{\max} / (t_{tot} - t_{slew}) \qquad \text{eqn. 4.6}$$

Substituting the value of t_{slew} from equation 4.3 in equation 4.6, gives the expression

$$(4V_{\max}^2 / A_{peak}) - 3V_{\max} t_{tot} + 3P_{tot} \geq 0 \qquad \text{eqn. 4.7}$$

Solving for the boundaries of the quadratic equations, the limits of $V_{\max}$ are found. The following steps are used to calculate different moves for each P_{tot} and t_{tot} being tested:

i. Using equation 4.4 and 4.7, the limits of V_{max} are calculated.

ii. A value of $V_{\max}$, within these limits, is chosen.

iii. Using that $V_{\max}$ in equation 4.3, the value of t_{slew} is calculated.

iv. Using equation 3.27 P_{slew} is then calculated.

v. From equation 4.2, $t_{accel/decel}$ is calculated.

vi. $P_{accel/decel}$ is then calculated using equation 4.1.

vii. To calculated a different move, steps 2-6 are repeated.

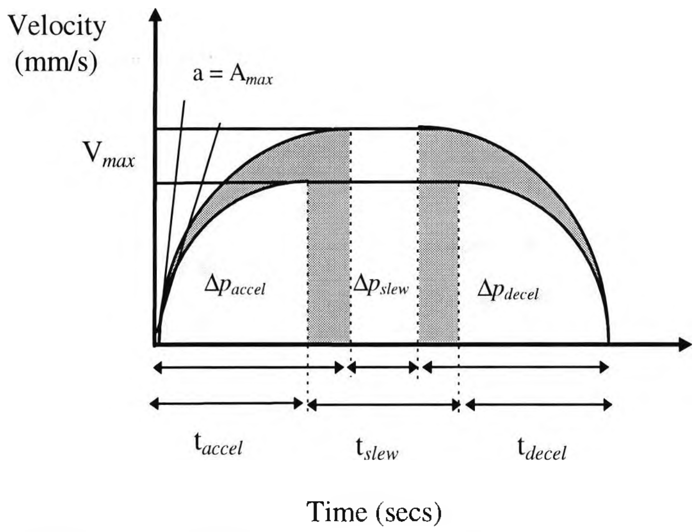


Figure 4.3 *Different parabolic moves with the same time and move length*

Table 4.3 shows an example of the various moves calculated covering a distance of 991 mm in 5 seconds. For a listing of all the segmented parabolic moves tested, refer to appendix C.2.2.

Table 4.3 *Segmented parabolic moves of Δp = 991 mm and Δt =5 seconds*

V_{max} (mm/s)	A_{max} (mm/s^2)	$t_{accel/decel}$ (secs)	$p_{accel/decel}$ (mm)	t_{slew} (secs)	p_{slew} (mm)
201.5	3229	0.12	16.8	4.75	957
205.8	1482	0.28	38.1	4.44	914
213.4	797	0.54	76.2	3.93	838
243.9	347	1.41	228.7	2.19	533
259.1	294	1.76	304.9	1.47	381
274.4	263	2.08	381.1	0.83	228.7
289.6	245	2.37	457.3	0.26	76.2
295.7	239	2.47	487.8	0.05	15.2

A minimum time move, for a distance P_{tot}, can be calculated by using the systems limits, V_{peak} and A_{peak}. The value of $t_{accel/decel}$ can be calculated first, using equation 3.26, to be 0.61 seconds. Secondly, solving equation 4.2 for $t_{accel/decel}$ and substituting that in equation 4.3 results in equation 4.8. Using this equation we find the time for t_{slew}.

$$t_{slew} = (P_{tot}/V_{peak}) - (4/3 \times t_{accel/decel}) \qquad \text{eqn. 4.8}$$

Finally the total minimum time for the move will be calculated using equation 4.2

4.3.3 S-curve moves

4.3.3.1 Single part S-curve move

The way to programme this type of move is to set a time *t* and move length Δp, and our system will add two segments at the start and end of each time increment t, to ensure a gentle start and stop to the move as shown in figure 3.12. This makes the total time of the move t_{tot} to be equal to 3 x *t* (refer to section 3.5 for a description on this type of move). For each unique move length tested, the maximum acceleration (A_{max}) of the move is varied to produce different moves. From the acceleration values, and the chosen Δp, the value of the time increment *t* is calculated using equation 3.49. Figure 4.4 shows moves covering the same distance, using different times. To avoid exceeding the systems maximum velocity, using equation 3.51, the limit of A_{max} is set to be

$$A_{\max} \leq (16/9) \times V_{peak}^2 / \Delta p \qquad \text{eqn. 4.9}$$

If the value concluded is greater than the systems limit then the latter is used as the limit instead. After setting A_{max}, the value of the maximum velocity is calculated from equation 3.50. Table 4.4 shows an example of the various moves covering a distance of 991 mm. A listing of all the single part S-curve moves tested, is included in appendix C.3.2 and C.3.4.

Table 4.4 *Single part S-curve move with Δp of 991mm*

Move increment Δp (mm)	t (secs)	t_{tot} (secs)	A_{max} (mm/s^2)	V_{max} (mm/s)
991	0.66	1.97	2287	1129
991	0.81	2.42	1524	922
991	1.18	3.53	716	632
991	1.80	5.41	305	412
991	2.35	7.04	180	317

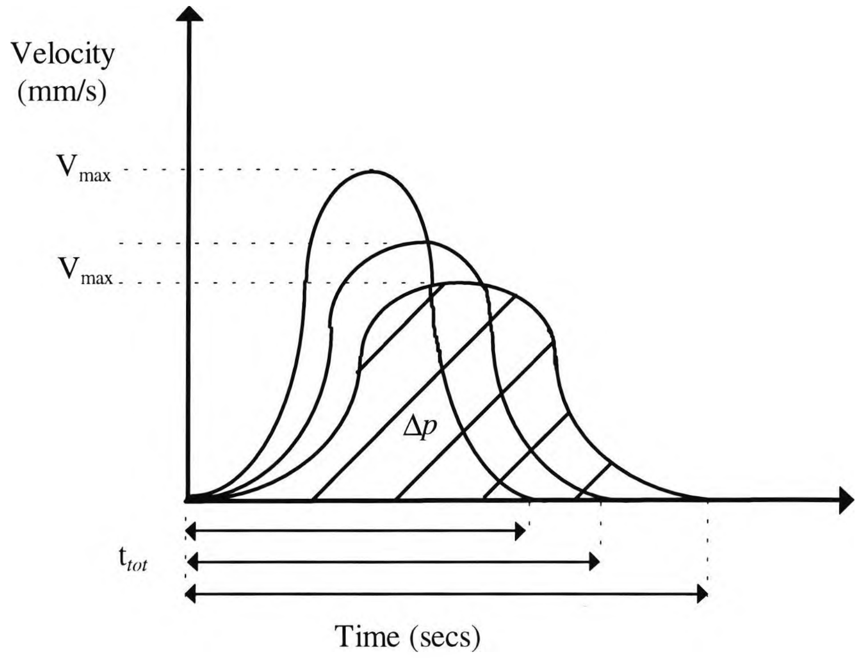


Figure 4.4 *Single part S-curve moves of the same increment*

Moves covering short distances will never reach V_{max}, so the only constraint is A_{max}. The minimum time moves can be calculated using equation 3.49 (refer to section 3.5.1 for details on minimum time moves). The maximum length possible for a minimum time move using systems limits calculated using equation 3.51 is 550 mm. A minimum time move that is greater than 550 mm, will be constrained by V_{peak}, and will never reach A_{peak}. The time for this move is calculated using equation 3.54. A minimum time move covering a distance greater than 639 mm will have to use three part S-curve move using A_{peak} and V_{peak} .

4.3.3.2 Three part S-curve move

For each move length Δp and time increment Δt, a set of moves were calculated that vary in their A_{max} and V_{max} values. A detailed description of this type of move is given in section 3.5. For all the moves tested, the value of N (the ratio of the velocity at the point of maximum acceleration, to the maximum velocity of the move) is set to 1/2 i.e. $t_1=t_2$ (refer to equation 3.37), with the acceleration time equalling the deceleration time as shown in figure 4.5. For each distance Δp and time Δt, the various moves are calculated using the following steps :

i. A value for V_{max} and A_{max} is chosen, making sure they do not exceed systems limit.

ii. Using equations. 3.31 and 3.33, with the A_{max} and V_{max} values the times of acceleration t_1 and t_2 are computed.

iii. Knowing the value of the total time of the move Δt and the acceleration/deceleration times, the time the move will travel at a constant speed t_3 is calculated.

iv. Using equation 4.10 from equation 3.39, the value of P_s (the distance travelled when accelerating) is calculated.

$$P_S = V_{max}^2 / A_{max} \qquad \text{eqn. 4.10}$$

v. Using the value of P_s and V_{max} the distance needed when travelling at a constant speed P_{slew} is calculated from equation 3.46.

Table 4.5 shows an example of a set of moves these calculated for a distance of 1981 mm in 3.32 seconds, using the method described. A listing of all the three part S-curve moves tested, is included in appendix C.3.3.

Table 4.5 *Three part S-curve moves of Δp = 1981 mm and Δt =3.32 seconds*

V_{max} (mm/s)	A_{max} (mm/s^2)	$t_{accel/decel}$ (secs)	$p_{accel/decel}$ (mm)	t_{slew} (secs)	p_{slew} (mm)
690	3073	0.22	155	2.42	1672
725	2474	0.29	212	2.15	1557
846	1732	0.49	413	1.37	1156
1128	1443	0.78	881	0.20	220

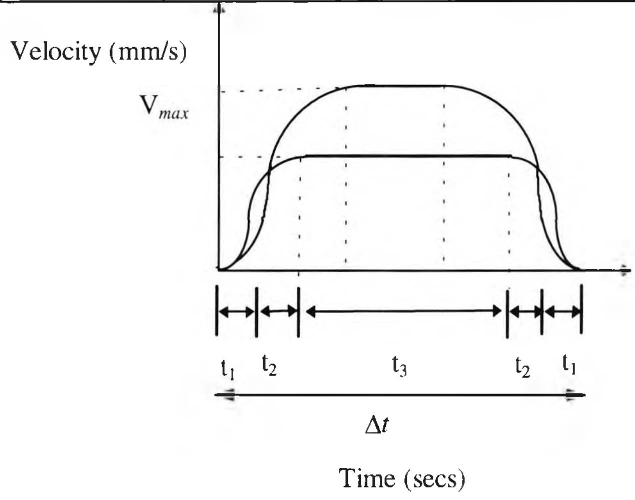


Figure 4.5 *Different 3-part S-curve moves with the same time and move length*

The minimum time move for a given Δp and Δt is calculated using the A_{peak} and V_{peak} values of our system. First the time of acceleration t is calculated, from equation 3.43. Using the resulting value in equation 3.44, the distance P_s is calculated. The slew distance P_{slew} is then calculated using equation 3.45, and the t_{slew} value from equation 3.46. As stated in section 3.5.1, to be able to make this move we need to have a distance of greater than 639 mm. For a minimum time move for less than 639 mm, the velocity will not reach the system maximum velocity V_{peak}..

4.4 Results of robot vibration experiments

For each move made a data file of the combined transitional and rotational accelerations of the robot end-effector is captured using an accelerometer described in section 2.8. The data file is analysed, using the method described in section 2.8.3, to determine the duration and amplitudes of the sideway displacements of the end-effector during and after the move. The settling time, which will determine our effective move time, and the maximum lateral acceleration amplitude, which will determine the overall quality of the move, are noted. The maximum velocity V_{max} and the maximum acceleration A_{max} reached are both calculated for each move. The results for each move length tested is analysed separately. For a full listing of the move results refer to appendix C.

4.4.1 Performance results of the moves

4.4.1.1 Long moves

The result of the various 2 m long moves are discussed and compared in this section. Tables 4.6 and 4.7 are a summary of the results with the calculated V_{max} and A_{max} of the moves, as well as the determined settling time and maximum lateral acceleration for each move. A full listing of the results is included in appendix C. From figures 4.6 and 4.7 and tables 4.6 and 4.7 it can be seen that moves reaching high velocities (>800 mm/s) and moderately high accelerations (>1000 mm/s^2) result in significant settling time as well as violent vibrations during motion. This is apparent in various moves made in under 4 seconds, including the minimum time moves. Those moves have the combined effect of high velocities and accelerations leading to violent starts and stops to the moves, as well as short move times resulting in inadequate time for the vibration from the start of the move to die down. These factors make these type of moves of inferior quality to moves made with lower accelerations and velocities values, in comparison to the values mentioned above.

For moves longer than 4 seconds, it is clear that some, depending on the type of move, need to reach high velocity values, or acceleration, or both, to be able to perform the move. However, this does not necessarily lead to bad vibration moves. To make this point clearer, the performance of the different moves made in 5 seconds are compared. The linear move type and single part S-curve move type both need to reach high velocities to perform the move in this time, resulting in violent vibrations during motion but zero settling time. The 3 part S-curve move needs to use high acceleration values, relative to the other types of moves, but nonetheless results in a good quality move, i.e. moderate vibration during motion but zero settling time. This is due to its S-curve acceleration and deceleration capabilities, which reduces the shock on the equipment. Comparing the result of the segmented parabolic type move with that of the parabolic U type move, where both need relatively low V_{max} and A_{max} values, it can be seen that both moves have zero settling time, with the segmented parabolic move having less vibration during motion, making it a more desirable move. The reasoning behind this is explained in section 4.4.2.

Table 4.6 *Results of 2 m moves*

Move type	**V_{max} (mm/s)**	**A_{max} (mm/s^2)**	**Settling time (secs)**	**Max. lateral acceleration (mm/s^2)**	**Effective move time (secs)**	**File name**
Results of 10 second	*moves*					
Linear	383	79	0	542	10	dy13
Parabolic U move	297	119	0	547	10	p3
Segmented parabolic	255	153	0	472	10	p135
Single part S-curve	442	175	0	551	10	p322
3 part S-curve	325	167	0	429	10	p276
Results of 5 second	*moves*					
Linear	725	320	0	915	5	dy8
Parabolic U move	595	476	0	691	5	p2
Segmented parabolic	591	478	0	618	5	p190
Single part S-curve	894	716	0	829	5	p321
3 part S-curve	493	1009	0	732	5	p280
Results of 3.32 second	*moves*					
Parabolic U move	895	1079	0.68	1321	5	p1
Segmented parabolic	838	1169	0.16	1800	3.48	p195
3 part S-curve	846	1732	0	857	3.32	p285

Table 4.7 *Results of minimum time moves of 2 m*

Move type	**Move time (secs)**	**V_{max} (mm/s)**	**A_{max} mm/s^2**	**Settling time (secs)**	**Max. lateral acceleration (mm/s^2)**	**Effective move time (secs)**	**File name**
Linear	3.32	1015	742	0.76	1461	4.08	dy2
Parabolic U move	2.93	1015	1385	0.80	1695	3.73	p106
Segmented parabolic	2.73	932	2045	1.16	2324	3.89	p138
Segmented parabolic	2.92	873	1793	0.82	2215	3.74	p139
Single part S-curve	3.98	1121	1128	0.4	7153	4.38	p319
3 part S-curve	2.63	1067	2744	1.82	5590	4.45	p265

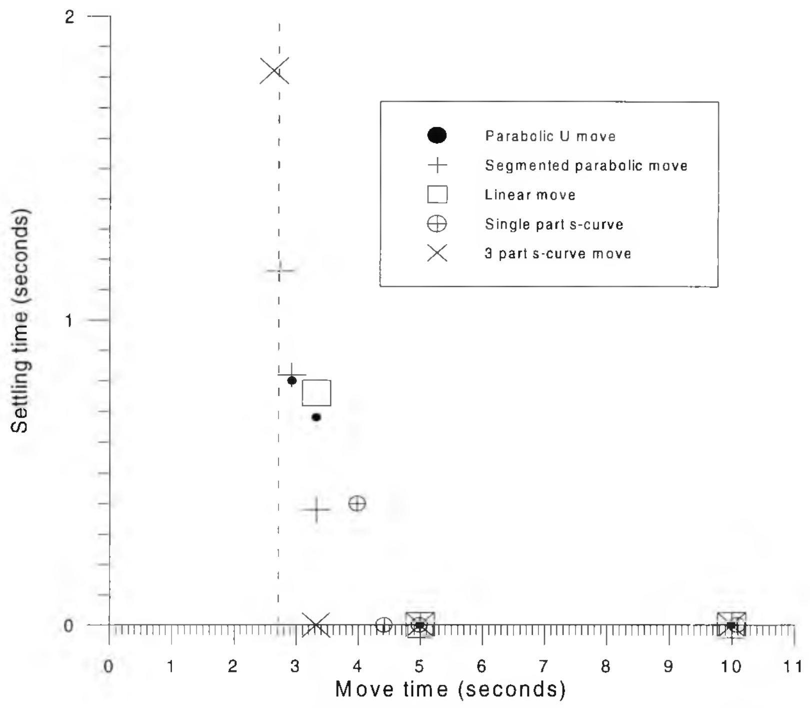


Figure 4.6 *Settling time of 2 m moves with respect to move time*

In conclusion, figures 4.6 and 4.7 show that 2 m long moves, with sufficient time (> 7 secs) to perform the moves using low velocities and accelerations, will result in negligible settling time and scarcely any vibration during motion. The recommended move in that time range would be the 3 part S-curve type move. It is a preferred move to make, as it produces relatively lower stresses and strains within the robot. For moves made using between 7-5 seconds move time the 3 part S-curve move type will require higher acceleration and deceleration values in comparison to other types of moves. These moves will still perform well under these conditions, but the segmented parabolic move will be a better move, having less vibration while performing the move, as well as needing lower acceleration/deceleration values and benefiting from the smooth positive inflection at the start and stop. For 2 m long moves made in less than 5 seconds, the 3 part S-curve move is clearly the best to use, down to a time of 3.32 seconds. Further reductions in the move time yields very bad quality moves with increasing settling time, which makes the move effective times sub-optimal.

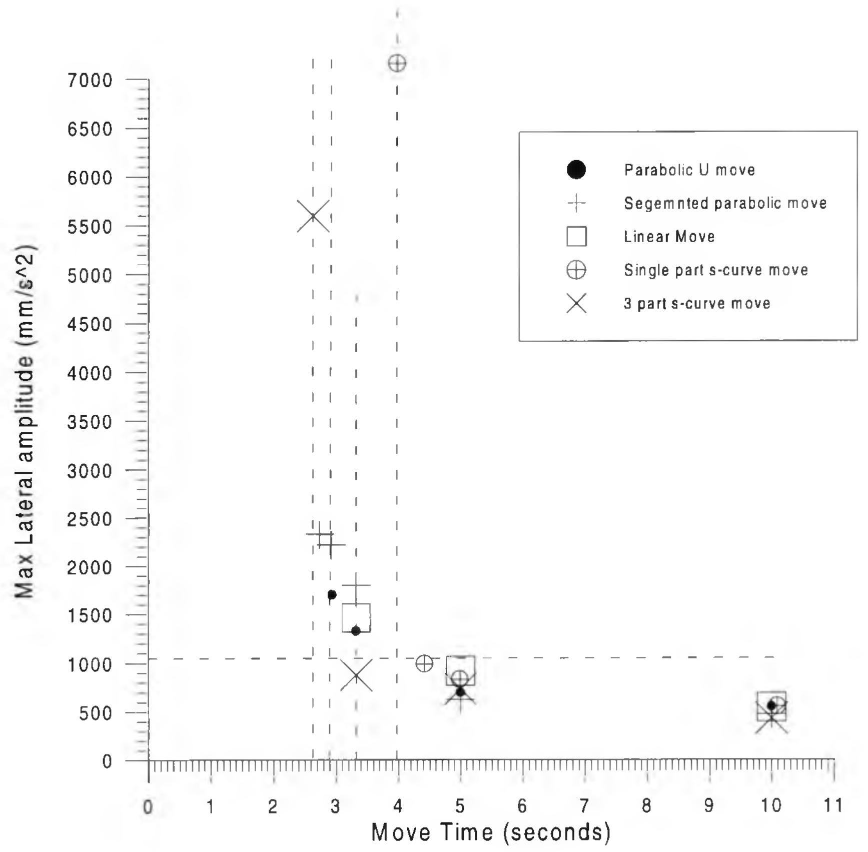


Figure 4.7 *The quality of 2 m moves with respect to move time*

4.4.1.2 Medium length moves

The result of the various 1 m long moves are reviewed in this section. Tables 4.8 and 4.9 are a summary of the results with the calculated V_{max} and A_{max}, as well as the determined settling time and maximum lateral acceleration for each move. A full listing of the results is included in appendix C. From these results and figures 4.8 and 4.9, it can be seen that all the moves are satisfactory when the time allowed to make the move is sufficient (>3.32 seconds). This means that moderately low acceleration values (< 600 mm/s^2) and velocities (< 300 mm/s) are required. These values result in good quality moves, with relatively small vibration during the move and zero settling time. As the move times decrease some move types will need to reach higher values of acceleration, or velocities, or both to complete the move in those times. From the results in tables 4.8-

9 it can be seen that the single part and the 3 part S-curve move types will always need to reach higher V_{max} and A_{max} values relative to other types of moves. This will not necessarily result in moves with more vibration, due to the abilities of that type of move to build up motor power at the start of the move and reduce it to zero at the end of the move, resulting in low vibrations. The exceptions is when these types of moves reach extremely high velocities (> 700 mm/s) and accelerations (> 800 mm/s^2), as in the case of the minimum time move (refer to table 4.9), then they will have violent vibrations at the start and end which will result in large settling times, sometime as long as the move duration.

In conclusion, it is clear that the moves with enough time (>3.5 seconds) all perform well. The S-curve type moves, with their abilities to smooth acceleration/deceleration, resulting in less strain on the equipment, are preferable in this case. The segmented S-curve move has the advantage of part of the move travelling at a constant speed, which helps suppress some of the vibrations. Moves performed in short times (around 3.5 seconds) all perform well with no settling time and acceptable vibrations during their moves. The exceptions are the single part S-curve move and the linear move. The former will require relatively high velocities and accelerations to perform the move in those time sets, resulting in higher vibration during the move relative to the other types of moves. The later does not perform well when given a short time to perform the move. Minimum time moves (<2 seconds), shown in table 4.9 are of extremely bad quality with significant settling times and violent start and stops to the moves as a result of reaching high velocities with high accelerations. The quality of the single and 3 part S-curve move types will definitely be the worst as they will reach even higher accelerations and velocities compared to the rest. The parabolic and 3 part parabolic move types performance will be similar, requiring lower velocities. In this, the 3 part parabolic move types performance preferred, but still producing bad vibrations and a settling time of around 2 seconds. All the minimum time moves are, in general, ineffective on account of the settling times being nearly equal to the time of the move. They are also violent moves which are bad for the robot system.

Table 4.8 *Results of 1 m moves*

Move type	**V_{max} (mm/s)**	**A_{max} (mm/s^2)**	**Settling time (secs)**	**Max. lateral acceleration (mm/s^2)**	**Effective move time (secs)**	**File name**
Results of 10 second	*moves*					
Segmented parabolic	143	73	0	323	10	p203
3 part S-curve	140	96	0	156	10	p292
Results of 7 second	*moves*					
Linear	178	122	0	426	7	dy31
Parabolic U move	211	120	0	481	7	p6
Segmented parabolic	181	155	0	430	7	p141
Single part S-curve	180	317	0	454	7	p326
Results of 5 second	*moves*					
Segmented parabolic	290	245	0	412	5	p211
3 part S-curve	246	504	0	441	5	p296
Results of 3.32 second	*moves*					
Linear	441	347	0	758	3.32	dy25
Parabolic U move	423	481	0	634	3.32	p5
Segmented parabolic	396	528	0	447	3.32	p217
Single part S-curve	632	716	0	766	3.32	p325
3 part S-curve	559	647	0	539	3.32	p301

Table 4.9 *Results of minimum time moves of 1 m*

Move type	**Move time (secs)**	**V_{max} (mm/s)**	**A_{max} (mm/s^2)**	**Settling time (secs)**	**Max. lateral acceleration (mm/s^2)**	**Effective move time (secs)**	**File name**
Linear	1.66	922	1574	1.56	3994	3.22	dy19
Parabolic U move	1.66	895	2157	2.9	2838	4.56	p4
Parabolic U move	1.46	1015	2771	1.8	7922	3.26	p107
Segmented parabolic	1.66	884	2186	2.12	2526	3.78	p225
Segmented parabolic	1.46	873	3586	2.94	5598	4.40	p144
Single part S-curve	1.97	1129	2287	2.1	8599	4.07	p323
3 part S-curve	1.66	1015	2969	2.18	9017	3.84	p303

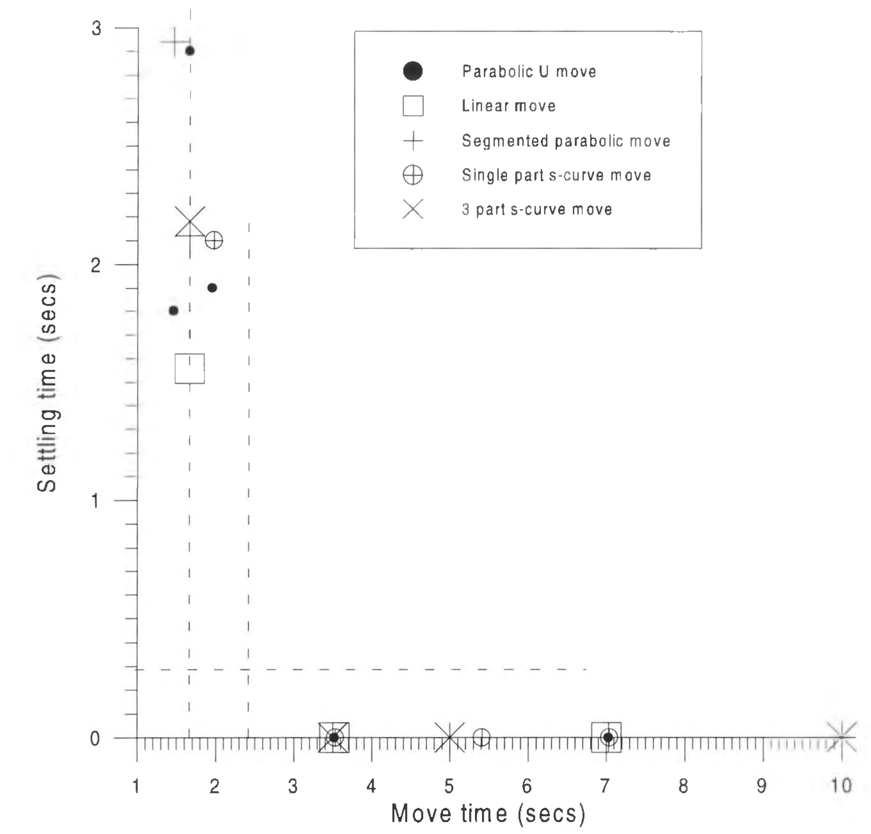


Figure 4.8 *Settling time of 1 m moves with respect to move time*

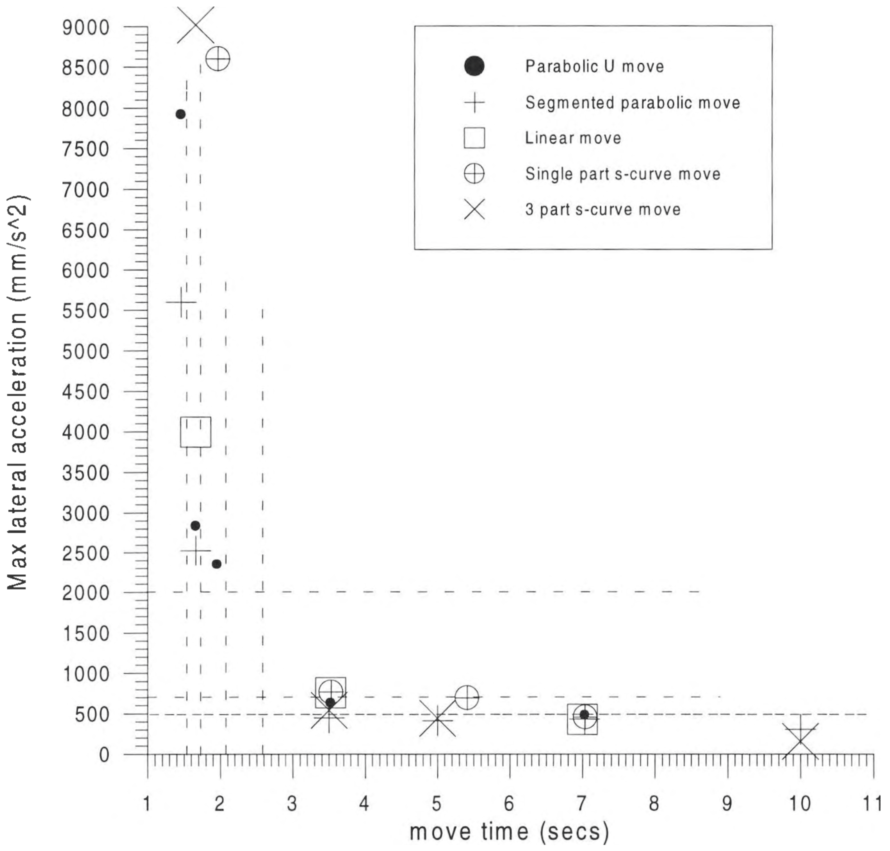


Figure 4.9 *The quality of 1 m moves with respect to move time*

4.4.1.3 Very short moves

Short moves of 15 mm and 4 mm lengths were tested. Unfortunately, not enough results were established for the 15 mm move to enable detailed conclusion. As for the 4 mm moves, the results are discussed and compared in this section. Table 4.10 summarises the results, with the calculated V_{max} and A_{max} values, as well as the determined settling time and maximum lateral acceleration for each move. A full listing of the results is included in appendix C. From these results, a plot of the settling time and the maximum lateral amplitude of the various moves are shown in figures 4.10 and 4.11. Given enough time($\geq$0.35 seconds) all the moves are relatively good, with no settling time and minimum vibration during motion. As the time of move is decreased, the quality of the parabolic U move declines drastically, having an effective move time of 0.89 seconds for a 0.17 second move, for example. The 3 part S-curve, on the other hand, was found to be a complex move for this short distance and did not perform particularly well. The fact that it is made of many segments, with minimum time for each to be taken into consideration, did not allow for moves of less than 0.2 seconds. For moves of times around 0.2 seconds, the linear and segmented parabolic moves had the best performance, with a satisfactory effective move time but with a certain amount of vibrations during the move. The single part S-curve move performed reasonably well with minimum move settling times and some vibration during motion compared to the rest of the moves. In the case of the minimum time move, the systems maximum velocity is reached, thus resulting in a move with considerable amounts of vibration during motion. The settling time of the minimum time moves was found to be twice as long as the move time, thus resulting in the effective move time to be comparatively high. The linear move's performance, as well as that of the segmented parabolic move, were found to be the best overall, with the linear move able to make shorter time moves relative to the rest of the moves. Zero settling time, found in all these moves, with reasonable vibrations during motion in relation to the rest of the move. This can be seen clearly in figure 4.11.

Table 4.10 *Results of 4 mm moves*

Move type	V_{max} (mm/s)	A_{max} (mm/s^2)	Settling time (secs)	Max. lateral acceleration (mm/s^2)	Effective move time (secs)	File name
Results of 0.78 second	*moves*					
Linear	10	26	0	129	0.78	dy53
Parabolic U move	8	39	0	147	0.78	p109
Segmented parabolic	7	50	0	173	0.78	p145
Single part S-curve	11	59	0	154	0.78	p341
Results of 0.39 second	*moves*					
Parabolic U move	15	156	0	201	0.39	p110
Segmented parabolic	15	160	0	174	0.39	p259
Single part S-curve	23	229	0.02	257	0.41	p340
3 part S-curve	21	230	0	259	0.39	p315
Results of 0.29 second	*moves*					
Parabolic U move	20	277	0	310	0.29	p11
Segmented parabolic	17	356	0	224	0.29	p147
Results of 0.195 second	*moves*					
Linear	37	420	0	412	0.195	dy49
Segmented parabolic	26	802	0	391	0.195	p148
Single part S-curve	46	930	0.04	404	0.235	p338
Results of 0.17seconds	*moves*					
Parabolic U move	36	862	0.62	1186	0.89	p10
Results of 0.15 seconds	*moves*					
Single part S-curve	58	1524	0	512	0.15	p337
Results of 0.12seconds	*moves*					
Single part S-curve	71	2287	0	614	0.12	p336
Results of 0.11seconds	*moves*					
Linear	68	1154	0	597	0.11	dy45
Single part S-curve	1143	2744	0.18	675	0.29	p269

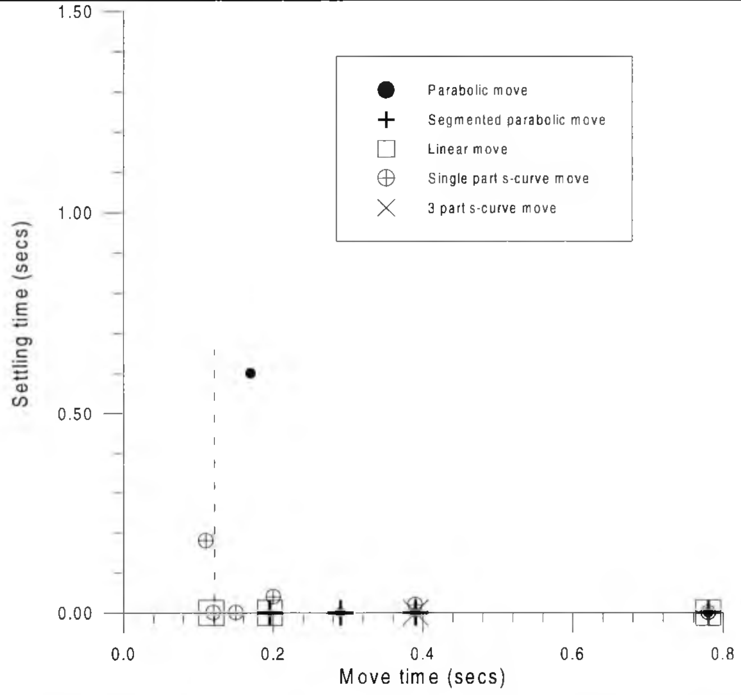


Figure 4.10 *Settling time of 4 mm moves with respect to move time*

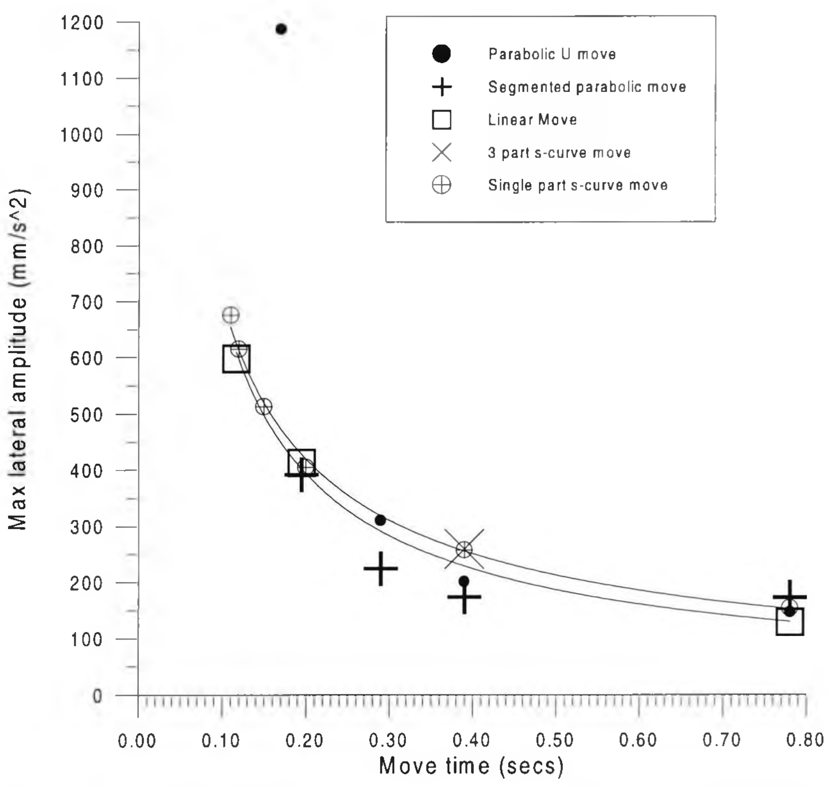


Figure 4.11 *The quality of 4 mm moves with respect to move time*

4.4.2 Natural frequency vibration

Some moves were found to have shorter settling times in comparison to others, even though they were expected to be worse moves, inducing more vibration. An example of this was investigated further by comparing the resulting settling time of two parabolic U moves described in table 4.11. These moves both travelled a total distance ΔP of 991 mm in different times. The first move, having a total time of 1.46 seconds, is expected to be worse than the second move. With a total time of 1.66 seconds it has to reach higher V_{max} and A_{max} values. This is clearly evident when comparing the maximum lateral acceleration reached by both moves. The settling time of the 1.46 second move was also shorter than that of the 1.66 seconds move (figure 4.12 and figure 4.13). This can be explained by the fact that the complex vibration pattern induced by the 1.46 seconds move, resulted in some frequencies cancelling each other out (figure 4.12). Whereas the 1.66 seconds move induced the vibrations of the natural frequency of the robot end-effector, calculated to be around 2.5 Hz from the results shown in figure 4.13.

Table 4.11 *Parabolic U moves investigated*

Δt (secs)	V_{max} (mm/s)	A_{max} (mm/s^2)	Maximum lateral acceleration (mm/s^2)	Settling time (secs)	File name
1.46	1015	2771	7922	1.8	p107
1.66	895	2157	2838	2.9	p4

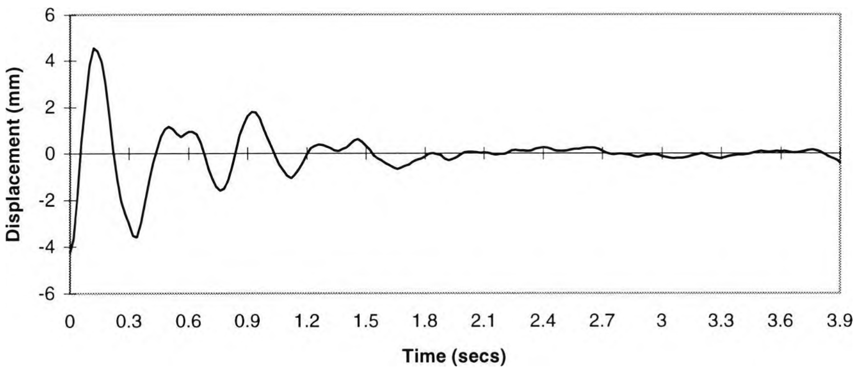


Figure 4.12 *Complex frequency vibrations of end-effector settling*

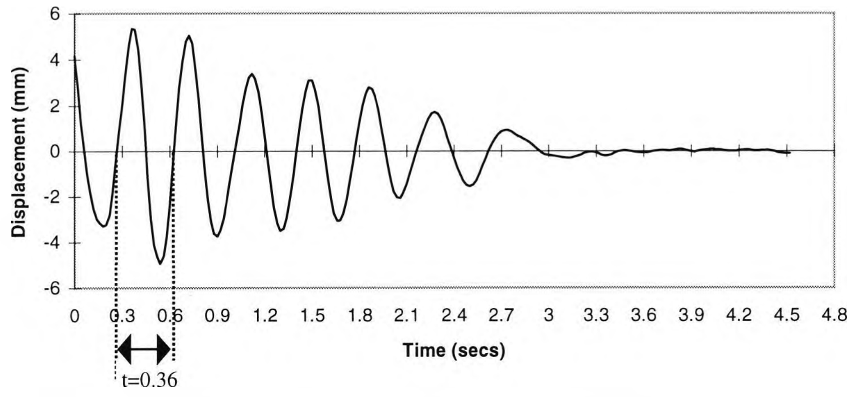


Figure 4.13 *Natural frequency vibration of end-effector settling*

4.4.3 Comparing the performance of parabolic moves

The segmented parabolic move, relatively speaking, uses higher velocities and higher accelerations than the parabolic U type move. In spite of that, the results in section 4.4.1 showed the parabolic U type move's performance to be, in general, inferior to that of the segmented parabolic move, when comparing similar moves. The examples shown in figure 4.14 and 4.15 is the lateral acceleration of moves made using the same time (5 seconds) and distance (2 m) increment. The first is a parabolic U type move and the second a segmented parabolic move. The move profile of each is sketched below the acceleration signal, to show the accelerating and decelerating parts of the signal. Table

4.12 shows the results of the moves. From that, it can be seen that the segmented parabolic move performs relatively better with lower lateral acceleration values. From figures 4.14 and 4.15 it can be seen that both moves start with similar vibrations. In the case of the parabolic U type move, the vibration after reaching maximum velocity will start to increase, and its effect will accumulate while decelerating to the end of the move. As for the segmented parabolic move, the constant speed segment of the move allows some of the vibration to decay during that segment, which results in less vibration at the end of the move. Segments of constant speed, as short as 0.4 seconds, will still be of benefit, considering the natural frequency vibrations of the end-effector is 2.5 Hz, (as described in section 4.4.2). It can be assumed that any move type composed of segments, e.g. the 3-part S-curve move, can benefit from the segment of constant velocity. This will help to reduce vibration acquired at the start of the move.

Table 4.12 *Moves of 1981 mm made in 5 seconds*

Parabolic U move								
V_{max} (mm/s)	A_{max} (mm/s^2)	Δt_{accel} (secs)	Δp_{accel} (mm)	Δt_{decel} (secs)	Δp_{decel} (mm)	Max. lateral acceleration (mm/s^2)	Settling time (secs)	File name
595	476	2.5	990.5	2.5	990.5	691	0.00	p2
Segmented parabolic move								
V_{max} (mm/s)	A_{max} (mm/s^2)	$\Delta t_{accel/decel}$ (secs)	$\Delta p_{accel/decel}$ (mm)	Δt_{slew} (secs)	Δp_{slew} (mm)	Max. lateral acceleration (mm/s^2)	Settling time (secs)	File name
564	506	2.23	838	0.54	305	624	0	p189

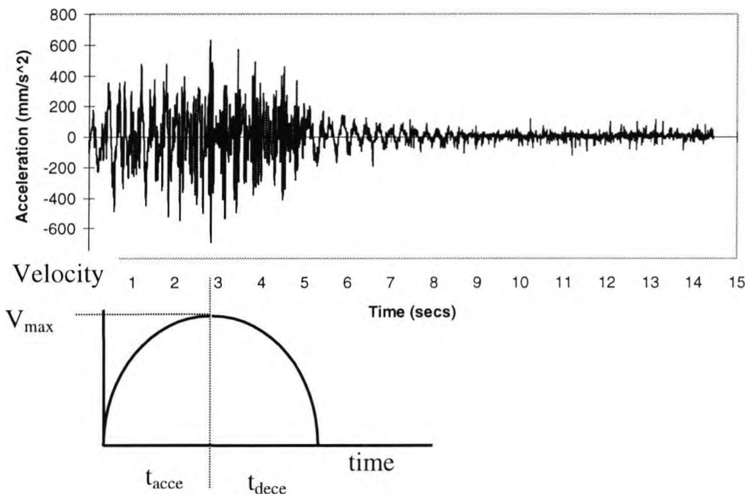
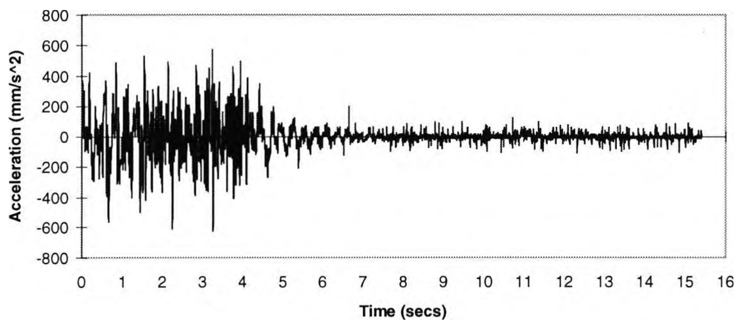


Figure 4.14 *End-effector lateral acceleration result of parabolic U move*



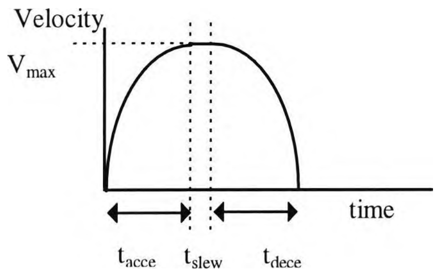


Figure 4.15 *End-effector lateral acceleration result of segmented parabolic move*

4.5 Conclusion

While examining the robot's vibration in a single plane for moves in the x-axis, we have seen how a variety of factors influence the vibration and the settling time of the robot end-effector. The principles and methods used in this study to establish optimal settings for the robot in this single axis can equally well be applied to the other axes, and to different orientations of the end-effector. Further experimentation would enable the robot to be programmed to carry out moves at a higher level of performance. Nonetheless, the study gives a good indication of the overall behaviour of the different types of moves.

The moves tested were designed and programmed using the theory described in chapter 3, where a description of the method of programming each move, ways to avoid impossible moves and programming minimum time moves, are given.

From the results, it can be seen that, in general, moves made within a shorter period will induce stronger vibration and increased settling times at the end of the move. This is explained by the need for such moves to be carried out utilising higher velocities and accelerations. Moves with relatively short periods of time do not have the opportunity for the initial surge of the start of the move to decay during the move. These factors make effective move time unacceptably high due to high settling times and also result in excessive wear and tear on the robot motors. Therefore, these short, fast moves are ineffective for use. The worst performance for these types of moves came from the S-curve type moves. This was due to the complex start and stop they possess, which require higher acceleration and velocity values.

Given adequate time to make a move ($\geq$7 seconds for the long move and $\geq$5 seconds for the medium size move), the S-curve type moves were found to be the most desirable type of move, with the 3 part S-curve move performing better than the single part S-curve. This is due to the fact that the V_{max} and A_{max} of the former move can be adjusted to improvement the quality of the move, thus allowing control of the acceleration/deceleration part of the move. The 3 part S-curve move also has the benefit of a constant velocity segment, which was found to reduce the vibrations acquired at the start of the move. These moves are preferable, in the cases mentioned, due to their S-curve acceleration and deceleration capabilities.

Overall, the parabolic moves performed well, with the advantage of not requiring high values of acceleration and velocities compared to other types of moves. This makes these moves preferable when short time moves were required for long and medium length moves (i.e. < 7 to 3.32 second for the long move and <5 to 2 seconds for the medium length move).

The segmented parabolic move allows vibration induced at the start of the move to decay during the constant speed segment of the move. This results in less vibration at the end of the move, making it perform better than the parabolic U type move. This type of move is important because it causes minimum energy dissipation in the motor (for covering a given distance in a given time). The parabolic profile, by providing the maximum acceleration at the lowest velocities, enables the bulk of the distance to be covered at low values of acceleration and, therefore, low power dissipation.

Depending on the time available to perform the short moves tested (4 mm), the robot performance varied. Given enough time, $\geq$ 0.35 seconds, all the move types tested were found to perform relatively good, with zero settling time and minimum vibration during motion. As the time of the move is decreased the performance of the parabolic U move declines dramatically. The 3 part S-curve, on the other hand, was found to be a complex move for this short distance and did not perform well. As it is made of many segments, with a finite time required for each part, it did not allow for moves of less than 0.2 seconds. The linear and segmented parabolic moves had the best performance for these short distances, with a satisfactory effective move time. However, there was a certain amount of vibration during the move. The single part S-curve move performed reasonably well, with some significant move settling times and vibration during motion. The linear move's performance, as well as that of the segmented parabolic move, was found to be the best overall, with the linear move able to make shorter time moves in relation to the rest of the moves. Zero settling time was found with all the short length (4 mm) linear moves tested, with reasonable vibrations during motion.

In summary, a much better understanding of the preferred type of move for a given set of parameters and requirements has been achieved. The results from this section are used to develop a RBES (described in chapter 9.4.3) which recommends the best type of move for given move specification. Again, this will be limited to moves on the x-axis while the rest of the axes are at zero positions. The same rule base will also check for the ability of the robot to perform the different types of moves requested. This will help the user to operate the robot more efficiently.

Chapter 5: Conveyor Location

5.1 Introduction

For construction sites applications, robots will need to deal with imprecision in equipment location and material dimensions. Measures, therefore, need to be taken for the robot to recognise changes and accommodate these in equipment and material. Failure to do so could have devastating results for the robot and it's workspace.

This chapter covers the process of developing and testing the method used to locate the conveyor in the robot work area. Such a requirement is in keeping with the advanced robot concept, a robot that is able to apply artificial intelligence to the identification and position determination of objects in its workspace. The need to locate the conveyor's exact position will ultimately enable determination of the position of blocks on the conveyor, when they are ready for pick-up. This position is called the BPP. It is shown in figure 2.3.12 and described in chapter 2.3.1.3. The process of locating the conveyor and from that calculating the BPP, make up stage 1 of the building process illustrated in figure 2.2. The location of the conveyor will be accomplished using a target mounted on the far end of the conveyor and aligned to it (figure 2.3.12). The target's design, shape, and dimensions are described in chapter 2.3.2. The two ultrasonic sensors mounted on the robot end-effector, seen in figure 2.3.11, will be used to find and position the target in the robot's workspace.

The complete process of locating the conveyor, calculating the BPP, will be carried out in four defined stages. The first stage, will carry out an area scan to identify the target from its shape and dimensions, and determine its approximate angle and centroid. The second stage will determine the target angle more accurately by searching closer to the target, using the first stage's approximate angle and centroid values. The third stage uses the corrected angle of the target, found as a result of the second stage, and the first stage's approximated centroid value, to search closer for the target centroid. The fourth and final stage will use the results of the second and third stages to calculate the BPP on the conveyor.

Using the ultrasonic sensors on the robot end-effector, a scan of the plan area is first made as the first stage (using the vertically pointing sensor). This determines the approximate position and angle of the target. For the purpose of demonstrating the effectiveness of the method developed, the search area covered is limited to the far right hand corner of the robot work space, with the reasonable assumption that the target is in that area. The second stage is to search for the target's exact position, based on the approximate target position. This is accomplished using a search algorithm devised through experimentation, which uses the horizontal ultrasonic sensor mounted on the robot end-effector. From this, the conveyor location and BPP can be calculated. A description and verification of the methods developed to locate the conveyor will be covered along with experiments for each stage of the process. Observations are made on accuracy and reliability. Details of the exact location of the BPP on the conveyor, and its use when picking up blocks, are given in chapter 2.3.1.3.

5.2 Review

The approach to automation of masonry construction depends largely on the dimensional tolerances that are accepted for the actual masonry units. Where precise materials are employed (e.g. 400 mm ±1 mm), there is the possibility of pre-programming the assembly task, starting with the delivery of a precisely prepared pallet of masonry units. Early in the research, this was the preferred approach (Chamberlain *et al.*, 1990). However, it became apparent that there was no ready supply of precise masonry units in the UK and, furthermore, that masonry units where almost invariably delivered in a shrink wrapped pack, without a base pallet. Precise masonry material could be specially produced or imported, however, the cost would be significantly high.

Considering these factors, and the inevitable need for object size and position confirmation, even with the precise approach, it was decided that imprecise material delivery in imprecise stacks would be catered for. This is clearly a more generic and flexible approach, and thus of greater value in the wider sense. It is noted that, elsewhere (Bock *et al.*, 1993) and (Pritschow *et al.*, 1994), the approach is based on the use of precise masonry material delivered in a precise formation on a loading pallet. This allows the automation task to be largely pre-programmed. In the author's case, the task is pre-programmed as a theoretical task that is subject to run time accommodations for dimensional and positional variations. This is achieved through the application of intelligence to sensor signals. Our method uses a conveyor to supply the building materials (blocks) and a gripper mounted on the robot end-effector to pick them. These blocks need only be fed onto the conveyor, in a typical hand loading operation. Therefore, there is a need to determine the precise positioning of the block on the conveyor. This is done using ultrasonic sensors and a target mounted on the conveyor. This method is subsequently incorporated into the expert system programme for picking up a block, as described in chapter 9. This takes into consideration different cases, for example, when the robot's intelligence may suspect that there is a conveyor present but concludes that pick-up is not safe due to uncertainty in the block location.

5.3 Conveyor Location using Target

The target is mounted on the far end of the conveyor and is aligned to it. Principal dimensions are given in figure 2.3.10. For the purpose of our experiments, it is assumed that the target could be found anywhere in the top right hand corner of the robot workspace. This area is shown in the plan view of the robot workspace (figure 5.1). The height of the target was decided for two reasons. The first is to remain within the working range of the vertically pointing sensor when the robot's z-axis is raised, and the second to be the highest object in the workspace, hence reducing the possibility of it being confused with other objects. When scanning for the target, an assumption is made that no unplanned obstacles will be in the way of the robot, consequently no measures have been taken to account for any. This is thought to be a reasonable point of view. In reality, some form of policing of the workspace for unknown objects and human intruders would be necessary.

The search for the target location is done in three stages. In the first stage, the robot identifies the target from its estimated shape and dimensions and, when confirmed, approximately locates the target in its own co-ordinate system. In the second and third stages, the exact position of the target is determined using a search algorithm developed for that purpose.

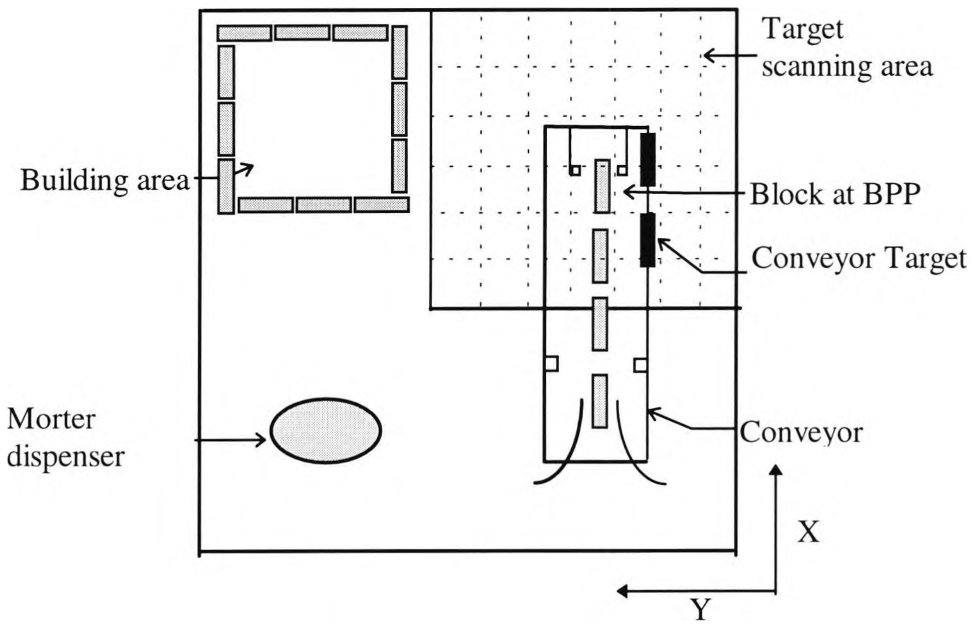


Figure 5.1 *Top view of target scanning area within the robot workspace*

5.3.1 Scanning target shape and centre (stage 1)

A scan is made of the area shown in figure 5.1 using the vertical sensor. The zero point of this sensor is made the TCP of the robot during the scan procedure, which is called '*Sensor_scan_TCP*' and described in chapter 2.2. During the scan, the robot's z-axis is raised, and the yaw axis is aligned to the direction of motion. The scan starts with moving the robot in the x-axis direction and then in the y-axis direction along the grid sketched in figure 5.1. A grid width of 130 mm was chosen to ensure that, irrespective of the target angle, the sensor crosses the target at least twice. The normal distance from the sensor to the top of the target was measured as shown in figure 2.3.11. During the scan, at positions where a valid sensor reading occurs i.e. not open circuit reading and equals the distance mentioned earlier, the (x,y) co-ordinates are recorded. Once the scan is finished, all the data points collected are analysed to estimate the shape of the object scanned, its angle relative to the robot co-ordinates, and its centre. This was achieved by adopting a representation method for the region to determine its boundary description. This employs the method of object moments (Gonzalez, 1987). This technique made it possible to quantify the shape of the object for recognition. Generally, only the first few moments are required to differentiate between signatures of clearly distinct shapes. The second moments, described in appendix D.1, quantify the spread of the data points about the mean value (i.e. the centroid). The advantage of moments over other techniques is that it is straightforward to implement, and extremely effective. This was confirmed through experimentation.

5.3.1.1 Implementation of method

A C program, '*target.c*', was implemented using the theory of moments to analyse the sensor data on the object. The theory of moments is given in appendix D.1. The method was developed using the procedure described in this section. The radius of gyration was used to give an idea about the shape of the object, and the centroid of the object was calculated to give an estimate of position. The orientation of the shape was determined from the principle axis direction.

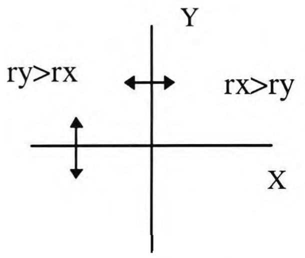


Figure 5.2a *The radius of gyration indicates shape*

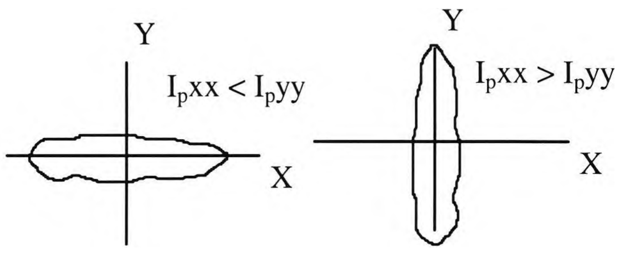


Figure 5.2b *The second moments indicates direction*

The application procedure follows:

(i)- Enter (x,y) co-ordinates of each data point.

(ii)- Find the centroid $(\bar{x}, \bar{y})$ of the set of data points.

(iii)- Find the second moments Ixx, Iyy, and Ixy from equations d.3, d.4 and d.5 in appendix D.1 with respect to the robot system axis.

(iv)- Find the principle axis direction, θ using equation d.13 in appendix D.1.

(v)- Using θ, calculate the values of the principle I_pxx, I_pyy, and I_pxy values i.e. where Ixy and Iyx are zero, as shown in appendix D.1.

(vi)-Find the radius of gyration, using equations d.16 and d.17in appendix D.1, to confirm the shape of the object.

5.3.1.2 Verification of method

Two types of experiments were carried out to verify the method of moments. The first was to simulate the target detection using AutoCAD. The second was to use the position data of the object, collected from the ultrasonic sensors on the robot. The two experiments, and their results, are presented in the following section.

5.3.1.2.1 Target detection simulation

The simulation experiment was done by generating the scanning area as a grid that the robot follows in the scanning operation and representing the target in various locations and orientations within it. AutoCAD was used for this, a plot from which is given in figure 5.3. At the points where the target intersects the grid, the co-ordinates were recorded. These points were then analysed using the program '*target.c*' to calculate the centroid $(\bar{x}, \bar{y})$ of the object, the principle direction angle θ and the radius of gyration (r_x, r_y). Different target positions were tested, the worst case found to be where the grid crosses the target three times only. This worst case can be seen in figure 5.3, where the target is at -30° to the x-axis and intersects the grid at intersection point identifier numbers 46, 47, and 48. The best case is where the target cuts the grid six times, as shown in figure 5.3, for example when the direction is 0° to the x-axis and the intersection point identifier numbers are 40, 41, 42, 43, 44, and 45. For each case, 20 co-ordinate positions, chosen from the various intersections of the grid of the case, are analysed using program '*target.c*', and the results covered in tables 5.1 and 5.2. From these results, it can be seen that the worst error in the position of the centroid was 81.6 mm. The worst error found in the target angle was less than 2°. These indicate that the method is satisfactory for detection and estimation of the target. Table 5.2 shows the radius of gyration results for each set of data points, which indicates that the object analysed is always clustered on one axis. This confirms that the object is slender, information which is beneficial in distinguishing the object from other objects that may have been detected in the scanning operation.

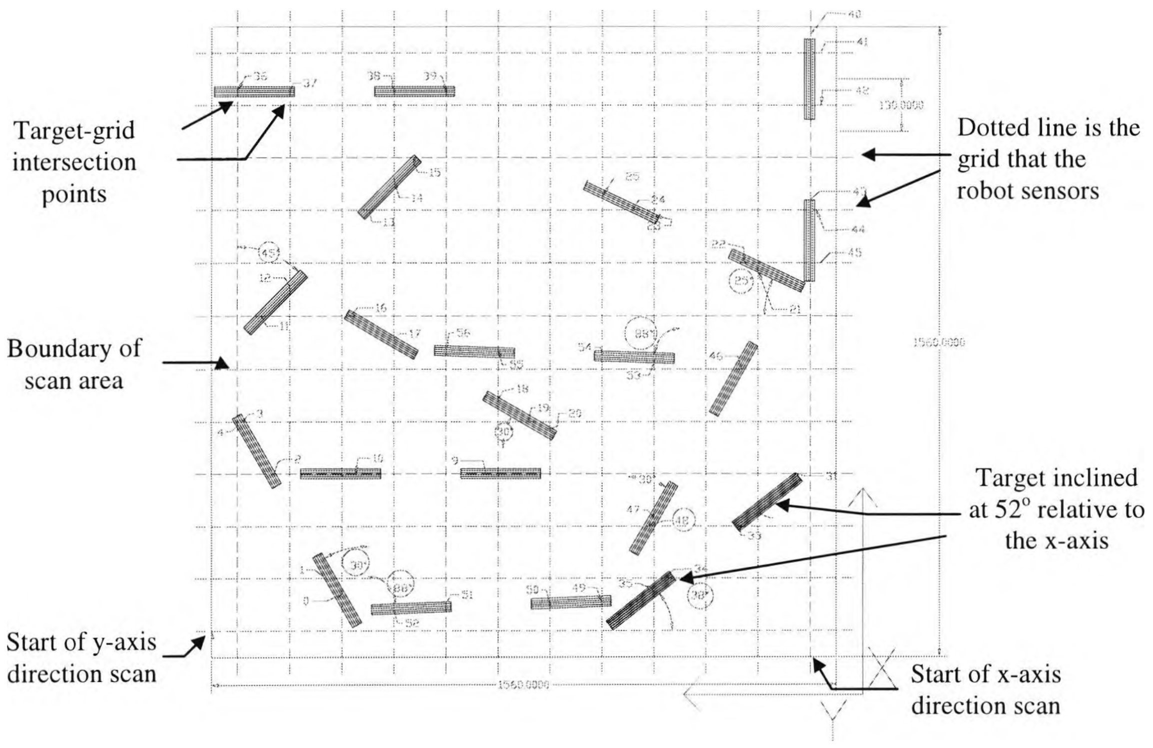

Target-grid intersection points
Boundary of scan area
Start of y-axis direction scan
Dotted line is the grid that the robot sensors
Target inclined at 52° relative to the x-axis
Start of x-axis direction scan
1560.0000
130.0000

Figure 5.3 *AutoCAD drawing of the scanned area and targets at different location*

Table 5.1 *Reference to targets in figure 5.3*

Target number	Intersection point identifiers in figure 5.3	Actual centroid $\bar{x}$ (mm)	Actual centroid $\bar{y}$ (mm)	Calculated centroid $\bar{x}$ (mm)	Calculated centroid $\bar{y}$ (mm)	Vector error in centroid (mm)
1	36-39	3849.6	1254.7	3848.7	1260.5	5.9
2	11-15	3470.4	1257.6	3463.8	1269.1	13.3
3	31-35	2711.8	327	2741.9	294.3	44.4
4	16-19	3149.5	963.2	3110.2	891.7	81.6
5	40-45	3678	65.4	3725.8	66.0	47.8
6	9-10	2905	1043.7	2903.9	1024.65	19.1
7	21-25	3490.1	356.5	3484.0	345.35	12.7
8	0-4	2789.1	1346.9	2779.7	1341.2	11.0
9	46-48	2968.5	357.7	2899.3	391.7	77.1
10	53-56	3200.3	703.3	3201.0	715.0	11.7
11	49-52	2580.7	862.6	2578.1	825.2	37.5

Table 5.2 *Results of target direction estimation using moments*

Target no.	**Actual principle direction (degrees)**	**Principle axis direction θ (degrees)**	**2nd moments Ipxx (mm^2)**	**2nd moments Ipyy (mm^2)**	**Principle axis direction**	**Radios of Gyration rx**	**ry**	**Error in estimated direction θ**
1	0	-0.01	930857	1560	y-axis	215	8	0.01
2	-45	-43.51	6291	84082	x-axis	17	205	1.49
3	38	38.14	519175	1624	y-axis	161	9	0.14
4	-30	-30.23	871116	1622	y-axis	208	9	0.23
5	0	-0.07	1070	854768	x-axis	7	206	0.07
6	0	0.12	836144	875	y-axis	204	6	0.12
7	-25	-24.32	487795	1192	y-axis	156	7	0.68
8	30	30.40	1978	906298	x-axis	10	212	0.40
9	-30	-30.41	1328	817569	x-axis	8	202	0.41
10	-2	-2.35	846428	1392	y-axis	205	8	0.35
11	2	1.88	891947	1445	y-axis	211	8	0.12

5.3.1.2.2 Target detection by robot

For the scan, the TCP of the robot is chosen to be at the point of the vertically directed sensor mounted on the robot end-effector, called '*sensor_scan_TCP*' (figure 2.3.12). When performing the scan the robot was moved with the TCP set directly above the known target level and within the designated area for the search. The commencement points of the scan in the x and y directions are shown in figure 5.3. As stated previously, the target height is known, and the distance from the sensor to the top of the target is within the working range of the sensor.

The C program '*targ_fun.mak*', developed to scan the area and analyse the result of the scan, is based on the procedure described below. The robot moves in a linear fashion which allows the scan to be performed at a constant speed. The limits of the yaw axis were incorporated and respected in the search.

The application procedure follows (including pseudo code):

```
(i)- Configure robot, and set-up sensor communications drivers.
(ii)- Scan for target detection:
    • Set TCP at sensor_scan_TCP
    • Open file 'scan_results' to store scan results
    • Go to start of x-axis, scan grid with the yaw_axis pointing in the
      direction of motion
    • Do x-direction scan
    while moving on the grid
    If (sensor reading) = (distance from sensor to target top ± tolerance),
 then
    robot (x) position = target_x_pos[z]
    robot (y) position = target_y_pos[z]
 increment z
• Go to start of y-axis scan grid with the yaw_axis pointing in the direction
  of motion
• Do y-scan
  while moving on the grid
    If (sensor reading) = (distance from sensor to target top ±3 mm),
  then
    robot (x) position = target_x_pos[z]
    robot (y) position = target_y_pos[z]
    increment z
• Store all target_x_pos and target_y_pos co-ordinate in file 'scan_results'
```

(iii)- Read in data collected and convert to *sensor_scan_TCP* position

(iv)- Find centroid $(\bar{x}, \bar{y})$, direction of principle axis (θ) and 2nd moments (I_pxx, I_pyy, and I_pxy), using procedure described in section(5.2.1.1).

(v)- Find direction of target θ relative to robot axes

```
If principle direction is y-axis ($I_pxx < I_pyy$) then Angle = θ
If principle direction is x-axis ($I_pyy < I_pxx$) and θ is -ve, then Angle = 90 + θ
If principle direction is x-axis ($I_pyy < I_pxx$) and θ is +ve, then Angle = θ - 90
```

(vi)- Convert $(\bar{x}, \bar{y})$ to *'block_pick_TCP'* co-ordinate system

(vii)-Calculate two possible robot end-effector angles to face target for the purpose of searching target centre point:

```
    1st option yaw_angle = Angle +90
    2nd option yaw_angle = Angle - 90
```

The conveyor could be at either side of the target, hence there is a need to record the two yaw_axis angle options for further investigation. Determination would be a simple procedure with scan detection of other features and objects belonging to the conveyor, such as height of belt relative to target height. Due to the lack of time and the difficulty in changing the conveyor's position for experiments it is guided by the user's choice of yaw angle for the robot end-effector to face the target front.

Experiments were carried out to check the accuracy of the method and procedure described in this section. For these experiments, it was necessary to be able to move the target to different positions and set different orientations in the area designated for the scan. To overcome the problem of relocating the conveyor within the robot cell, the target was mounted on a portable rotary table, which was first aligned to the robot axis system, as seen in figure 5.4. Different angles and positions of the target were tested, and the accuracy of the method determined. This approach was vital in the development of the next stage of the procedure for the precise location of the target, as described in section 5.3.2.

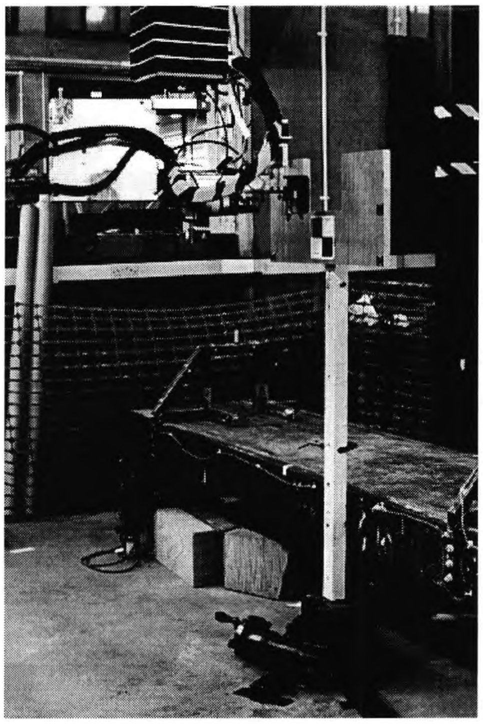


Figure 5.4 *Target set-up on portable table*

5.3.1.2.2.1 Accuracy of angle and centroid

Experiments were carried out to determine the accuracy of the scanning detection method. This was done by rotating the target to known angles, and comparing them with the scan detected values. The table was also placed at different positions relative to the grid system. This was done to test the effect of the number of times the target was traversed during the scan on the accuracy of the detected angle and centroid position. The rate of data captured from the sensors was fixed. The effect of the speed of the robot on the number of data points read was also investigated. This was done by varying the speed of the linear moves for each scan, for the same target position. The worst case was included for i.e. when the robot traverses the target four times only. The best case was when it traversed it more than eight times.

In the first experiment, the target was placed at 90° to the robot x-axis and the effect of changing the speed of the scan tested to determine the accuracy and repeatability of the results achieved. Two cases were investigated. The first was the worse case and the results of that are shown in tables 5.3 and 5.5. The better results of the second are shown in tables 5.4 and 5.6. The angles and centroid were calculated using the procedure described earlier. From these results, the largest maximum errors in the angle and centroid were found to be 5.72° and 95 mm, as seen in tables 5.3 and 5.5 respectively. These errors correspond to the combination of the target traversed a minimum number of times and the robot making the scan in a minimum time of 52 seconds. The largest mean errors of the angle and centroid approximation from the worst case test were found to be 2° and 26 mm. The largest maximum errors in the angle and centroid approximation, for the best case tested, were found to be 1.52° and 30 mm, as can be seen in tables 5.4 and 5.6 respectively. These errors were due to the combined effects of having the target traversed a maximum number of times during the scan, and the robot making the scan in a minimum time of 52 seconds (robot moving at an average speed of 800 mm/sec). The largest mean error of the angle and centroid estimation of the best case test were found to be 1.43° and 18.4 mm.

This error is as anticipated due to the method adopted for processing the data and the minimum time in which the scan was required to be completed. These rough approximations of the angle and centroid of the target, and the maximum errors found, will be used as a starting point to finding the target more accurately, described in section 5.3.2.

Table 5.3 *First stage detection of target angle: worst case scan*

Total Time of	**scan is 122 secs**	
No. data points read	**Target angel θ (degrees)**	**Maximum angle error (degrees)**
10	-1.55	2.42
16	0.44	**Mean of absolute angle**
18	0.81	1.25
14	-2.42	**Standard deviation of angle**
16	-1.01	0.77
Total Time of	**scan is 75 secs**	
No. data points read	**Target angel θ (degrees)**	**Maximum angle error (degrees)**
10	-0.36	1.36
8	-1.27	**Mean of absolute angle**
7	1.2	1.01
9	-1.36	**Standard deviation of angle**
8	0.87	0.41
Total Time of	**scan is 52 secs**	
No. data points read	**Target angel θ (degrees)**	**Maximum angle error (degrees)**
6	-2.77	5.72
7	0.88	**Mean of absolute angle**
6	-1.67	2.06
6	-2.09	Standard deviation of angle
6	-1.56	1.41
4	5.72	
5	1.14	
6	-2.17	
6	-1.01	
6	-1.57	

Table 5.4 *First stage detection of target angle: best case scan*

Total Time of	**scan is 122 secs**	
No. data points read	**Target angel θ (degrees)**	**Maximum angle error (degrees)**
50	-0.66	0.66
50	-0.39	**Mean of absolute angle**
54	0.22	0.36
54	0.13	**Standard deviation of angle**
51	-0.54	0.22
Total Time of	**scan is 75 secs**	
No. data points read	**Target angel θ (degrees)**	**Maximum angle error (degrees)**
53	-0.66	0.85
51	-0.85	**Mean of absolute angle**
53	-0.62	0.74
52	-0.78	**Standard deviation of angle**
53	-0.8	0.10
Total Time of	**scan is 52 secs**	
No. data points read	**Target angel θ (degrees)**	**Maximum angle error (degrees)**
17	-1.43	1.52
17	-1.45	**Mean of absolute angle**
17	-1.39	1.43
17	-1.38	**Standard deviation of angle**
17	-1.52	0.06

Table 5.5 *First stage detection of target centroid: worst case scan*

Total Time of scan is 122 secs

No. data points read	**Actual centroid** $\bar{x}$ (mm)	y (mm)	**Detected centroid** $\bar{x}$ (mm)	y (mm)	**Difference in centroid (mm)**
10	2382	719	2388	692	27.7
16	2382	719	2389	697	23.1
18	2382	719	2390	707	14.4
14	2382	719	2383	714	5.1
16	2382	719	2381	714	5.1
				mean of error	**15.08**

Total Time of scan is 75 secs

No. data points read	**Actual centroid** $\bar{x}$ (mm)	y (mm)	**Detected centroid** $\bar{x}$ (mm)	y (mm)	**Difference in centroid (mm)**
10	2382	719	2386	714	6.4
8	2382	719	2388	763	44.4
7	2382	719	2378	752	33.2
9	2382	719	2385	700	19.2
8	2382	719	2390	731	14.4
				mean of error	**23.52**

Total Time of scan is 52 secs

No. data points read	**Actual centroid** $\bar{x}$ (mm)	$\bar{y}$ (mm)	**Detected centroid** $\bar{x}$ (mm)	$\bar{y}$ (mm)	**Difference in centroid (mm)**
6	2382	719	2402	714	20.6
7	2382	719	2383	752	33.0
6	2382	719	2395	714	13.9
6	2382	719	2390	714	9.4
6	2382	719	2393	714	12.1
4	2382	719	2402	812	95.1
5	2382	719	2393	688	32.9
6	2382	719	2390	714	9.4
6	2382	719	2399	714	17.7
6	2382	719	2398	714	16.8
				mean of error	**26**

Table 5.6 *First stage detection of target centroid: best case scan*

Total Time of scan is 122 secs					
No. data points read	Actual centroid		Detected centroid		Difference in centroid (mm)
	$\bar{x}$ (mm)	y	$\bar{x}$ (mm)	y	
50	2382	719	2383	701	18.0
50	2382	719	2382	704	15.0
54	2382	719	2380	715	4.5
54	2382	719	2381	716	3.2
51	2382	719	2382	705	14.0
				mean of error	**10.94**
Total Time of scan is 75 secs					
No. data points read	Actual centroid		Detected centroid		Difference in centroid (mm)
	$\bar{x}$ (mm)	y	$\bar{x}$ (mm)	y	
53	2382	719	2383	700	19.0
51	2382	719	2382	711	8.0
53	2382	719	2382	700	19.0
52	2382	719	2381	707	12.0
53	2382	719	2382	701	18.0
				mean of error	**15.2**
Total Time of scan is 52 secs					
No. data points read	Actual centroid		Detected centroid		Difference in centroid (mm)
	$\bar{x}$ (mm)	y	$\bar{x}$ (mm)	y	
17	2382	719	2382	702	17.0
17	2382	719	2381	704	15.0
17	2382	719	2381	704	15.0
17	2382	719	2381	704	15.0
17	2382	719	2381	689	30.0
				mean of error	**18.4**

The second stage of the experiment was to check the accuracy of the previously detected angle of the target for different orientations to the robot axes. This was done by turning the table to different angles and comparing this angle with that detected from the scanning process. The results of the experiments are shown in table 5.7. The maximum error found was 5.28°, and the mean error 2.41°. Again, as the angle results are only meant to be an approximate of the actual angle, they give an acceptable basis for the closer search of the target in stages two and three of the process.

Table 5.7 *Second stage detection of target angle*

Set target angles to the robot x-axis (degrees)	Angle of target from scan result (degrees)	difference in angle estimation (degrees)
-60,120	-65.28	5.28
-60,120	-60.37	0.37
-40,140	-37.72	2.28
-30,150	-27.95	2.05
0,-180	-1.55	1.55
20,-160	15.71	4.29
45,-135	48.03	3.03
-70,110	69.57	0.43
	mean of error	**2.41**

5.3.2 Final location of the target centre (stage 2 and 3)

Using the approximate angle and centroid of the target detected with the aid of the moments method, the robot searches closer to the target to detect the precise position of the target centre point. This is done in two steps, using the horizontally pointing sensor, shown in figure 5.6, and an algorithm which was developed through experimentation. The first step, which is the second stage of the process, will be to correct the first approximation of the angle, thus achieving a more accurate angle of the target, hence the conveyor.

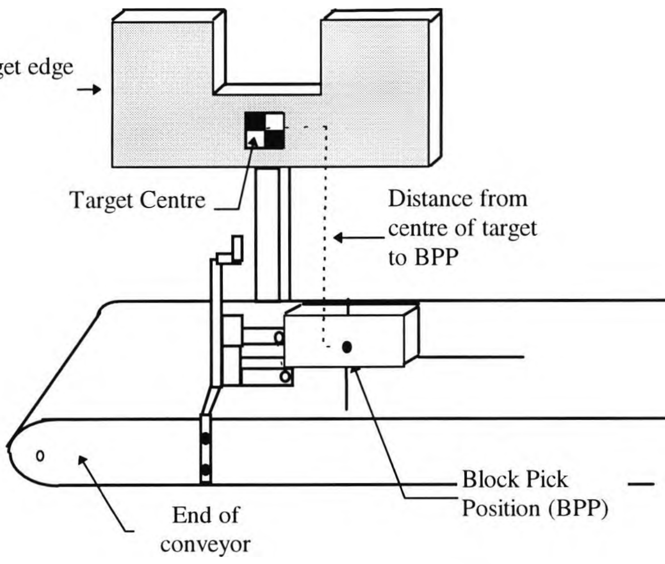

Target edge
Target Centre
Distance from centre of target to BPP
End of conveyor
Block Pick Position (BPP)

Figure 5.5 *Block pick position relative to target centre*

This will help to eliminate the error incurred when picking up a block, which has to be done very accurately. The second step, which is the third stage of the process, will be to find the target centre point, in the '*pick_block_tcp*' co-ordinate systems, which will ultimately allow the location of the BPP on the conveyor shown in figure 5.5.

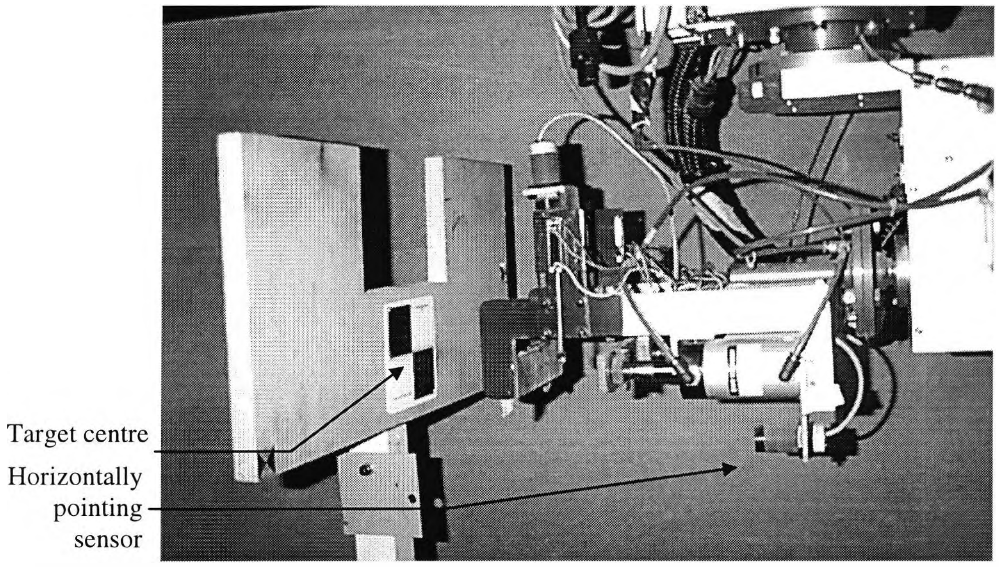


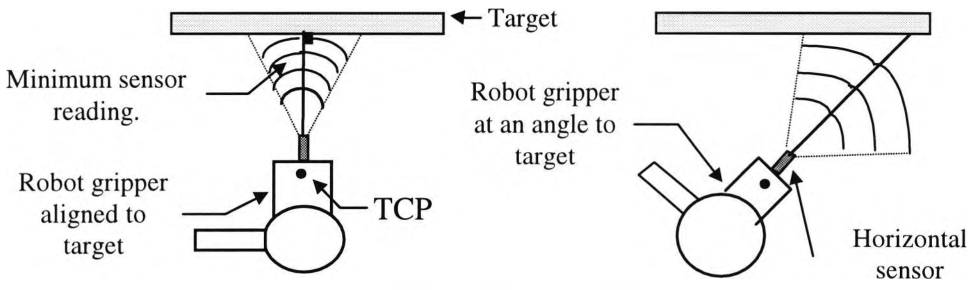
Figure 5.6 *Searching for the centre of the target using the horizontally pointing sensor*

5.3.2.1 Correction of detected target angle (stage 2)

To detect the angle of the target more accurately, the horizontally pointing sensor is used. This method exploits the distance accuracy of the ultrasonic sensors, which was confirmed in experiments carried out to determine the behaviour of the sensors.

5.3.2.1.1 Sensor readings related to direction angle

The ultrasonic sensors have an approximate cone shaped working detection envelop. It was thus necessary to find the working range of angles and their corresponding sensor readings when pointing towards the target surface. This was done by first aligning the robot end-effector to the target middle and normal to it. From this position, the robot gripper was moved, using the *'block_pick_TCP'*, to different angles (±18°) and the sensor readings recorded (figure 5.6). This was repeated for different distances of the end-effector from the target. Results in table 5.8 are of a case where the sensor is at a distance of 350 mm from the target (distance used for our algorithm) and the end-effector normal to the target. From this, it can be concluded that at around ±8°-9° the reading is at a minimum for that distance. At angles of -12° and +14° from the normal, the sensor reading is out of range. The experiment was repeated a few times, and the angle at which the sensor has an out of range reading was concluded to be, on average, ±12.7°. The distance of the sensor from the target affects this value.



Target
Minimum sensor reading.
Robot gripper aligned to target
TCP
Robot gripper at an angle to target
Horizontal sensor

Figure 5.7 *Determining sensor reading at varying angle to the target*

Table 5.8 *Results of sensor reading when end-effector is at -90° and aligned to target*

End-effector angle (degrees)	H_sensor reading (mm)	End-effector angle (degrees)	H_sensor reading (mm)
-90→ -82	350	-92 → -101	350
-81	351	-102	351
-80	354	-103	352
-79	362	-104 onwards	out of range
-78 onward	out of range	----	----

5.3.2.1.2 Method of angle correction

The approximate angle and centroid of the target, both determined in stage 1 of the process using procedure described in section 5.3.1.1, are used at the start of the method described in this section to find the target angle more accurately. The robot end-effector is first set to face the target (figure 5.6), using the approximated angle and centroid. To search for the target closer the horizontal sensor (H_sensor) were used. Using the results from the sensor experiments in section 5.3.2.1.1, which investigate how the sensor readings vary for different target angles, an algorithm was developed. This algorithm, which is the second stage of the process, started by locating the target centre roughly, then aligning the robot end-effector to it more accurately. Function '*correct_ang()*', which is part of the 'C' program '*targ_fun.mak*', is based on the procedure developed for this purpose. The procedure is described in the following.

The resulting worse accuracy of the detected target angle and centroid from first stage, investigated in section 5.3.1.2.2.1 are $\pm 5.72^{\circ}$ and ± 95 mm, respectively. Using the detected angle and the known height of the target, the robot is first lowered at a safe vertical offset distance of 400 mm from the detected target centroid. This is done with the assumption that there are no unknown objects in the area of the target.

All the moves made were in the *'block_pick_TCP'* co-ordinates system. Functions were written to move the robot end-effector sideways, and backwards and forwards in the orientation of the angle the yaw is at. The function's names are *move_forward(angle,distance)* and *move_side(angle, distance)*, both included in file *'Find_target.c'*.

The application procedure for 'correct_angle' follows (stage 2):

```
(i)- Using Yaw angle based on the stage 1 target angle found in section (5.3.1.2.2),
      move yaw to that angle using 'block_pick_TCP'.
(ii)- Move away from the target approximate centroid position $(\bar{x},\bar{y})$, from stage 1,
   by 400 mm.
(iii)- Lower z-axis to a position where H_sensor is pointing at lower half of target.
(iv)- Do while (H_sensor = out_of_range)
          • move end-effector $-18^{\circ}$, then move forward in that direction 15 mm
          • if (H_sensor = out_of_range)
                  • move end-effector $+18^{\circ}$, then move forward in that direction
                    15 mm
                  • read H_sensor
(v)- Adjust end-effector to be roughly aligned to target (angle estimate depends on
      distance away from target)
          • move end-effector $20^{\circ}$
          • move end-effector $60^{\circ}$ slowly in the opposite direction
            while moving
            record (H_sensor[z], x_pos[z], y_pos[z], yaw_angle[z])
          • Find Min(H_sensor[z]) that is in_range
          • Move end-effector to yaw_angle[z + 5]
(vi)- Read H_sens and move towards target by (350- H_sensor)
(vii)- Find a estimate of target mid point
          • Find estimate of target edge
             Move end-effector 30 mm sideways
                 Repeat Until (H_Sensor = out_of_range)
             Move end-effector sideways to opposite direction 10 mm
                 Repeat Until (H_Sensor $\neq$ out_of_range)
          • Move 1/2 (length of target) sideways
```

```
(viii)- Find Angle $\pm 1^o$ accuracy
    • if (H_sensor = out_of_range)
          { move end_effector -$13^o$
             if (H_sensor = out_of_range) try other side
                 move end_effector $26^o$
                 while (H_sensor = in_range)
                     move end_effector -1 .$5^o$
                 move end_effector $12^o$
             otherwise if (H_sensor = in_range)
                 while (H_sensor = in_range)
                     move end_effector $1.5^o$
                 move end_effector -$12^o$ }
    • if (H_sensor = in_range)
          while (H_sensor = in_range)
              move end_effector -$1.5^o$
          move end_effector $12^o$
(iv)- Find Angle accurately ($\pm 0.5^o$)
    • move end-effector one side of target 200 mm
    • read (H_sensor = Side1)
    • move end-effector one side of target -200 mm
    • read (H_sensor = Side2)
    • find gradient = (Side1-Side2)/400
    • Angle of difference = atan(gradient)
    • move end_effector by Angle of difference
    • do step 8 until angle of difference < $\pm 0.5^o$
```

Experiments to verify the accuracy of the corrected angle were carried out. These are described together with the results in section 5.4.1.

5.3.2.2 Target centre location (stage 3)

After aligning the robot end-effector successfully with the target, using the procedure described in section 5.3.2.1, the next step (stage 3) is to find the target centre accurately. Finding the co-ordinates of the target's centre allow determination of the BPP on the conveyor, using the *'block_pick_TCP'* co-ordinates system. The heights of the target, target centre and conveyor, are all known through the use of the robot positional readings, all determined at the start of this stage. The target centre is detected relative to the TCP of the robot end-effector (refer to section 2.2.1 for details on the TCP position).

The (x,y) co-ordinates of the target centre are found from the accurate determination of the target edge position. To find the edge of the target, some experiments have to be carried out first, to check the kind of values expected from the ultrasonic sensors when used for reading edges. Once that is done, the concluded values are used in a procedure for finding the target centre.

5.3.2.2.1 Sensor readings for edges

Because the ultrasonic sensors have a large beam width and are used normal to the surface, edges cannot be determined easily. To enable the determination of the target edges accurately, an experiment needed to be carried out. The target was placed at 0° relative to the robot x-axis. The robot end-effector was aligned to the target (i.e. yaw axis being at angle -90°) making the pointing sensor normal to the target. The end-effector was then moved to a position roughly the middle of the target and at an offset of 330 mm from it. The end-effector was then moved slowly along the target, until it passed the target edge (i.e. the sensor reads out of range). From here, the end-effector was moved back towards the target in small steps, while maintaining it's alignment to the target. At the first sensor reading not out of range, the distance from the TCP point is measured to the edge of the target, this distance called 'D'. 'D' will be the offset from the edge of the target at the first instance of sensor reading being out of range, as illustrated in figure 5.8. The position of the robot at that point is noted, in the *'block_pick_TCP'* co-ordinate system. The robot end-effector is then moved to the estimated mid-point of the target. This is done by moving it a distance of Edge_offset + 1/2(length of the target), while still aligned with the target. At this point, the TCP is at the mid point of the target. The position of the end-effector is recorded at this point. The experiment was repeated 10 times, the results shown in table 5.9. The resulting distance of the edge offset 'D' from the TCP was found to be, on average, 39.92 mm.

Table 5.9 *Results of target edge offset*

Distance of TCP from edge 'Edge_offset' (mm)
40.0
39.0
39.8
39.9
39.5
40.0
39.9
40.5
40.3
40.3
Average value Of
(D) = 39.92 mm

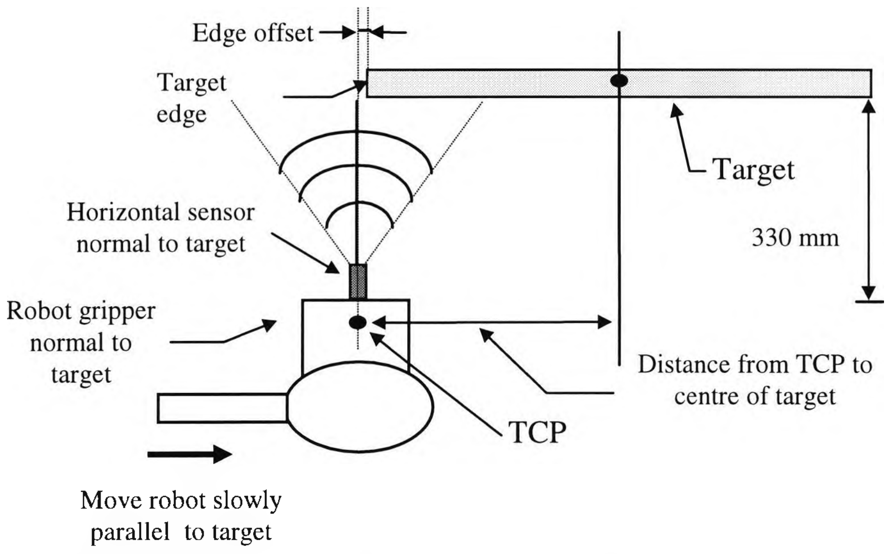

Edge offset
Target edge
Horizontal sensor normal to target
Robot gripper normal to target
Target
330 mm
Distance from TCP to centre of target
TCP
Move robot slowly parallel to target

Figure 5.8 *Determining target edge using horizontal sensor*

5.3.2.2.2 Procedure for locating target centre

Using the Edge_offset 'D', found in section 5.3.2.2.1, and the corrected angle of the target found in section 5.3.2.1, a procedure was developed to find the target centre (stage 3). The edge of the target is found first, then the Edge_offset 'D' is added to it as well as half the length of the target (300 mm). The procedure is given below and can be followed by reference to figure 5.8.

The application procedure (stage 3):

(i)- Using the approximate target edge location, and corrected angle, (both from stage 2), the end-effector is set at normal to the target, at an offset of 330 mm (figure 5.8).

(ii)- while (H_sensor = in_range)

```
          move end-effector 10 mm sideways
(iii)- while(H_sensor = out_of_range)
          move end-effector -0.5 mm sideways
(iv)- move end_effector -(1/2 target length + D)
(v)- move z-axis so that TCP at mid point of target
```

Experiments were carried out to determine the accuracy of the method. They are described with their results in section 5.4.1.

5.3.3 Block Pick Point Location (stage 4)

After the target was mounted and aligned to the conveyor, the distance from the BPP, (for more details on the BPP see chapter 2.3.1.3) to the target centre was measured accurately. This arrangement is shown in figures 5.5 and 2.3.12. From the concluded target centre, found by methods described in sections 5.3.2.2, the BPP position is calculated. This plays a central role in finding the precise pick position of a block on the conveyor, according to the method described in chapter 2.4.2.

5.4 Target location experiments

Two types of experiments were made. The first was to investigate the accuracy of the target location using the whole process (stages 1, 2 and 3). The second was to test the point of failure of the search algorithm, relative to the errors in the first estimations of the angle and centroid of the target. Both tests were done using the table upon which the target was mounted, and aligned with the robot axes, as shown in figure 5.4. Using the table will allow varying the position and angle of the target for the experiments.

5.4.1 Accuracy of target location

The target was mounted on a portable table, as described in section 5.3.1.2.2. For each experiment the angle of the target was changed. For each angle a scan of the area was done, and the results analysed to detect the target angle and centroid. The target location is then searched for, using the algorithm described in section 5.3.2. The correction in the angle resulting from the *correct_ang()* procedure (stage 2) is recorded, as well as the difference between the actual target centre point and the centre point detected at stage 3. This was done a number of times, each time using different angles for the target. The results are shown in table 5.10. One angle position was tested a number of times, to check the repeatability of the results. These results are shown in table 5.11. From this, it can be seen that the average error in the target angle is 0.4°, with the maximum error of 1.26°. On average, the error in the centre of target results was 1.3 mm, with the maximum error of 2.3 mm. The effect of these error on the BPP location, and its effect on the picking up a block from the conveyor procedure are discussed in section 5.4.3.

Table 5.11 gives the experiment results for the repeatability test. The results indicate that they are highly repeatable.

Table 5.10 *Target angle experiments: testing at different positions*

Actual target angles (degrees)	Approximate angle of target (stage 1) (degrees)	Actual Yaw angle (degrees)	yaw angle (degrees)	Corrected yaw angle (stage 2) (degrees)	Error in angle (degrees)	Error in corrected angle (degrees)	Error in target mid point location (mm)
(-60),(120)	-65.28	30	24.72	30.02	5.28	0.02	2.3
(-60),(120)	-60.37	30	29.63	29.88	0.37	0.12	1
(-40),(140)	-37.72	50	52.73	49.83	2.73	0.17	1.5
(-30),(150)	-27.95	60	62.05	58.74	2.05	1.26	0.3
(0),(-180)	-1.55	90	88.45	89.46	1.55	0.54	0.1
(20),(-160)	15.71	-70	-74.29	-69.54	4.29	0.46	0.5
(45),(-135)	48.03	-45	-41.97	-44.63	3.03	0.37	2.2
(-70),(110)	69.57	-20	-20.43	-19.69	0.43	0.31	2.3
				Mean error =	**2.5**	**0.4**	**1.3**
				Maximum error =	**5.28**	**1.26**	**2.3**

Table 5.11 *Target angle experiments: repeatability test*

Actual target angles (degrees)	Approximate angle of target (stage 1) (degrees)	Actual Yaw angle (degrees)	yaw angle (degrees)	Corrected yaw angle (stage 2) (degrees)	Error in angle (degrees)	Error in corrected angle (degrees)	Error in target mid point location (mm)
(-20),(160)	-16.63	70	73.36	69.92	3.36	0.08	0.3
(-20),(160)	-18.06	70	71.94	70.05	1.94	0.05	0.8
(-20),(160)	-16.45	70	73.55	69.97	3.55	0.03	0.3
(-20),(160)	-15.94	70	74.06	69.83	4.06	0.17	0.8
(-20),(160)	-16.16	70	73.84	69.9	3.84	0.10	0.3
				Mean error =	**3.35**	**0.09**	**0.50**
				Maximum error =	**4.06**	**0.17**	**0.80**

5.4.2 Search algorithm failure experiment

Experiments were carried out to determine the point of failure of the search algorithm, as a judgement on its reliability. As the algorithm depends on the approximate results from stage 1 for the target angle and centroid, these values are varied in order to find the point of failure. Using the same arrangement of the target, as described in section 5.3.1.2.2, the target is scanned and the results noted (stage 1). Two observations were made, the first to find the effect of the error in the stage 1 approximated angle on the failure of the algorithm. The second is to observe the effect of the error in the stage 1 detected centroid, on the failure of the algorithm.

5.4.2.1 Effect of error in stage 1 detected angle

A check on the point of failure of the search algorithm, and how robust it is using the (stage 1) detected target angle when it is fairly inaccurate in its approximation. This will be carried out by setting the position and an angle of the target relative to the robots x-axis. A scan of the area is made, and the stage 1 detected target angle and centroid calculated using the C program '*target.c*' (described in section 5.3.1.1). The algorithm for searching more accurately for the centre of the target (stages 2 and 3, both described in section 5.3.2) was tested by varying the value of the stage 1 detected angle, which the search algorithm uses as a starting point. The actual target angle was deliberately offset for each case, and this used instead of the stage 1 detected angle. The offset was increased until the search algorithm failed to locate the target. This experiment (tables 5.12 and 5.13) shows that, an error in the initial target angle can be as high as 20° in the first case and 40° in the second case, and still the search algorithm was able to accurately locate the target centre.

Table 5.12 *Testing the effectiveness of the search algorithm (target at 0° angle to the x-axis)*

Offset Angle (degrees)	**Target angle entered (degrees)**	**Yaw angle entered (degrees)**	**Corrected Yaw angle (stage 2) (degrees)**	**Error in target angle concluded (degrees)**	**Centroid $\bar{x}$ (mm)**	**y**	**Result centroid $\bar{x}$ (mm)**	**y**
-0.97	-0.97	-90.97	-90.92	0.91	2311	857	2120	718
-5	-5	-95	-90.82	0.82	2312	858	2120	718
5	5	-85	-90.82	0.82	2312	857	2120	718
-10	-10	-100	-90.80	0.80	2311	858	2120	718
10	10	-80	-90.73	0.73	2311	858	2120	718
-15	-15	-105	-90.56	0.56	2309	863	2120	718
15	15	-75	-90.50	0.50	2309	863	2120	718
20	20	-70	-90.56	0.56	2309	863	2120	718
25	25	-65	Failed	Failed	Failed	Failed	Failed	Failed
			Mean angle error =	**0.71**	**Measured error in**			
			Maximum angle error =	**0.91**	**target center =1.2 mm**			

Table 5.13 *Testing the effectiveness of the search algorithm (target at 90° angle to the x-axis)*

Offset Angle (degrees)	Target angle entered (degrees)	Yaw angle entered (degrees)	Corrected Yaw angle (stage 2) (degrees)	Error in target angle concluded (degrees)	centroid		Result centroid	
					$\bar{X}$ (mm)	$\bar{Y}$	$\bar{X}$ (mm)	$\bar{Y}$
-1.36	88.64	-1.36	0.25	0.25	2135	687	2257	725
-5	85	-5	0.12	0.12	2128	687	2257	725
5	95	5	-0.06	0.06	2130	686	2257	725
10	100	10	0.15	0.15	2135	686	2257	725
-10	80	-10	0.07	0.07	2135	686	2257	725
15	105	15	0.07	0.07	2135	686	2257	725
-15	75	-15	0.22	0.22	2135	686	2257	725
20	110	20	0.00	0.00	2135	686	2257	725
-20	70	-20	-0.36	0.36	2134	686	2257	725
25	115	25	-0.08	0.08	2135	686	2257	725
-25	65	-25	0.28	0.28	2134	686	2257	725
30	120	30	-0.24	0.24	2135	686	2257	725
-30	60	-30	-0.12	0.12	2129	686	2257	725
35	125	35	0.30	0.30	2135	686	2257	725
-35	55	-35	0.07	0.07	2134	686	2257	725
40	130	40	0.45	0.45	2135	686	2257	725
-40	50	-40	0.50	0.50	2129	732	2257	725
45	135	45	Failed	Failed	Failed	Failed	Failed	Failed
-45	45	-45	Failed	Failed	Failed	Failed	Failed	Failed
			Mean angle error =	**0.095**	**Measured error in**			
			Maximum angle error =	**0.50**	**target center = 1.1 mm**			

5.4.2.2 Effect of error in estimated centroid

A check on the point of failure of the search algorithm, and how robust it is, using the stage 1 detected target centroid, when it is fairly inaccurate in its approximation, was carried out. For this, the target was set in a position and the angle of the target set relative to the robot x-axis. A scan of the area is made, and the stage 1 detected target angle and centroid found using C program '*target.c*' (described in section 5.3.1.1). The algorithm to search more accurately for the centre of the target (stages 2 and 3, described in section 5.3.2) was used to find the centre of the target. This was then repeatedly tested, with the centre of target offset varied. The offset value was increased to determine the point of failure of the algorithm. Table 5.14 gives the results of this experiment. These results show that the error in the centre of target detection, used at the start, can be as high as ±320 mm for the search algorithm to still accurately locate the target.

Table 5.14 *Testing the effectiveness of the search algorithm (target at 90° angle to the x-axis)*

Offset (mm)	**Offset target centroid** $\bar{x}$ (mm)	y	**Result target centroid (stage 3)** $\bar{x}$ (mm)	y	**Corrected Yaw angle (stage 2) (degrees)**
-150	2122	584	2129	734	0.83
150	2122	884	2129	733	0.69
-200	2122	534	2129	733	0.45
200	2122	934	2130	733	0.36
-250	2122	484	2130	733	0.29
250	2122	984	2129	734	0.49
-280	2122	454	2129	734	0.49
280	2122	1014	2129	734	-0.01
-300	2122	434	2129	733	0.23
300	2122	1034	2129	735	0.49
-320	2122	414	2130	734	0.15
320	2122	1054	2129	734	0.12

5.4.2.3 Effect of error on stage 1 detected angle and centroid

The same type of experiments described in sections 5.4.2.1 and 5.4.2.2 were carried out. This time, the combined effect of the errors in the angle and centroid on the search algorithm are investigated. The results of the experiments are shown in table 5.15. These show that the algorithm was not reliable, when the error of the angle reached ±20° and the error in the centre detection reached ±280 mm.

Table 5.15 *Testing the effectiveness of the search algorithm (target at 90° angle to the x-axis)*

Angle offset (degrees)	**Centroid Offset (mm)**	**Offset target centroid** $\bar{x}$ (mm)	y	**Result target centroid (stage 3)** $\bar{x}$ (mm)	y	**Corrected Yaw angle (stage 2) (degrees)**
-20.0	-280	2122	454	2128	734	0.23
-20.0	280	2122	1014	2129	734	0.1
20.0	-280	2122	454	Failed	Failed	Failed
20.0	280	2122	1014	2128	734	0.12
-20.0	-300	2122	434	2128	734	0
-20.0	300	2122	1034	2129	734	0.08
20.0	-300	2122	434	Failed	Failed	Failed
20.0	300	2122	1034	2128	734	0.02

5.4.3 Results of block pick point detection experiments

The target location accuracy has an effect on BPP location. From the experiments to determine the accuracy of the method (section 5.4.1), the target angle error was found to be at a maximum of 1.26°. The maximum error in the target centre detection was found to be 2.3 mm. As the distance '*l*' from the centre of rotation of the yaw axis to the '*block_pick_TCP'* is 257.2 mm, the effect of the error in angle is not significant. This is because the estimated effect of the error in angle on picking up a block to be a lateral offset of maximum ±0.06 mm, calculated using equation 5.1.

$$offset = l \sin\theta \times \sin(\theta/2) \qquad \text{eqn 5.1}$$

The error in the target angle detection, which is the error in the conveyor angle, does not effect the success of the block pick-up procedure. That is because an independent method of detecting the block angle on the conveyor was devised that can align the gripper to the block on the conveyor even when it is at 9° angle to it (refer to chapter 6.2.2). A set of three ultrasonic sensors are mounted in a special frame under which the block material passes, as shown in figure 2.3.4. This arrangement enables each block to be precisely measured and located in centroid and inclination relative to the conveyor.

5.5 Conclusion

The process of locating the target was developed in stages, with each stage tested for accuracy. Experiments were also done to establish the accuracy of the ultrasonic sensors with consideration of their use in detection work. The target design and the method of moments, adopted to detect and locate the target shape and position relative to robot axes, were both found to be extremely effective. The target position was always successfully detected (stage 1), with a worse case average error of 2° in the angle and 26 mm in the target centre. The effect of the speed of the scan was found to be reliable using an average speed of 800 mm/sec. The largest error, found from the worst case experiments, with the average scan speed of 800 mm/sec, was found to be 5.72° and 95 mm in the initial angle and the target centre approximation, detected in the first stage, respectively.

To locate the target more accurately, stages 2 and 3, a search algorithm was developed. This algorithm relied, at the start, on the target angle and centre detection of stage 1. Experiments were carried out to establish the reliability of this algorithm. These proved that the algorithm was reliable in accurately locating the target, even if the error in the stage 1 angle detection was as high as ±20°. Furthermore, the algorithm was still reliable when the error in the stage 1 target centroid detection was as high as ±320 mm. But, when both type of errors occur at the same time (i.e. ±20° in the angle estimation as well as ±320 mm in the centre estimation), the algorithm failed. This case will, however, never happen, as the error in the stage 1 detection, at the start, are at maximum, ± 5.72° of the target angle and ±95 mm of the target centre.

In the experiments carried out to determine the accuracy of the complete process, i.e. stages 1, 2, and 3 together, the average error of ±0.4° for the target angle, and ±1.3 mm for the target mid point location were found. Also a maximum error of ±1.26° for the target angle, and ±2.3 mm for the target mid point location were found. Using sensors that are more accurate, the error could be reduced. The effects of these errors on the final calculation of the block pick position (stage 4) from the target angle and centre causes, at maximum, a lateral offset of ±0.06 mm. This offsets is negligible, and on its own, will not cause an unsafe block pick-up operation.

Chapter 6: Experiments In Block Sensing

6.1 Introduction

The purpose of this chapter is to determine the accuracy of the method developed to measure the block dimensions and position on the conveyor, which is described in chapter 2. This enables the necessary run time adjustments to be made for the building process by determination of the accurate block pick-up position on the conveyor. These adjustments were made using the information gathered from the conveyor sensors and the gripper sensors. A detailed description of the types of sensors used on the conveyor, and the process developed to analyse this information, was covered in section 2.3.1. A description of the gripper sensors, and the way they were used to make the necessary final adjustments, for the safe block pick-up, was covered in section 2.4. The 'C' programmes described in sections 2.3.1 and 2.4 were used to carry out the experiments described in this chapter. For the purpose of these experiments, a wooden calibration block was used. This has a length of 440.0 mm, width of 99.9 mm, and height of 214.5 mm and provides a consistent reference to compare results. The experiments first check the accuracy of the dimension measurements of blocks according to stage 2 of the building process (figure 2.2). The overall procedure of picking-up a block from the conveyor, stage 3 of the building process (figure 2.2), was then tested, to check the rates of success or failure.

For the experiments carried out to test the block pick-up procedure, a few assumptions had to be made, the first being that the conveyor position, which affects the BPP, (both determined at the start), was the same throughout the experiments. The method developed to determine the BPP was covered in chapter 5. The second assumption is that the block arrives at the pick-up point according to the position and angle calculated from the conveyor sensor data i.e. the block is not grossly disturbed. These assumptions, are considered reasonable for the purpose of these particular experiments.

For safety of the robot, and extreme reliability of the procedures, there is, however, a need to avoid making any of the assumptions previously stated. Each step of the process, therefore, needs to be confirmed safe and possible variations and consequences anticipated and overcome. This is accomplished using a RBES, which gives intelligence to the robot. The rule-base, which is described in chapter 9, integrates the real-time sensing and the robot functionality. This enables the robot to anticipate different situations and act accordingly, conducting its tasks in a safe manner.

To check the position and orientation of the gripped block relative to the gripper, and the success of the gripping process, an independent check is carried out using a laser profiler. This, along with the different sensors on the gripper, allows final adjustments to the 'theoretical task'. The theory of the process, and a description of the laser profiler, is covered in section 2.5. Experiments carried out to test the method are covered in chapter 7.

6.2 Verification of block sensing method

Experiments were conducted to test the conveyor sensing methods. The process developed to handle the sensor block data and the block pick-up procedure, were also studied. The calibration block was used throughout these experiments. The BPP was calculated, at the start of these experiments, using the conveyor locating results, as described in section 2.3.1.3. A detailed description of the method of locating the conveyor is given in chapters 2.3.2 and 5.

6.2.1 Accuracy of sensor measurement of block

In this section the accuracy of the method developed to calculate the block length 'TL', width 'W', and height 'H' is investigated. For this, the calibration block was passed several times through the different stages of sensing on the conveyor. Each run resulted in a block file, from which the block dimensions were deduced.

6.2.1.1 Verification of block length 'L' sensing

The first experiment checks the accuracy of the conveyor's Senr2 reading of the apparent length 'L'. After a few tests, it became clear that the reading from Senr2, of the block length 'L', was not accurate. This was found to be due to the nature of the type of sensor used. For this sensor to start reading, three quarters of its sensing window has to be covered, which results in an error at the start and end of the block. An experiment was carried out to determine the constant '*Length_Correct*', needed to adjust the value of 'L'. For this experiment, the calibration block was passed through the different sensing stages on the conveyor 40 times. Each time the block file was captured and analysed, to establish the block dimensions and its angle of inclination to the conveyor. Using that angle 'ϕ' as well as the known width of the calibration block (99.92 mm), and the adjusted value of the apparent length 'L' (i.e. L+ *Length_Correct*), the 'TL' of the block was calculated using equations 2.3.3 and 2.3.4. The '*Length_Correct*' value was adjusted, until the average value of the 'TL', from the 40 readings equal to the length of the calibration block. This correction value was found to be 4.17 mm, as shown in table 6.1.

6.2.1.2 Accuracy of block measurements

Experiments were carried out to confirm the accuracy and reliability of the method devised for calculating the block dimensions (TL, H, W), using the conveyor's sensors readings, as described in section 2.3.1. The calibration block was passed through the different sensing stages on the conveyor 34 times. Each time the block file was analysed to establish the block dimensions and the angle of inclination to the conveyor 'ϕ'. When calculating the 'TL', the value of 4.17 mm was used for the '*Length_Correct*' constant as concluded in section 6.2.1.1.

The results of these experiments are shown in tables 6.2 and 6.3. From table 6.3 it can be seen that the readings are reliable. The block length measurement had an average reading of 440.1 mm, which is a difference of 0.1 mm from the original reading. The standard deviation of the length measurements were 0.7 mm, with a maximum error of ±1.2 mm, making it a maximum percentage error of 0.3%. The height measurement were found to be on average 214.4 mm, which is a difference of 0.1 mm from the original reading. The standard deviation of the height reading was 0.5 mm, with a maximum difference of 1 mm, making it a maximum percentage error reading of 0.5%. The worst results were of the width reading. On average that was found to be

100.3 mm, which is a difference of 0.4 mm from the original reading. The standard deviation of the height reading was 0.6 mm, with a maximum difference of 1.3 mm, making it a maximum percentage error reading of 1.3%.

Table 6.1 *Calibration block used to measure the 'Length_Correct' for Senr2's reading*

Block Pass No.	Apparent Length 'L'+ ('Length_Correct' = 4.17 mm) (mm)	Angel of block to conveyor 'ϕ' (degrees)	True Length 'TL' (mm)	Difference of 'TL' reading to the real block length 440 mm (mm)
1	442.32	4.31	440.22	0.22
2	441.98	4.23	439.98	-0.02
3	439.59	3.01	439.12	-0.88
4	438.91	2.51	439.12	-0.88
5	439.25	2.49	439.49	-0.51
6	438.91	2.48	439.17	-0.83
7	438.91	2.27	439.47	-0.53
8	438.57	2.26	439.14	-0.86
9	438.23	1.99	439.18	-0.82
10	438.23	1.97	439.21	-0.79
11	437.20	1.05	439.61	-0.39
12	437.20	0.99	439.71	-0.29
13	436.86	0.90	439.52	-0.48
14	436.86	0.69	439.86	-0.14
15	436.52	0.58	439.70	-0.30
16	435.50	0.50	438.81	-1.19
17	435.84	0.30	439.50	-0.50
18	436.18	0.22	439.97	-0.03
19	436.52	0.15	440.43	0.43
20	436.18	0.08	440.20	0.20
21	436.52	0.01	440.68	0.68
22	436.86	-0.05	440.94	0.94
23	436.86	-0.10	440.86	0.86
24	437.54	-0.34	441.12	1.12
25	437.54	-0.58	440.72	0.72
26	438.57	-1.37	440.47	0.47
27	438.91	-1.48	440.65	0.65
28	438.91	-1.59	440.47	0.47
29	438.91	-1.65	440.39	0.39
30	439.25	-1.83	440.45	0.45
31	438.91	-1.91	439.99	-0.01
32	439.25	-2.16	439.97	-0.03
33	439.25	-2.37	439.67	-0.33
34	440.61	-2.46	440.90	0.90
35	440.96	-2.66	440.97	0.97
36	440.96	-2.78	440.79	0.79
37	440.61	-3.20	439.90	-0.10
38	440.96	-3.30	440.10	0.10
39	440.96	-3.55	439.79	-0.21
40	436.52	0.60	439.67	-0.33

Table 6.2 *Experiments using conveyor sensing on the calibration block*

Block Pass No.	Angel of block to conveyor 'ϕ' (degrees)	True Length 'TL' using ('*Length_Correct* = 4.17 mm) (mm)	Block Height 'H' (mm)	Block width 'W' (mm)
1	-3.55	440.96	214.48	99.03
2	2.51	439.10	214.01	100.40
3	-2.16	439.96	213.93	99.86
4	2.48	439.14	214.08	100.56
5	-0.10	440.86	213.72	101.08
6	0.08	440.20	213.74	101.12
7	0.99	439.69	213.78	100.61
8	-1.83	440.45	213.65	99.65
9	3.01	439.11	213.81	100.00
10	-2.46	440.91	213.81	99.42
11	0.30	439.49	214.04	100.82
12	4.23	440.04	213.54	99.10
13	4.31	440.26	213.66	99.44
14	-1.65	440.38	214.66	99.89
15	0.22	439.97	214.30	99.92
16	-2.66	440.97	214.73	99.83
17	1.99	439.15	214.58	100.73
18	0.50	438.80	214.35	100.90
19	-3.20	439.93	215.06	99.45
20	-2.37	439.67	214.47	99.83
21	2.27	439.47	214.73	99.73
22	2.26	439.13	214.59	100.12
23	0.90	439.50	214.54	101.11
24	-0.34	441.12	214.91	100.80
25	-1.59	440.45	214.75	100.57
26	-0.58	440.71	214.45	100.91
27	-1.37	440.47	214.72	99.79
28	1.05	439.60	214.67	100.69
29	0.01	440.68	215.21	100.44
30	2.49	439.44	214.94	101.01
31	-1.91	439.97	215.11	100.46
32	0.58	439.68	214.86	101.24
33	0.15	440.43	214.97	100.59
34	-1.48	440.63	214.56	100.73

Table 6.3 *Results of experiments on block dimension measurements*

Statistical results	length 'TL' results (mm)	Difference in 'TL' from actual length (mm)	height 'H' results (mm)	Difference in 'H' from actual height (mm)	Width 'W' results (mm)	Difference in 'W' from actual weight (mm)
Measurement	440.0	------	214.5	-----	99.92	-----
Average reading	440.1	0.1	214.4	0.1	100.3	0.4
Maximum reading	441.1	1.1	215.2	0.7	101.2	1.3
Minimum reading	438.8	1.2	213.5	1.0	99.0	0.9
Standard deviation of readings	0.7	-----	0.5	-----	0.6	-----

6.2.2 Gripper to block alignment

A method of determining the alignment of the block to the robot gripper was developed and used to make the final adjustments to the block pick-up position, making it safe to grip. The procedure relies on the horizontal ultrasonic sensor 'H_sensor' mounted on the gripper (figure 2.4.1). Using this sensor the robot measures the horizontal angle between the gripper to the block 'β', which it then applies to an adjustment on the yaw-axis, in the *'block_pick_TCP'* co-ordinate system. A detailed description of this procedure was covered in section 2.4.2 A test was carried out to find the maximum offset angle the gripper could be at and still manage to complete the procedure successfully. The accuracy of the angle determined was also worked out.

At commencement, the robot gripper was positioned at a set point, facing the block with its mid-point near the mid-point of the block. The block was then rotated at different angles about its centre, and at each angle the procedure to measure that offset angle was carried out. To accurately rotate the block, a scaled drawing of the block was prepared using AutoCAD. This offset the block from its mid-point for a number of angles up to 9° (figure 6.1). This was placed on the ground, with the zero angle of the drawing aligned accurately to the y-axis of the robot. At the start, the block was placed at the zero angle, and the robot's yaw-angle set to zero (i.e. angle $\beta = 0^{\circ}$), whilst making sure the gripper was at a safe distance from the block, yet still in sensing range.

Table 6.4 gives the results of the experiment, indicating that the gripper can accurately detect the angle of block up to an offset angle of 9°. The accuracy of the angle measure was on average $\pm0.8^{\circ}$, with a maximum error of $\pm1^{\circ}$. This level of accuracy in the measurement is vital when gripping a block, as the gripper opening is small compared to the block width used (figure 2.4.5). The calculations in section 2.4.2 show that the maximum safe angle offset of the gripper to the block, was 3.26°. This gives a 0.5 mm clear way of the gripper edges to the block, making it safe to go down and grip the block. This value was calculated assuming the gripper centre is at the mid point of the block. If, for example, the gripper centre was 1.5 mm offset at either sides of the gripper fork, from the middle of the block, the error in the angle 'β' could be up to 2°, and still leave a clearance of 0.5 mm.

Measurement of the angle of the block to the gripper 'β' guarantees alignment of the gripper to the block to ±1°. This, in turn, guarantees the accurate adjustments needed to position the gripper centre on top of the block centre (i.e. to ensure safety in the block gripping action).

Table 6.4 *Adjusting gripper to block experiment results*

Gripper to block angle 'β' (degrees)	**Left side H_sensor reading (mm)**	**Right side H_sensor reading (mm)**	**Resulting Angle measure (degrees)**	**Error in estimate (degrees)**
0	122	124	0.4	0.4
3	113	134	4.0	1.0
6	104	140	6.8	0.8
9	97	149	9.8	0.8

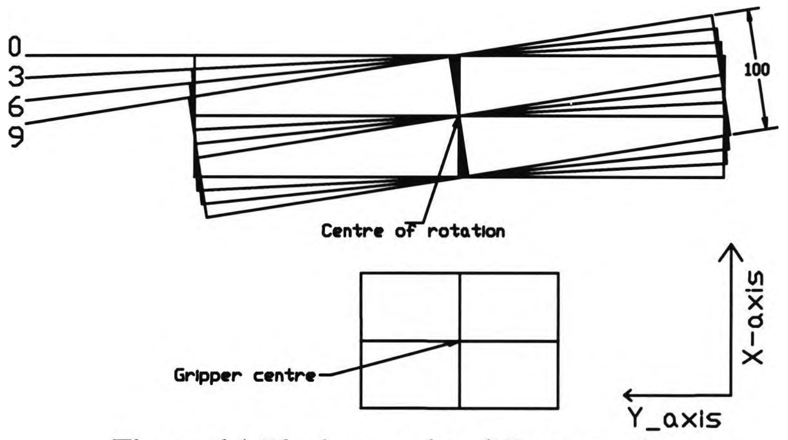


Figure 6.1 *Block rotated at different angles*

6.2.3 Accuracy of block pick-up

The complete process developed to calculate the SBPP on the conveyor, using the BPP, was tested. These makes up stages 1, 2, and 3 of the building process, shown in figure 2.2 The BPP used was calculated as a result of locating the conveyor at the start of these experiments (stage 1 of the building process). The 'C' programme '*targ_fun.mak*', described in chapter 5, was used for this purpose. The rate of success in locating and gripping the blocks, was determined, using the complete process, which is completed in two steps. The first step uses the conveyor sensor information, from which the block dimensions and the BCP are calculated and used to find the NBPP (stage 2 of the building process). These processes are described in sections 2.3.1.2 and 2.3.1.3 consecutively. The second step (stage 2 of the building process), is to use the gripper sensors to make the final adjustments, which arrive at the SBPP (described in section 2.4.2). The 'C' programme '*Block_pick.mak*' was used for these experiments, which integrates the complete process.

For this experiment, the calibration block was passed through the different sensing stages on the conveyor, 12 times. Each time the block file was captured and analysed to establish the SBPP. The block was gripped safely each time.

The maximum angle adjustment to angle 'ϕ', which was needed for aligning the gripper to the block, was found to be $< \pm 1^\circ$. This proves that the determined angle of the block to the conveyor ϕ, found using the conveyor sensor information, was accurate to $\pm 1^\circ$.

6.3 Conclusion

The use of the conveyor, with its various sensing stages, for supplying material was found to be effective. Potentially, it allows concurrent processing of the different tasks of the building process. Due to the fact that the masonry units vary in dimension (table A.1), this prompted the need for an accurate measurements of the block dimension in order to make the run-time adjustments to the 'theoretical task'. The conveyor sensing operation, and the method developed to analyse the block sensor profile, proved to be accurate in measuring the block dimensions, and locating the block ready for pick-up. On average, the accuracy of the block dimension measurements were ±0.1 mm for the length, ±0.1 mm for the height, and ±0.4 mm for the width. The maximum errors were found to be ±1.2 mm for the length, ±1.0 mm for the height, and ±1.3 mm for the width. These errors were not found to effect the success of the block pick-up process.

The use of the gripper sensors and the methods devised to make the final adjustments to the block pick-up position, proved to be vital and necessary to secure the safety of the block pick-up operation. The process developed to align the block to the gripper was found to be accurate to $\pm1^{\circ}$. This ensures the alignment of the gripper to block with that same accuracy.

From the experiments in this chapter, it was noted that the maximum angle adjustment needed to correct the concluded angle ϕ, was less than $\pm1^{\circ}$. As the same BPP was used throughout these experiments (i.e. giving the same conveyor location), this eliminated any error occurring from the determination of the conveyor position (using the target). Thus it can be assumed that the error arose from the determination of the angle ϕ, using the conveyor block profile information. Experiments in chapter 5 proved that the maximum possible error in determining the conveyor orientation was 5.7°. Combining this error with the error in determining the angle of the block to the conveyor ϕ, makes a maximum gripper to block angle 'β' to be $\pm6.7^{\circ}$. The method developed to align the gripper to the block proved accurate, even when the angle 'β' was as large as 9°. This makes the error of $\pm6.7^{\circ}$, mentioned above, to be of no real significant to the success of the block pick-up operation.

In conclusion, the end result of the various real-time sensing procedures enabling the gripping of the block from the conveyor, in a safe and secure fashion, proved satisfactory, with a success rate of 100%.

Using a RBES, which added intelligence to the robot, helped to confirm and check each step of the process tested in this chapter. The expert system, described in chapter 9, integrates the real-time sensing and the robot functionality with various rules. This enables the robot to anticipate different situations and act on them accordingly, to conduct its tasks in the safe manner.

Chapter 7: Block Manipulation Experiments

7.1 Introduction

The method developed to safely pick-up a block from the conveyor, using the conveyor and gripper sensors, described in chapters 2.3.1 and 2.4, was tested and found to be effective (chapter 6). The rationale for this is that it is impossible to guarantee the robot picking a block in the exact same way each time. This prompted the need to confirm the position of the block in the gripper, in order to make the necessary run-time adjustments for the procedures of applying mortar on the block and lay the block in its pre-defined position, according to stages 5 and 6 of the building process (figure 2.2). A method was developed and described in chapter 2.5, which uses a laser profiler (figure 2.5.1) to determine the block's position in the gripper. This chapter determines the accuracy of this method.

Two types of experiments were conducted. The first was to find the accuracy of the laser profiler's measurements of the various lengths in the gripper and block profile. These measurements are used in determining the offset values for the *Mortar_dispensing_TCP*'s position (figure 2.7) i.e. the reference point 'P'. This particular TCP will be used in the mortar dispensing process, as well as the block laying process, according to stages 5 and 6 of the building process (figure 2.1). Consequently, the second type of experiment conducted was to test the accuracy of the block laying process. This was done using the block and gripper measurements from the laser profiler, in the calculations of the block to gripper geometry, as developed in chapter 2.5. For these tests, a pre-defined location was used to lay the blocks, as there was not enough time to test a whole building project. Experiments were carried out using the calibration block (figure 2.5.3) and 'Celcon' blocks, gripped in a variety of ways to test different situations, for example, varying the angle to the gripper (figure 2.5.2). For each test, the laser profiler scans the block a number of times and the resulting data is then processed using the method described in chapter 2.5.3. The final result for each test is then taken as the average of six scans.

7.2 Experiments in laser profiler measurement

The use of the laser profiler for scanning the gripped block, to determine the way the block was picked, was tested and its accuracy determined. The block was deliberately gripped at different angles (figure 2.5.2), different heights, and different central locations relative to the gripper. For each test, a calliper micrometer was used to measure the block's position in the gripper, which was then compared with the laser profile's measurements. The robot roll-axis was set at zero position, and the laser profiler horizontal, making the angle, about the roll axis, between the laser strike and the gripper 'α' to be 0° (figure 2.5.7-2.5.8). The gripper was sent repeatedly to the same location, positioned normal to the laser profiler, with its mid point aligned to the middle range of the laser scanner axis, and within the range of the laser displacement sensor (figure 2.5.3). Its profile axis range is limited to 300 mm, which limited the possible length of block material. As well as the special calibration block, built from wood, cut 'Celcon' blocks were prepared to 250 mm length.

Experiments were carried out with the calibration block picked horizontally, making the angle of the block to the gripper 'γ' as close as possible to 0° (figure 2.5.6). This was done by finding a location on the ground which was level, making sure the robot roll-axis was at zero position (i.e. horizontal), and then checking the transducer readings before gripping the block. As a result of the gripping operation, the gripped block's angle to the gripper varied slightly. This experiment was repeated a number of times, with each time the horizontal distance of the block in the gripper was varied as well as the location of the centre of the block in the gripper. Other experiments were carried out by placing the blocks in the gripper at different angles. The results of these experiments are described in section 7.4.2.

7.3 Block placing experiments

Experiments were carried out to determine the accuracy achieved in the block laying process. The Calibration block and 'Celcon' blocks were both used. Using the processed data of the laser profiler, and the displacement transducers on the gripper, adjustments are made to the reference point 'P' on the block (figure 2.5.4). In theory, this reference point ‘P’ is the ‘*Mortar_dispensing_TCP’* , described in chapter 2.2 (figure 2.7), which is used for the mortar dispensing move described in chapter 2.6. After this, it is used to adjust the pre-determined block laying position. A description of these process, how they are related, and their order in the building process, is illustrated in 2.2.

Due to time limitations and the limited mortar dispensing pump availability, it was not possible to test the continuous process of picking-up a block, dispensing mortar on it, and laying the block. The experiments carried out were restricted to the process of picking-up a block, scanning the block using the laser profiler, making the necessary adjustments, and then laying the block. This meant there was no mortar on the block, making the reference point ‘P’ directly on the block i.e. t_b and t_p were equal to zero (figure 7.1).

The accuracy of the values of B and B_2 from the laser profile’s readings, effected the accuracy of the whole process. It became evident that the problem was due to the poor resolution on the encoder of the motion axis of the laser profiler. Their measurements had an average error of ±0.58 mm and a maximum error of around ±1.3 mm (table 7.3). Furthermore, the laser profiler’s measurements of the angle, about the roll axis, between the laser strike and the gripper α, had an average error of $\pm7.7^{\circ}$. The magnitude of these errors, made the results of these variables extremely unreliable to use in the calculation of the reference point ‘P’. For the purpose of our experiment, the gripper was aligned to the laser profiler. This allowed for the assumption of the angle α to be 0°, which meant the actual measurements of (9.7 mm) could be used for the variables B and B_2 (figure 2.5.11). The horizontal position of the gripper to the laser profiler was set making the variable E_2 equal 31.1 mm (figure 2.5.8). For each experiment the result of 6 block scans was averaged, and used to find the a1 and a3 offset for the reference point ‘P’. This was calculated using the ‘C’ programme ‘*block_lay.c*’, which implemented the method described in chapter 2.5.3.

For the block placing location, a target of the block's length and width was drawn to scale and placed on the ground in the robot work space, in the orientation shown in figure 7.2. The reference point 'P', shown in figure 8.1, was used as the TCP in these experiments, which was adjusted in each case by calculating the 'a_1' and 'a_3' offset values (refer to section 2.5.3.1). First the calibration block was gripped at its middle, making the measurements of A and A_2 equal to 43 mm. The values of 'a_1', and 'a_3' for this case were then concluded. a_3 was calculated, using the value of O_2 calculated from the displacement sensor reading and the measurement of the height of the block '*h*' in equation 7.1 (figures 7.1 and 2.5.9):

$$a_3 = -h - (O_2 + R) \qquad \text{eqn 7.1}$$

, *R* being the distance from the centre of rotation of the robot to the bottom of the gripper (figure 2.5.10 and 7.1).

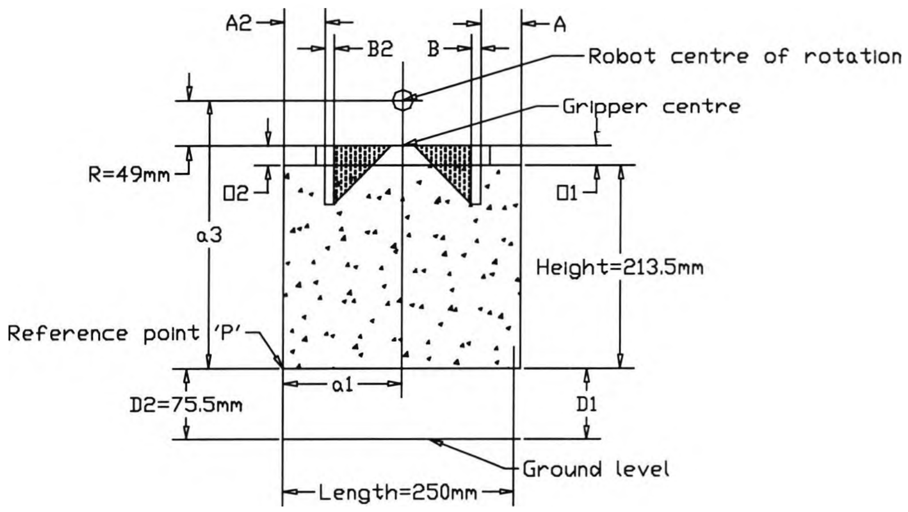


Figure 7.1 *Example of a block placing experiment set-up*

Using the 'a_1' and 'a_3' values determined with the 'C' programme, '*tcp2.c*', the robot was sent to a position on top of the target, with the gripped block at a distance of 75.5 mm from the ground. The position of the robot, relative to the reference point 'P', was then noted and used as the reference position for all the following moves. The steps, set out below, were then repeated for each different gripping of the block.

(i)- First step: for each test, the block was gripped differently and sensor measurements, made of the variables which determine the way the block is picked, used to calculated the 'a_1' and 'a_3' offsets of the reference point 'P'.

(ii)- Second step: the 'C' programme '*tcp2.c*' was used, with the new adjusted 'a_1' and 'a_3' values from step 1, to move the reference point 'P' to the location mentioned earlier. The roll angle is adjusted using the angle 'γ' calculated using equation 2.5.4.

(iii)- Third step: the offset of the block to the target are noted for each case (figure 7.2).

The results of these experiments are discussed in section 7.4.2.

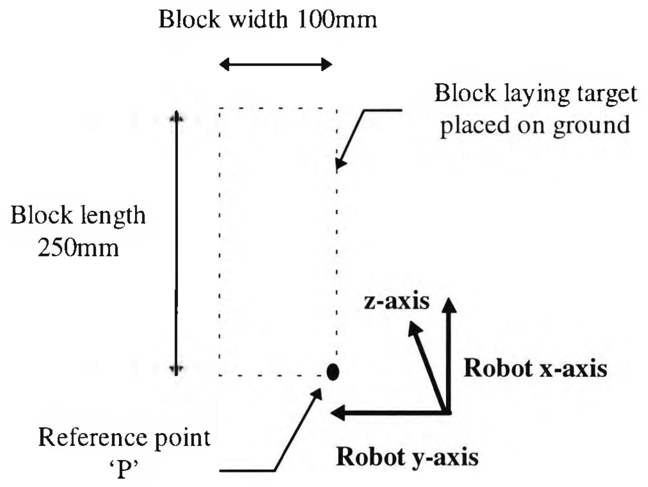


Figure 7.2 *Target positioning for block placing experiment set-up*

7.4 Results of block manipulation experiments

7.4.1 Accuracy of laser profiler measurements

The results of the six experiments are covered in this section. Experiments 1-3 were of a block picked horizontally, and experiments 4-6 of a block picked at an inclination. In each case, the laser profile was processed using the method described in section 2.5.2. For each scan, the edges were determined (figure 2.5.6). Using these edges, the various parameters are determined according to table 2.5.1. The complete results of the edge values, and their corresponding variable calculations, are included in appendix E.1. The average values of these measurements for each experiment (table 7.1) are used to calculate the necessary parameters. Each result was compared to its direct measurement, determined using a calliper micrometer. Table 7.2 shows a comparison between these measurements of A, and A_2 and those determined by the laser profile. The results show a root mean error of 1 mm, a maximum error of ±1.6 mm, with a percentage error of around 4%. Table 7.4 shows the error in the length results, where the laser profile results of the length was calculated using equation 2.5.1. These results show a root mean error of 0.75 mm, a maximum negative error of 1.5 mm, and a percentage error of 0.6%.

The angle, about the roll axis, between the laser strike and the gripper α is set to 0°, allowing the various gripper measurement results (W_2, B, B_2,) to be compared by direct measurement (illustrated in figure 2.5.11). Table 7.5 shows the results of comparing the direct W_2 measurement, to that from the laser profile, calculated using equation 7.2 (figure 2.5.8). The root mean error in this measurement was found to be 1.52 mm, with a maximum error of 1.6 mm, making this measurement unreliable. The effect of this error is apparent in the laser profile's measurements of angle α, calculated using equation 2.5.5. This is shown in table 7.5, to be as height as 13°, with a root mean error of 7.7°. Table 7.3 gives a comparison of the direct measurements of B, and B_2 with their values measured by the laser profile. The results show a root mean error of 0.58 mm, a maximum error of ± 1.3 mm, and a percentage error of 14%.

$$W_2 = (C_2 + D + C)/\cos(\alpha) \qquad \text{eqn 7.2}$$

The O_1 and O_2 variables are 164 mm apart, which is the direct measure of W_2. The direct values for these variables are used to calculate the angle of the block to the gripper 'γ', and compare it with the results calculated using the sensed readings. Both measurements of the angle are calculated using equation 2.5.4, and the results shown table 7.6. From these it can be seen that the measurements of the displacement transducers are extremely accurate, resulting in a mean square error of only less than 0.1°, and a maximum error of 0.26°.

Table 7.1 *Results of each experiment calculated from mean of 6 block scan results*

Average Scan results	A_2 (mm)	A (mm)	B_2 (mm)	B (mm)	C_2 (mm)	C (mm)	D (mm)
Experiment 1	46.3	39.7	10.3	10.6	32.6	31.2	79.4
Experiment 2	27.4	59.4	9.6	9.1	32.7	30.3	80.0
Experiment 3	42.2	43.8	10.3	9.7	34.0	31.2	77.7
Experiment 4	42.2	44.3	9.2	8.4	37.3	35.9	71.9
Experiment 5	49.4	36.4	10.0	9.1	32.7	31.0	81.1
Experiment 6	57.9	29.6	10.3	10.6	20.8	18.4	101.7

Table 7.2 *A and A_2 direct and sensed values*

Experiment number	Variable name	Direct measure (mm)	Result from profile data (mm)	Mean error of scans (mm)
1	A_2	45.5	46.3	0.8
1	A	41.0	39.7	-1.3
2	A_2	26.2	27.4	1.2
2	A	59.8	59.4	-0.4
3	A_2	40.6	42.2	1.6
3	A	45.4	43.8	-1.6
4	A_2	41.2	42.2	1.0
4	A	44.8	44.3	-0.5
5	A_2	48.2	49.4	1.2
5	A	38.0	36.4	-1.6
6	A_2	57.0	57.9	0.9
6	A	29.3	29.6	0.3
		The root mean	square error	**1.03**
		maximum	-ve error	**-1.6**
		maximum	+ve error	**1.6**

Table 7.3 *B and B_2 direct and sensed values*

Experiment number	Variable name	Direct measure (mm)	Result from profile data (mm)	Mean error of scans (mm)
1	B_2	9.7	10.3	0.6
1	B	9.7	10.6	0.9
2	B_2	9.7	9.6	-0.1
2	B	9.7	9.1	-0.6
3	B_2	9.7	10.3	0.6
3	B	9.7	9.7	0.0
4	B_2	9.7	9.2	-0.5
4	B	9.7	8.4	-1.3
5	B_2	9.7	10.0	0.3
5	B	9.7	9.1	-0.6
6	B_2	9.7	10.3	0.6
6	B	9.7	10.6	0.9
		The root mean	square error	**0.58**
		maximum	-ve error	**-1.3**
		maximum	+ve error	**0.9**

Table 7.4 *Block length 'l' direct and sensed values*

Experiment number	Variable name	Direct measure (mm)	Result from profile data (mm)	Mean error of scans (mm)
1	*l*	250	247.7	0.1
2	*l*	250	248.5	-1.5
3	*l*	250	248.9	-1.1
4	*l*	250	249.2	-0.8
5	*l*	250	249.8	-0.3
6	*l*	250	249.2	-0.7
		The root mean	square error	**0.75**
		maximum	-ve error	**-1.5**

Table 7.5 *Gripper width 'W_2' ,and 'α' angle's direct and sensed values*

Experiment number	Direct (W_2) measure (mm)	Result from (W_2) profile data (mm)	Mean error (mm)	Direct (α) measure (degrees)	Result from (α) profile data (degrees)	Mean error (degrees)
1	144.6	143.2	-1.44	0	8.10	8.10
2	144.6	143.0	-1.61	0	8.56	8.56
3	144.6	142.9	-1.65	0	8.66	8.66
4	144.6	145.1	0.48	0	4.66	4.66
5	144.6	144.8	0.26	0	3.43	3.43
6	144.6	140.9	-3.7	0	12.99	12.99
	The root	square error	**1.52**	The root	square error	**7.7**
	maximum	-ve error	**-1.61**	maximum	error	**12.99**
	maximum	+ve error	**0.48**			

Table 7.6 *Block to gripper 'γ' angle's direct and sensed values*

Experiment number	Variable name	Direct measure (degrees)	Result from sensor data (degrees)	Mean error (degrees)
1	γ	0.25	0.23	-0.02
2	γ	0.21	0.17	-0.04
3	γ	0.29	0.15	-0.14
4	γ	-1.61	-1.59	0.02
5	γ	-3.97	-4.23	-0.26
6	γ	4.89	4.92	0.03
		The root mean	square error	**0.09**
		maximum mean	-ve error	**-0.26**
		maximum mean	+ve error	**0.03**

7.4.2 Results of block placing experiment

The results for the experiments described in section 7.3 are reviewed in this section. This section covers the experiments carried out using the calibration block and a 'Celcon' type block. The calibration block used in these experiments is described in chapter 2.5.1 and shown in figure 2.5.3. This block, and a 'Celcon' block cut to a length of 250 mm, were both used. These blocks were laid in the direction of the robots x-axis. For each test the values of the variables A, A_2, O_1, and O_2 were measured directly, and their results compared to their corresponding sensor derived values. An assessment of the errors from the sensor readings was made, and their effect on the accuracy of the process of laying a block. The error in the block laying position were measured directly, and the results shown in table 7.9. D_1 and D_2 are the vertical offset of the block from the ground (figure 7.1). The error in the x and y axis offsets are determined relative to the target location for the block laying position shown in figure 7.2.

7.4.2.1 Tests using calibration block

These experiment were carried out using the calibration block. The results in table 7.7 indicate the error in the A, and A_2 values have average values of ±1.05 mm and ±1.4 mm respectively, and maximum values of +2.29 mm and +2.67 mm respectively. This error directly affects the calculated a_1 and a_3 offsets of the reference point 'P' (refer to equations 2.5.21 and 2.5.22), hence the adjustments to the position when laying the blocks, determined using these offsets. The error in the A_2 values are influenced more by the a_1 value, however, resulting in a lateral average error of 1.6 mm and maximum 3 mm (table 7.9). In the experiment this lateral error was in the x-axis direction. The error in the horizontal angle used to lay the block was a maximum of 0.57° (table 7.9). This error can be directly linked to the errors in the O_1 and O_2 values i.e. the error in the γ angle measurement (table 7.8). This caused an average vertical offset of around 0.6 mm with a maximum error of 1.0 mm. The errors in the offsets found for the y-axis direction were not the result of any errors in measurements or calculations. A closer examination of the way the blocks were gripped showed, that when the gripper was not central to the block, the block was caused to slant. This error could be corrected if the robot had a pitch axis, by scanning the block side and correcting for it. The complete results of the A and A_2 measurements are included in appendix E.2.2.

Table 7.7 *Error in* (A,A_2) *values*

Experiment no.	**Direct values**		**Laser scanned**		**Error in results**	
	A	**A_2 (mm)**	**A**	**A_2 (mm)**	**A**	**A_2 (mm)**
1	43.0	43.0	43.73	45.67	0.73	2.67
2	38.3	47.7	37.47	48.74	-0.83	1.04
3	46.0	40.5	47.07	42.59	1.07	2.09
4	47.7	37.9	48.18	37.61	0.48	-0.29
5	42.5	44.5	43.90	46.04	1.40	1.54
6	51.7	36.0	53.40	37.93	1.70	1.93
7	28.0	58.6	30.29	60.20	2.29	1.60
8	38.0	48.2	38.21	49.97	0.21	1.77
9	42.0	44.5	43.71	46.07	1.71	1.57
10	57.9	29.5	57.97	29.64	0.07	0.14
Root mean square error					**1.05**	**1.41**
Maximum +ve error					**2.29**	**2.67**
Maximum -ve error					**-0.83**	**-0.26**

Table 7.8 *Error in the block to gripper angle measurements*

Experiment no.	**Direct values O (mm)**	**O_2**	**Sensor Readings O (mm)**	**O_2**	**Error in results O (mm)**	**O_2**	**γ Angle measurement using sensor (degrees)**	**Error in γ angle measurement (degrees)**
1	18.3	17.8	18.5	17.8	0.20	0	0.22	0.05
2	10.8	9.70	10.87	9.72	0.07	0.02	0.34	-0.04
3	23.4	11.8	23.78	10.92	0.38	-0.88	3.99	-0.05
4	16.1	22.0	16.72	22.58	0.62	0.58	-1.85	-0.20
5	4.60	6.50	4.96	6.10	0.36	-0.42	-0.35	-0.31
6	21.3	5.30	21.89	4.0	0.59	-1.30	5.55	-0.02
7	6.7	14.8	6.47	15.38	-0.23	0.58	-2.77	-0.06
8	16.9	12.1	16.6	11.27	-0.32	-0.83	1.65	-0.03
9	8.20	10.8	8.3	11.03	0.1	0.23	-0.85	-0.06
10	15.4	6.10	15.63	5.98	0.23	-0.12	3.0	-0.25
Root mean square error					**0.31**	**0.49**	**-----**	**0.11**
Maximum +ve error					**0.62**	**0.58**		**0.05**
Maximum -ve error					**-0.32**	**-1.3**		**-0.31**

Table 7.9 *Error in block laying*

Experiment no.	**D_1 (mm)**	**D_2 (mm)**	**Angle of block to ground (degrees)**	**Z-axis offset from mid block (mm)**	**x-axis offset (mm)**	**y-axis offset (mm)**
1	75	74	0.23	1.0	3.0	0
2	75	74	0.23	1.0	0.5	-0.5
3	76.5	74	0.57	0.25	1.5	0
4	75.5	75	0.12	0.25	0.5	0
5	74.5	75	-0.12	0.75	1.5	-2
6	74.3	76	-0.39	0.35	2.5	0
7	75	75	0	0.5	2.5	0
8	74	75	-0.23	1.0	0.5	-3.0
9	74	76	-0.46	0.5	2.0	0
10	73.5	75.5	-0.46	1.0	1.8	0
Root mean square error			**0.28**	**0.66**	**1.63**	**0.55**
Maximum +ve error			**0.57**	**1.0**	**3.0**	**0**
Maximum -ve error			**-0.46**	**none**	**none**	**-3.0**

7.4.2.2 Tests using 'Celcon' blocks

These experiments were carried out using a 'Celcon' block. The results in table 7.10 indicate the inaccuracy in the measurements A and A_2 values with the laser profiler. The complete results of the A and A_2 measurements are included in appendix E.2.2. The maximum error was found to be 3.5 mm, with an average error of 2.9 mm. These errors are higher than those found in the experiments done using the calibration wooden block (section 7.4.2.1), this is because the edges of these blocks are not as well defined as the wooden block. These errors directly effect the offset when placing the blocks, this was found to be on average of 1 mm with a maximum of 3 mm in the x-axis direction. Offset errors on the y-axis were found to be similar magnitude.

Table 7.10 *Error in (A,A_2) direct and sensed values*

Experiment no.	Direct values A	A_2 (mm)	Laser scanned A	A_2 (mm)	Error in results A	A_2 (mm)
1	32	54	33.55	57.5	1.55	3.5
2	31	54.5	32.81	57.88	1.81	3.38
3	59.5	27.5	60.86	28.80	1.36	1.3
4	35	50.5	35.70	53.96	0.7	3.46
Root mean square error					**1.355**	**2.91**
Maximum +ve error					**1.86**	**3.50**

Table 7.11 *Error in the block to gripper angle measurements*

Experiment no.	Direct values O (mm)	O_2	Sensor Readings O (mm)	O_2	Error in results O (mm)	O_2	γ Angle measurement using sensor (degrees)	Error in γ angle measurement (degrees)
1	23.2	16.4	23.21	16.44	0.01	0.04	2.11	-0.26
2	16.4	11.6	17.12	10.84	0.72	-0.76	1.96	0.28
3	15	9.3	15.17	8.05	0.17	-1.25	2.22	0.23
4	11.8	12.4	11.69	12.47	-0.11	0.07	-0.24	0.04
Root mean square error					**0.25**	**0.53**	-----	**0.20**
Maximum +ve error					**0.72**	**0.07**	-----	**0.28**
Maximum -ve error					**-0.11**	**-1.25**	-----	**-0.26**

Table 7.12 *Error in block laying*

Experiment no.	D_1 (mm)	D_2 (mm)	Angle of block to ground (degrees)	Z-axis offset from mid block (mm)	x-axis offset (mm)	y-axis offset (mm)
1	74	71.6	0.55	2.7	0.5	0
2	74.5	74	0.12	1.25	0.5	0
3	75	72	0.69	2.0	3.0	-1.0
4	75	73	0.46	1.5	0	2
Root mean square error			**0.46**	**1.86**	**1.00**	**0.75**
Maximum +ve error			**0.69**	**2.7**	**3.0**	**2.0**
Maximum -ve error			----	---		**-1.0**

7.5 Conclusion

The LADS device used in the laser profile's measurement is accurate (0.01 mm) in the ranging measurement for the different depths of the gripper, block, and target triangles. On the other hand, the imprecision of the LADS's motion axis position caused significant errors in-spite of the application of an edge correction. The errors in the corresponding positional value of detected edges was about ±1 mm, mainly on account of imprecision in the positional encoder of the profile motion axis.

The block length measurements were found to have an average error of 0.75 mm, compared to the measurement carried out using the conveyor sensing (table 6.3), which were found to be about 0.1 mm (results of the experiments in chapter 6). The measurement the angle, about the roll axis, between the laser strike and the gripper 'α' was found to have an average error of around 8°, and a maximum error of around 13°. These values are extremely unreliable, therefore this angle was not used when calculating adjustments for the block laying operation. This error was due to the inaccurate laser profiler's measurements of C, D, and C_2 which resulted in an accumulated error of the edge detection.

The lateral error in the block laying operation was found to be, on average, ±1.63 mm with a maximum value of 3 mm. This can be directly linked to the laser profiler's measurements of the A_2 variable used in calculations to adjustment the block laying position. This error was found to be, on average, 1 mm and 1.4 mm, with a maximum value of 2.67 mm, which is of a similar magnitude to that of the block laying offset error. The errors in the horizontal angle for the block laying, was found to be a maximum of 0.57°. This error can be linked to the errors in the O_1 and O_2 values (i.e. the error in the γ angle measurement). The error in the block laying horizontal angle caused an average vertical offset of around 1.8 mm, and a maximum error of 2.7 mm. The offsets found in the y-axis where mostly due to the middle of the block being pushed to one side, causing it to slant during the gripping action.

Error in the A_2 measurements with the ‘Celcon’ block were found to be slightly higher than those found using the calibration block. This was due to the nature of the edges of the ‘Celcon’ block, which are not as well defined as those of the calibration block.

Chapter 8: Experiments in Mortar Dispensing

8.1 Introduction

Following a review of other work in masonry automation with wet bonding (section 8.2), this chapter covers the experiments carried out to test the automation of the mortar dispensing process. One of the main requirements is to have controlled dispensing. The main factors considered in these investigations were the choice of pump type, mortar mix material, the dispensing nozzle shape and design, the dispensing rate, the angle and offset of the nozzle, and the masonry unit preparation. These factors have been investigated by various studies at City University. The experiments of mortar dispensing described in this chapter are built on the findings of these studies.

Section 8.2 covers a review of the different methods and materials used for bonding in the various other research projects. For this research, options for the mortar dispensing process were discussed in chapter 2.6.3. The options in using the robot in applying the mortar to the block/bricks and presenting and manipulating the masonry units under the nozzle were investigated. A special move, designed for dispensing mortar continuously on the edges of the block (described in chapter 2.6.3.1) was tested. The results of these experiments are incomplete, due to the fact that the pump was only available for a limited amount of time, during which the robot's z-axis motor failed and had to be serviced. Only first trials of dispensing and building small assemblies were thus possible. This resulted in it not being possible to integrate this stage of the building process (stage 5) with the stages before and after it (figure 2.2).

8.2 Review

In the various research projects carried out in automated masonry, world wide, a variety of different materials and bonding methods are apparent. This work tends to reflect the different national building characteristics. Different prototypes and solutions have been suggested.

A robot design at the Massachusetts Institute of Technology 'Blockbot' (Solcum *et al.*, 1988) used a method of dry-stacking standard concrete blocks, which are later surface reinforced using a spray-applied fibreglass-reinforced bonding cement on both wall faces. Only a small partial prototype of the designed system was built. In Germany, there have been and are existing a number of ongoing research projects. Early work (Anliker *et al.*, 1988) involved a system for assembling prefabricated walls using standard mortar and special masonry blocks and the walls surfaces being covered. In that project, only the mortar supply is fully automated. Other German projects are covered later in this section. In Finland (Lehtinen *et al.*, 1989) an automated system has been designed that uses lime bricks with strict minimum dimensional tolerances, which allows the use of thin bonds (<3 mm). Here, the traditional UK mortar is replaced by a glue-like material placed in a bowl that is designed to allow the two sides of the bricks to be dipped, resulting in an even spread of mortar. The bricks are then grooved to remove surplus mortar. In Russia, part of the design of a robotic-complex for brick-laying (Malinovsky *et al.*, 1990) is the application of a layer of mortar which is extruded on a continuous line of bricks delivered by a conveyor. Vertical joints are manually injected after a course of bricks are placed. The bricks are presented under the extruder at constant speed, with the thickness of the mortar pre-set. Results show that by adding plasticizer to the mortar an enhanced bond strength can be achieved. A 10% to 15% increase in strength, compared to units placed by hand, was reported.

In Israel (Rosenfeld *et al.*, 1990 and 1991), a robot developed for interior finishing used light-weight large gypsum blocks, which are placed dry and then have a fibre-reinforced plasters applied to the wall. Work at City University (Chamberlain *et al.*, 1991) carried out various experiments on mortar dispensing to try and achieved a mix that was pumpable as well as workable. At the early stages of the project a mobile, Metrix pressure pointing mixer, which is an auger-type pump, was used to mix and deliver the mortar. The operational tolerance requirements to achieve the best bond were determined, which allow the required accuracy of the robot to be defined. These requirements included the thickness and eveness of the mortar bed in relation to the dimensional tolerance of the masonry units used. Later work (Chamberlain *et al.*, 1992), was done using a peristaltic type 'smart-pump' that could be controlled from the main PC via a serial link. This work included experiments carried out at City University (Ahmed, 1993) and (Charles, 1993). These are covered in section 8.3. They form the background of the work carried out by the author.

A project in Germany (Bohm *et al.*, 1991) developed a partially mechanised solution using a clamp as part of a working platform that assists the human worker in handling blocks (figure 8.1). The worker dips the blocks into the mortar and a vibrator, built into it, taps the blocks onto the layer of mortar. The mortar pump is suggested to be placed on the platform and is connected to the mortar sledge, thus providing ready-mix mortar continuously. Another project in Luxembourg called FAMOS brick (Wurth,1992), which is semi-automated, assists the human worker, who places the bricks under the mortar dispenser. This involves use of traditional mortar.

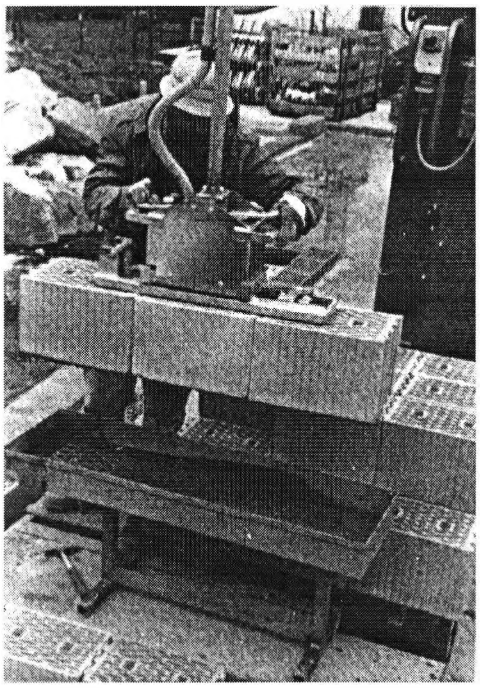


Figure 8.1 *Dipping blocks into mortar before setting them onto the wall (Bohm et al., 1991)*

In the United States (Bernold *et al.*, 1992), a system was developed that emphasised the need for a computer integrated control strategy for the delivery of controlled amounts of mortar. In this work, it was suggested that to compensate for the difference in height of masonry units, the current layer of the blocks should be levelled and thus absorbed the level variations in the mortar joins below that layer. A force sensor on the gripper is used to produce the required pressure when placing the bricks, thus achieving the required bond strength. There is, however, a contradiction between the need to be level and possible settlement under the applied pressure. Further work on the suggested system was covered previously (Atobelli *et al.*, 1993). The general concept is that mortar is mixed on demand and delivered fresh, thus eliminating the problems of clogging (figure 8.2). The experiments to test the bond strength and accuracy of robotically placed bricks were conducted using a scaled prototype of the system. To control the amounts of mortar being pumped, the mortar was placed in a vertical pipe section which was fitted to a flat head, milled to shape, to produce a smooth thin flow of mortar making up a displacement type pump. A robot pushed the plunger at constant speed, which forced the mortar out of the head while a conveyor, moving at constant speed, delivered the bricks under the head. The blocks were then placed using another robot whose gripper had a forced sensor to determine the force applied when placing the blocks. The mix used was of 3:1 ratio of screened sand to type N mortar (USA standard), which was found to pump reasonably well. When the consistency of the mortar was normal outside this, mixes caused blockage of the head. Bond-wrench tests showed a flexural bond strength achieved in range of 149.8 MP to 874.1 MP (2173 psi to 126.78 psi), with a mean value of 84.05 MP and standard deviation of 31.25 MP for using an average placement force of 18.9 N (4.25 lb.).

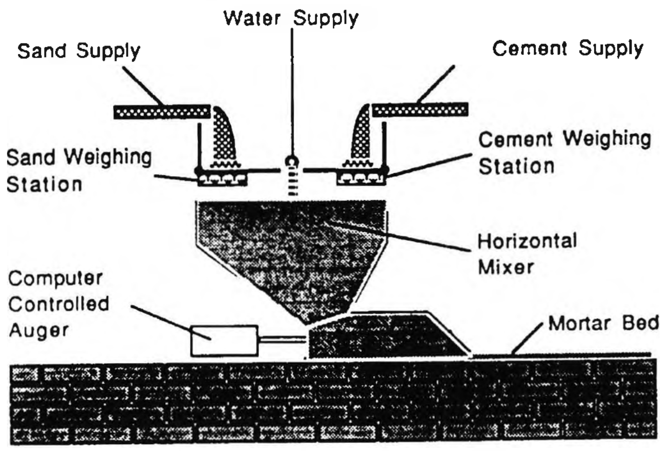


Figure 8.2 *Computer controlled mortar supply (Atobelli et al., 1993)*

A further German project (Bock *et al.*, 1993), restricted the use of materials to ones with small dimensional tolerances (sand-lime and cellular concrete blocks), which allowed the use of thin mortar beds for bonding. Inaccuracies in the floor had to be levelled out, however. Again, the bricks are dipped in the mortar. A separate German project (Pritschow *et al.*, 1995) also uses building material with 1 mm tolerance with a thin mortar applied by dipping the blocks into a mortar tub and then stripping the excess mortar off. Initial experiments, with no sensors used, are reported to have produced good results. The mortar tray (figure 8.3) is placed on the mobile robot as part of a functional technology unit. There is no mention of how a continuous supply of mortar was to be achieved, however.

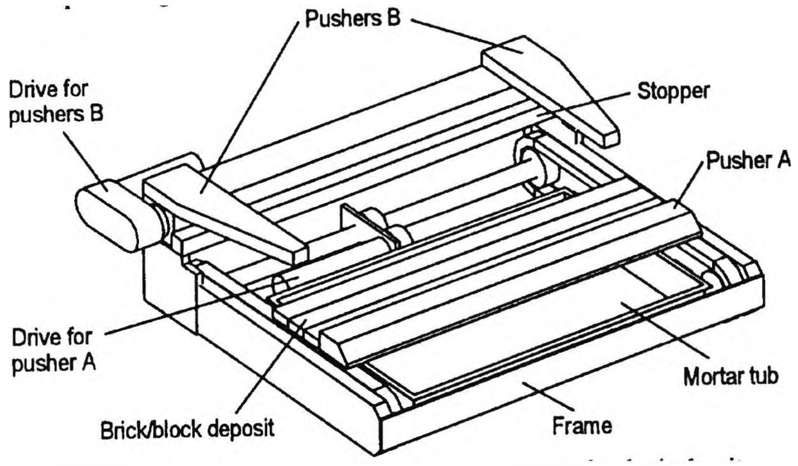


Figure 8.3 *Multi-functional technology unit (Pritschow et al., 1995)*

At North Carolina State University, a study on the automation of mortar application and adaptive control of the brick laying operation was conducted as part of a concept Experimental Robotics Masonry System (ERMaS) (Rihani *et al.*, 1994). This is illustrated in figure 1.5. A piston type pump, having the ability to be driven at various speeds, was used to apply mortar to bed and head joints. An experimental cell was built to produce partial assemblies as an initial study. To simulate the mortar application of the bed joint of the system, a pump and a conveyor were used, as well as a small robot. The pump and conveyor were integrated, and their speeds controlled, as well as the speed of the robot arm. The results of the tests showed that mortar thickness could be controlled with adequate precision by varying the mortar pump speed. This also showed that there was a relation between force application when laying the brick and the position of the block relative to the gripper for different mortar types. The force was measured using force/torque sensors mounted on the gripper, and a photoelectric sensor for measuring the gripper position.

In conclusion, from the many project covered, early attempts of using traditional method of mortar dispensing and extrusion aimed at showing the possibility of automating such procedures. Further work, however, is needed to achieve a complete solution to this complex task. To avoid that complex problem, many projects have adopted the use of building units with minimum tolerance, which allows for the use of thin mortar. It has been suggested that the use of this approach has been found to be necessary and sensible in achieving a technically possible and profitability system (Drees *et al.*, 1991). Counter to this point of view, the research of the author, in-line with the approach adopted at City University takes the point of view that advanced robotics technology should enable the use of traditional materials and bonding methods with great overall accuracy (Chamberlain, 1994).

8.3 Background work

Different studies were conducted at City University (Charles 1993, Ahmed 1993), aimed at determining the method of automating the mortar dispensing process. The main concern is to find a suitable mortar mix, that has acceptable mortar properties and is pumpable. The peristaltic pump, described in chapter 2.6, was used for these studies. This was subsequently modified to be controlled by a PC. The nozzle, designed at City University (described in chapter 2.6), was also used during these studies. This section describes the mortar mixes experimented with, and the various tests carried out with them.

8.3.1 Mortar qualities

The qualities of the mortar mix needed for the automation process, are for it to be workable, and at the same time pumpable. The suitability of the mix is expressed in terms of the various properties such as adhesion, cohesion, density, flowability, plasticity and viscosity. Also, when deciding on the design of the mix, there is a need to have a pumpable mix whose proportions can be easily reproduced in the building site. Compliance with existing Codes of Practice for that mortar produced is also sought. In the mortar pumping studies it was found that no particular mix could produce a mortar with all the desirable mortar properties and that improvement in one property was often to the detriment of another. A wide range of Tilcon and Pozament based mixes were examined for their suitability for pumping using the peristaltic pump. From this, two mixes were selected (table 8.1), as they offered reliable pumping for relatively prolonged periods, with the minimised occurrence of blockage in the system. Their selection and suitability for masonry construction was further confirmed by extensive property tests, based on BS 5441 methods of testing mortar, as well as giving satisfactory settlements. Tests such as the dropping ball test, which measures the consistence of fresh mortar were carried out. The consistence of mortar affects other qualities such as plasticity, stiffening time and strength. Entrained air tests, as well as tests for consistence and water retentivity were carried out. Flow table tests were carried out to asses the flowability of mortar, which is considered a main property when deciding on the degree of a mix's workability. This test was used for the purpose of our experiments, and is covered in section 8.4.1.1. Compressive strength tests for hardened mortar were also done. A non-standard test was also conducted to predict how much a brick settles when placed on freshly laid mortar (Chamberlain, 1994).

As mentioned earlier, from the two types of mortars tested, only two were found to be suitable for pumping. Results of the flow table tests showed that both mixes were workable when they had a minimum flow table value of 170 mm for fresh mortar. On pumping however, the flow of both the Tilcon and Pozament mortars was found to decrease, typically from 193 mm and 180 mm to 178 mm and 170 mm. This was thought to be due to the action of the pump used, which compacted the mortar and caused some water content to be separated out in the process. The flowability and pumpability of both mortars was also found to be time dependent with the Tilcon's decreasing sharply after 40 minutes, and the Pozament's after 90 minutes. The enhancement of the characteristics of the Tilcon mix was tried by increasing its water content by 15%, however, this resulted in a subsequent loss of 40% in its compressive strength and segregation of the mix components. It was then decided to use the less workable mix, describe in table 8.1, with the option of retempering during the pumping cycle. From the consistence tests, it was clear that both mixes showed loss of consistence (or increase in plasticity) on pumping, and with time.

From the settlement tests, the results showed that the settlement for a mortar bed of 12 mm thickness, for both mixes, with the brick dropped from a height of 15 mm were 1.5 mm and 1.4 mm for Tilcon and Pozament mortar respectively. This is considered satisfactory for practical use. The cured strength of the mortar was found to be reduced drastically (40%) after pumping. This is consistent with the apparent loss in workability of the pumped mortar and the separation of the mix water leading to insufficient hydration water for the cement. There is need for further research on this to understand the relationship between strength and the pumping of mortar.

Table 8.1 *Mortar mixes suitable for pumping*

Mortar type	Mix proportions ratio	Values of mix proportions	Flow table value (before pumping) (mm)	Pump speed (rpm)
Tilcon	Sand/Lime/Cement/Water	6.0/1.0/0.93/1.026	190	20-40
Pozament	Solids/water	5.0/1.0	180	20+

8.3.2 Pump and nozzle settings

At the start, calibrations of the peristaltic pump's delivery of water and the two mixes used was carried out, with the results shown in figure 8.4. From these results, it is clear that the more workable Tilcon mix covered a wider range of delivery rate, as opposed to the stiffer Pozament mix which covered a more narrow range of deliveries. This quality gives a wider range of control for the bead size and dispensing rate.

The pulsating action of the peristaltic pumping method created a rippled effect on the mortar bed delivered (figure 8.5). The different mortars showed that a diverse bead geometry can be achieved for combinations of offsets, nozzle angle and pump settings.
Various test were conducted to investigate the effects of the different combination of settings: nozzle angle, nozzle offset, pumping rate, and traverse speed of nozzle. This is necessary information, vital for the automation of the dispensing operation (i.e.

choosing the shape and size of bed needed). Table F.1, and figure F.1, in appendix F are for the tests using the Tilcon mix. The results shown in the table can be used to control the thickness and quality of bed. They show the effect of increasing the nozzle offset on the geometry of the bed. Variation in the nozzle angle, however, does not affect the shape of the bed. These observations were based on a set of curves, each curve showing the effects in bed size for increasing offsets at particular angle of nozzle. As for the effects of increasing the nozzle offset, this tended to produce narrower beads of increasing thickness, with more pronounced rippling effect. This can be seen clearly in figure F.1 (included in appendix F). A similar effect was found for higher pump settings.

The Pozament mortar exhibited similar variations in bead thickness and width. However, large offsets prompted more pronounced rippling in the bed. This mix was found to have a lower rate of delivery for a given pump setting, thus the speed of the conveyor or robot gripper would have to be less than with the Tilcon mix. Another difference was found to be that the ripples were less pronounced, except where large offsets are used, as shown in figure F.2 (included in appendix F). Table F.2 in appendix F shows the effects of increasing the offsets on the geometry of the bead.

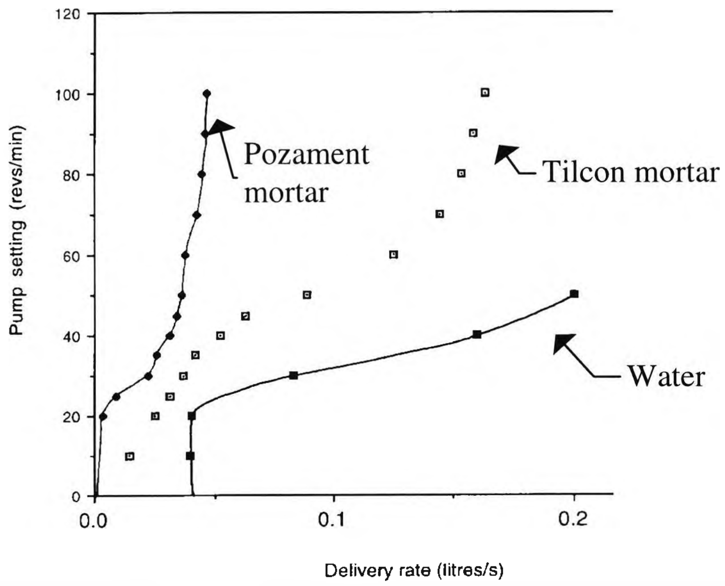


Figure 8.4 *Calibration of pump at delivery height of 400 mm (Ahmed 1993)*

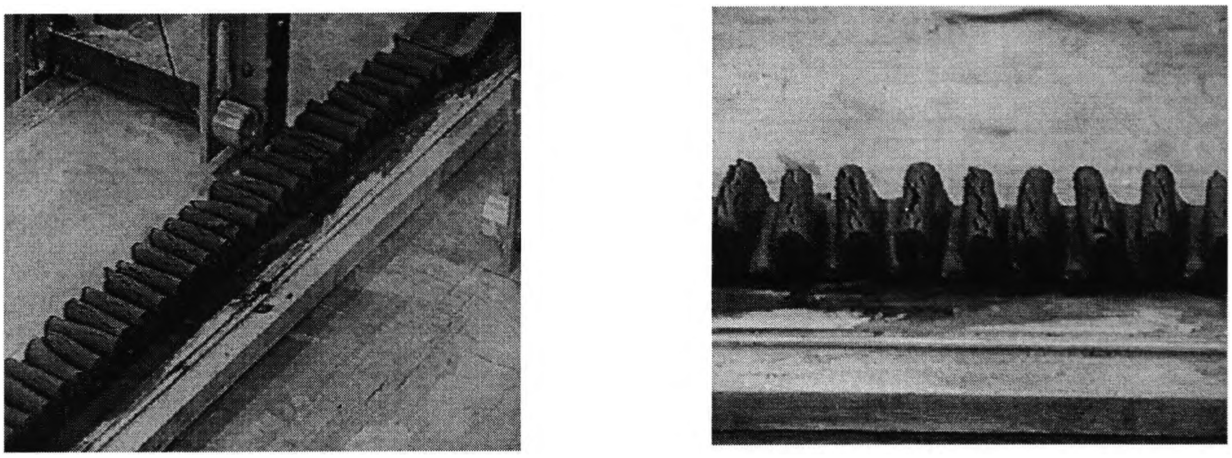


Figure 8.5 *Rippled effect on mortar bed due to nature of peristaltic pump used*

8.3.3 Application of mortar

Both techniques of automated masonry construction discussed in chapter 2.6.3, were tested, the first being the buttering of individual units and the second the continuos delivery on pre-laid units. These cater for the two distinctly different approaches to the automation of masonry building. In the first case, individual masonry units need to be manipulated either by the robot or, more effectively, on the supply conveyor during mortar application. In the second case, the ends of masonry units would need to be bonded by some form of vertical injection. A problem with this approach is that great care would be needed to ensure that the bedded down unit were not disturbed during the vertical injection.

The continuous delivery method was tested by placing a series of brick/block units on the conveyor bed, and clamping the nozzle at a fixed point over the conveyor (figures 8.6). The results showed negligible settlement or spread of mortar to the sides of the bead, which indicates potential for this technique to be used in erecting full scale walls. This technique also provided a continues bed of pre-determined size and quality. The immediate benefits generated from this method for a semi-automated environments is a substantial saving of time and money, made possible by reducing the necessity for field experiments during the erection stage. The only distinct difference in realisation would be to have a mobile pump unit, perhaps moving on tracks. On the other hand, the unsuitability of this method is that the units have to be presented at a pre-determined and precise offset and speed. Some of these problems would be eliminated using a fully automated system, where the units are presented to the nozzle at the required offset and speed under microprocessor control. Experiments carried out in this project include a limited study on this as described in section 8.4.

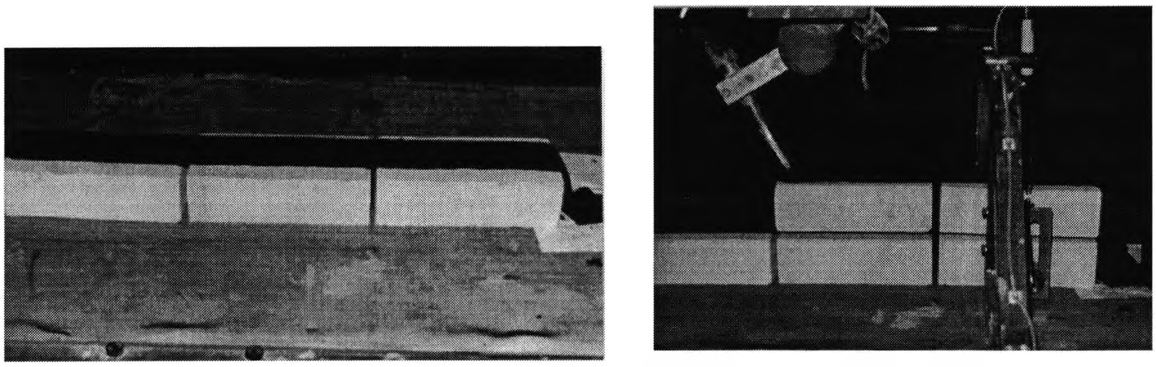


Figure 8.6 *Continues application of mortar on pre-laid bricks*

8.4 Robotic mortar dispensing

From the study of the types of mortar to be used for our application (described in chapter 8.3.1), the Pozament mix was chosen. The main reason for this choice is that it takes longer for this mix's flowability and pumpability to deteriorate, making it more appropriate for large batch continuous dispensing. Experiments were carried out to test the dispensing method of buttering individual blocks, and building small assemblies using the robot to manipulate the blocks. These experiments were conducted using a modified version of the peristaltic pump described in chapter 2.6. This is the same pump used for the experiments covered in section 8.3. Microprocessor control was added to the pump, allowing it to be operated from the main PC, using a library of 'C' commands. This enabled integration of its functions within the robot cell. A detailed description of the pump is given in chapter 2.6.

8.4.1 Mortar preparation

The mix was first weighed (figure 8.8), and the correct amount of water was added according to the proportions given in table 8.1. It was vital to achieve the right amounts of water content, as too little water content resulted in the mix being not pumpable and the occurrence of blockages. Too much water resulted in a segregated mix with little cohesion. To enable quality control it was necessary to carry out a Flow-table test, using the recommended results as a guide (table 8.1) to achieving a workable mix. Details of the flow test are described below.

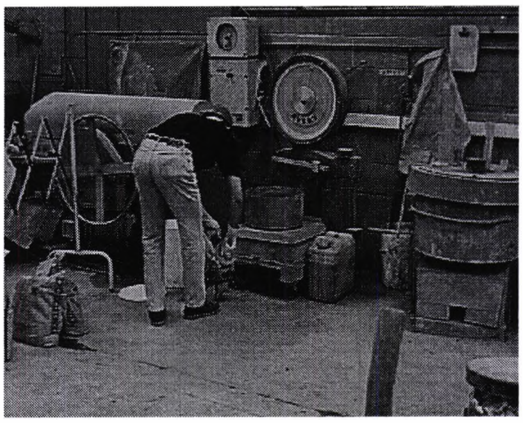


Figure 8.7 *The mix components are first weighed*

8.4.1.1 Flow-table test

The Flow-table attempts to measure one aspect of the workability of a mix, its flowability. The Flow-table is shown in figure 8.8. It comprises of a horizontal brass table mounted on a vertical shaft, which is raised to allow it to fall freely under the action of a cam, the falling distance being 12.75 ±0.13 mm. The test is conducted on a sample of mortar which is placed in a mould on the flow table's centre (figure 8.8). With the mould removed, the cam is turned twenty-five times in fifteen seconds (figure 8.8). The average diameter of the resulting flow is then determined, using four diameters measured at equal intervals. The mortar flow is then defined as the increase in the average diameter of the mortar, expressed as a percentage of the diameter of the mould, recorded to the nearest 5 mm. To try and make the results consistent it is important that the same operator performs all tests.

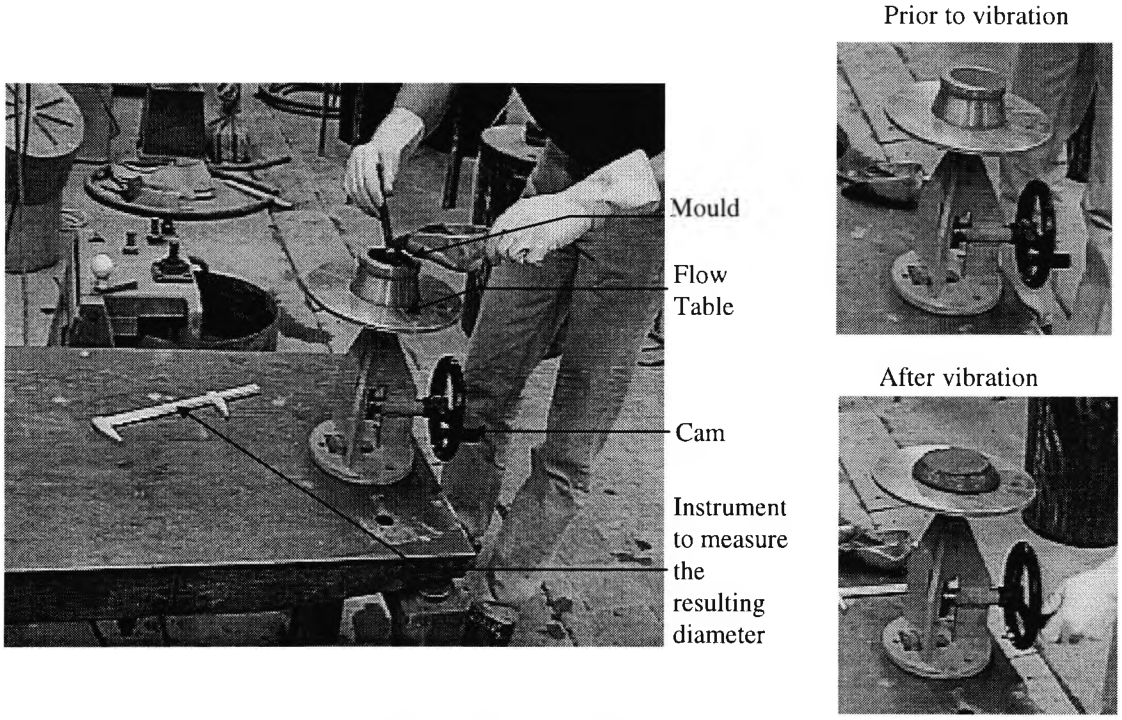


Figure 8.8 *The Flow-test*

8.4.2 Dispensing moves

The robot was used to present and move the blocks relative to the dispensing nozzle, for buttering a single side (figure 8.9) or two sides (figures 8.16- 8.21) of a block. Use of the robot allowed automatic setting of the nozzle offset, as well as control of the bead application speed. Also, being able to control the speed of the pump by the main controlling software facilitated control of the shape of the bead. The settings described in section 8.3 were used as a guide to achieving this, (e.g. tables F.1 and F.2 in appendix F). A number of experiments were carried out to test the fully automated process. In this, small constructions were made as three block assemblies, with the middle block staggered, as illustrated in figure 8.10. This was done to allow the middle block to be punched out in a shear strength testing arrangement. Unfortunately circumstances did not permit this punching test to be carried out on more that one specimen.

For bedding down the blocks, two methods were tried. The first method was to locate the block using a combined sideways and downward movement (figure 8.11), to its required position. This offset depended on the mortar thickness that was used. The method adds some pressure to the mortar and assists the bond. The second method was to simply locate the block downwards in a slow motion until it touched the surface it was laid on. In the case where it had mortar on both its sides, it was then moved sideways, again in a slow fashion, until it touched the adjacent block.

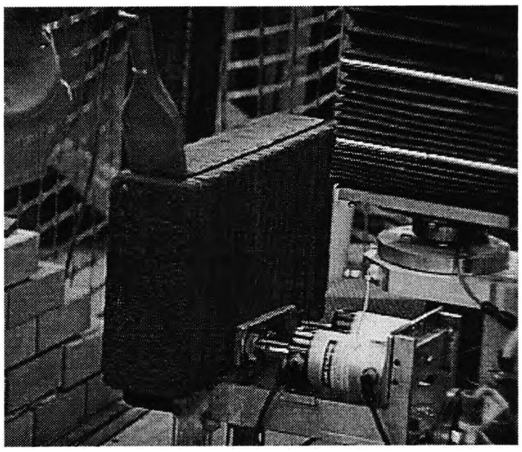


Figure 8.9 *Mortar application on the bedding side of the block*

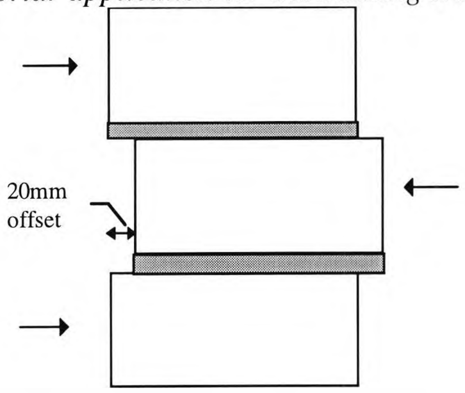


Figure 8.10 *Three block assembly with the middle block offset by 20 mm (to be punched out)*

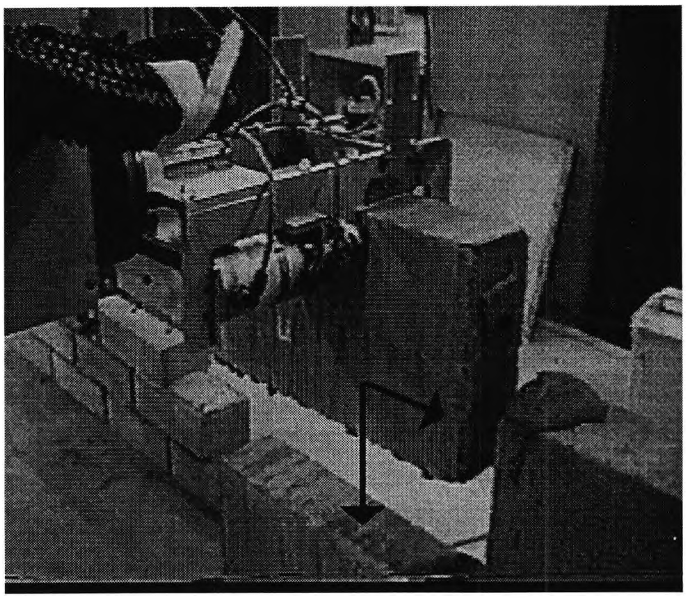


Figure 8.11 *When laying the block, a sideways and down movement is used after lowering the block*

For buttering the two sides of the block, a single pass dispensing move was devised that allowed the robot to rotate the block relative to the dispensing nozzle and give a constant bead application velocity. This represents stage 5 of the building process as defined in figure 2.2. A detailed description of this move was covered in chapter 2.6.3.1. A 'C' programme was implemented for generating the parts of the complete move. This allowed the variables to be tested, including the speed of robot motion for required mortar thickness. In theory, these variables are calculated automatically from the mortar thickness needed, which was, in turn, calculated using the real-time adjustments described in chapter 2.5.3.1. The adjustments calculated, use the information gathered in the previous stages of the building process (figure 2.2). The accelerating part of the move is calculated to ensure the robot starts moving at the required constant speed (i.e. not accelerating) when the block arrives at the start position for dispensing (i.e. the fixed point of the nozzle). The 'C' programme executes a segmented parabolic move, which has a maximum positive inflection at the start and end of the moves (described in chapter 3.4.2). The programme is integrated with the dispensing pump's functional library. The start and end of the move are synchronised with the pump.

8.5 Wall constructions and dispensing results

Small assemblies were successfully built using the robot to lay the buttered blocks (figures 8.12-8.15). From initial trials, before the robot's movement was synchronised successfully with the pump, it was found that the end of the move resulted in the block being stationary for a few seconds with mortar accumulated at the end of the block. An example of this is shown in figure 8.12. To resolve that problem, a snuff (suck back) option was added to the pump, which reversed the pumping motion briefly at the end of the move. Also, an extra move was added at the end of the dispensing process, moving the block away from the nozzle quickly. Starting the pumping at exactly the edge of the block was also necessary. Results showed that any excess, as shown in figure 8.17 on the side of the block, tended to make the whole bead of dispensed mortar peal away from the block face.

Preliminary results also showed that the location accuracy and orientation of individual units in the simple placement, reflect the high percussion of the robot and thus is a promising result. However, the error in the lateral offsets when laying the block, as a consequence of the gripping operation of the block (chapter 7), results in an overall misalignment (± 5 mm). A method to eliminate this error was suggested in chapter 7.

A satisfactory bead of mortar was achieved, proving the ability to control thickness and shape by varying the pump speed and nozzle offset (tables 8.2-8.3). Through trials, it was found that the best results were achieved when the nozzle angle was set at 30°, at an offset of less than 1 mm and using a pumping rate of 60 (revs/sec), with the blocks wet but surface dry. Using these settings gave a satisfactory mortar bead with a thickness of 10 mm, that adhered well to the block. To achieve the nozzle offset of less than 1 mm, the nozzle had to be pressed down against the block surface during the dispensing move, thus risking impact with the block. To avoid possible damage, the nozzle was mounted from the flexible rubber delivery hose, which connects to the pump unit.

A failure stresses of 0.19 N/mm^2 was achieved in a shear test using the three block assembly (double shear arrangement) shown in figure 8.15. The permissible shear strength is 0.21 N/mm^2 (CP111 Structure recommendation for load bearing wall). The above figure of 0.19 N/mm^2 as a result of the tests made in this research is a little short of the desired strength, however further improvements made to the mortar mix design as well as further experiments are needed to address that.

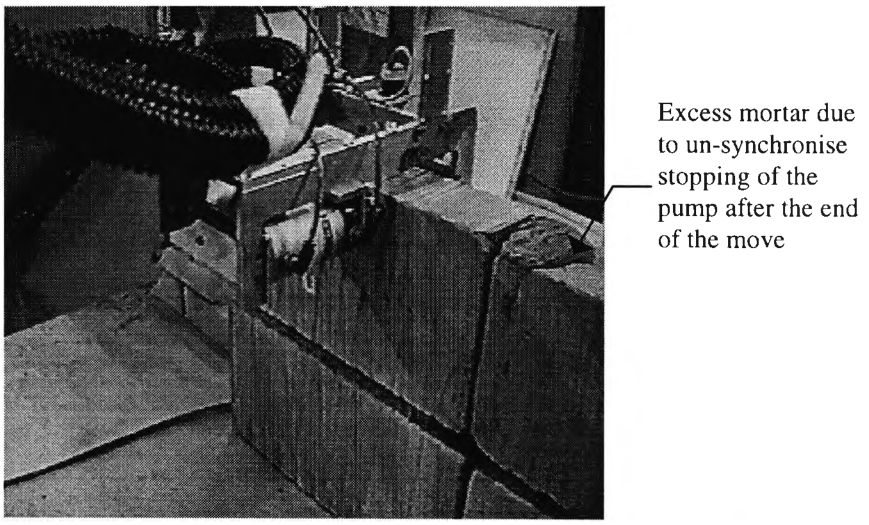

Excess mortar due to un-synchronise stopping of the pump after the end of the move

Figure 8.12 *Showing the robot laying a block after mortar has been dispensed on it*

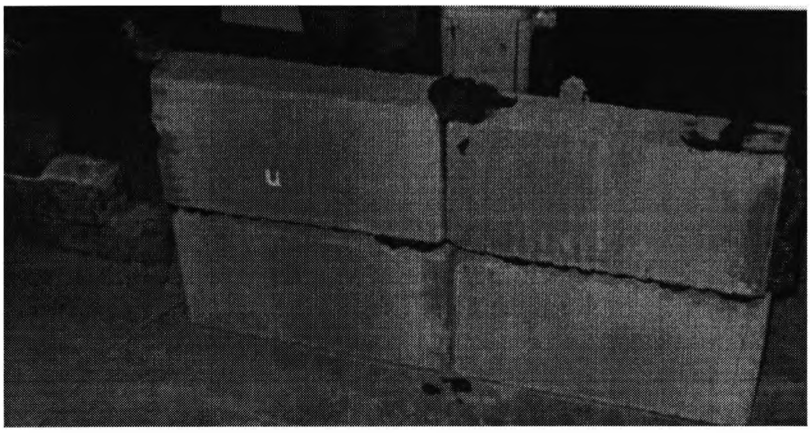


Figure 8.13 *Buttered blocks used in building small walls*

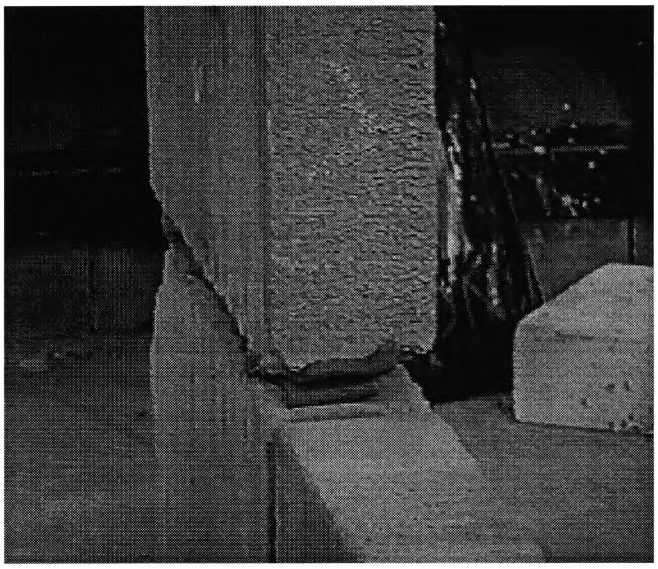


Figure 8.14 *Example of a laid block with mortar on the bedding side only*

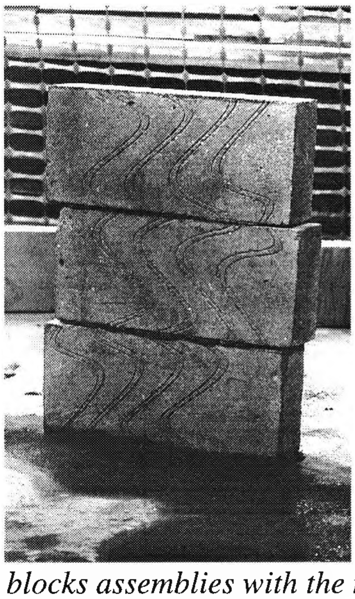


Figure 8.15 *Three blocks assemblies with the middle block offset*

Table 8.2 *Results of dispensing mortar on blocks using single pass move using robot speed of 43 mm/sec*

Pump speed (rpm)	Offset (mm)	Block condition before application	Nozzle inclination (degrees)	Outcome test (did the mortar stick to the block ?)
4.3	0	DRY	45	Yes, but too thin
5.5	0	DRY	45	Yes, but too thin
6.0	10	DRY	45	No
6.0	3	WET	45	No
6.3	0-1	WET	45	No
6.0	0-1	WET (surface dry)	30	Yes
6.0	0-1	WET	30	Yes
6.0	0-1	DRY	45	Side dropped
6.0	0-1	DRY	45	Yes
6.0	0-1	Very Wet	45	NO
6.3	0-1	Very Wet	45	NO
6.3	0-1	WET (surface dry)	30	No

Table 8.3 *Results of dispensing mortar on bricks using single pass move using robot speed of 43 mm/sec*

Pump speed (rpm)	Offset (mm)	Block condition before application	Nozzle inclination (degrees)	Outcome test (did the mortar stick to the block ?)
6.3	0-1	DRY	30	NO
6.3	0-1	WET	30	YES
6.3	2	VERY WET	30	YES
6.3	0-1	DRY	30	NO

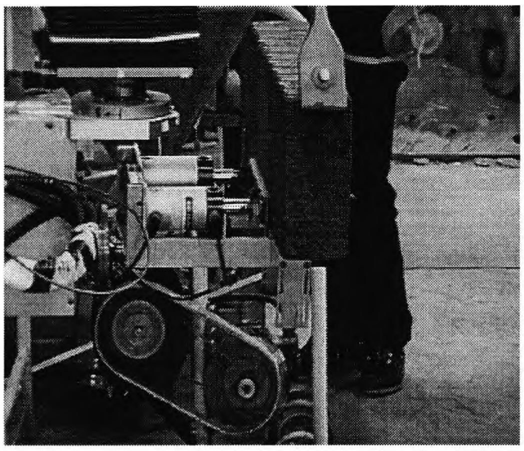
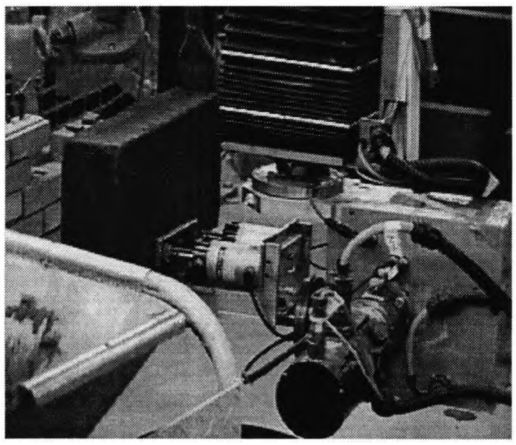



Figure 8.16 *Two views of the starting position of the mortar dispensing move (the nozzle is touching the block)*

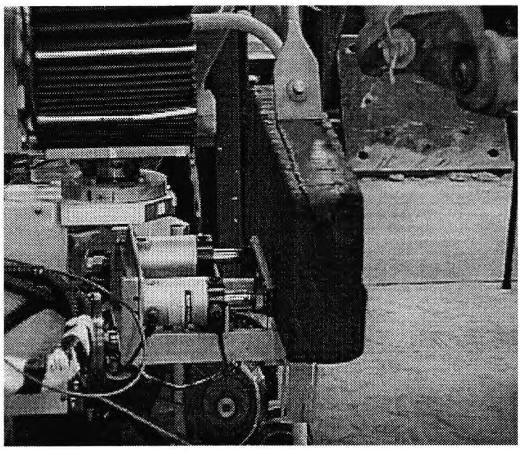


Figure 8.17 *During stage 1 of the dispensing move*

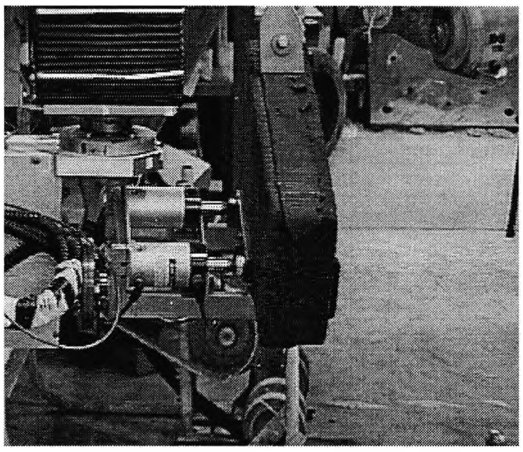


Figure 8.18 *Near the end of stage1, of the dispensing move*

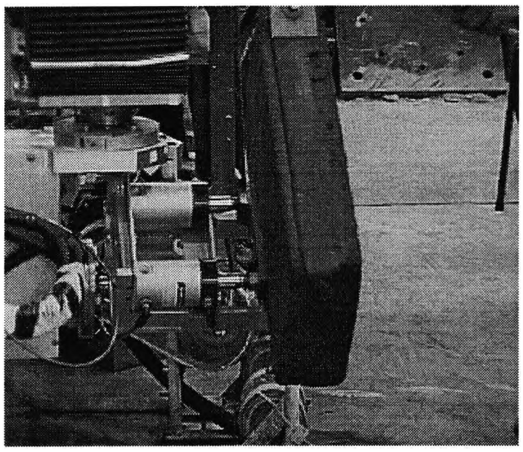


Figure 8.19 *End of stage 1, start of stage 2 dispensing mortar round the corner of the block*

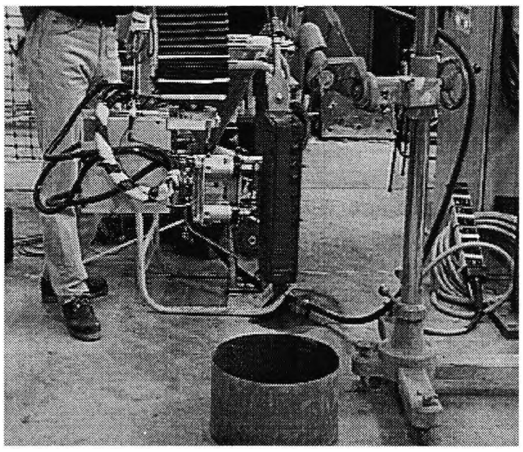


Figure 8.20 *During stage 3 of the dispensing move, applying mortar on the side of the block*

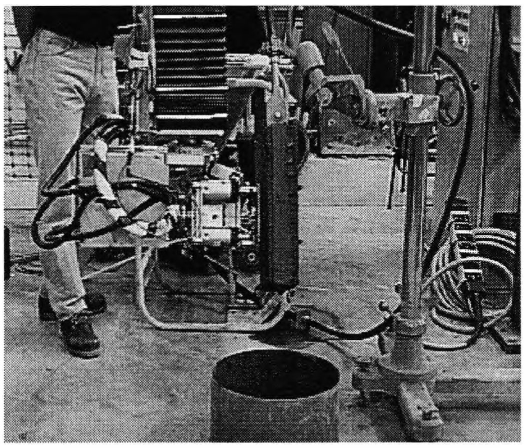


Figure 8.21 *At the end of stage 3 of the dispensing move*

8.6 Conclusion

The tests on mortar dispensing showed that a variety of bead geometry were possible, for combinations of offset and angle of the nozzle, as well as pump settings and velocity of the robot's motion. A clear trend of decreasing widths combined with increasing thickness was noticed for gradual increase in offsets at a given nozzle angle. The design of the nozzle was found to be effective in the extrusion of the desired bead. Mortar was shown to be effectively pumped and dispensed for robot wall construction using a Pozament mix. A weight had to be applied to top of the mix in the pump's feed hopper. Also, the mortar was stirred occasionally to minimise the chance of blockages. This reflects the thixatropic nature of the mix. Inadequate cleaning of the nozzle caused the shape of the mortar bead to be disfigured. Thorough cleaning of the pumping elements was also necessary for the same reason.

From the 18 assemblies built, a promising result was achieved in the accuracy using the fully automated robotised building process. Even when the blocks were placed with large errors (±5 mm offsets), the assemblies achieved were found to be stable. The method of laying the blocks, where pressure was applied with combined sideways and downwards motion, was found to be less accurate on account of the partial compliance of the yaw-axis. Here, ±5 mm location errors (centre) and approximately ±1° orientation error accrued. While applying pressure by the robot in placing a block, a side way motion is still needed to close the perpendicular joint. Part of the ±5 mm location errors was due to the block sometimes slanting when gripped. From the experiments covered in chapter 8, this was found to cause an error of ±3 mm in some cases. This tends to occur when the centre of the gripper does not coincide with the centre of the block just prior to gripping. A method of scanning the side of the block, using a laser profiler to measure the slant, was suggested in chapter 8. However, it would be necessary to add a 6th (pitch) rotational axis to the robot to implement corrections for the laying process.

In the pumping operation, the need to synchronise the starting and stopping of the mortar issue from the nozzle with the robot's movement of the blocks is vital for the success and accuracy of the process.

Extensive measurements of the settlement of the mortar, when the blocks are placed using the robot, have not been carried out. It was typically 1-2 mm. This is an important factor in automating the building process. Further studies are needed in order to understand and document accuracy in building larger assemblies, and the ability of producing a horizontal course of blocks which are also vertically aligned, as well as subject to suitable bonding pressure.

Whilst circumstances (short availability of the lent pump) did not allow further testing, a result of 0.19 N/mm^2 was achieved. This figure is a little short of the desired strength of 0.21 N/mm^2. Whilst a small modification of the mix, without sacrificing other essential properties, the mix strength could be easily adjusted to suite CP111 or any other requirement.

Chapter 9: Robot Intelligence

9.1 Introduction

To enable robots to work on construction sites, it is necessary that they sense unplanned events and react to them. This research is committed to the development of robotic automation that exhibits intelligence for working in such hostile and dynamic environments. This will allow the robot to pursue the goals required of the construction task, rather than play-out scripts, and thus transform the robot to becoming perceptive and its actions to be interactive. Thus, its performance becomes responsive to the state of the environments it is working in. Also, to truly automate the construction task requires intelligent programming, for the integration of the design phase with the construction phase, thus allowing the integration of building designs appropriate for automated construction. The first step for this process is to identify and classify types of designs, processes, and the tools with respect to how they can be automated and integrated to achieve the main goal.

In this research, rule-base expert systems (RBESs) and IF..THEN constructs are used to provided knowledge representation for the intelligent behaviour of the robot. Such RBESs respond to specific input, and produce output based on the knowledge embedded in the objects with respect to an interpretation of the rules. The rules are causal relationships between the objects. To build the RBES, a shell, KShop, was used. This made possible control of the inference engine (backward or forward chaining) in a relatively flexible manner, and also building the knowledge into a productive rule network. The main reason for the choice of this expert system shell is the facility to translate the developed rule-bases into 'C' code, thus enabling integration with the main robot functional programme, as illustrated in figure 9.1. This enabled the development of real-time intelligence for the robot, allowing handling of inaccuracies in materials and operation in the dynamic environment of the construction site.

Three separate RBESs were developed (figure 9.1). The knowledge elicitation for each RBES was tackled in different ways. The first RBES pre-processes the CAD design of a building project into a 'theoretical task', using assembly rules (figure 1.7). The project design is created using a CAD programme which allows project designs to be surface modelled. These rules process each project design into a symbolic representation and determine the order in which a project is to be built. They take into consideration the design of the robot end-effector and collision zones, which clearly affect the order of building at the corners in the design project. Knowledge elicitation for this RBES was provided by the author, through investigation of block and robot geometry, as well as trial and error work. The limitations of this system, described in chapter 1.5.2, are the limited project designs the system can order, as well as not having the intelligence to decide which end of the block is to be buttered in the construction of corners. Accommodation of sophisticated designs would only require the addition of more rules to the current system. The experimental RBES is tested and verified using an example masonry project design, produced using the CAD programme.

The second RBES deals with the run-time adjustments implemented through sensing. This was developed to perform the task of safely picking up a block in an intelligent way. The RBES is integrated with the robot functions and sensor data (figure 9.1). The 'block-pick' RBES uses the process knowledge developed in chapter 2. The knowledge elicitation method for this RBES was experimentation and observation. For the block picking task, no assumptions were made about the conveyor location and hence the block location. The procedures developed to locate the conveyor, described in chapter 5, are represented in this RBES. With the conveyor and gripper sensor information applied to the rules, the robot is able to assess a variety of situations, and react accordingly to them.

The third RBES determines the optimum moves for the robot. It detects impossible moves and advises the user against their use (e.g. if the move requested exceeds the system's maximum velocity). Knowledge elicitation for this RBES was from the theory of robot programming (chapter 3), as well as experimentation (chapter 4). The knowledge gathered from the investigation into the vibration of the robot end-effector is used in the development of the rule base. The experiments for this were limited to investigation of moves on the x-axis. The method developed to determine the robot end-effector vibration was covered in chapter 2.8, and the experiments and results presented in chapter 4. Data on the maximum velocity and acceleration of the robot, as well as the theory of the different move types, both used in the development of the RBES block pick, are given in chapter 3.

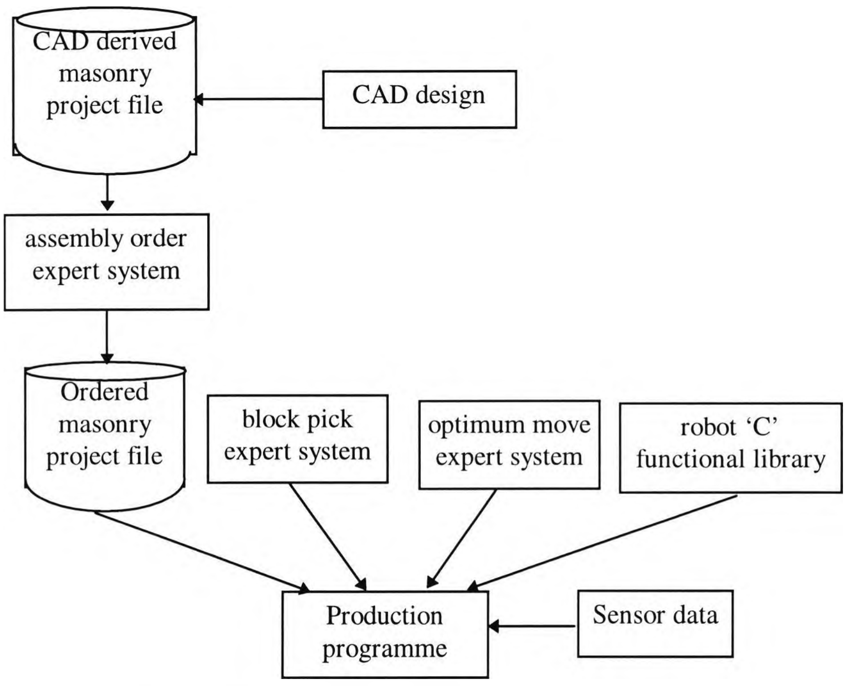

CAD derived masonry project file
CAD design
assembly order expert system
Ordered masonry project file
block pick expert system
optimum move expert system
robot 'C' functional library
Production programme
Sensor data

Figure 9.1 *Creating the production programme*

9.2 Construction robot intelligence

Robots in industry are commonly used to achieve one of four tasks, pick-and-place of large objects, peg-in-hole assembly, grinding, and finer assemblies such as nut-onto-bolt (Abu-Hamdan and El-Gizawy, 1992). A large number of those industrial robots are considered "first generation" robots. These type of industrial robots carry out tasks that are stored in the system's memory, and may have basic sensors thus making them slightly aware of their surroundings. "Second generation" robots, which are being introduced in industrial environments, are sensor-based robots that supply the robot controller with sensory data, thus enabling the robot to react to a changing environment. As technology becomes increasingly complex, there is a growing need for even more sophisticated industrial robots. To meet these needs, a "third generation" robot has been developed. These robots are considered to be super-intelligent machines, are completely autonomous, and described as having “decision making” capabilities.

Intelligent behaviour requires knowledge representation, and a framework for evaluating information (Stein, 1991). Several methods for approximating intelligent behaviour have been produced through research in the field of artificial intelligence. The choice of a knowledge representation scheme is of fundamental importance to a-life simulations. Rule base expert systems are common working examples of knowledge representations, having the ability to model aspects of behaviour. The rules fire according to input conditions, producing output based on the knowledge embedded in the data structures, objects, events, and other entities with respect to an interpretation of the rules. Furthermore, an expert system may generate widely different outputs for slightly different inputs, a by-product of the conditional constraints embedded within the rule structure.

The expert system tools most recently available, provide the user with the ability to control the inference engine in a relatively flexible manner. Most of these shells are rule-based systems, were knowledge is not represented explicitly but is built into the production rule network. Regarding these, Steiger-Garcao and Camarinha-Matos (reviewed by Thien and Hill, 1991) claim that for an expert system development tool to be useful, it should have the following features. Firstly, it should possess the ability to represent concepts, objects and rules to perform inheritance among attributes. Secondly, it should have the facility to interface with an external programming language, and also execute programs and routines external to the tool enabling access to device drivers, databases and CAD systems. Thirdly it should posses capabilities associated with altering the knowledge-base, dynamically and automatically. Also, it should provide a controllable inference engine, which can reason in a forward or backward fashion. Finally, it should include a good graphical user interface, with the ability to exhibit rule and object hierarchies visually, allowing debugging to be performed simply. The features mentioned above are included in the expert system shell chosen for this research.

The construction industry, compared to other industries, represents one of the most complex sectors for the utilization of robotics (Lowe and Campus, 1990). This comes as a result of the hostile working environment of the construction industry that suggest that any robot used in that industry will need to be 'smart' (Whittaker, 1986). To implement such intelligent robots, there is a need to provide a very sophisticated sensing system. This must satisfy the demand for a range of planning techniques to deal with the different situations likely to be encountered.

A robot working in the construction industry must survive in its environment to accomplish goals on its own (Flemming *et al.*, 1986). Further, the robot design must anticipate the extremes and impediments of a work site and incorporate devices and techniques for handling them. Cognitive robots sense, model, plan, and act towards achieving working goals. Cognitive robots pursue goals rather than play out scripts: they move towards goals and notions rather than to prescriptions and recipes. Although software driven, they are not programmed in the classical sense. Cognitive robots are perceptive and their actions are interactive: they take action in the face of vagaries and contingencies of the world. Performance is responsive to the state of the environment and the robot itself.

In the mid eighties a number of attempts were carried out to use artificial intelligence techniques in software automation of a variety of fields in the construction industry. Attempts were carried out to automate areas such as evaluation of construction schedules (Askew *et al.*, 1989), construction site layout (Tommelein *et al.*, 1989), and construction management, (Shohet *et al.*, 1991). In construction management there is a growing movement towards 'intelligent' planning programs that utilise expert systems and other artificial intelligence computing techniques (Booth *et al.*, 1991). Another research effort in construction, that uses artificial intelligent methods, is the development of an intelligent budget planing model in the building products industry

(Dawood, 1995). This is a computer-based factory simulator which automates the process of budget planning using factory attributes and intelligent production rules. The intelligent production rules, which are knowledge-based rules, which administrate the planning process. The rule-driven nature of this approach enables it to mimic the decision making of a human planner. Among other research efforts using expert systems in the field of construction applications, five small prototypes have been developed. These include expert systems for fire regulations, repainting of wooden facades, renovation loan guidance, pavement design and selection of concrete mixes (Koskela *et al.*, 1986). It is understood that the many attempts at software automation in the areas mentioned above have not progressed beyond demonstrator systems. Software liability in commercial decision making is one of the significant obstacles to fuller development and industry take-up.

The TAMIR, Technion Autonomous Multipurpose Interior Robot, has been developed in Israel (Rosenfeld *et al.*, 1990) and (Rosenfeld *et al.*, 1991). For this interior-finishing robot, an intelligent task-planning system was developed (Shohet *et al.*, 1994). Autonomous task-planning was achieved using mathematical algorithms. This approach provides a feasible solution, as it guarantees a near-optimal solution, is fully automated with fast calculation times, and the user interface is fairly simple. Several test-runs of the algorithm proved that it can be used efficiently and reliably for autonomous-task planning.

Research was carried out at City University, to examine the use of an expert system shell LEONARDO for the development of a RBES to investigate the planning provisions for a the wall construction project carried out by a masonry construction robot (Chamberlain *et al.*, 1990). This system could be linked to a CAD block description system to be used in planning the assembly task, using the dedicated CAD facility that enables the project to be solid modelled (Chamberlain *et al*, 1993). In the programme, rule-based reasoning is used for cavity wall construction, to avoid small block lengths and unwanted vertical joint alignment. On conclusion, a project description is generated. To derive the theoretical task, a generalised rule-base was used to order the building task with respect to potential collision zones arising with the end-effector. Also, rules are used for planning optimum moves and giving runtime intelligence for executing the theoretical task with sensor determined adjustments.

Research at Lancaster University has developed an intelligent robot excavator, "Lancaster University Computerized Intelligence Excavator" LUCIE, capable of safely and efficiently executing a range of excavation tasks with superior performance to that of the human operator. For this, a knowledge base is used to allow the autonomous robotic excavator to be intelligently controlled. In the development of this, emphasis had to be placed on observation as a means to knowledge gathering. (Green *et al.*, 1990). LUCIE is capable of autonomously digging a trench to a controlled depth in a variety of ground types and conditions. A production system approach is used for the intelligence in this. The production system is made up of rules which determine the behaviour of the system, and a working memory which contains updated system data used as the conditional part of these rules. These variables are updated by the low-level controller, sensors or rules. The other part of the production system is the inference engine, which determines the robot's actions, concluded as a result of checking the conditions of rules (Seward *et al.*, 1992). The data structures for the production system are written in the Ada language. This language is considered ideal for its flexibility and logical structure. (Seward and Quayle, 1996). Extensive field trials have been conducted from which it was established that LUCIE can dig in a whole range of soils with no human intervention, producing a good quality flat-bottomed trench. A task centred, goal oriented structure has been adopted as the operation method for LUCIE, which can be further subdivided into activities (Bradley *et al.*, 1994). Using the real-time production rule-base, the current goal of the system is first identified, and an activity is concluded that will achieve that goal (Bradley and Seward, 1995). The latest version, LUCIE 2 is equipped with a sophisticated scanning laser used for the detection of objects (Seward and Quayle, 1996). In future work on this project, the intention is to translate CAD drawings into robot operations, using expert system type rules derived from site observation knowledge.

Rule-based intelligence has been used for robot path planning of construction processes, in a research project carried out at Carnegie Mellon University (Stouffs *et al.*, 1993). A rule-based simulation programme has been developed that generates a motion path for each robot action, for each specific construction task. The simulation programme translates each task from a task description plan into a robot motion plan using a rule-based description of the robot agents. The task description consists of a detailed plan of the construction processes and construction elements. Use of a knowledge-based expert system called PLANEX, a planing software, is examined. This automatically generates a project activity plan, which is a description of the sequence of activities involved in the process. The robot's capabilities is described by a motion rule set which is invoked for decision making, according to the order in which the motion steps for a construction task should be processed. These rules take into consideration the many constrains on the robot's behaviour as a result of issues such as obstacles and safety. The simulation programme was demonstrated using two types of robots for a residential building example constructed with precast concrete panels. The simulation was successful in recognising impossible plans, and highlighting inefficiencies in the task plan including the influences of crane placement.

Using a BRAWAL excavator, produced by a polish company, research has been carried out to implement elements of artificial intelligence (Walczewski, 1995). In this, a monitoring system has in-built elements of artificial intelligence. This is done by monitoring several parameters, such as temperature and level of working fluids. It also warns the operator if permissible levels are exceeded. The control system consists of a microprocessor and a set of electronic sensors for location of the excavator. The control system consists of ultrasonic sensors for positioning of the collecting cart, and pressure gauges and tensometers for measurement of force. The BRAWAL 1611 excavator, equipped with the described system, has demonstrated intelligence, by taking over many working functions, diagnosing the working situation, and analyzing and optimising work parameters.

At the Technical University in Brno, Czech Republic effort is focused on development of alternative intelligence for multi-axis ALR robots (Belohoubek *et al.*, 1995). Here, intelligence is used for management control and robot system control. For robot movement, graphic simulation has been developed in C++ language. To introduce intelligence for obstacle avoidance, fuzzy logic is used (Belohoubek *et al.*, 1996). This method is used to providing fast desicion making for trajectory changes, using the solutions from time and space robot movement analysis. Fuzzy logic is also used for the servodrive control optimisation. In commercial systems fuzzy logic is a more established method than rule based expert systems.

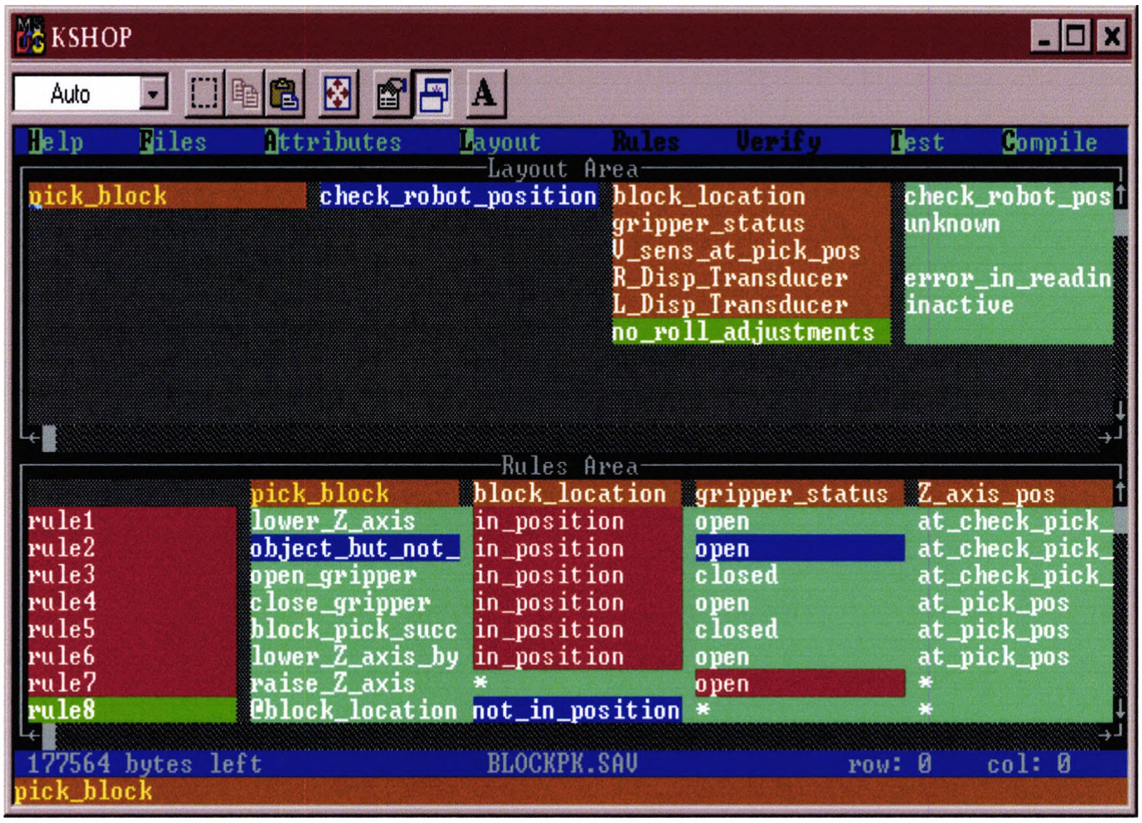
Another development, using intelligence in construction automation, is the use of virtual reality tools and AI techniques. To this end, research is underway at the University of Liverpool to develop a theoretical model for an intelligent VR system for planning layout of construction sites in a virtual environment (Boussabaine, 1996). Here, AI methods are employed to classify facilities that are used on construction sites and represent constraints between them. Rules on facilities locations, give advice on sizes of typical facilities, and provide a description of the layouts used on similar projects. The geometrical shape of the site facilities are created by CAD or a virtual reality software called Superscape. IF..THEN rules, written in a low level programming language, are used to control the size and position of objects in a virtual site layout. The rules and object models are then used to automate the creation of a site-layout.

9.3 Expert system shell

A RBES shell was chosen for the development of the RBESs. It was used for the purpose of knowledge representation and formulation of the inference strategies. It includes a user interface that is menu driven. This supports the user when building and testing an expert system. The rule layout area, the top window on the screen shown in figure 9.2, is used to create and link attributes together into a reasoning network. Two main types of attributes can be defined, conclusion or premise attributes, to form a network which describes the relationship between the conclusion attribute and its premise. Attributes represent the variables needed to solve the problem at hand. The rule area, the bottom window on the screen, is shown in figure 9.2. This is used to describe how to determine a value for the selected conclusion attribute from its premise attribute. Attributes can hold numeric values, character string values, a variety of functions and formulas, or a combination of these. Also, they can hold comparison values such as equal, not equal or less than, as well as using other concluded attribute values as premises, interval values and functions. This allows rules to be very flexible in how they are used and what they can test and calculate. An example is shown below:

```
if current_level is number
    then which_level is fmod(@current_level,2)
```

, where 'number' is any integer value entered, and conclusion of attribute *which_level* uses the numerical function fmod(x,y) which is available in the ES shell. By placing @ before attribute *current_level*, the value of that attribute is used i.e the entered value of 'number' in the above case.



```
KSHOP
Auto
Help     Files     Attributes     Layout     Rules     Verify     Test     Compile
                              Layout Area
pick_block          check_robot_position  block_location       check_robot_pos
                                          gripper_status       unknown
                                          U_sens_at_pick_pos
                                          R_Disp_Transducer    error_in_readin
                                          L_Disp_Transducer    inactive
                                          no_roll_adjustments
                              Rules Area
                 pick_block        block_location   gripper_status   Z_axis_pos
rule1            lower_Z_axis      in_position      open             at_check_pick_
rule2            object_but_not_   in_position      open             at_check_pick_
rule3            open_gripper      in_position      closed           at_check_pick_
rule4            close_gripper     in_position      open             at_pick_pos
rule5            block_pick_succ   in_position      closed           at_pick_pos
rule6            lower_Z_axis_by   in_position      open             at_pick_pos
rule7            raise_Z_axis      *                open             *
rule8            @block_location   not_in_position  *                *
177564 bytes left                   BLOCKPK.SAV                row: 0     col: 0
pick_block
```

Figure 9.2 *Knowledge Shop main screen*

The ES shell automatically checks to make sure that there is a rule for every possible combination of inputs, and lists the missing ones. This saves a great deal of time during decision module construction and testing. Testing rules involves making sure that rules succeed when expected to, concluding appropriate values for the conclusion attribute. These are useful features when developing a RBES for real-time monitoring of activities.

Forward and backward chaining inferencing are utilized. Forward chaining inferencing starts from an attribute whose current values have changed. Each of the conclusions of the attributes are then checked to see whether their values may also have changed. This process is useful in updating a network when an input attribute's current values have changed e.g. 'a sensor reading'. The input attribute and the changes are automatically propagated throughout the network. Forward chaining is an inference method used to propagate the effect of a new input value for a premise attribute to all the conclusion attributes that depend on it. It overwrites current values and concludes new values for the conclusion attribute that depends on the selected premise attribute. They are used to monitor new inputs and then propagate effects throughout the network.

Backward chaining is a way of controlling reasoning, by way of starting from an attribute whose value we wish to conclude. All rules that conclude values for that attribute were tried, in an effort to conclude values for it. In an embedded application, the application programme would supply the values. Backward chaining was used when we wish to know the value(s) of a goal or output attribute, given the value(s) of an input attribute. Backward chaining does not disturb the current values of any attributes. However, forward chaining will erase values from the conclusions if different values have been concluded while propagating from the changed inputs. The backward chaining command is used to find values(s) for the current attributes, given the values(s) to its premise (input) attribute. Backward chaining is done on goal attributes, and the systems attempts to find the state of the current situation. Once all relevant rules have been attempted, the goal attribute(s) will contain the results of the consultation.

A facility to trace the reasoning is used, with colour coded results of what occurred. These are displayed in the current set of rules in the Rule Area (figure 9.2). If a rule is green, then all premises are true and the rule has succeeded. If a rule is red this means either it has no conclusion value or a premise, indicated in red, has caused the rule to fail. A purple colour indicates a syntax error, and red or green colours will apply if a rule is succeedes or fails respectively.

The main reason for choosing this expert system shell is the facility for generating 'C' source code after building the knowledge system, which can, in turn, be embedded and complied with other programs (figure 9.3). This provides performance speed as 'C' is close to the machine hardware, and is designed to translated efficiently into assembly code or object code. The source code file contains the implementation of each of the interface functions, it also initialises the data structures that hold information about attributes, such as the premises and conclusions of the attributes and the rules used to conclude values for that attribute.

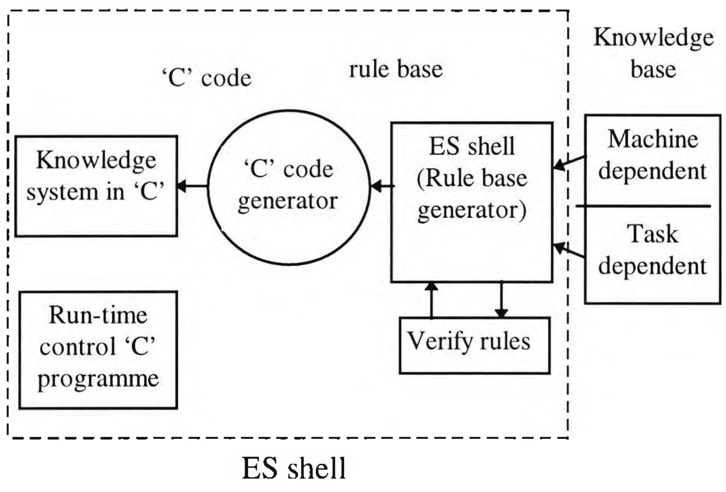


Figure 9.3 *Expert systems creation process*

Input, output and error hooks are provided so that the calling application programme can have a great deal of control over the behaviour of the developed ES module. For example, if an input for a certain attribute should read from a sensor, a new input hook function is installed that reads the sensor for that attribute. On output, if concluding a value that should, for example, control an actuator (switch), then an output function can be written that tests the conclusion value for that attribute and takes the necessary action. Being able to link the ES modules into real-world sensors and actuators, and the ability to process the complied rules very quickly, gives the ES modules the ability to perform intelligent decision making in real-time monitoring and control. These modules provide a network with inputs and outputs, and internal nodes between them (figure 9.4). Each node in this network has links, created by the user, to other nodes in the network. Each link has a set of rules defined by the user, that it uses to calculate its own values from the values of the input. The structure of those links are all user-defined. The user has complete control over the reasoning strategy that the network uses. Backward chaining can be done from one or more outputs, or internal nodes, or forward chaining from one or more input nodes. A mixture of backward and forward chaining can also be used.

The network of attributes implemented by the ES modules is a Directed Acyclic Graph. This means that information flows from inputs to internal attributes to output attributes and that no cycles between these attributes are allowed. Such cycles would allow the reasoning process to get caught in endless loops.

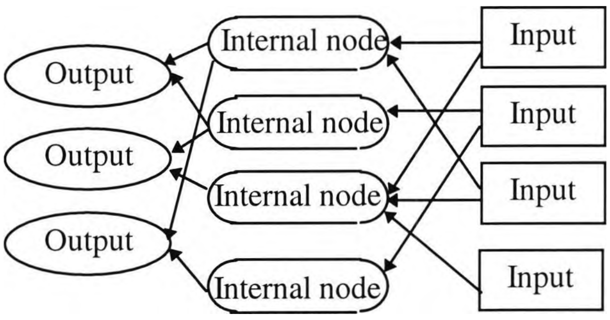


Figure 9.4 *Knowledge Shop system structure*

Some advantages of this approach over more "traditional" code for monitoring and controlling applications are:

(i)-The rules that monitor a situation can be automatically checked for completeness.

(ii)-Rules can be thoroughly tested and debugged in an environment that is suited for that purpose, before being integrated into the calling program.

(iii)-The calling programme does not need to be changed if changes need to be made to the rules or internal attributes of the rule-base module.

(iv)- New rules can be readily imported into a current rule-base. Rules can thus be designed and tested on a small scale and then added to the main rule-base.

9.4 Expert systems developed

9.4.1 Expert system: Assembly order

Using a dedicated CAD facility, project designs are modelled, and a corresponding 'project description' file generated. This enabled the transfer of essential information for robot construction, and linking of the design stage with the construction stage. When constructing the masonry project, using the project description file, it is necessary to consider the method of assembly, making sure it is carried out without collisions. To achieve that, assembly rules were developed to partake in the translation of the 'project description' into a 'theoretical task' for the robot (figure 1.7). These assembly rules take into consideration the design of a project and the collision zones arising from the robot end-effector. This clearly effects the order of building at the corners in a design project. To tackle this task, a range of possible designs and shapes of the masonry projects was considered. Assemblies were for single cavity blockwork, with a limited number of wall legs in a masonry project.

The process of developing the rules was carried out in steps. Firstly, potentially destructive situations for the robot, or the assembled part of the project, had to be identified. Using this information, rules were developed in stages to avoid these cases and determine the best method of assembly. To apply these rules to the designed masonry project, and determine their order of building, they were integrated with 'C' coded procedures that process the data which describes the masonry project. Once the rules were completed, tested and verified, and their 'C' code generated, the resulting 'C' code for the rule-base was integrated with the calling programme.

Figure 9.5 shows the steps taken to determine the assembly order for projects, starting with the first step of generating the project design using the CAD design. The project description file generated is then expanded (step 2), adding more information on each entity (brick or block), by defining its location relative to other entities in the project. This is carried out using rules integrated with 'C' coded procedures.

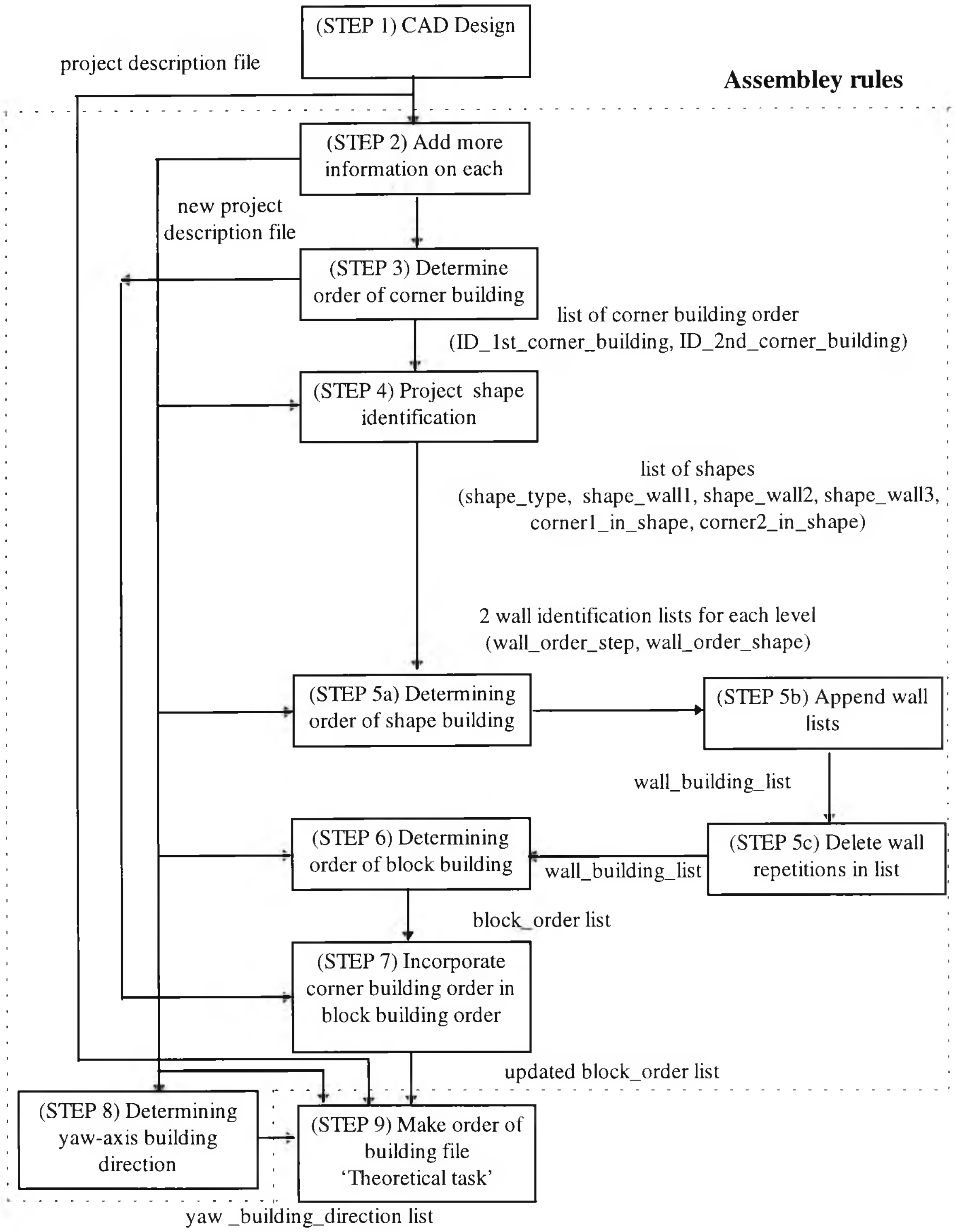

(STEP 1) CAD Design
project description file
Assembley rules
(STEP 2) Add more information on each
new project description file
(STEP 3) Determine order of corner building
list of corner building order (ID_1st_corner_building, ID_2nd_corner_building)
(STEP 4) Project shape identification
list of shapes (shape_type, shape_wall1, shape_wall2, shape_wall3, corner1_in_shape, corner2_in_shape)
2 wall identification lists for each level (wall_order_step, wall_order_shape)
(STEP 5a) Determining order of shape building
(STEP 5b) Append wall lists
wall_building_list
(STEP 6) Determining order of block building
wall_building_list
(STEP 5c) Delete wall repetitions in list
block_order list
(STEP 7) Incorporate corner building order in block building order
updated block_order list
(STEP 8) Determining yaw-axis building direction
(STEP 9) Make order of building file 'Theoretical task'
yaw _building_direction list

Figure 9.5 *Steps for determining the order of building masonry project*

Due to the shape of the robot end-effector, the order of laying the blocks in the corner without causing any damage, had to be determined (step 3). A method of describing the design of masonry projects was developed, using various predetermined plan shapes, which establish the design of projects using plan shape identification rules (steps 4).

After this was established, the order in which they are to be built was resolved. This was done using rules which enable the robot to build the project, whilst avoiding collision zones (step 5a). The result of this is a wall identification number list, ordered in the manner in which they have to be laid (steps 5b and 5c). From this, the order of the blocks is established (step 6), after which the previously determined order of building the corners is incorporated (step 7). Another extremely important factor in the determination of the method of building is the orientation of the yaw axis when laying each block. Using the measurements of the robot end-effector and side motor (figure 9.6), and from trial observation on the right and wrong manner of laying blocks, in different situations, rules were developed to conclude the orientation of the robot yaw axis when laying blocks (step 8). The results from this are combined with the order of building the blocks, to produce the 'theoretical task' (step 9).

A brief description of the complete process of ordering the building task has been covered in this section. A more detailed explanation of the methods, rules, and procedure devised to conclude each step of the process, is covered in the following sections. Examples of these rules are also given. The complete rule set is included in appendix G.1.

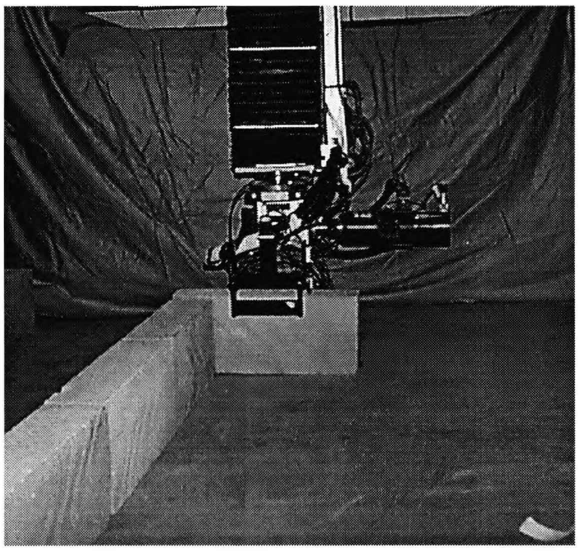


Figure 9.6 *Example of robot laying a corner block*

9.4.1.1 CAD design generation: step1

For the project design, a CAD facility, which has been programmed in AutoLISP, is used (Chamberlain *et al.*, 1993, and Chamberlain 1994). This was developed further by Warren Breen (Breen, 1993). The programme allows a variety of projects to be surface modelled as either single skin brickwork, blockwork or cavity construction comprising both bricks and blocks. Standard window and door openings can be introduced and the width of the cavity can be varied. Rule-based reasoning is used to help avoid small block lengths and vertical joint alignment in the cavity wall. After the user designs the project required, a project description file is generated. This file contains a list of information on each masonry unit used. The list is built up in a continuous spiral order, using a clockwise direction, starting from the lowest level upwards. The description list contains a description of each masonry unit as follows:

$$(X, Y, Z, XYSET, ID, TYPE, SPLENG)$$

, where X, Y, Z are the centroid co-ordinates of the target location of a unit, relative to a ground datum. XYSET is the orientation index (2= Y-direction, 1=X-direction) and ID is the block identification number. TYPE is the type of block (e.g. standard, half block, or special block) and SPLENG is the length, in the case of a special block. An example of a project description file, generated for the masonry project design shown in figure 9.7, is included in appendix G.1.2.

A user friendly interface was set up with pull down icon menus that prompt the user to interact and respond with the interface. They guide the user through the necessary functions, while maintaining all the AutoCAD menu functions. The main problem associated with alternate courses of blocks is the vertical alignment of mortar joints. For most construction purposes such alignment is not permitted. To prevent this alignment occurring, a number of tests are carried out to establish whether the mortar joints will align with those in the next course. If they do, then measures are taken to rectify the problem. The result is that joints are staggered over alternate courses, as can be seen in figure 9.7, thus making it easier for the project description file to be processed.

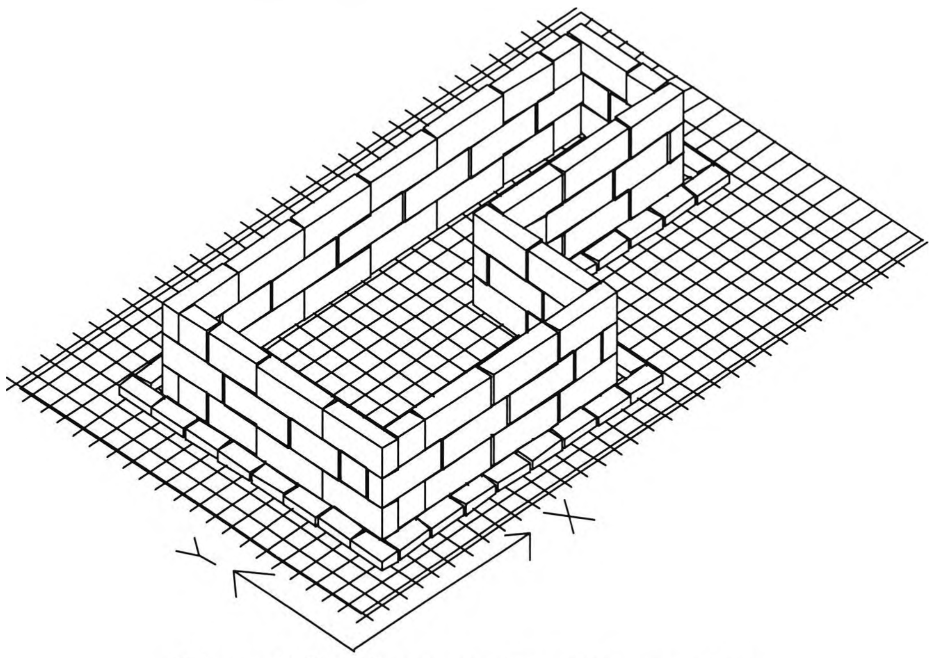


Figure 9.7 *CAD output of a masonry project*

9.4.1.2 Project description: step 2

Using rules integrated with 'C' procedures, expansion of the information describing each entity (brick or block) in the original project description file is carried out (step 2 figure 9.4) to provide the knowledge needed to determine the order of construction. This extra information will describe the location of each entity relative to the project as a whole, as well as its relation to neighbouring entities. The 'NEW project description' file is generated using the same order as the original project description file, i.e. spiralling upwards, in a clockwise direction, starting from the lowest level. Each entity in the new list is a description of each masonry unit as follows:

$$(block\_ID, wall, block\_wall\_ID, block\_direction, length, block, block\_wall\_pos, level)$$

, where *block_ID* is the block identification number, *wall* is the wall identification number the block is a part of, *block_wall_ID* is the position of the block relative to the wall it is in (1= first block in the wall, 2= middle block in the wall and 3= last block in the wall). Both are clearly labelled in figure 9.8. *block_direction* is determined using the same x and y directions, XYSET, generated in the original project file, with the addition of a sign to its direction (1= +ve y direction, 2= +ve x direction, 3= -ve y direction, 4= -ve x direction). The sign of the *block_direction* is determined using the direction in which the project file was processed (i.e. clockwise) as illustrated in figure 9.8. *length* is the length of the block, calculated using the TYPE and SPLENG description details from the original project file. *block* describes whether the block is a corner block or not, (1= corner block at start of the wall, e.g. block_id 1 in figure 9.8, 2= corner block at the end of the wall, e.g. block_id 4 in figure 9.8, 3= not a corner block, e.g. block_id 2 in figure 9.8). *block_wall_pos* is the block position relative to the wall it belongs to, i.e. 1st, 2nd and so on, block in the wall. Finally, *level* is the course-level the block belongs to.

When processing the project description file, three assumptions are made, all in view of the project designs generated, using the special AutoCAD programme and the way the project description files were determined. The first assumption is that the even numbered courses all have the same block lengths and layout (figures 9.8 and 9.10), and the odd numbered courses also. The second assumption is that the first block in each file, which is the first block of the lowest level built in the direction of the y axis of the design programme, will always be of type *block* = face_to_end (figure 9.8). This automatically makes the first block of all the even numbered levels to be of the same block type (figure 9.10). As a consequence of this, the first block in the second course will always be of type *block* end_to_face (figure 9.9). The third assumption is that the first block in a course, going round in a clockwise direction, will determine the configuration of the corner blocks in that course-level. An example of this is that the configuration of blocks 4, 5, 11, 12, and 13, 14 is determined by the configuration of blocks 22, and 1 (figure 10.8).

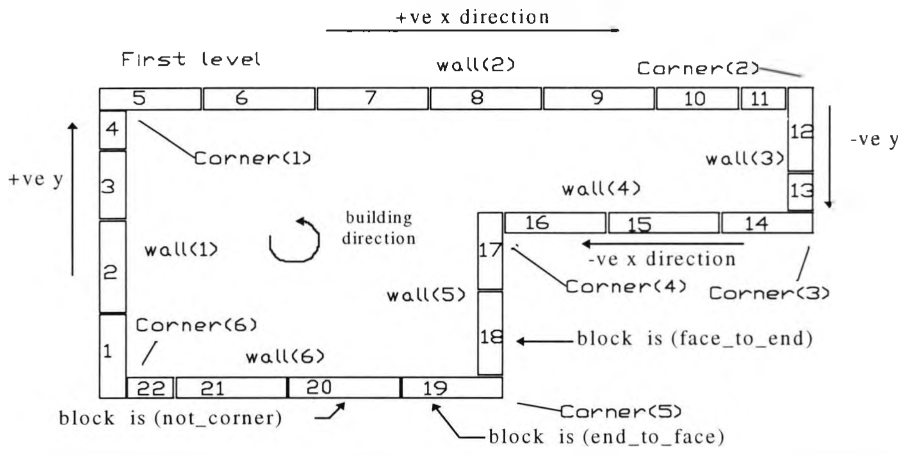


Figure 9.8 *First course of the masonry project shown in figure 9.7*

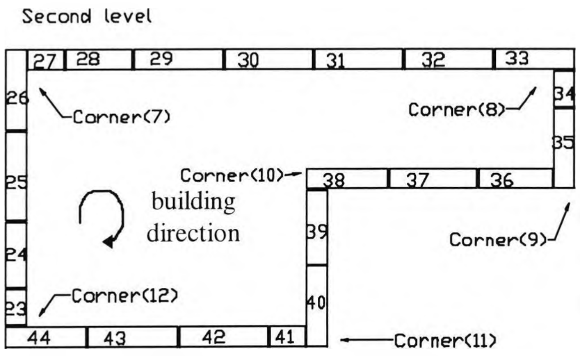


Figure 9.9 *Second course of the masonry project shown in figure 9.7*

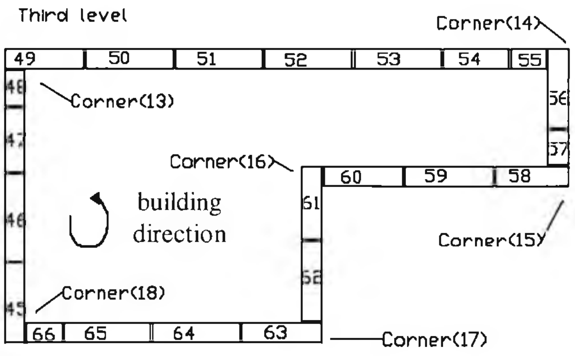


Figure 9.10 *Third course of the masonry project shown in figure 9.7*

9.4.1.2.1 Generating 'NEW project description' file

The new items describing each entity in the original project description file are determined using a combination of 'C' coded procedures and rules integrated with them.

The procedure described below determines the *block_ID*, *wall*, *level*, and *block_wall_pos* descriptions for each block from the AutoCAD generated project file (step2).

Procedure to determine *block_ID, wall*, *level*, and *block_wall_pos*:

```
(i)- Read block details from AutoCAD generated project description file.
(ii)- Using SPLENG and TYPE find block length for all block.
(iii)- Convert rest of details
        -for block_ID =1, wall=1, direction = XYSET(block_ID), level =1,
            block_wall_pos =1
        - block_wall_pos(block_ID) = block_wall_pos
        -Repeat until block_ID = number of blocks in project
                - block_ID = block_ID +1
                - if  Z(block_ID) > Z(block_ID -1)
                        - level = level + 1 , wall =0
                -if direction = XYSET(block_ID)
                        - block_wall_pos = block_wall_pos +1
                -if direction ≠ XYSET(block_ID)
                        - block_wall_pos = 1 , wall = wall +1
                        - direction = XYSET(block_ID)
                - block_wall_pos(block_ID) = block_wall_pos
                -wall(block_ID) = wall , level(block_ID) = level
        - no_walls in the project = wall
```

The remaining items, *block, block_direction, block_to_wall_direction,* and *block_to_wall_pos*, described earlier, are determined. These rules make up part of the complete RBES developed to determine the order of building a masonry project. Figure 9.11 shows these attributes and those used to conclude their values. A detailed description of each attribute and its determination is explained below:

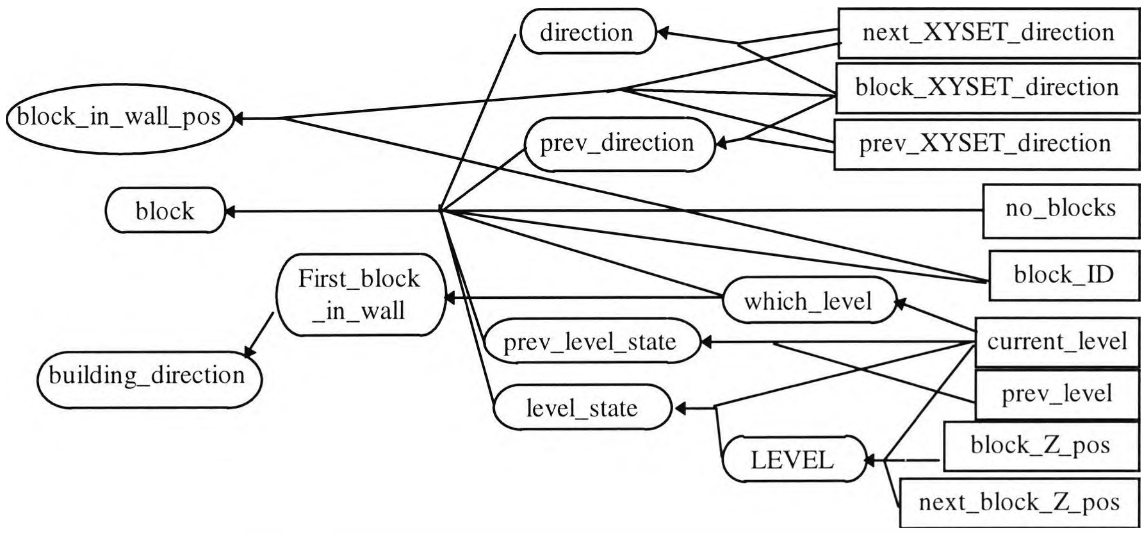


Figure 9.11 *The dependencies of attributes block, block_direction, and block_in_wall_pos*

9.4.1.2.1.1 Determining item *block*

A way of describing a block, is to determine wether it is a corner block or not, and, if so, what type of a corner block (face_to_end, or end_to_face). Examples of the three types of blocks, shown in figure 9.8, are block number 21, which is of type *block* = not_corner, block number 1, which is of type *block* = face_to_end, and block number 22, which is of type *block* = end_to_face. Rules were created to determine the type of each block in the project. Attributes used to conclude the value of attribute *block* are listed in table 9.1. To understand the rules concluding attribute *block*, the attributes in the list will be described first, in the following sections.

Table 9.1 *List of attributes that are immediately linked to attribute 'block'*

Attribute name	Attribute description
block	What type of a block is it? A corner block, or not. If it is a corner block then which block in the corner is it?
direction	Indicates if the direction of the block being examined is different than that of the next block position
prev_direction	Indicates if the direction of the block being examined is different than that of the previous one
which_level	Which level the block belongs to, indicating either an odd numbered level, or a an even numbered level
level_state	Has the level which the block belongs to just changed? (i.e. is the level of the block being examined the same as the one after it)
prev_level_state	Has the level which the block belongs to just changed? (i.e. is the level of the block being examined the same as the previous one)
Block_ID	Block identification number relative to the project description
no_blocks	Number of blocks in the building projects

9.4.1.2.1.1.1 Block level description attributes

Attributes are used to describe the course-level a block belongs to and how it relates to the levels of the blocks before and after it in the description file, thus enabling detection of the block possibly being the first or last block in a course. These block course description attributes are used throughout the RBES.

Attribute *level_size* is the number of blocks in each level, calculated using the number of blocks divided by the number of levels in a project file. Attribute *LEVEL* is used to determine the level of the block which comes after the one currently being examined. Rules determining this attribute rely on the z centroid co-ordinate of the block centre relative to a ground datum, given in the 'project description' file. Also, the level of the blocks currently being examined is used (i.e. attribute *current_level*). An example of these rules is given in rule_AO_1.

```
rule_set_AO_1:  if NEXT_BLOCK_Z_POS is > BLOCK_Z_POS
                then LEVEL  = current_level+1
```

As stated previously, an inherent part of the generated design is that the even and odd courses each have the same block sizes and layout (figures 9.8-9.10). Attribute *which_level* is used to give an indication of whether the block being processed belongs to an even or odd numbered course-level. This is determined from the remainder value resulting from the division of the course-level of the block by two. If the result is zero, this indicates an even numbered course-level, and any other value indicates that the current course-level is an odd numbered course. The rule concluding attribute, *which_level*, is shown in rule_AO_2, where the conclusion value is determined using the numerical function fmod(x,y) available in the ES.

rule_set_AO_2: if *current_level* is number
then ***which_level*** is fmod(@*current_level*,2)

An example of the use of attribute *which_level* is when the first block in the wall has to be determined. The rules, rule_AO_3 and rule_AO_4, created to conclude this uses the assumptions on the way the project designs are created, as discussed at the start of section 9.4.1.2.

rule_set_AO_3: if *which_level* is ≠ 0 then ***First_Block_in_wall*** is *face_to_end*

rule_set_AO_4: if *which_level* is =0 then ***First_Block_in_wall*** is *end_to_face*

Attribute *level_state* indicates if the course-level of the next block position to be examined is different from the one currently being examined (0 = level of next block position is same as current level, otherwise it indicates that the level will change). Attribute *level_state* is concluded using the expression in rule_AO_5.

rule_set_AO_5: if *LEVEL* is number and *current_level* is number
then ***level_state*** is @*LEVEL*-@*current_level*

Attribute *prev_level_state* is similar to attribute *level_state*, but instead it compares the course-level of the block being processed with that of the block previous to it. Attributes *direction* and *prev_direction* indicate if a change in the direction of a block had occurred, relative to the block after it, or the block previous to it, respectively. This is achieved using the value XYSET of a block, given in the project description file (section 9.4.1.1). Attribute *block_XYSET_direction* is the XYSET value of the block being processed, and attributes *prev_XYSET_direction* and *next_XYSET_direction* are those of the two blocks before and after it, when processing the project description file in a clockwise direction.

9.4.1.2.1.1.2 Attribute *block* rules

Rules were created to determine the value of attribute *block* for each block in a project, using the attributes listed in table 9.1 and illustrated in figure 9.11.

Rule_AO_6, given below, indicates that if the course-level of the block being examined is the same as that of the blocks before and after it, and the direction of the block compared to that of the blocks before and after it is unchanged, and it is not the last block in the project, then the block is not a corner block (e.g. blocks 2,3 in figure 9.8). Rule_AO_7 indicates that if the level of the block being examined is the same as that of the blocks before and after it, and the block belongs to an odd numbered course-level (i.e. first block in that level is of type face_to_end), also if the direction of that block, compared to that of the previous block is unchanged (i.e. the block is not the first block in a wall), but the direction of the block after it is changed (i.e. the block is the last block in a wall), and the block is the same level as the block previous to it (i.e. not the last block in a level), and it is not the last block in the project, then the block type is an end_to_face.

```
rule_set_AO_6:  if level_state is =0
                   and  prev_direction is =0
                   and  direction is =0
                   and  block_ID is ≠ no_blocks
               then block is not_corner

rule_set_AO_7:  if level_state is ==0
                   and  which_level is ≠ 0
                   and  prev_direction is =0
                   and  direction is ≠ 0
                   and prev_level_state is =0
                   and  block_ID is ≠ no_blocks
              then block is end_to_face
```

9.4.1.2.1.2 Determining item *block_direction*

Rules to determine the value of attribute *block_direction* for each block, rely on the x and y centroid co-ordinate of the block centre relative to a ground datum blocks. These rules rely on the fact that the blocks in the project are processed in a clockwise direction. An example of these rules is shown below. Rule_AO_8 is a direct consequence of the first assumption discussed in section 9.4.1.2, regarding the way the 'project description' file is generated. An example of rule_ AO_9 is block number 15 and rule_ AO_10 block number 12, both shown in figure 9.8.

rule_set_AO_8: if *block_ID* is 1 then ***block_direction*** is *Y_pos_dirct*

rule_set_AO_9: if *block_XYSET_direction* is 1
and *block_ID* is ≠1
and *BLOCK_X_POS* is < *PREV_BLOCK_X_POS*
then ***block_direction*** is X*_neg_dirct*

rule_set_AO_10: if *block_XYSET_direction* is 2
and *block_ID* is ≠1
and *BLOCK_Y_POS* is < *PREV_BLOCK_Y_POS*
then ***block_direction*** is *Y_neg_dirct*

9.4.1.2.1.3 Determining item *block_to_wall_pos*

The position of the block is concluded using rules which determine if the block is the first, middle, or last block in that wall. These rules rely on the block XYSET direction given in the 'project description' file, and the fact that they will be processed in a clockwise direction. An example of rule_ AO_11 is block number 5, rule_ AO_12 is block number 6, and rule_ AO_13 is of block number 11, all shown in figure 9.8.

rule_set_AO_11: if *block_XYSET_direction* is ≠ *prev_XYSET_direction*
and *block_ID* is ≠1
then ***block_to_wall_pos*** is *first_block*

rule_set_AO_12: if *block_XYSET_direction* is = *prev_XYSET_direction*
and *next_XYSET_direction* is = *block_XYSET_direction*
and *block_ID* is ≠1
then ***block_to_wall_pos*** is *middle_block*

```
rule_set_AO_13:  if  next_XYSET_direction is ≠ block_XYSET_direction
                         and  block_ID is ≠1
                    then block_to_wall_pos is last_block
```

9.4.1.3 Corner building order: step 3

The next stage of ordering the building task, is to check the order of building each corner.

9.4.1.3.1 General building direction

The direction of building was determined first from the design of the corners and the robot end-effector. The various restrictions on the order of constructing the project, while avoiding collisions, will be used to reorder the building project. These restriction are in the form of rules discussed in this and subsequent sections.

For safety, the corner blocks are laid in the order of block type end_to_face first, and then face_to_end. Once this was decided, and using the information on the design of the corners in a project from the 'project description' file, discussed in section 9.4.1.2, then the building direction for each course-level can be determined. Rule_ AO_14 and rule_set_AO_15 were created to conclude attribute *Building_direction* using attribute *First_Block_in_wall* discussed in section 9.4.1.2.

```
rule_set_AO_14:   if First_Block_in_wall is end_to_face
                     then Building_direction is clockwise

rule_set_AO_15:   if First_Block_in_wall is face_to_end
                     then Building_direction is anticlockwise
```

9.4.1.3.2 Corner building rules

The order of building each corner is determined by the size of the block and the gripper width. An example of the robot laying a corner block is shown in figure 9.6. It is clear that if the second block to be laid in such a corner was a small block of type face_to_end, then the robot gripper would collide with the block that is already laid. This is a consequence of the gripper design, as illustrated in figures 2.5.10 and 2.5.11. The size of a block, which could cause such a crash, had to be determined from both the block and the gripper widths. If a block is gripped exactly at its centre, the width of the block already laid is 100 ±2 mm, and the width of the gripper 164 mm (figure 2.5.10), then the maximum block size that could be laid safely without causing any damage to the block or robot is 364 mm. This is calculated using equation 9.1. Allowing for the block to be gripped at a lateral offset of 20 mm from its centre, with the tolerance in the widths of the blocks used, then the effective block size would be 408 mm.

$$block\_size = ((gripper\_width / 2) + block\_width) \times 2 \qquad \text{eqn 9.1}$$

From this calculation, a small block can be defined to be one which is less than 410 mm, allowing for an extra 2 mm as a safety margin. To define the block size for the corner blocks, first rules rule_ AO_16 and rule_ AO_17 are used to define attribute *block_size.*

rule_set_AO_16:	if *size* is =>410 then ***block_size*** is large
rule_set_AO_17:	if *size* is <410 then ***block_size*** is small

Rules using the information on each block size and type, and the assumptions on the generated project design, as discussed in section 9.4.1.2, were created to conclude attribute *corner_placing_order* (figure 9.12). These rules are integrated with 'C' procedures to process the project description files, and conclude the order of building in the project. The method of processing the project description files is incorporated into these rules, thus making *block_ID* of attribute *next_block* greater than *block_ID* of attribute *block* (i.e. moving in a clockwise direction for all levels). Only the order of laying the first block in a corner is determined, from which the second is concluded. An example of these rules is shown below, with figure 9.13 illustrating each rule. The full rule-set is included in appendix 9b.

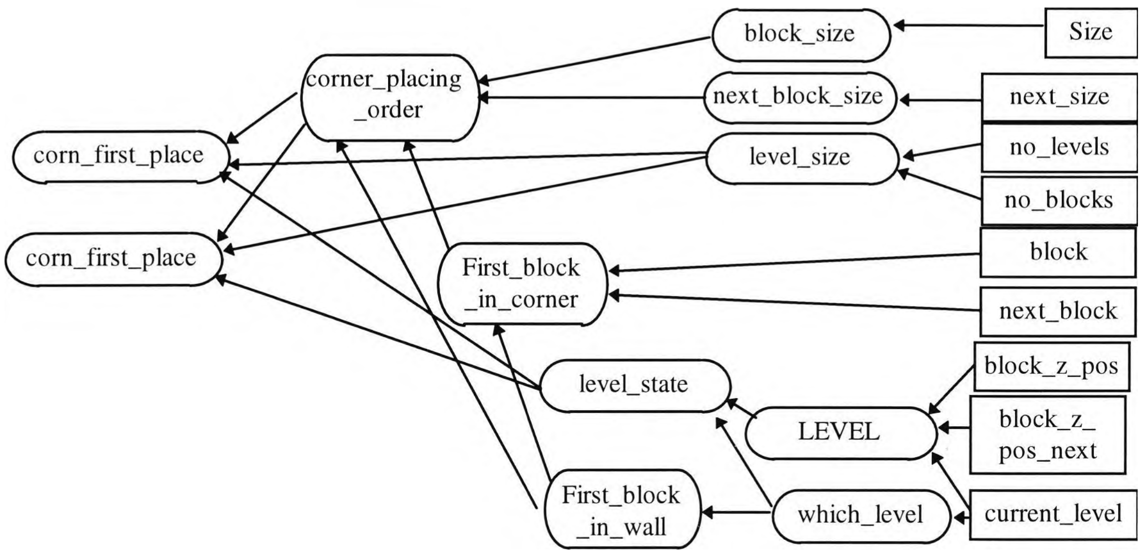


Figure 9.12 *Attribute list used to determine blocks in corner placing order*

rule_set_AO_18: if *next_block_size* is *small*
and *block_size* is *large*
and *first_block_in_corner* is *face_to_end*
and *First_Block_in_wall* is *face_to_end* or *end_to_face*
then ***corner_placing_order*** is *first*

rule_set_AO_19: if *next_block_size* is *large*
and *block_size* is *small*
and *first_block_in_corner* is *face_to_end*
and *First_Block_in_wall* is *face_to_end* or *end_to_face*
then ***corner_placing_order*** is *first*

rule_set_AO_20: if *next_block_size* is *small*
and *block_size* is *large*
and *first_block_in_corner* is *end_to_face*
and *First_Block_in_wall* is *face_to_end* or *end_to_face*
then ***corner_placing_order*** is *second*

rule_set_AO_21: if *next_block_size* is large
and *block_size* is large
and *first_block_in_corner* is *end_to_face*
and *First_Block_in_wall* is *face_to_end*
then ***corner_placing_order*** is *second*

rule_set_AO_22: if *first_block_in_corner* is *not_corner*
then ***corner_placing_order*** is *not_corner*

, where attribute *first_block_in_corner* is used to find the block type of the first block at a corner. This is determined using the assumptions discussed in section 9.4.1.2. An example of the rules concluding this attribute are shown below:

```
rule_set_AO_23: if First_Block_in_wall is face_to_end
                    and  block is end_to_face
                    and  next_block is face_to_end
               then first_block_in_corner is end_to_face

rule_set_AO_24: if First_Block_in_wall is face_to_end
                    and  block is face_to_end
                    and  next_block is face_to_end or end_to_face or not_corner
               then first_block_in_corner is not_corner
```

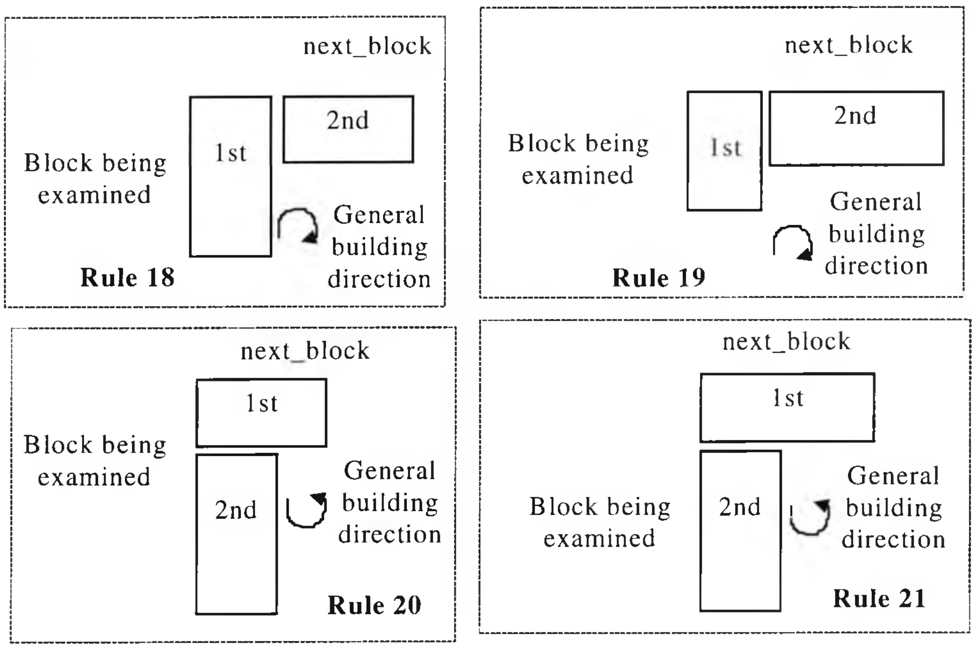


Figure 9.13 *Example of corner building order*

A 'C' procedure was written to produce a list of corners, each with the block_IDs of the first and the second blocks to be laid. This procedure uses the corner building rules, and the 'NEW project description' files, to generate the list. When processing a 'NEW project description' file, the first corner to be examined is the corner between the first two wall legs (figure 9.8). To determine the block_ID's for this list, special rules concluding the *block_ID* of the first block to be laid in each corner are presented as attribute *corn_first_place*. The *block_ID* of the second block of the same corner is presented as *corn_second_place*. An example of these rules is given below. These rules take into account the last corner of a level (e.g. rule_set_AO_28 for attribute *corn_second_place*).

```
rule_set_AO_25:  if corner_placing_order is first
                   then corn_first_place is block_ID

rule_set_AO_26:  if corner_placing_order is  second
                         and  level_state is = 0
                   then corn_first_place is block_ID+1

rule_set_AO_27:   if corner_placing_order is second
                  then corn_second_place is block_ID

rule_set_AO_28:  if corner_placing_order is first
                      and  level_state is ≠ 0
                 then corn_second_place is (block_ID+1)- level_size
```

Procedure for the building order of corner blocks (step 3, figure 9.5):

```
(i)-Get data from 'NEW project description' file
(ii)- attribute no_blocks = number of blocks in file, attribute no_levels=
   level(no_blocks), backward chain on attribute level_size.
(iii)-corner_ID = 1, x=1
(iv)-Repeat , Until x = no_blocks + 1
        - attributes block_ID =x, and current_level = level(x)
        - backward chain on attribute First_block_in_wall
        - get values from 'NEW project description' file for attributes
                -size = length(x), next_size = length(x+1), current_level= level(x)
                 Block_Z_pos = Z(x), next_block_Z_pos = Z(x+1),
                block = block(x), next_block = block(x+1)
        - forward chain on new attribute values
        - backward chain on attribute corner_placing_order
        - if value of attribute corner_placing_order = 'first' or 'second' then
              - backward chain on attributes corn_first_place and corn_second_place
             -corner = corner_ID
             -ID_1st_corner_build_block(corner) = value of attribute corn_first_place
              -ID_2nd_corner_build_block(corner)= value of attribute
                corn_second_place
             -corner_ID = corner_ID +1
        - x = x + 1
```

9.4.1.4 Masonry project symbolic representation

To enable the process of ordering the building task for a variety of project designs, a system for generating a symbolic representation describing these designs had to be established. Also, to establish the background knowledge for the order building rules, potentially destructive situations that may cause a collision consequently harming the robot and demolishing the assembled parts, also needed to be determined (figure 9.14). Examining the various possible project designs, which the RBES is expected to cover, it was clear that to avoid destructive scenarios, as in the case shown in figure 9.14, there is a need to build what is referred to as a step-shape followed by other shapes. From this observation, the various shapes had to be defined, each made up of three adjacent walls. These shapes are illustrated in figure 9.15, where each arrow is considered to be a wall leg.

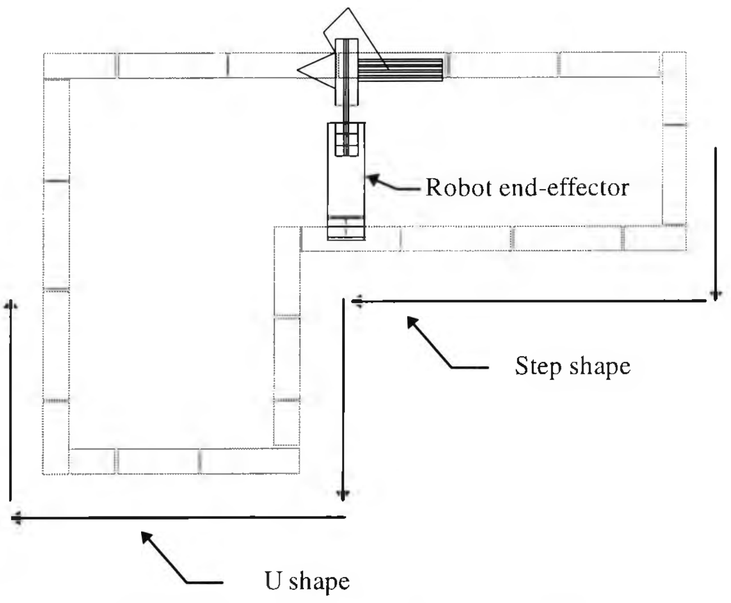


Figure 9.14 *Collision zone to be avoided*

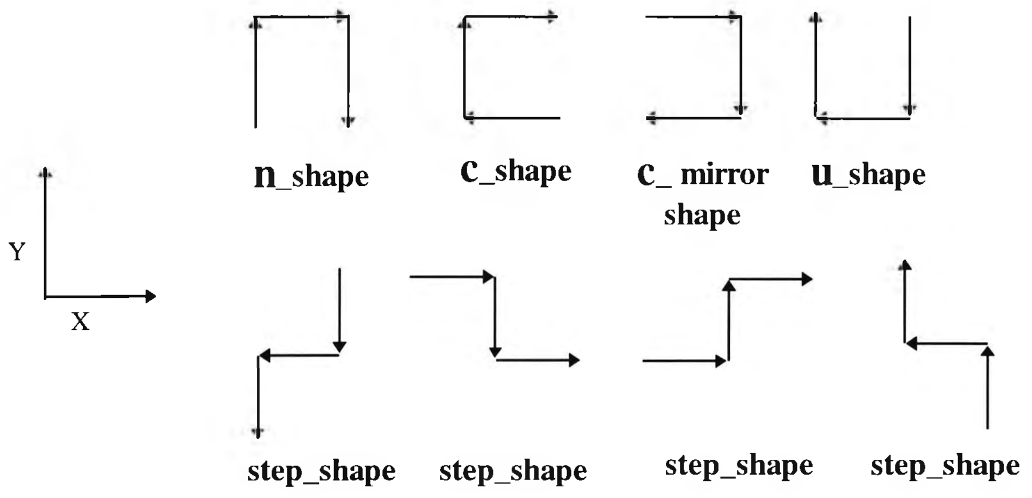


Figure 9.15 *Illustrations of the different shapes expected in the project designs and their names*

9.4.1.4.1 Project shape identification: step 4

Each masonry project design is translated into shapes, using the symbolic representation illustrated in figure 9.15. Using the 'NEW project description' file, described in section 9.4.1.2, the wall identification numbers and their directions for a project can be determined. This wall direction represents the orientation of the wall legs, determined when processing the project, starting from the first wall leg and proceeding in a clockwise direction (figure 9.8). These wall leg sets enable translation of the design project into shapes, by processing the description file in a clockwise direction, and forming three adjacent wall legs into a shape representation. For any project design, the first shape would be made up of wall identification numbers (1, 2 and 3), the second shape would be of wall legs (3, 4 and 5), and the last shape would be of walls (N-1, N and 1), where N is the number of wall legs in a project design. The shapes will be the same for each level. An example of rules written to determine the shapes in a project, using the direction of the wall legs, are rule _AO_29 and rule_ AO_30. The attributes used to conclude attribute *three_wall_shape* are shown in figure 9.16. The conclusion of attribute *three_wall_shape* will be the name of the shape and the wall identification numbers making up that shape:-

rule_set_AO_29: if *prev_wall_dirct* is *Y_pos_dirct*
and *wall_direction* is *X_pos_dirct*
and *next_wall_dirct* is *Y_neg_dirct*
and *wall_id* is ≠ *No_walls*
then ***three_wall_shape*** is *n_shape* and *wall_id*-1
and *wall_id* and *wall_id*+1

rule_set_AO_30: if *prev_wall_dirct* is *X_pos_dirct*
and *wall_direction* is *Y_neg_dirct*
and *next_wall_dirct* is *X_neg_dirct*
and *wall_id* is ≠ *No_walls*
then ***three_wall_shape*** is *c_mirr_shape*
and *wall_id*-1 and *wall_id* and *wall_id*+1

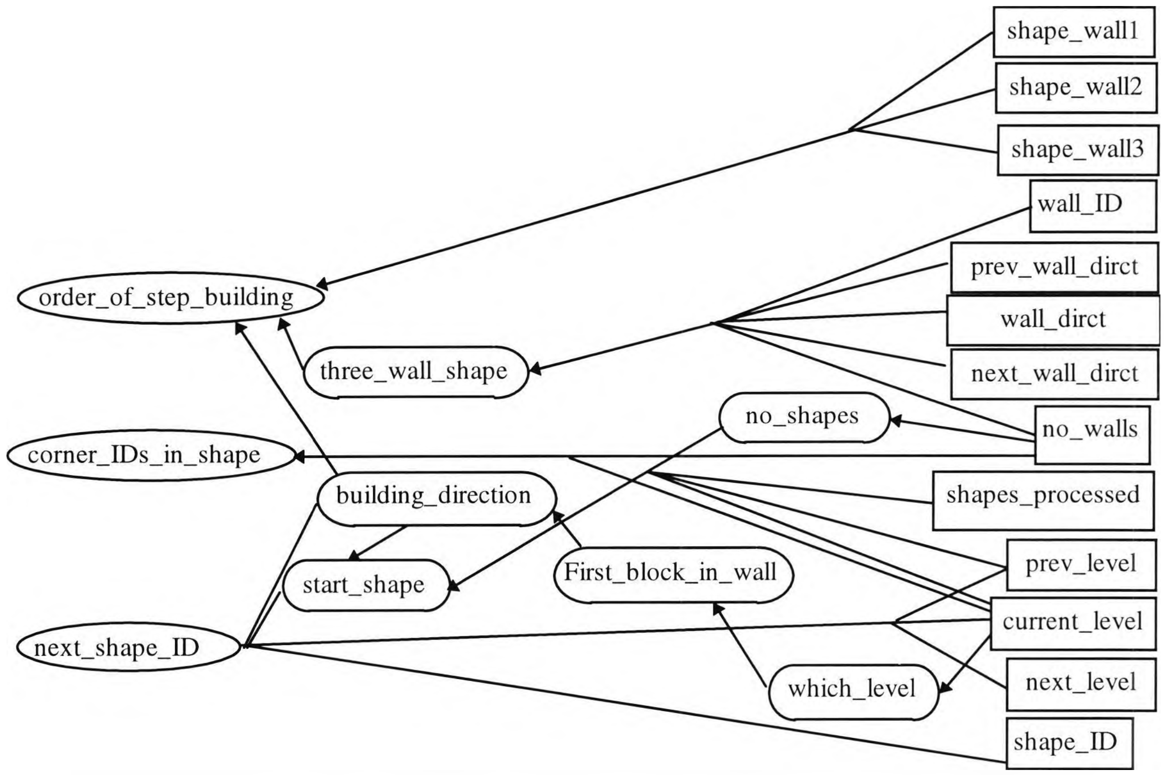


Figure 9.16 *Shape identification, and order of building the shapes attributes*

To gather more information on each shape, which is of benefit when ordering the building task, the corner identification numbers in each shape are determined first. For this, a rule with conclusion attribute *corner_IDs_in_shape* was developed. This uses an expression to calculate the result of that attribute, which relies on the first wall identification numbers of a shape concluded from attribute *three_wall_shape*. The rule is described as follows:

```
rule_set_AO_31:  if three_wall_shape is number
                    and  No_walls is number
                    and current_level is number
                 then corner_ids_in_shape is
                       @three_wall_shape + (@No_walls × (@current_level-1))
                   and  @three_wall_shape +(@No_walls × (@current_level-1))+1
```

To use the shape identification rules on designs of masonry projects, they were integrated into 'C' procedure developed to processes the data describing the project, to produce a shape list defining the design (step 4, figure 9.5). The result of this procedure is a list of shapes, each having information on its shape type, wall identification number of the shape, level and the corner_IDs of the shape.

Procedure to determine shape details of building project (step 4):

```
- attribute No_walls = number of walls in each level
- x =1 , shape_no =1, no_blocks = number of blocks in level
- attribute prev_wall_direction(x) = block_direction(x)
- Repeat, until block_direction(x) ≠ block_direction(x + 1)
     - x = x + 1
- x = x + 1
- attribute wall_direction = block_direction(x)
- Repeat, until x = no_blocks + 1
    - if level(x -1) ≠ level(x)
            - Repeat, until block_direction(x) ≠ block_direction(x + 1)
                    - x = x + 1
            - attribute prev_wall_direction = block_direction(x)
            - attribute wall_direction = block_direction(x + 1)
            - x = x + 1
    - shape_level[shape_no]  = level(x),
    - attributes current_level  = level(x)
    - Repeat, until block_direction(x) ≠ block_direction(x + 1)
            - x = x + 1
    - wall_ID = wall(x)
    - if x+1 > no_blocks
            - attribute next_ wall_direction = block_direction(1)
    - if x+1 < = no_blocks
    - attribute next_ wall_direction = block_direction(x + 1)
     - backward chain on attributes three_wall_shape
     - shape_type[shape_no] = conclusion 0 of attribute three_wall_shape
     - shape_wall1[shape_no] = conclusion 1 of attribute three_wall_shape
     - shape_wall2[shape_no] = conclusion 2 of attribute three_wall_shape
     - shape_wall3[shape_no] = conclusion 3 of attribute three_wall_shape
     - backward chain on attributes corner_IDs_in_shape
     - corner1_in_shape[shape_no] =conclusion 0 of attribute corner_IDs_in_shape
```

- corner2_in_shape[shape_no] =conclusion 1 of attribute *corner_IDs_in_shape*
- shape_no = shape_no + 1
- attribute *prev_ wall_direction* = value of attribute *wall_direction*
- attribute *wall_direction* = value of attribute *next_wall_direction*

9.4.1.4.2 Order of building shapes: step 5

After making a list of the shapes for a designed project, the next step is to decide on the order in which they should be built, which automatically concludes the order in which the wall legs should be built. It was decided that the step-shapes should be built first for each course-level, thus allowing the robot enough space to manoeuvre itself free of collisions (figure 9.14). Rules to conclude the order of building the steps were created. These rules use the shape list, produced from the shape identification rules (step 4, figure 9.5) and the procedure described in section 9.4.1.4.2, in conjunction with a 'C' procedure, described later in this section, developed to produce the order of building the wall legs. The result of this procedure is two wall identification lists for each level. The first list is the building order for the part walls making up shapes that are of type 'step', called wall_order_step. The second list is the building order for the part walls making up shapes that are anything other than shape type 'step' (e.g. n_shape), called wall_order_shape. The procedure and rules were developed with consideration of the method in which the shape list was created (i.e. in a clockwise direction, starting from the lowest course-level upwards).

Attributes needed for determining the building order for the shapes are given in figure 9.16. Attributes *start_shape* and *next_shape_ID* determine the shape identification number, using the shape list in the order which they should be built. This is achieved using the information on the general building direction for each level, using attribute *Building_direction*. Using the identification number of the next shape to be built, attribute *order_of_step_building* determines the wall identification numbers of that shape, in the order they should be built. The result is added to the wall_order_step or the wall_order_shape list, depending on the shape type.

Attribute *start_shape* determines the number of the first shape to start processing, for each level, concluded using the following rules:

```
rule_set_AO_32: if Building_direction is clockwise
                    and shapes_processed is >= no_shapes
                then start_shape is (@no_shapes × @prev_level)+1

rule_set_AO_33: if Building_direction is anticlockwise
                    and shapes_processed is >= no_shapes
                then start_shape is @no_shapes× (@current_level+1)
```

, where no_shapes is the number of shapes for each course-level, and prev_level is the course-level number previous to the one being processed. After, determining the shape to start processing for each course-level, attribute *next_shape_ID* concludes the next shape to be processed in that level. The rules concluding attribute *next_shape_ID* takes into consideration the change in the level being processed, as shown in rule_AO_34 below. If a change in the level is detected, then the shape to be processed next will be the shape determined using rules_AO_32-3 for attribute *start_shape*:

```
rule_set_AO_34: if Building_direction is clockwise or anticlockwise
                    and shape_ID is = 0
                    and start_shape is number
                then next_shape_ID is @start_shape

rule_set_AO_35: if Building_direction is anticlockwise
                    and shape_ID is !=0
                    and current_level is number
                    and next_level is =@current_level
                then next_shape_ID is @shape_ID-1
```

Attribute *order_of_step_building* uses the shape_type and the wall identification numbers of that shape, and determines the order of building. These rules are listed below:

```
rule_set_AO_36:  if Building_direction is clockwise
                    and shape_wall1 is number
                    and shape_wall2 is number
                    and shape_wall3 is number
                    and three_wall_shape is string
                 then order_of_step_building is @three_wall_shape and
                         @shape_wall1 and @shape_wall2 and @shape_wall3
```

```
rule_set_AO_37: if Building_direction is anticlockwise
                    and shape_wall1 is number
                    and shape_wall2 is number
                    and shape_wall3 is number
                    and three_wall_shape is string
                  then order_of_step_building is @three_wall_shape and
                      @shape_wall3 and @shape_wall2 and @shape_wall1
```

The way in which the rules determining the order of building the shapes and wall legs is described in the procedure below:

Procedure for ordering the shapes (step 5a, figure 9.5):

```
-shapes_processed = 0, x = 0, no_shapes = number of shapes in the building project
-attribute shape_ID = 0
-attribute No_walls = number of walls in each level
-attribute shapes_processed = shapes_processed
-attributes current_level = shape_level(1) , prev_level = shape_level(1)
            next_level = shape_level(1),
-backward chain on attributes building_direction, start_shape, next_shape_ID
-x = result of attribute next_shape_ID
-Repeat, Until shapes_processed = no_shapes
        -attribute Three_wall_shape = shape_type[x]
        -attributes shape_wall1 = shape_wall1[x] , shape_wall2= shape_wall2[x]
                    shape_wall3 = shape_wall3[x]
        - backward chain on attribute order_of_step_building
        -y=0 , level = shape_level[shapes_processed],
        -shape_type = conclusion 0 of attribute order_of_step_building
        -if shape_type = 'step'
          -Repeat, Until y =3
          -wall_order_step[level][y]= conclusion y of attribute order_of_step_building
        -if shape_type ≠ 'step'
          -Repeat, Until y =3
          -wall_order_shape[level][y]= conclusion y of attribute order_of_step_building
-attribute shape_ID = x
-shapes_processed = shapes_processed + 1
-attribute shapes_processed = shapes_processed
-attribute current_level = shape_level[shapes_processed+1]
-forward chain current_level (to determine building direction of the next level to be
built)
-attribute prev_level = shape_level[shapes_processed -1]
-attribute current_level = shape_level[shapes_processed]
-attribute next_level = shape_level[shapes_processed +1]
-backward chain attributes start_shape, next_shape_ID
- x = result of attribute next_shape_ID
```

The result of this procedure is the two lists of wall identification numbers. These lists, wall_order_step and wall_order_shape, are then appended, in that order. This is followed by appending the resulting list for each level together, starting with the lowest level and going upwards (step 5b, figure 9.5). There will be repetition of wall numbers, the reason being the way the shapes were determined with lapping walls, as described in section 9.4.1.4. This is taken care of by deleting any repetitions in the final list, keeping the first instance of each wall number that is repeated (step 5c, figure 9.5). The result is stored in a list called wall_building_list.

9.4.1.5 Order of block building: step 6

The next step is to make a list of the block identification numbers in the order they should be laid, using the list of walls 'wall_building_list'. Rules determining the block laying order were created using the attributes shown in figure 9.17. A 'C' procedure, described later in this section, developed to process the wall_building_list, uses these rules and the 'NEW project description' file to create a list of block identification numbers.

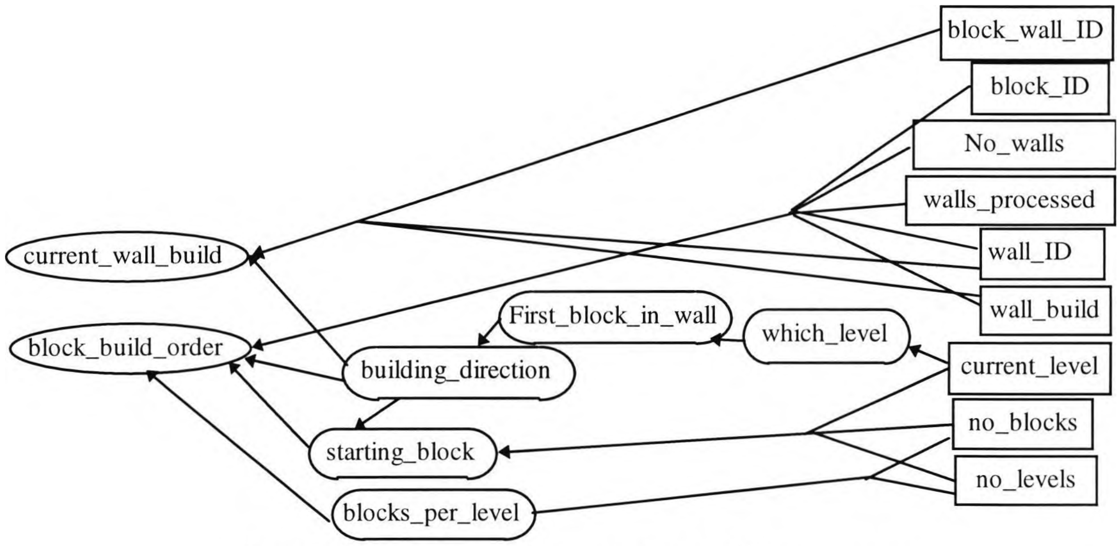


Figure 9.17 *Block building order attribute network*

Each block position in the project description file is examined to check whether it is to be the next block position to be laid, when determining the order in which the blocks will be laid. This is determined using rules for attribute *block_build_order*. The order of examining the blocks in the project description file is decide on using the conclusion to attributes *starting_block* and *block_build_order*. Attribute *starting_block* gives the block identification number of the first block to be examined in each level. This is concluded from the general building direction for that level, and information in the 'NEW project description' list. An example of these rules is rule_AO_38:

```
rule_set_AO_38: if current_level is number and no_levels is number
                   and  no_blocks is number
                   and Building_direction is clockwise
              then starting_block is  1+  (@current_level-1) ×
                              (@no_blocks/@no_levels)
```

Attribute *block_build_order* is used to conclude the block laying order for each course-level, using the wall leg order list 'wall_building_list', described in section 9.4.1.5. The order of laying the blocks in each wall leg is determined form the general building direction of each level (described in section 9.4.1.3.1). Using the procedure described below, each wall leg is processed in the list order. The rules created to conclude attribute *block_build_order* determine whether the block currently being examined, in the project description file, is the block identification number of the next block position to be laid, as well as the block identification number of the next block position to be examined. The first conclusion value to attribute *block_build_order* will always be the Block_ID of the next block position to be examined. If the current block position being processed is concluded to be the next block position to be laid (e.g. rule _AO_39 below), this is represented as the value of the second conclusion to attribute *block_build_order*, which is added to the block order list, called 'block_order'. The rules concluding this attribute detect the end of processing each course-level, and use the value of attribute *starting_block* to determine the next block position to be processed in the next level up (rule_set_AO_41 below):

```
rule_set_AO_39:  if Building_direction is clockwise
                     and wall_id is number and starting_block is number
                     and No_walls is number
                     and current_level is number and blocks_per_level is number
                     and wall_build is =@wall_id
                     and walls_processed is <@No_walls
                     and block_ID is >=1+((@current_level-1) × @blocks_per_level)
                          and     blocks_per_level × @current_level
                 then block_build_order is @block_ID+1 and @block_ID

rule_set_AO_40:  if Building_direction is clockwise
                     and wall_id is number and starting_block is number
                     and No_walls is number
                     and current_level is number and blocks_per_level is number
                     and wall_build is !=@wall_id
                     and walls_processed is <@No_walls
                     and block_ID is >=1+((@current_level-1) × @blocks_per_level)
                          and <@blocks_per_level x @current_level
                 then block_build_order is @block_ID+1

rule_set_AO_41:  if Building_direction is clockwise and wall_id is number
                     and No_walls is number and starting_block is number
                     and current_level is number and blocks_per_level is number
                     and wall_build is ≠ @wall_id
                     and walls_processed is <@No_walls
                     and block_ID is >=@blocks_per_level × @current_level
                 then block_build_order is @starting_block
```

Attribute *current_wall_build* is used in the process of examining each wall leg. Rules were created to give an indication of the status of the current wall being examined. If all the blocks in the wall being processed have been examined, then the conclusion is value 'next' (rule _AO_42 below), which indicates the need to start examining the next wall leg in the wall list. Conclusion value 'same' indicates that the blocks in that wall have not all been processed (rule _AO_43 below):

```
rule_set_AO_42: if block_wall_id is last_block
                     and Building_direction is clockwise
                     and wall_id is number
                     and wall_build is = @wall_id
                then current_wall_build is next
```

```
rule_set_AO_43: if block_wall_id is first_block or  middle_block
                   and Building_direction is clockwise
                   and wall_id is number
                   and wall_build is = @wall_id
               then current_wall_build is same
```

Using the 'NEW project description' file (section 9.4.1.2), the wall_build_list, and order of block building rules, the sequential procedure to produce the block laying order list is described below:

Procedure to determine the block building order (step 6):

```
- level =1 , no_blocks = number of block in the project
- walls = number of walls in each level , z = 0
-attribute No_walls = walls
-attribute current_level = level, forward chain on current_level
-attributes no_levels = level[no_blocks], no_blocks = no_blocks
-backward chain on attribute starting_block
- x = conclusion value of attribute starting_block
-attribute block_ID = x
-walls_processed =0 , y = 0
-attribute wall_build = wall_building_list[z]
-Repeat, Until walls_processed = (no_walls × no_levels)
    -attributes  block_wall_ID = block_wall_ID(x),  wall_ID = wall(x)
    -backward chain on attribute block_build_order
    - x = value of conclusion 0 of attribute block_build_order
    - if number of conclusion of attribute block_build_order > 1
        -block_order[y] = value of conclusion 1 of attribute block_build_order
        -y = y +1
    -set output hook to myfunction
    - backward chain on attribute current_wall_build
    -set output hook to the ES shell output function
    -if level < level(no_blocks) and value of attribute walls_processed = walls
        -level = level +1
        -attribute current_level = level, forward chain on new value
        -backward chain on block_build_order
        - x = value of conclusion attribute block_build_order
        - attribute walls_processed = 0
    if level = level(no_blocks) and value of attribute walls_processed = walls
        -attribute walls_processed = walls +1
    - attribute block_ID = x
```

One of the main advantages of the adopted approach is the ability to install user defined functions in strategic parts of the code modules. The ES shell function

Ks_set_outputhook(*myoutput*) installs an output hook into the complied module. The output function *myoutput*, is called whenever any value is concluded for an attribute. In the main calling procedure, the output hook is installed at a point so that function *myoutput* is called whenever attribute *current_wall_build* is concluded. Function *myoutput* was written for the purpose of updating the value of walls_processed, used in the calling function, as well as obtaining the next wall identification number from the order of building the wall legs list (wall_building_list[]), to update attributes *wall_build*: Function *myoutput*:

```
- if value of attribute current_wall_build  = 'next'
        - get next wall wall_ID to be built from wall_building_list[]
        -attribute wall_build = wall_ID
        -wall_processed = value of attribute walls_processed
        - attribute walls_processed = walls_processed + 1
```

9.4.1.6 Re-order block building list: step 7

The next step, after creating the 'block_order' list (step 6, figure 9.5), is the incorporation of the order of building the corners. This is accomplished using the corner list, which contains the block identification numbers of each corner, and the order in which they should be built. The method of creating the corner list (step 3, figure 9.5) was described in section 9.4.1.3. The reordering of the 'block_order' list is carried out using a procedure written in 'C' programming language. This procedure, described below, first creates a link list, in the 'C' programming language, of the 'block_order' list, and searches for corner blocks and repositions them in the 'block_order' list. The use of linked lists facilitates the search for, insertion, and deletion of items in the list:

Procedure reorder block building list (step 7, figure 9.5):

```
-x= 1, no_walls = number of walls in each level
-make a linked list from block_order[] list, block_list
-Repeat, Until x = no_blocks in project
        -x = ID of first item in block_list
        -if block(x) = 'not_corner'
                x = ID of next item in block_list
        -if block(x) = 'face_to_end' or 'end_to_face'
                - wall = wall(x) + (level(x) -1) × no_walls
                -if x = ID_1st_corner_build_block(wall)
```

```
            or x = ID_1st_corner_build_block(wall)
              -if x = ID of last block in block_list
                  -search block_list for
block_ID=ID_2nd_corner_build_block(wall)
              -if x $\neq$ ID of last block in block_list
                   -if x = ID_1st_corner_build_block(wall-1)
                       -search block_list for block_ID
                            =ID_2nd_corner_build_block(wall-1)
        -delete it from its position in the list and add it after current
          position of block_list,
        - x = next item in block_list
        -if x = ID_2nd_corner_build_block(wall)
               -search block_list for block_ID =ID_1st_corner_build_block(wall)
        -if x $\neq$ ID_2nd_corner_build_block(wall)
               - if wall $\neq$ 1
                    -if  x = ID_2nd_corner_build_block(wall-1)
                        -search block_list for block_ID
                               =ID_1st_corner_build_block(wall-1)
        -delete it from its position in the list and add it before current
          position of block_list,
        - x = next item in block_list
```

9.4.1.7 Yaw-axis building direction: step 8

Due to the design of the robot end-effector, which has a large motor attached to its side, as shown in figure 9.18, care has to be taken to accommodate it. An example of the robot laying the block in a way that can damage it is shown in figure 9.18. In this case, if the robot is lowered to lay the gripped block the robot motor could collide with the block below it. For the case shown in figure 9.18, the correct way of laying is by first rotating the robot yaw-axis by an angle of 180°. Assuming all wall legs are parallel to either the robot's x and y axes, there are only four yaw angles that the robot can use when laying blocks. These angles are yaw = 0°, 90°, -90°, 180°, as illustrated in figure 9.19.

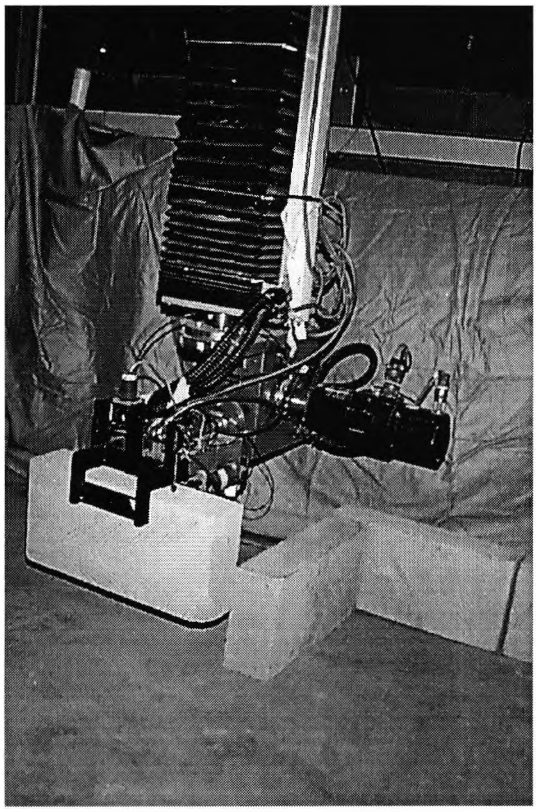


Figure 9.18 *Robot placing brick using the wrong yaw-axis angle, which can impact the robot motor on to the already laid block*

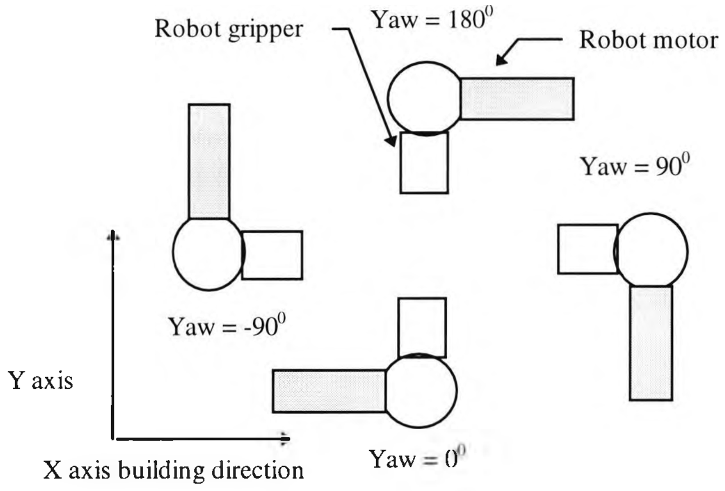


Figure 9.19 *Robot yaw-axis orientations*

To determine the yaw angle to be used, the critical yaw angle has to be determined first. Using the attributes shown in figure 9.20, rules were written to conclude the yaw angle used to place each block. These rules rely on the information compiled on each block in the project, stored in the 'NEW project description' file, which was described in section 9.4.1.2. A 'C' code procedure was developed to process the information from this file using these rules, to conclude the yaw angle of laying a block. The result of this procedure is a pre-processed list of yaw angles for each block, called 'yaw_building_direction' (step 8, figure 9.5).

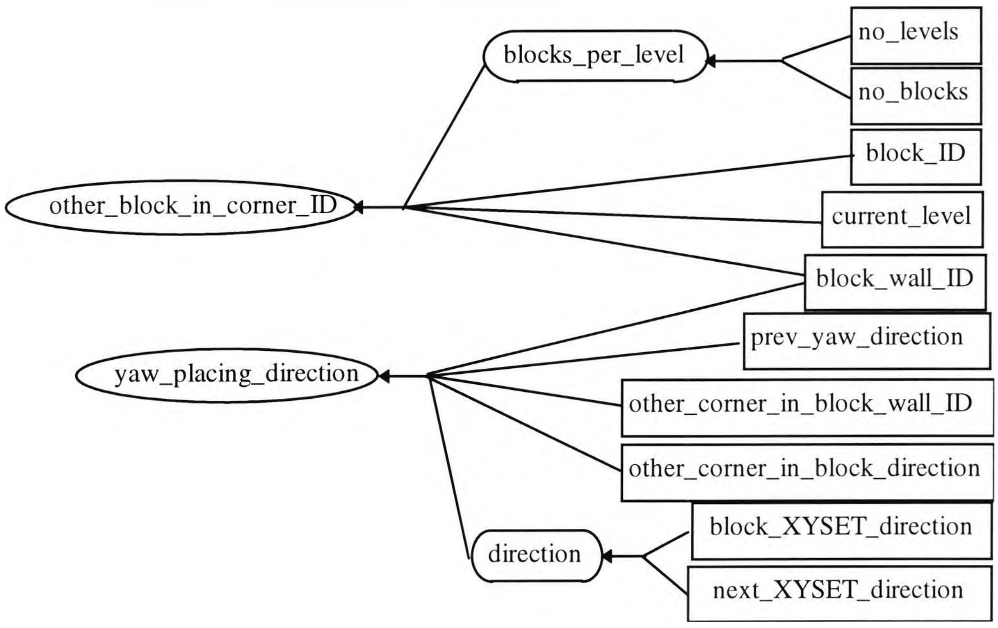


Figure 9.20 *Yaw-axis building direction network of attributes*

To create the rules concluding attribute *yaw_placing_direction*, various scenarios had to be considered. An example of this is shown in figure 9.21a and 9.21b. These rules take into consideration the order in which the blocks are processed in the 'C' code procedure which calls them. The rules use the block direction of the corner blocks to determine the yaw angle used to lay each block in that corner. An example is shown in rule_AO_44 and rule_AO_45 below, which are illustrated in figures 9.21a and 9.21b respectively. Once the direction of building a block corner is determined, the other blocks in the same wall will be laid using the same yaw angle e.g. rule_AO_46 below:

```
rule_set_AO_44:  if *direction* is *Y_pos_dirct*
                    and *block_wall_id* is *first_block*
                    and *other_corner_block_direction* is *X_neg_dirct*
                    and *other_corner_block_wall_id* is *last_block*
                 then ***yaw_placing_direction*** is -90
```

rule_set_AO_45: if *direction* is *X_neg_dirct*
and *block_wall_id* is *last_block*
and *other_corner_block_direction* is *Y_pos_dirct*
and *other_corner_block_wall_id* is *first_block*
then ***yaw_placing_direction*** is 0

rule_set_AO_46: if *block_wall_id* is *middle_block*
and *Prev_yaw_direction* is number
then ***yaw_placing_direction*** is *@prev_yaw_direction*

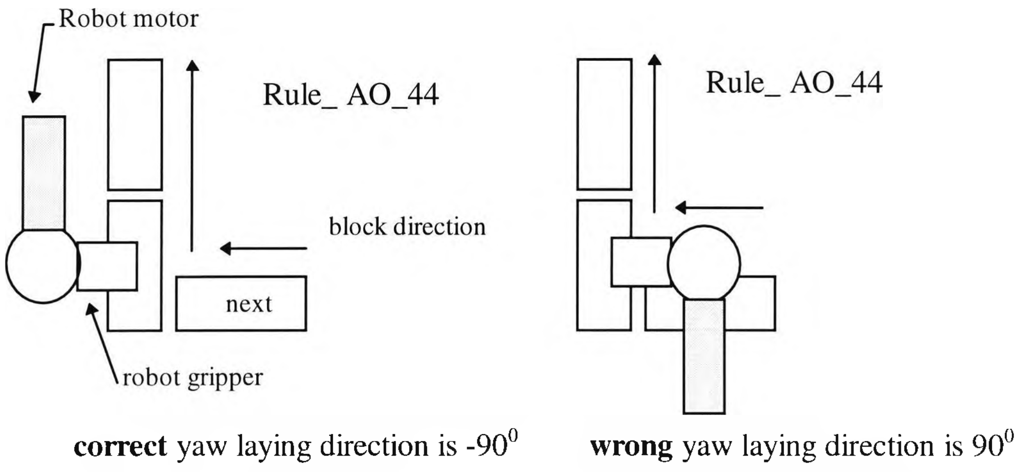

Robot motor
Rule_ AO_44
block direction
next
robot gripper
correct yaw laying direction is -90⁰
Rule_ AO_44
wrong yaw laying direction is 90⁰

Figure 9.21a *Illustration of (Rule_ AO_44) showing the correct and the wrong way to lay a block*

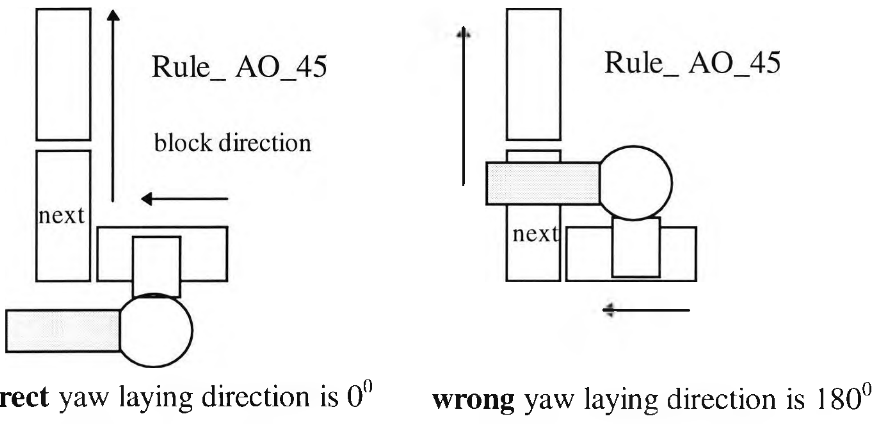

Rule_ AO_45
block direction
next
correct yaw laying direction is 0⁰
Rule_ AO_45
next
wrong yaw laying direction is 180⁰

Figure 9.21b *Illustration of (Rule_ AO_45) showing the correct and the wrong way to lay a block*

To assist the 'C' code procedure in processing the 'NEW project description' file, rules were created to identify corner blocks. Given the location of a block in the wall (*block_wall_id*), these rules, listed below, will conclude whether the block is a corner block or not. If it is, then the block identification number is determined (rule_ AO_48 below), otherwise a result of 'not_corner' is given (rule_ AO_47 below) as conclusions to attribute *other_block_in_corner_id*. The use of these rules are made clear in the description of the following procedure below:

```
rule_set_AO_47:  if block_wall_id is middle_block
                    then other_block_in_corner_id is not_corner

rule_set_AO_48:  if block_wall_id is first_block
                    and  blocks_per_level is number  and current_level is number
                    and block_ID is >1+ ((@current_level 1) × @blocks_per_level)
                            and <@blocks_per_level x @current_level
                 then other_block_in_corner_id is @block_ID-1
```

The procedure to determine the robot yaw building direction for each block (step 8) is:

```
-no_walls =  number of walls for each block , x = 1
-no_blocks = number of blocks for building project
-attributes No_walls = no_walls, No_levels = level(no_blocks), current_level = 1
-backward chain on attribute starting_block
-attribute block_ID = value of attribute starting_block
- Repeat, Until x = no_blocks
        -block_ID = attribute block_ID , x = x + 1
        -attributes direction = block_direction(block_ID), current_level =
        level(block_ID)
        -attribute block_to_wall_ID = block_to_wall_ID(block_ID)
        -backward chain on attribute other_block_in_corner_ID
        -if value of attribute other_block_in_corner_ID ≠ 'not_corner'
                - y = value of attribute other_block_in_corner_ID
        -attribute other_block_in_corner_direction = block_direction(y)
        -attribute other_corner_block_wall_ID = block_wall_ID(y)
        -backward chain on attribute yaw_placing_direction
        -yaw_building_direction[block_ID] = value of attribute yaw_placing_direction
        -attribute prev_yaw_placing_direction = yaw_building_direction[block_ID]
        -attribute blocks_processed = x
        -y = value of attributes (blocks_per_level × current_level)
        -if  x = y
                -level = value of attribute current_level
                -attribute current_level = level +1
                -forward chain attribute current_level (this will change the value of
                        attribute starting_block)
        -attribute current_level = level(block_ID)
```

-backward chain on attribute *next_block_ID*
-y = value of attribute *next_block_ID*
-attribute *block_ID* = y

9.4.1.8 Generating the theoretical task: step 9

Applying the assembly rules to the generated CAD design, using the various procedures described earlier (steps 2-8, figure 9.5), the information generated is used in generating the 'theoretical task' (step 9, figure 9.5. Also see figure 1.7).

Using the yaw_building_direction list, as described in section 9.4.1.7, the 'block_order' list described in sections 9.4.1.5 and 9.4.1.6, the 'project description' list generated using the CAD facility described in section 9.4.1.1, and the 'NEW project description' list described in section 9.4.1.2, a new description list is produced. This list is built in the order in which the blocks are laid and contains a description of each masonry unit as follows:

$$(block\_ID, X, Y, Z, yaw\_angle)$$

, where *block_ID* is the block identification number, X, Y, Z the centroid co-ordinates of the target location relative to a ground datum, and *yaw_angle* the yaw angle to be used by the robot when laying the block.

9.4.1.9 Testing the assembly rules

The assembly rules were tested in four stages. The first two stages were carried out in the ES shell environment. In the first stage of testing, each rule-set making up the complete assembly rules was verified individually, which resulted in a list of missing rules. A process of selecting and adding the missing rules from that list was then carried out. This ensured that every single possible combination of events was accounted for.

The second stage of testing the rule base was done by entering different combinations of input values in attributes, to simulating different scenarios, then using backward and forward chaining to check and verify the various conclusions.

The third stage of testing was carried out during the various stages of development of the rule sets and their calling ‘C’ coded procedures. Each rule-set was tested on a ‘project description’ file generated for the project shown in figure 9.7. This file is listed in appendix G.1.2. The forth and final stages of testing was carried out using the main ‘C’ coded procedure, in which the various procedures are integrated. Again, this was tested on the same ‘project description’, and the results checked and found to be satisfactory. The resulting ‘theoretical task’ file is included in appendix G.1.2.

9.4.2 Expert system: block picking

A RBES was developed to enable the robot to perform the task of picking up a block in an intelligent way, making run-time decisions. The ‘block-pick’ RBES uses the process developed to perform the block picking-up task, as described in chapter 2. The difference in this case is that no assumptions are made about the location of the conveyor and the block. Instead, every detail is intelligently checked with the aid of the conveyor and gripper sensors, thus enabling the robot to asses a variety of situations and react accordingly. This is done by using robot functions. These include a target detection function, from which the conveyor location and the BPP are determined prior to the start of the block picking-up process. These two procedures are described in chapter 5, and make up stage 1 of the building process illustrated in figure 2.2. Another function calculates the block dimensions and its orientation relative to the conveyor. This process is described in chapter 2.3.1.2 (figure 2.3.6). A function that adjusts the BPP is also used. This incorporates the conveyor sensor information, making up the first part of adjusting the BPP for safe pick-up (described in chapter 2.3.1.3). These two procedures make up stage 2 of the building process, as illustrated in figure 2.2. The result of this stage is the new block pick position (NBPP). The next function checks the gripper to the block angle, and adjusts for it. This is part of stage 3 of the building process, as illustrated in figure 2.2. The final adjustment is made to the horizontal distance of the gripper to the block, which is checked and adjusted using the ultrasonic H_sensor, as described in chapter 2.4.3.2.

The RBES is embedded in the calling programme, which performs the block picking process. The robot can perform the task with minimum human intervention, under constrained collision risk. Also, as the robot assess and reacts to variations in pre-programmed task, at run time, this minimises the risk of having to interrupt and restart the robot.

To make possible the RBES, information was established by experiments carried out to pick-up blocks, using the procedures mentioned above. Once this was carried out, rules were built to express the significance of the various combinations of events. An example of that is, if the gripper is open and the robot is confirmed to be at the calculated block pick-up position and all sensor readings indicate that a block is below the robot end effector and the conveyor confirms that a block has arrived at the block pick-up position on the conveyor, then all that information confirms that a block is ready to be picked safely.

In the following sections, a more detailed explanation of the way the RBES was developed, and integrated with the robot functionality, is provided.

9.4.2.1 Process of building the expert system

The main goal of the RBES is to pick a block, which is indicated as the conclusion attribute '*Pick_block*'. An illustration of the various attributes making up the RBES, and their dependencies, is shown in figure 9.22. This gives the various premise attributes that are immediately linked to the conclusion attribute, and the how they are networked. The attributes listed on the left hand side of the diagram are all input attributes. They are mainly information gathered by interrogating the robot for its position, or from the various sensors in the robot cell. The critical states of the input attributes had to be established, e.g. for the *Gripper_data* attribute, it was either open, closed or unknown. The meanings of the various inputs for robot positions and situations had to be defined. For example, when the robot was at a certain height, the vertical sensor reading is relevant (e.g. if its reading at that point indicates a level at which the conveyor might be). All other attributes were concluded from such values.

The process of building the RBES involved starting from the goal attribute and finding what conditions were needed for success, using the knowledge available. Below is an explanation of each attribute concluded, with examples of the rules that conclude them. The complete rule set is included in appendix G.

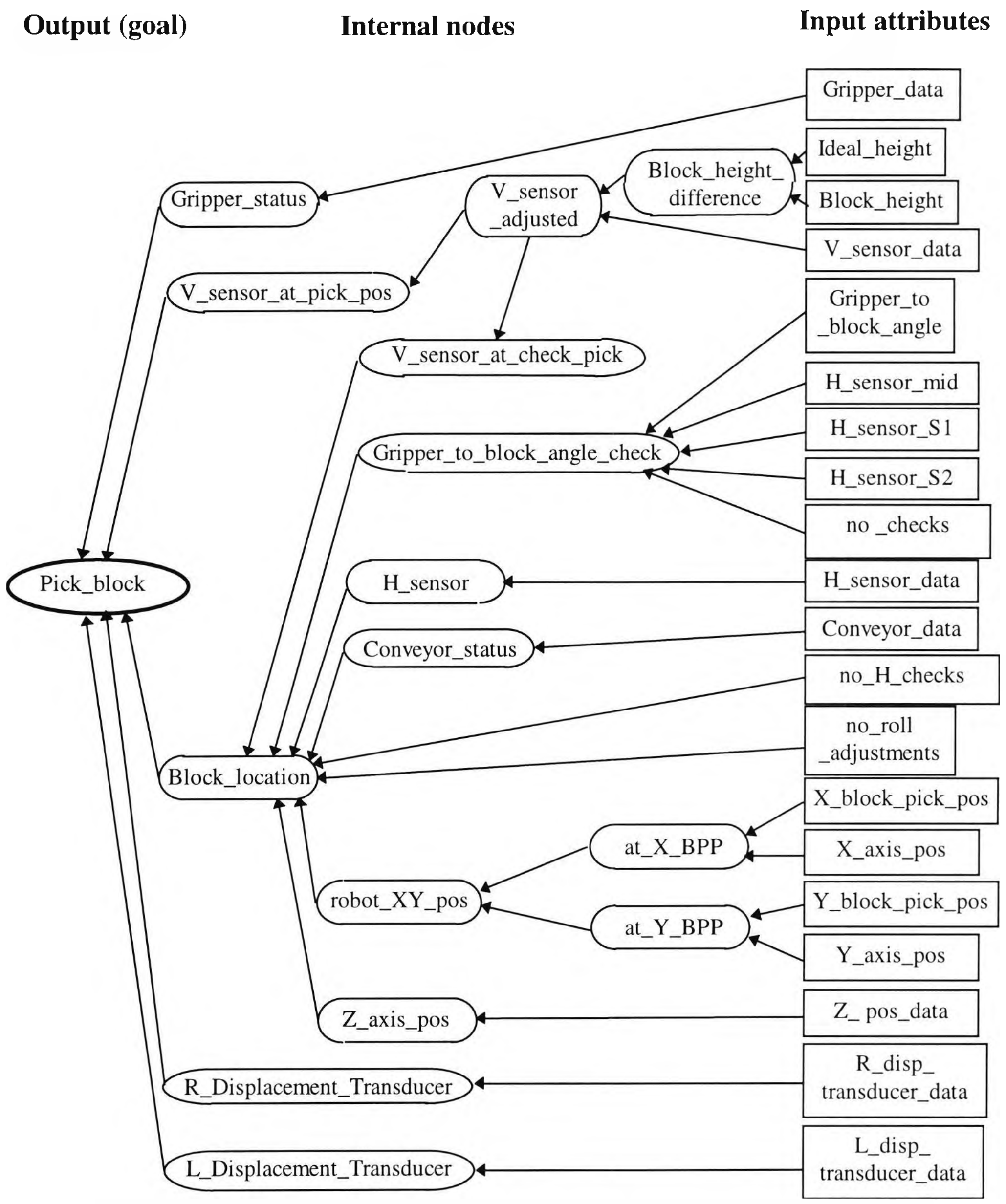


Figure 9.22 *Network of attributes for the block picking expert system*

9.4.2.1 Picking up a block

To start building the system, the conditions to successfully pick a block had to be established. That was done using the knowledge available on the robot position, block location, and the sensor information. This established the attributes concluding the goal attribute of the system, *Pick_block*. Table 9.2 shows a list of these attributes, with a brief explanation on each. How each of these attributes are concluded is described later on in this section. The rule indicating the success of the block picking up operation is as follows:

```
rule_set_BP_1:  if  block_location is in_position and  gripper_status is closed
                     and Z_axis_pos is at_pick_pos
                     and Horizontal_sensor is at_safe_to_pick
                     and V_sens_at_pick_pos is at_pick_pos_level
                     and R_Disp_Transducer is at_safe_height_to_pick
                     and L_Disp_Transducer is at_safe_height_to_pick
                then  pick_block is block_pick_successful
```

Table 9.2 *List of attributes that are immediately linked to attribute 'Pick_block'*

Attribute name	Attribute description
Pick_block	Our goal is to pick-up a block, this conclusion attribute will indicate the success, failure, or action to be taken in the block picking-up process.
Gripper_status	The status of the gripper, whether it is closed or opened or if the status is unknown
V_sensor_at_pick_pos	Reading of the vertical sensor at the point when the robot end-effector is at the block pick position
Block_location	This gives the status of the block location, e.g. if it is in position ready to be picked up, or not.
R_Displacement_transducer and, L_Displacement_transducer	Status or position of the transducer

Other rules that support the process of picking up a block are given below. The conclusions for each rule are acted on by the robot, using the methods described in section 9.4.2.3.

Rule_BP_2 triggers the function to open the gripper. At the time when it has been confirmed that there is a block in the correct position to be picked-up, the robot end-effector being at a height where it can detect the block, both displacement transducers inactive and the gripper is closed, we conclude that we need to open the gripper:

rule_set_BP_2: if *block_location* is *in_position* and *gripper_status* is *closed*
and *Z_axis_pos* is *at_check_pick_pos*
and *R_Disp_Transducer* is *inactive*
and *L_Disp_Transducer* is *inactive*
then ***pick_block*** is *open_gripper*

Rule_BP_3 indicates that the gripper is at the correct position to pick up a block, but not low enough, which concludes the need to lower the gripper:

rule_set_BP_3: if *block_location* is *in_position*
and *gripper_status* is open
and *Z_axis_pos* is *at_pick_pos*
and *H_sensor* is *at_safe_to_pick*
and *V_sens_at_pick_pos* is *at_pick_pos_level*
and *R_Disp_Transducer* is
touching_block_but_not_safe_to_gripp
and *L_Disp_Transducer* is
touching_block_but_not_safe_to_gripp
then ***pick_block*** is *lower_Z_axis_by_4mm*

Rule_BP_4 checks that the gripper is at the right position to grip the block, in which case the next step would be to close the gripper. In response, the robot acts on this conclusion and calls a function that closes the gripper:

rule_set_BP_4: if *block_location* is *in_position*
and *gripper_status* is *open*
and *Z_axis_pos* is *at_pick_pos*
and *Horizontal_sensor* is *at_safe_to_pick*
and *V_sens_at_pick_pos* is *at_pick_pos_level*
and *R_Disp_Transducer* is *at_safe_height_to_pick*
and *L_Disp_Transducer* is *at_safe_height_to_pick*
then ***pick_block*** is *close_gripper*

Other examples of the rules follow:

rule_set_BP_5: if *block_location* is *in_position*
and *gripper_status* is *closed*
and *Z_axis_pos* is *at_pick_pos*
and *H_sensor* is *not_safe_to_pick* or *at_safe_to_pick*
and *V_sens_at_pick_pos* is *at_pick_pos_level*
and *R_Disp_Transducer* is
touching_block_but_not_safe_to_gripp
and *L_Disp_Transducer* is
touching_block_but_not_safe_to_gripp
then ***pick_block*** is *block_picked_but_not_safely*

rule_set_BP_6: if *gripper_status* is *unknown*
then ***pick_block*** is *check_gripper_status* and *not_safe_abort*

9.4.2.2 Determining the various attributes

In the following section, an explanation of the various attributes used in the rule-base is given, starting with the attributes that are directly influenced by the input attribute.

9.4.2.2.1 Determining sensor readings

Data from the two ultrasonic sensors and the two displacement transducers mounted on the robot end-effector (figure 2.4.1) are used in the process of determining the different situations. The V_sensor and H_sensor are both described in chapter 2.4.2.1, and shown in figure 2.4.3.

9.4.2.2.1.1 V_sensor attributes

The V_sensor reading is used to determine what is directly under the gripper. Readings expected from the V_sensor, when it is at certain set heights, are established for different objects such as a block on the conveyor. This was done for two different heights, the first being a safe height above the block when it is on the conveyor, called the at_check_pick_position (figure 9.23). The second height, when the robot is ready to grip a block on the conveyor, is called the at_pick_position (figure 9.24). For each height, the level of the conveyor and the block (using the calibration block described in section 2.3.1.3) are determined by moving the robot end-effector and determining the value of V_sensor.

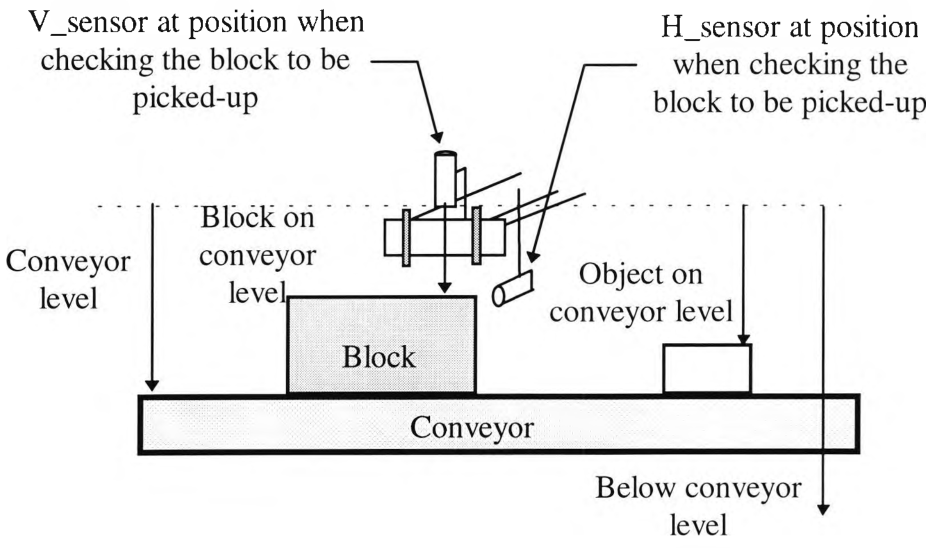

V_sensor at position when checking the block to be picked-up
H_sensor at position when checking the block to be picked-up
Conveyor level
Block on conveyor level
Object on conveyor level
Block
Conveyor
Below conveyor level

Figure 9.23 *Readings of sensors when the gripper is in position to check for the block pick up*

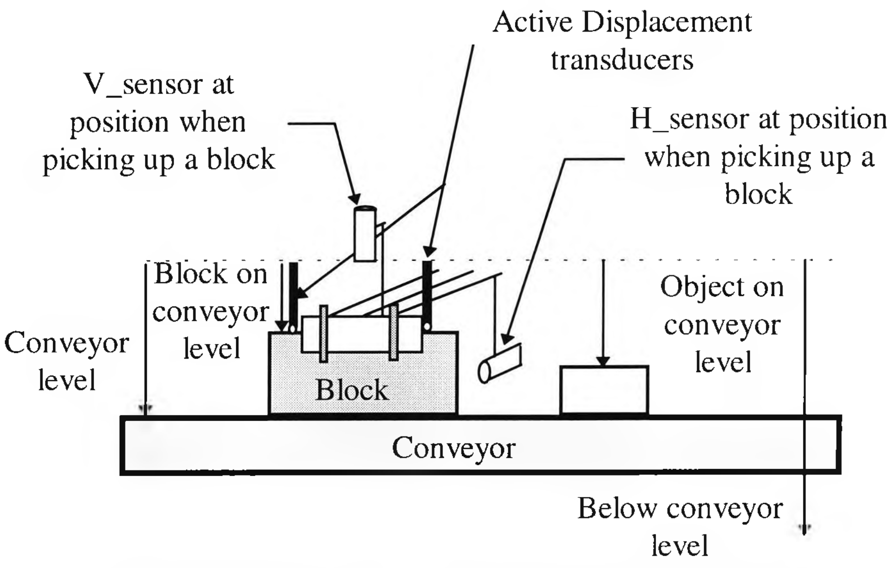

Active Displacement transducers
V_sensor at position when picking up a block
H_sensor at position when picking up a block
Block on conveyor level
Object on conveyor level
Conveyor level
Block
Conveyor
Below conveyor level

Figure 9.24 *Readings of V_sensor when the gripper is in position to pick up a block*

Rules were written to determine the value of the gripper sensor readings. The first stage uses the V_sensor to detect the presence of the block on the conveyor, this relies on the expected height of the building block to a tolerance. The ideal block has attribute '*Ideal_height*'. The height of the block to be picked has attribute '*Block_height*', and is determined from the calculations of the conveyor sensor information as described in section 2.3.1.2. This incorporates the discrepancies in the block height, as the meaning of the various levels the V_sensor reads are established using the calibration block. Attribute *V_sensor_adjusted* is a conclusion of rule:

V_sensor_adjusted = V_sensor_data - height_difference

Rules rule_BP_7-12 translate the information gathered from the various sensors. The rules determining the *V_sens_at_check_pick_pos* attribute are:

rule_set_BP_7: if *V_senosr_adjusted* is <100 or >600
then ***V_sens_at_check_pick_pos*** is *out_of_range*
rule_set_BP_8: if *V_senosr_adjusted* is > 434 and < = 437
then ***V_sens_at_check_pick_pos*** is *conveyor_level*
rule_set_BP_9: if *V_senosr_adjusted* is > = 217 and < 225
then ***V_sens_at_check_pick_pos*** is *block_on_conveyor_level*
rule_set_BP_10: if *V_senosr_adjusted* is 225.:235
then ***V_sens_at_check_pick_pos*** is *block_edge_on_conv_level*
rule_set_BP_11:if *V_senosr_adjusted* is between 235..434 inclusive or > =100and<217
then ***V_sens_at_check_pick_pos*** is *object_on_conveyor_level*
rule_set_BP_12: if *V_senosr_adjusted* is <600 and >437
then ***V_sens_at_check_pick_pos*** is *below_conveyor_level*

The rules determining the *V_sens_at_pick_pos* attribute are:

rule_set_BP_13: if *V_senosr_adjusted* is between 138 →142 inclusive
then ***V_sens_at_pick_pos*** is *at_pick_pos_level*
rule_set_BP_14: if *V_senosr_adjusted* is <100 or >600
then ***V_sens_at_pick_pos*** is *out_of_range*
rule_set_BP_15: if *V_senosr_adjusted* is between 352 → 354 inclusive
then ***V_sens_at_pick_pos*** is *conveyor_level*
rule_set_BP_16: if *V_senosr_adjusted* is <600 and >354
then ***V_sens_at_pick_pos*** is *below_conveyor_level*
rule_set_BP_17: if *V_senosr_adjusted* is >142 and <354
then ***V_sens_at_pick_pos*** is *object_on_conveyor_level*

9.4.2.2.1.2 H_sensor attributes

Prior to picking up a block, the gripper centre needs to be vertically above the centre of the block, thus ensuring that the gripper does not collide with the block, when lowered for the pick up. To achieve this, the H_sensor reading when the gripper is at the necessary horizontal distance from the block, to perform the safely picking up a block, is sampled. This is illustrated in figures 9.23 and 9.24. This value is incorporated into rules that will translated the H_sensor reading in order to determine the state of the *H_sensor* attribute:

rule_set_BP_18: if *H_sensor_data* is >=100 and <123 or >125and < = 500
then ***H_sensor*** is *not_safe_to_pick*
rule_set_BP_19: if *H_sensor_data* is =>123 and <=125
then ***H_sensor*** is *at_safe_to_pick*
rule_set_BP_20: if *H_sensor_data* is <100 or >500
then ***H_sensor*** is *out_of_range*

9.4.2.2.1.3 LVDT attributes

The two displacement transducer readings confirm the presence of a block in the gripper. From their values, the angle of the block in the gripper 'γ' can be determined. In chapter 2.4.3 these LVDTs were described, as well as the range of the values appropriate for the block to be safely gripped (figure 2.5.10). This information is necessary to prevent collision of the gripper with the block, to make sure the block is in a suitable position in the gripper, and compensate for the inclination by adjustment of the roll axis. Rules rule_BP_21-23 apply to the attribute *R_Disp_Transducer*:

rule_set_BP_21: if *R_Disp_transducer_data* is between 0.6→14.2 inclusive
then ***R_Disp_Transducer*** is *at_safe_height_to_pick*
rule_set_BP_22: if *R_Disp_transducer_data* is between 15.5→16.9 inclusive
then ***R_Disp_Transducer*** is inactive
rule_set_BP_23: if R_Disp_transducer_data is >= 0 and < 0.6
then ***R_Disp_Transducer*** is *at_block_close_to_gripper_top*

9.4.2.2.2 Checking the robot position

It is necessary to confirm the robot's requested x, y and z axes position. A robot function interrogates it for its position in absolute motor counts. This is converted using equations 2.2-2.6, and the offsets (a_1,a_2,a_3) for the *block_pick_TCP* (figure 2.5), as used for the BPP in to the *block_pick_TCP* co-ordinate system. The resulting values are the input attributes *X_axis_pos*, *Y_axis_pos*, and *Z_pos_data*.

Rules were written to check if the robot is at the theoretical block pick position BPP. The difference between where the robot should be e.g. attribute *Y_block_pick_pos* minus attribute *Y_axis_pos,* are calculated in the conclusion attribute *at_Y_BPP*:

rule_set_BP_24: if *Y_BLOCK_PICK_POS* is number
and *Y_axis_pos* is number
then ***at_Y_BPP*** is abs(@*Y_BLOCK_PICK_POS*-@*Y_axis_pos*)

The equivalent is conducted for the x-axis.

A detected difference of up to 15 mm is tolerated, as this can be corrected using procedure *check_pick_pos*(ф) that aligns the gripper to the block, as described in chapter 2.4.3.1. The following rules will determine if the robot is at the required x, y BPP:

rule_set_BP_25: if *at_X_BPP* is =<15 and *at_Y_BPP* is =<15
then ***robot_XY_position*** is *at_Block_Pick_position*
rule_set_BP_26: if *at_X_BPP* is >15 and *at_Y_BPP* is >15
then ***robot_XY_position*** is *not_at_block_pick_pos*
rule_set_BP_27: if *at_X_BPP* is >15 and *at_Y_BPP* is =<15
then ***robot_XY_position*** is *not_at_block_pick_pos*
rule_set_BP_28: if *at_X_BPP* is =<15 and *at_Y_BPP* is >15
then ***robot_XY_position*** is *not_at_block_pick_pos*

Rules were written to check if the robot was at the requested z position. The apparent z-axis position was firstly investigated for two situations. These two levels correspond to at_pick_pos (figure 9.24), and at_check_pick_position (figure 9.23), both described in section 9.4.2.2.1.1. The following rules will determine the location of the z axis:

rule_set_BP_29: if *Z_pos_data* is between -911→ -909 inclusive
then ***Z_axis_pos*** is *at_check_pick_pos*
rule_set_BP_30: if *Z_pos_data* is between -993→ -991 inclusive
then ***Z_axis_pos*** is a*t_pick_pos*
rule_set_BP_31: if *Z_pos_data* is >-909 or between -911→ -993
exclusive not including <-993
then ***Z_axis_pos*** is *not_at_check_pick_pos*

9.4.2.2.3 Checking conveyor status

Each stage of the block moving along the conveyor is communicated to the main computer. These stages are decided using the conveyor logic, which relies on the various sensors on the conveyor. The conveyor logic is shown in a flow diagram in appendix A.2. An explanation of the various conveyor sensors was given in chapter 2.3. This information is incorporated into the RBES which decides, using the set of rules shown below, the status of the conveyor and the block on the conveyor.

rule_set_BP_32: if *conveyor_data* is X then ***Conveyor_status*** *is block_in_pick_pos*
rule_set_BP_33: if *conveyor_data* is W then ***Conveyor_status*** *is ready_for_block*
rule_set_BP_34: if *conveyor_data* is R then ***Conveyor_status*** *is block_file_created*
rule_set_BP_35: if *conveyor_data* ***is*** Z then ***Conveyor_status*** *is block_file_downloaded*

9.4.2.2.4 Aligning gripper to block

The procedure that aligns the gripper to the block, uses the ultrasonic H_sensor on the gripper to find the block to gripper angle 'β'. The function developed for this is *check_pick_pos*(ϕ), described in chapter 2.4.3, where ϕ is the angle of the robot yaw axis. This procedure is part of stage 3 of the building process (figure 2.2). This procedure is incorporated into the RBES by first setting the input attribute *Gripp_to_block_angle* equal to 'β'. Below are a few examples of the type of rules concluding the success or failure of the procedure. The complete rule set is included in appendix G.2. The gripper is considered to be successfully aligned to the block when 'β'

angle is ±0.5° (rule_BP_38). The procedure is repeated up to a maximum of four times (e.g. rule_BP_37). If after the fourth attempt, the angle is still greater than 0.5°, then we can conclude that the gripper was not successfully aligned (e.g. rule_BP_39). The way the *check_pick_pos*(ϕ) procedure is integrated and called in the robot calling programme is explained in section 9.4.2.3.

rule_set_BP_36: if *no_checks* is <1
then ***Gripp_to_block_angle_check*** is *not_checked*

rule_set_BP_37: if *no_checks* is 1:.3
and *Hsens_mid* is between 100 and 400 inclusive
and *Hsens_side2* is between 100 and 400 inclusive
and *Hsens_side1* is between 100 and 400 inclusive
and *Gripp_to_block_angle* is <3 and >-3
then ***Gripp_to_block_angle_check*** is *redo_check*

rule_set_BP_38: if *no_checks* is >=1
and *Hsens_mid* is between 100 and 400 inclusive
and *Hsens_side2* is between 100 and 400 inclusive
and *Hsens_side1* is between 100 and 400 inclusive
and *Gripp_to_block_angle* is <=0.5 and >=-0.5
then ***Gripp_to_block_angle_check*** is *check_success*

rule_set_BP_39: if *no_checks* is >3
and *Hsens_mid* is between 100 and 400 inclusive
and *Hsens_side2* is between 100 and 400 inclusive
and *Hsens_side1* is between 100 and 400 inclusive
and *Gripp_to_block_angle* is >0.5 and <-0.5
then ***Gripp_to_block_angle_check*** is *check_failed*

9.4.2.2.5 Determining the block location

Rules were created to determine the status of the block location (i.e. whether it is in the expected block pick-up position on the conveyor, or elsewhere). These rules were based on the knowledge gathered from the robot sensor information, the robot position information and the conveyor sensor. Once this is determined, then the necessary action can be concluded and acted upon. The attribute values that conclude the *block_location* attribute are listed in table 9.3, and can be seen clearly in the RBES's attributes shown in figure 9.24. All these attributes are described in sections 9.4.2.2.1-9.4.2.2.4.

Table 9.3 *List of attributes that are immediately linked to attribute 'block_location'*

Attribute name	Attribute description
Block_location	Determining the block location status, e.g. in position, no block on conveyor
Gripper_to_block_angle_check	The state of the aligning of the block to the gripper using function *check_pick_pos*(ϕ)
H_sensor	What the H_sensor reading indicates
Conveyor_status	What stage the of logic the conveyor is at
Robot_XY_pos	what the position of the robot indicates
Z_axis_pos	what the Z axis position indicates
V_Sensor_at_check_pick	What the H_sensor reading indicates, at the position when the robot z axis is at a height to check for the block on the conveyor

The process of building these rules started by first determining the correct conditions for a block to be in its pick-up position on the conveyor. This is shown in the following rule :

```
rule_set_BP_40:  if conveyor_status is block_in_pick_pos
                        and V_sens_at_check_pick_pos is  block_on_conveyor_level
                        and H_sensor is at_safe_to_pick
                        and  Z_axis_pos is at_check_pick_pos
                        and  Gripp_to_block_angle_check is check_success
                        and robot_XY_position is  at_Block_Pick_position
                 then block_location is in_position
```

Other rules were built based on rule_BP_40, but using a combinations of different attribute values and deciding for each rule what is meant i.e. its conclusion. Some of these rules are listed below. The complete rule set is in appendix G.2.

rule_set_BP_41: if *Conveyor_status* is *block_in_pick_pos*
and *V_sens_at_check_pick_pos* is *conveyor_level* or
block_on_conveyor_level or *block_edge_on_conv_level*
and *H_sensor is not_safe_to_pick*
and *Z_axis_pos* is *at_check_pick_pos*
and *Gripp_to_block_angle_check* is *check_success*
and *robot_XY_position* is *at_Block_Pick_position* and
no_H_checks is <3
then ***block_location*** is *check_Rob_to_Blk_Hdistance*

rule_set_BP_42: if *Conveyor_status* is *block_in_pick_pos*
and V*_sens_at_check_pick_pos* is *conveyor_level* or
block_on_conveyor_level or block_edge_on_conv_level
and *H_sensor* is *not_safe_to_pick* or *at_safe_to_pick*
and *Z_axis_pos* is *at_check_pick_pos*
and *Gripp_to_block_angle_check* is *not_checked*
and *robot_XY_position* is *at_Block_Pick_position*
then ***block_location*** is *check_block_pos*

rule_set_BP_43: if *Conveyor_status* is *block_file_downloaded*
then ***block_location*** is *block_file_being_processed*

rule_set_BP_44: if *Conveyor_status* is *ready_for_block*
then ***block_location*** is *load_block_on_conveyor*

rule_set_BP_45: if *Conveyor_status* is *block_in_pick_pos*
and *V_sens_at_check_pick_pos* is *conveyor_level* or
block_edge_on_conv_level
and *H_sensor is at_safe_to_pick*
and *Z_axis_pos* is *at_check_pick_pos*
and *Gripp_to_block_angle_check* is *check_success*
and *robot_XY_position* is a*t_Block_Pick_position*
and *no_H_checks* is <3
then ***block_location*** is *not_in_position*

rule_set_BP_46: if *Conveyor_status* is *block_in_pick_pos*
and *V_sens_at_check_pick_pos* is *out_of_range*
and *H_sensor* is *not_safe_to_pick*
and Z_axis_pos is at_check_pick_pos
and *Gripp_to_block_angle_check* is *check_failed*
and *robot_XY_position* is *at_Block_Pick_position*
then ***block_location*** is *relocate_conveyor_position*

9.4.2.3 Running the expert system

After completing the rule-base, the 'C' code module of the RBES is then generated and embedded in the robot calling program. The various procedures coded for the stages of the block picking-up process, described in chapter 2 and 5, are used. The interface functions are embedded in the calling program, to pass information and commands to the code module of the RBES (figure 9.25). These functions allow the input attributes of the RBES to be embedded into the functions that make up the calling programme, to update them and conclude the attributes of the RBES.

The ES shell allows inclusion of user functions in the generated code. The one used in this programme is a function which enables an output hook, that is called whenever a new attribute conclusion is reached. In the case of the block pick RBES, the output hook '*myoutput(attribute, value)*' was written to react to each conclusion value of an attribute. As a result of concluding that, should it be necessary to make the robot move a certain distance or open the gripper, then the hook calls an appropriate robot function to deal with it. The output hook is described below. A description of the way that the attributes are networked (figure 9.22), and the rules that concludes their values are both given in section 9.4.2.2.

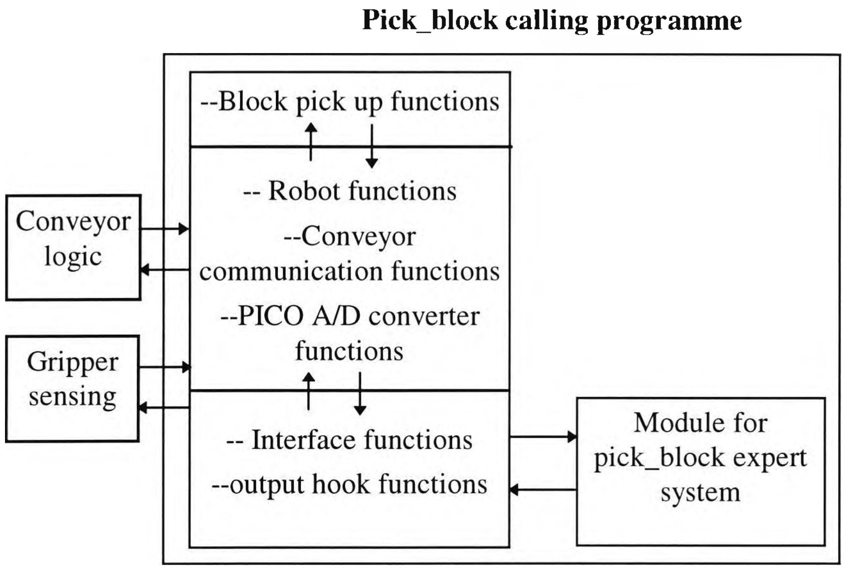

Pick_block calling programme
--Block pick up functions
-- Robot functions
--Conveyor communication functions
--PICO A/D converter functions
-- Interface functions
--output hook functions
Conveyor logic
Gripper sensing
Module for pick_block expert system

Figure 9.25 *The pick block 'C' code calling programme, integrated with the pick_block module and the gripper sensing and conveyor information*

The application procedure for the output hook *myoutput(attribute*, *value)* is now dealt with. In this procedure, whenever forward or backward chaining is carried out on an attribute, if the conclusion value of any attribute changes then the procedure is recursively called (i.e. goes back to step (i) of the procedure). This is given in the following procedure:

```
Procedure myoutput(attribute, value) :
(i)- If attribute = BLOCK_LOCATION then
        (i.i)- if value =  check_block_position then
         (i.i.i)-call function check_pick_pos($\phi$). Output of function is angle of
              gripper to block '$\beta$' and H_sensor at point S1, S2, at the point between
              position S1 and S2 all described in chapter 2.4.3.
              -no_checks_made = no_checks_made +1
              -attribute NO_CHECKS = no_checks_made
              -attributes HSENS_SIDE1 = S1, HSENS_SIDE2 = S2,
              HSENS_MID = H_sensor reading between points S1 and S2
              -Forward chain on each of attributes HSENS_SIDE1, HSENS_SIDE2,
              HSENS_MID, NO_CHECKS
              -update robot position attributes and backward chain attributes
              ROBOT_XY_POS, Z_AXIS_POS
              -attribute GRIPP_TO_BLOCK_ANGLE = $\beta$
              -forward chain attribute GRIPP_TO_BLOCK_ANGLE_CHECK
```

```
            -if value of GRIPP_TO_BLOCK_ANGLE_CHECK = redo_check & no_checks_made <4
                    then go to stage (i.i.i) of the procedure
    (i.ii)- if value = check_Rob_to_Blk_Hdistance then
            -adjust gripper position using block to centre alignment process described in chapter 2.4.3.2.
            -update robot position attributes and backward chain attributes ROBOT_XY_POS, Z_AXIS_POS
            -update V_sensor attributes and forward chain on its value
            -no_checks_horizontal_distance= no_checks _horizontal_distance+1
            -attribute NO_H_CHECKS = no_checks_horizontal_distance
            -backward chain attribute BLOCK_LOCATION which will take us back to step (i) of the procedure.
    (i.iii)- if value = check_robot_position then
            -update robot position attributes and backward chain attributes ROBOT_XY_POS, Z_AXIS_POS
            -backward chain attribute BLOCK_LOCATION which will take us back to step (i) of the procedure.
    (i.iv)- if value = relocate_conveyor_position then
            -execute procedure to locate conveyor, and calculate from it the BPP, described in chapter 6 (stage 1 of the building process)
            -execute procedure to adjust the BPP described in chapter 2.3.1.3
            -update robot position attributes and backward chain attributes ROBOT_XY_POS, Z_AXIS_POS
            -update sensor attributes H_SENSOR_DATA, V_SENSOR_DATA
            -backward chain attribute BLOCK_LOCATION
(ii)- If attribute = PICK_BLOCK then
    (ii.i)- if value = open_gripper then
            -open robot gripper
            -update attribute GRIPPER_STATUS values
            -forward chain attribute GRIPPER_STATUS
    (ii.ii)- if value = close_gripper then
            -close robot gripper
            -update attribute GRIPPER_STATUS values
            -forward chain attribute GRIPPER_STATUS
    (i.iii)- if value = lower_Z_axis_by_4_mm then
            -move Z axis down 4 mm
            -update robot position attributes and backward chain attributes ROBOT_XY_POS, Z_AXIS_POS
            -backward chain attribute PICK_BLOCK (i.e. go back to step (i))
```

```
(i.iv)- if value = raise_Z_axis then
        -move Z axis up
        -update robot position attributes and backward chain attributes
        ROBOT_XY_POS, Z_AXIS_POS
        -backward chain attribute PICK_BLOCK (i.e. go back to step (i))
(i.v)- if value = adjust_roll then
        -move Z axis up 30mm
        -update values for sensor attributes H_SENSOR_DATA,
        V_SENSOR_DATA, R_DISP_TRANSDUCER_DATA,
        L_DISP_TRANSDUCER_DATA
        -if L_DISP_TRANSDUCER_DATA < R_DISP_TRANSDUCER_DATA
                -then move roll axis 4° clockwise
        if L_DISP_TRANSDUCER_DATA > R_DISP_TRANSDUCER_DATA
                -then move roll axis 4° anti-clockwise
        -lower Z axis 30mm
        -update robot position attributes and backward chain attributes
        ROBOT_XY_POS, Z_AXIS_POS
        -update values for sensor attributes H_SENSOR_DATA,
        V_SENSOR_DATA, R_DISP_TRANSDUCER_DATA,
        L_DISP_TRANSDUCER_DATA
        -backward chain attribute BLOCK_PICK (i.e. go back to step (i))
        -no_roll_adjustments = no_roll_adjustments + 1
        -attribute NO_ROLL_ADJUSTMENTS = no_roll_adjustments
(i.vi)- if value = CHECK_ROBOT_Z_POS then
        -update robot position attributes and backward chain attributes
        ROBOT_XY_POS, Z_AXIS_POS
        -backward chain attribute PICK_BLOCK (i.e. go back to step (i))
(i.vii)- if value = BLOCK_PICK_SUCCESSFUL then
        -return to calling program
(i.vii)- if value = any other conclusion not listed above then
        -print error message = value
        -terminate any robot movement
        -exit calling program
```

9.4.2.4 Testing the expert system

The RBES was tested in three stages, the first two stages of the testing being carried out in the expert system shell environment. In the first stage of testing, the rules were verified, which resulted in a list of missing rules. A process of selecting and adding the valid missing rules from this list was then carried. This ensured that every possible combination of events was accounted for. For example, if there was no block to be picked-up at the block pick up position. The second stage of testing the rule-base was done by entering different combinations of input values for attributes, to simulate different situations, and using backward and forward chaining to check and verify the various conclusions of attributes.

The third stage of testing was carried out after creating the pick block calling programme, described in section 9.4.2.3. Experiments were carried out to test the resulting programme. The programme was run three times, and each time the block was picked-up successfully. The attributes of the RBES were monitored throughout the running of the programme, with the results included in appendix G.2.2.

9.4.3 Expert system: move optimisation

A further RBES was developed to assist in the programming of the robot motion. The knowledge gathered in chapter 3, on the theory and programming of the different move types of the robot, as well as the determined system limits, are used in the development of this. The RBES ensures a move requested is within the robot limits and capabilities. This rule-base was considered necessary in the translation of the project description into the theoretical task (figure 1.7). The system developed is considered to be a starting point in this translation, even though it has not been fully developed and integrated.

The rule-base detects impossible moves, and advises the user against their use (e.g if the move requested exceeds the system's maximum velocity). Also, knowledge gathered from the investigation into the vibration of the robot end-effector, which resulted in the establishment of optimum moves, is used in the development of the rule-base. The method developed to determine the robot end-effector vibration was covered in chapter 2.8, and experiments relating to this in chapter 4.

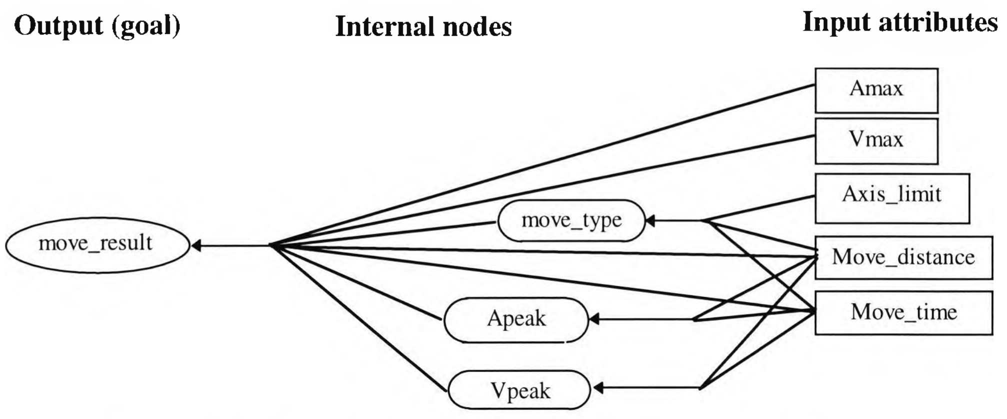


Figure 9.26 *Network of attributes for the move optimisation rules*

9.4.3.1 Optimum move rules

Rules were created to address the problem of the dynamic behaviour of the robot end-effector, which arises from the partial compliance of the yaw-axis. In chapter 4, the vibration of the robot end-effector, when performing different moves under various conditions was examined, using the method described and verified in chapter 2.8. This resulted in the determination of optimum types of move, with minimum settling time and overall vibration for a given set of conditions, enabling the robot to be used in an efficient and machine-safe manner. An example of such rules are shown in rule_MO_1 and rule_MO_2 below. In rule_MO_1, the requested move is for a distance 1000 mm. For this case the recommended optimum move is a 2 seconds single parabolic 'U' move, based on the finding of chapter 4.

rule_set_MO_1: if *Move_distance* is 1000
and *Move_time* is optimum
then ***Move_type*** is *parabolic_single* and 2

rule_set_MO_2: if *Move_distance* is 4
and *Move_time* is optimum
then ***Move_type*** is *parabolic_segmented* and 0.35

Other rules recommend the most desirable moves for a required distance and time, providing adequate time is allocated for the move duration. Again, these rules are based on the findings of the experiments carried out in chapter 4. An example follows:

```
rule_set_MO_3:  if Move_distance is 2000
                    and  Move_time is >=7
                then Move_type is SCurve_3_part and @Move_time
```

9.4.3.2 Move generation rules

Rules were created to enable efficient programming of robot moves, avoiding command trajectories that the robot is incapable of following. In chapter 3, the system's capabilities were checked through experimentation, and the theory behind the system's production of some of the command trajectories was described. The knowledge gathered from this is used in the production of the rule-base. Also, the outcomes of experiments carried out to find the system's maximum acceleration (Amax) and maximum velocity (Vmax), with the gripper attached for the x-axis (chapters 3.2 and 3.7) are used. These two values are represented in the RBES as attributes *Vmax* and *Amax* (figure 9.26). An example of these rules, which check for a requested single part parabolic 'U' type move, is given below in rule_MO_4 and rule_MO_5. In these rules, the values of peak acceleration (Apeak), and peak velocity (Vpeak), for the requested move, are calculated and compared to system's limits.

```
rule_set_MO_4:  if Move_type is parabolic_single
                    and  Move_distance is number
                    and  Move_time is number
                    and  Amax is < @Apeak
                    and  Apeak is number
                then move_result is move_exceeds_max_Acceleration

rule_set_MO_5:  if Move_type is parabolic_single
                    and Move_distance is number
                    and Move_time is number
                    and Vmax is <@Vpeak
                    and Vpeak is number
               then move_result is move_exceeds_max_Velocity
```

, where the value of attribute *Apeak* for a single part parabolic move is calculated using eqn. 3.12 and determined using rule_MO_6.

```
rule_set_MO_6:  if Move_distance is number
                    and Move_time is number
               then Apeak is (6 × @Move_distance)/(@Move_time × @Move_time)
```

, and the value of attribute *Vpeak* for a single part parabolic move is calculated from eqn. 3.8 and determined using rule_MO_7.

rule_set_MO_7: if *Move_distance* is number
and *Move_time* is number
then ***Vpeak*** is (3 $\times$ @*Move_distance*)/(2 $\times$ *Move_time*)

Other rules check if the move requested is within the range of the robot axes. An example of this type of rule is given in rule_MO_8 to check for the range of the robot's x-axis.

rule_set_MO_8: if *Move_distance* is >@*Axis_limit*
and *Axis_limit* is 3969
then ***Move_type*** is out_of_range

On completion of the rule-base, the system was verified and tested. Verifying the rules resulted in a list of missing rules. A process of selecting and adding the needed missing rules from that list was then carried out. Next, the rule-base was tested by entering different combinations of input values for the attributes, to simulate different situations.

9.5 Conclusion

The choice of the expert system shell proved to be ideal in developing the three rule-bases described. It allowed the development of rule-base intelligence in a user friendly environment, while providing the facilities to test and verify the rules, as well as generating the rule-base as a 'C' language code.

The rule-base developed to carry out the task of ordering the building of a designed masonry project was confirmed successful, and also the 'C' code to utilise it in the process of translating the CAD generated project description into a theoretical task. This rule-base enabled a method that allows integration appropriate for automation of the construction task. The resulting programme was tested using the example of a generated CAD design. The result of the test was checked to see if the system did produce the order of building without collision, which proved to be so. Even though the system developed had the capability of ordering the construction of limited design projects, the use of rule-base intelligence was shown to be effective in tackling the task. A limitation to the system is the intelligence of the decision on which end of the block is to be buttered in the construction of corners. Time did not permit coverage of this knowledge in the experiment rule-base.

The provision of real-time intelligence developed to enable the robot to carry out the task of picking up a block, was also achieved with success. The rule-base developed for this purpose was successfully integrated with sensor information and robot functionality. The resulting programme was tested twenty times to pick-up a block (all successful). This success, in the block pick-up operation, is confirmed using the sensor and rules. The provision of sensing intelligence allowed the task to be carried out using minimum assumptions on the exact conveyor and block locations.

A RBES that allows the programming of the robot in an efficient and effective way was successfully developed. This was carried out to demonstrate the principle behind the use of rule-base intelligence in the development of optimum moves for the robot, and also advise the user on best types of moves. The rule-base also advises the user on impossible moves that may exceed the system's capabilities. The rule-base was tested and the results checked and found to be correct. The step that needs to be considered is the integration of the generated 'C' code of this rule-base, with the order of building task and the robot cell configuration, for the automatic generation of the robot moves.

Chapter 10: Conclusion

The main objective of this research was to investigate the enabling technology for a masonry tasking robot, utilising an experimental robot cell. This was achieved with success using intelligent processing to carry out the various stages of the building process, from architectural planning through to execution of the construction work. The following conclusions correspond to the order of stated objectives in chapter 1.4. Each conclusion is preceded by the objective, to aid clarity.

10.1 Effectiveness of robot cell

To investigate the effectiveness of an experimental robot cell comprising a Cartesian robot, material conveyor, mortar dispensing pump, laser leveller and sensing system designed for masonry automation.

The experimental cell described in the introduction, section 1.6.1, proved to be effective and adequate in the process of testing the methods devised to automate the building task. The choice of working envelope dimensions enabled building of trial assemblies. Also, the design of the experimental cell was found to be effective in testing the methods devised to automate the building process, including masonry unit manipulation, mortar dispensing, sensing and equipment location.

The experimental cell comprised:

(i) The industrial conveyor, with sensing provisions, was found to be an efficient method for the delivery of masonry units. The various conveyor sensors ensured accurate measurement of the dimensions of each block, which is vital for the run time adjustments needed. In addition, the sensors' information enabled the synchronisation of the block delivery process, as well as supplying the information necessary to determine the location of the blocks at the pick-up position. This enabled material supply to work in parallel with the various other building tasks. Through the sensing provision, it was not found necessary to accurately place block material at the feed end of the conveyor.

Use of the conveyor, though with modifications, proved that low cost solutions derived from commonly available equipment are possible.

(ii) The conveyor had a target mounted on it, which enabled its automatic location. A design for this target was worked out through experimental trials.

(iii) The ‘clamp type’ gripper, constructed as the robot end-effector, was combined with on-board ultrasonic sensors and displacement transducers. This arrangement was found to be effective in enabling safe and intelligent manipulation of the blocks. The opening range of the gripper could have been greater, providing a less involved checking process. This however would have led to other problems in the way the blocks are gripped (described in section 10.11).

(iv) The purpose-built laser profiler provided an independent check on the position of a picked masonry unit in the gripper. It was found to be effective and accurate in calculating the necessary adjustments needed for applying mortar and laying the blocks in their theoretical position. A more detailed description of the finding are discussed in section 10.6.

(v) The mix dispensing station comprised a peristaltic pump, a mix feed line, an extruding nozzle and a control box which communicates with the main PC via a parallel port. This equipment enabled full automation of the mortar dispensing process. The robot was used to provide the motion and manipulation necessary for the application of mortar on both sides of a block. Building of assemblies was achieved and is discussed in detail in section 10.3. Separating mortar dispensing from the conveyor supply was more a matter of resource limitations than expediency. Ideally, dispensing would be carried out on the conveyor, leaving the robot to only lay the already buttered unit.

(vi) A vertical axis laser beacon was designed and built to provide horizontal referencing. This was not integrated in the cell.

10.2 Material choice

To investigate whether or not conventional, standard production masonry material, with significant dimensional tolerance, can be utilised in automated handling processes.

Throughout this research, standard 'Celcon' type blocks were used for the experiments testing the different stages of the building process. The manufactured dimensions of these are 440 ± 3 mm in length, 215 ± 2 mm in height, and 100 ± 2 mm in width. Refer to checks made on these dimensions by sampling 12 blocks (appendix A.4, table A.1). With such tolerances, it is clearly important that the dimensions are determined on a block by block basis. Through the use of intelligence and sensing, it was possible to determine the necessary run-time adjustments needed for safe manipulation of blocks and practical realisation of the theoretical task. This was successfully achieved by combining the use of conveyor sensors with advanced computer programming to accurately measure the dimensions of the masonry units and propagate the various discrepancies. Experiments, detailed in chapter 6, determine the accuracy of the method proposed in chapter 2. These covered computations of block dimensions using the conveyor mounted block profile sensors, which proved to be an accurate measure of block dimensions and location of the block ready for pick-up. On average, the accuracy of the block dimension measurements were ±0.1 mm for the length, ±0.1 mm for the height, and ±0.4 mm for the width. The maximum errors were found to be ±1.2 mm for the length, ±1.0 mm for the height, and ±1.3 mm for the width. These errors were not detrimental to the block pick-up process.

Gripper sensors were used to make the final adjustments to the block pick-up position. This proved to be vital and necessary in securing the safety of the block pick-up operation. The process developed to align the block to the gripper was found to be accurate to ±1°. Experiments, covered in chapter 6, were carried out to test the efficiency of this method. From the experiments carried out on block picking, it was noted that the maximum angle adjustment needed to correct the concluded angle of the block to the conveyor, was less than ±1°. It was shown that this error arose from the determination of the angle of the block to the conveyor, using the conveyor block profile information. Combining this error with the maximum possible error in determining the conveyor orientation, which was ±5.7° (determined from the experiments carried out in chapter 5) made the maximum gripper to block angle error to be ±6.7°. Other experiments were conducted to test the point of failure of the method, i.e. the ability to align the gripper to the block with the error too large for automatic correction. The method developed to align the gripper to the block proved successful, even when the angle was as large as 9°. This renders the error of ±6.7° insignificant to the success of the block pick-up operation.

The result of the various real-time sensing procedures enabling the gripping of the block from the conveyor, in a safe and secure fashion, proved satisfactory, with a success rate of 100%. A set of twelve different blocks was used to test the picking operation. Despite their significant dimension tolerances, the standard production 'Celcon' blocks were found to be suitable for the automation cell. This runs contrary to research elsewhere in which high precision or saw-cut prepared units are employed to make possible the automation process.

10.3 Mortar dispensing

To automate the mortar dispensing operation and investigate whether conventional or modified mortar can be used for bonding. Also, to determine the effectiveness of the robot in forming strong bonds, when used to lay the buttered blocks.

The process of mortar dispensing was fully automated successfully using a peristaltic pump controlled by the main computer. A method was developed that allowed the robot to manipulate the masonry units under the nozzle, covered in chapter 2. It was possible to dispense mortar by using the peristaltic pump and a specially designed nozzle, both described in chapter 8. The mortar used was a Pozament mix, dispensed in a controlled fashion to produce a variety of bead geometry. The ability to do this allowed the utilisation of standard production masonry units, by run time adjustments to compensate for their significant dimensional tolerances. From the experiments carried out in mortar dispensing, covered in chapter 8, it was noted that a weight had to be applied to top of the mix in the pump's feed hopper. Also, the mortar was stirred occasionally to minimise the chance of blockages. This reflects the thixatropic nature of the mix. It was also noted that inadequate cleaning of the nozzle caused the shape of the mortar bead to be disfigured. Thorough cleaning of the pumping elements was also necessary for the same reason. These actions could be added to the design of the pump, making the pumping of mortar extremely reliable.

During experimentation, eighteen assemblies were built, the results of which are covered in chapter 8. These assemblies showed a promising result achieved using the fully automated robotised building process. It was found that even when the blocks were placed with large errors (±5 mm offsets), the assemblies achieved were found to be stable. The method of laying the blocks, where pressure was applied with combined sideways and downwards motion, was found to be less accurate on account of the partial compliance of the yaw axis. Here, ±5 mm location errors (centre) and approximately $\pm 1^\circ$ orientation error accrued. While applying pressure by the robot in placing a block, a sideway motion is still needed to close the perpendicular joint. Part of the ±5 mm location errors were due to the block occasional slanting, when gripped. From the experiments covered in chapter 7, this was found to cause an error of ±3 mm in some cases. This occurs when the centre of the gripper does not coincide with the centre of the block just prior to gripping. A method of scanning the side of the block, using a laser profiler to measure the slant, was suggested in chapter 7.4.2.1. However, it would be necessary to add a 6th (pitch) rotational axis to the robot to implement corrections for the laying process.

In the pumping operation, the need to synchronise the starting and stopping of the mortar issue from the nozzle, with the robot's movement of the blocks, is vital for the success and accuracy of the process. The snuff-back action that was developed to prevent nozzle leakage between applications proved successful. However, for greater reliability, it would be necessary to sense the first emergence of the mortar on restarting the pump.

Whilst extensive measurements of the settlement of the mortar, when the blocks are placed using the robot, have not been carried out, investigation showed it to be typically 1-2 mm. This is an important factor in automating the building process. Since circumstances (short availability of the lent pump) did not allow further testing, a result of 0.19 N/mm^2 was achieved. This figure is a little short of the desired strength of 0.21 N/mm^2. Whilst a small modification of the mix, without sacrificing other essential properties, the mix strength could be easily adjusted to suite CP111 or any other requirement.

The results of these experiments are incomplete, due to the fact that the pump was only available for a limited amount of time, during which the robot's z-axis motor failed and had to be serviced. Only first trials of dispensing and building small assemblies were thus possible. We were, therefore, unable to integrate this stage of the building process (stage 5) with the stages that proceeded and followed it (figure 1.1). In spite of these setbacks, the experiments indicate that standard masonry units together with 'thick layer' mortar are a building system suitable for the robotic application.

10.4 System performance and end-effector vibration

To investigate the performance of the robot, as affected by the vibration of the end-effector, for various move types.

There are three parts to this conclusion. The first part covers investigations into system performance, the second investigation of end-effector vibration, and the third, determination of optimum moves for the robot.

The theory behind the different moves the robot is capable of was covered in chapter 3, where a description of the method of programming each move, ways to avoid impossible moves and programming minimum time moves, were given. This was used in programming the various robot moves throughout this research. To investigate the robot performance, the system limits were first determined through experiments. The experiments carried out covered investigation of the x-axis alone. Results of both V_{max}

and A_{max} are shown in table 3.6. The maximum velocity of the system varies with the maximum acceleration committed, and vice-versa. The use of extreme velocities and acceleration were found to be unreliable and resulted in violent moves. The most reliable value for V_{max} was found to be 1000 mm/sec which had the corresponding A_{max} of 3232 mm/sec^2. Moves made using these limits induce high levels of vibration on the robot, but did not cause the logic system to hang-up. Other experiments were undertaken to compare the accuracy of actual moves with theoretical moves. A maximum of 0.4% error on velocity data was found. The inability to interrogate the motion control card at a fast enough rate, however, did not allow high-speed moves to be checked.

The chosen method of investigating the robot end-effector vibration involved using an accelerometer, placed on the robot end-effector, to quantify the settling time and amplitude of deflection of the end-effector, enabling the overall 'quality' of the move to be determined. Verification of the method and the software programmes developed, as well as their accuracy in processing the accelerometer data from the acceleration-time domain to the displacement-time domain was completed. An independent check on the accuracy of the displacement results was also carried out. This was accomplished by comparing results of the displacement of the end-effector achieved from processing the accelerometer data with that using a laser analogue displacement sensor (LADS). Results of the experiments showed that the worst error based on the accelerometer signal was ±0.8 mm. On average, the readings were accurate to ±0.18 mm, with a maximum error of ±0.7 mm.

Using the above method, it was observed that a variety of factors influence the vibration and the settling time of the robot end-effector. This helped to give a good indication of the overall behaviour of the different types of moves, thus enabling the determination of optimum move. Moves tested were designed and programmed using the theory developed in chapter 3. From the results covered in chapter 4, it can be seen that, in general, moves made within a shorter period induced stronger vibration and increased settling times at the end of the move. This is explained by the need for such a move to be carried out utilising high velocities and accelerations. Moves, starting from rest, with relatively short periods of time do not have the opportunity for their initial impulse to decay during the move. These factors make the 'effective move time' (defined as move time + settling time) unacceptably long due to high settling times. They also result in excessive wear and tear on the robot system. Therefore, these short, fast moves are ineffective for use. The worst performance for these short, fast moves was found in the single S-curve move. This was due to the complex start and stop they possess, which incur higher acceleration and velocity values during the move. But, given adequate time to make ($\geq$7 seconds for the long distance move, and $\geq$5 seconds for the medium distance move), the S-curve type moves were found to be the most desirable type of move, with the 3 part S-curve move performing better than the single part S-curve. The 3 part S-curve move has the additional benefit of a constant velocity segment, which was found to reduce the vibrations acquired at the start of the move. These moves are preferable, in the cases mentioned, due to their S-curve acceleration and deceleration capabilities.

Overall, the parabolic moves performed well, with the advantage of not requiring high values of acceleration and velocities, compared to other types of moves. This made these moves preferable for fast, long and medium distance moves (i.e. < 7 to 3.32 seconds for the long distance move, and <5 to 2 seconds for the medium distance move). The segmented parabolic move allows vibration induced at the start of the move to decay during the constant speed segment of the move. This resulted in it performing better than the parabolic U type move. This type of move is important because it causes minimum energy dissipation in the motor (for covering a given distance in a given time). The parabolic profile, by providing the maximum acceleration at the lowest velocities, enables the bulk of the distance to be covered at low values of acceleration, and therefore, lower power dissipation.

Depending on the time available to perform short moves (<4 mm), the robot performance varied. Given enough time ($\geq$0.35 seconds) all the move types tested were found to perform relatively well, with zero settling time and minimum vibration during motion. However, as the time of the move is decreased the performance of the parabolic U move and the 3 part S-curve, declines dramatically. The linear move's performance, as well as that of the segmented parabolic move, was found to be the best overall. Zero settling time was found with all the short length (4 mm) linear moves tested, with reasonable vibrations during motion. From all the experiments, a much better understanding on the preferred type of move for a given set of parameters and requirements has been achieved. These results were used in the development of the rule-based expert system (discussed in section 10.9), which recommends the best type of move for given move specification, aiding efficient use of the robot.

10.5 Automatic location

To investigate the means for automatic location of the peripheral equipment of the robot cell. This is necessary in enabling the robot to deal with the unstructured environment of the construction site.

This part of the research was investigated for automatic location of the material supply conveyor. A specially designed target (described in chapter 2) and search algorithm were developed, utilising the ultrasonic sensors on the gripper. Following investigation of several target designs, a satisfactory shape of target was decided on. This design ensured the target was large enough and wide enough not to be missed, with a unique shape so as not to confuse it with other objects. It served for location in both the horizontal and vertical planes. The need to locate the conveyor's exact position was to ultimately enable determination of the position of blocks on the conveyor, when they are ready for pick-up. The procedure of locating the target was developed in stages. This approach was adopted to make the search and location process efficient i.e. separating detection of the target from its use for accurate alignment. Through a sequence of investigations, the accuracy of each stage of the target location was investigated. The accuracy of the ultrasonic sensors was also investigated with consideration of their use in detection work.

The target design (chapter 2.3.2) and the method of moments (appendix D.1), adopted to detect and locate the target shape and position relative to robot axes, were both found to be extremely effective (chapter 5). In the first stage of target location (chapter 5), a plan scan of the area was conducted and the target position calculated. The target position was always successfully detected, with a worse case average error of ±2° in horizontal angle and ±26 mm in the target centre. The scan was found to be reliable using an average speed of up to 800 mm/sec. The largest first stage error, with the average scan speed of 800 mm/sec, was found to be ±5.72° and ±95 mm in the angle and the target centre respectively. The next stage was to locate the target more accurately, and a search algorithm was developed to do that. This algorithm relied, on the approximate target angle and centre detection from the first stage search. Experiments were carried out to establish the reliability of this algorithm. These proved that the algorithm was reliable in accurately locating the target (chapter 5.4.1), even if the error in the first stage angle detection was as high as ±20°. Furthermore, the algorithm was still reliable when the error in the first stage target centroid detection was as high as ±320 mm. However, when both type of errors occur at the same time (i.e. ±20° in the angle estimation as well as ±320 mm in the centre estimation), the algorithm failed (chapter 5.4.2). This case will never happen however, as these far exceed the maximum first stage errors.

To determine the accuracy of the complete process (i.e. all the different stages together), further experiments were carried out. From these experiments, an average error of ±0.4° for the target angle, and ±1.3 mm for the target mid point location were found. Also a maximum error of ±1.26° for the target angle, and ±2.3 mm for the target mid point location were found. Using sensors that are more accurate, this error could be reduced. The effects of these errors on the final calculation of the block pick position from the target angle and centre, causes at maximum, a lateral offset of ±0.06 mm (determined in chapter 5.4.3). This offset is negligible, and on its own, will not cause an unsafe block pick-up operation.

The ultrasonic sensors used for this experiment were, highly focused, near range ultrasonic sensors of ±0.25 mm accuracy and 400 mm working range. They were mounted on the clamp type gripper. The ultrasonic sensors have an approximate cone shaped working detection envelope. It was thus necessary to find the working range of angles and their corresponding sensor readings when pointing towards the target surface (section 5.3.2.1.1). Results of the experiment were of a case where the sensor is at a distance of 350 mm from the target (distance used for the search algorithm) and the end-effector normal to the target. From this, it was concluded that at around ±8°-9° the reading is at a minimum for that distance. At angles of -12° and +14° from the normal, the sensor reading was out of range. The experiment was repeated a few times, and the angle at which the sensor has an out of range reading was concluded to be, on average, ±12.7°. The distance of the sensor from the target affects this value.

To find the edge of the target, some experiments had to be carried out first, to investigate the behaviour of the ultrasonic sensors when used for edge detection. Once that was carried out, the concluded values were used in a procedure for finding the target centre (section 5.3.2.2.1).

In conclusion, the target designed proved to be satisfactory in locating the conveyor. The same device could be applied to the other peripherals in the robot cell for their automatic location.

10.6 Laying masonry units

To determine the accuracy and efficiency in the laying of masonry units in a predetermined position, while incorporating the necessary run time adjustments.

To carry out the necessary run-time adjustments needed for the procedures of laying blocks in a pre-defined position, two types of measurements had to be determined. The first was the dimensions of the block being laid, and the second to confirm the position of the block in the gripper. A method is described in chapter 2.5, which uses a laser profiler to determine the block's length and it's position in the gripper. This was designed to scan the gripped block, relying mainly on a pair of triangular target pieces mounted on the gripper (figure 2.5.3) and the detection of the sides of the gripped block. By detecting the various edges on this profile the relationship between the gripper and the gripped block was determined. Also, the use of gripper mounted LVDTs was used as a means of determining the angle and offset of a gripped block in the vertical plane.

Two types of experiments were conducted. The first was to find the accuracy of the laser profiler's measurements of the various elements of the gripper and block profile. These measurements were used in determining the offset values for the *Mortar_dispensing_TCP*'s position. This particular TCP is used in the mortar dispensing process, as well as the block laying process. Consequently, the second type of experiment conducted was to test the accuracy of the block laying process. This was done using the block and gripper measurements from the laser profiler, in calculations of the block to gripper geometry, as developed in chapter 2.5. The LADS device used in the laser profile's measurement is accurate (±0.01 mm) in the ranging measurement for the different depths of the gripper, block, and target triangles. In spite of that, the imprecision of the LADS's motion axis position caused significant errors even with of the application of an edge correction. The errors in the corresponding positional value of detected edges was about ±1 mm, mainly on account of imprecision in the positional encoder of the profile motion axis. Consequently this effected the accuracy of the whole process.

Results of the first experiments proved that the block length measurements were found to have an average error of ±0.75 mm, compared to that by measurement using the conveyor sensing (table 6.3), which were found to be about ±0.1 mm (results covered in section 10.2). Also the angle measurement, about the roll axis, between the laser strike and the gripper 'α' was found to have an average error of around ±8°, and a maximum error of around ±13°. These values were extremely unreliable, therefore they were not used when calculating adjustments for the block laying operation. Instead the angle 'α' was set to equal zero in the experiment layout, and hence the experiment calculations. This error was due to the laser profiler's detection of the small target triangles mounted on the gripper.

From the block laying experiments, the average lateral error in the block laying operation was found to be ±1.63 mm, with a maximum value of 3 mm. This can be directly linked to the laser profiler's determination of the distance of the block edge to the gripper, which is used in calculations to adjust the block laying position. The average error in these measurements were ±1 mm and ±1.4 mm, with maximum values of +2.29 mm and +2.67 mm respectively, which is of a similar magnitude to that of the block laying offset error. The maximum error in the horizontal angle for the block laying, was found to be +0.57°. This error can be linked to the errors in the LVDT measurements (i.e. the error in the angle measurement of the block to the gripper when gripped 'γ'). The error in the block laying horizontal angle caused an average vertical offset of around 1.8 mm, and a maximum error of 2.7 mm. The offsets found in the y-axis where mostly due to the middle of the block having a lateral offset, causing it to slant during the gripping action.

The error in the measurements of the distance from the edge of 'Celcon' blocks to the edge of the gripper, were found to be slightly higher than those found using the wooden calibration block. This was due to the nature of the edges of the 'Celcon' block, which are not as well defined as those of the calibration block.

10.7 Rule-base intelligence for construction

To investigate the effectiveness of applying intelligence, integrated with sensing, to the performance of the robot in the actualisation of the building process.

The provision of real-time intelligence developed to enable the robot to carry out the task of picking up a block, was achieved with success (chapter 9.4.2). The rule-base expert system developed for this purpose, was made up of 135 rules. To create the rule-base expert system, information established from experiments carried out to pick-up blocks (section 10.2), was established. Once this was carried out, rules were built to express the significance of the various combinations of events. It was successfully integrated with sensor information and robot functionality. The rule-base expert system was embedded in the calling programme, which performs the block picking process. The robot can perform the task with minimum human intervention, under constrained collision risk. The resulting programme was tested twenty times to pick-up a block. All these tests were successful. This success, in the block pick-up operation, was confirmed using the sensor feedback and rules. The provision of sensing intelligence allowed the task to be carried out using approximate assumptions on the conveyor and block locations. As the robot evaluates and reacts to variations in the pre-programmed task, at run time, this also minimises the risk of having to interrupt and restart the robot.

10.8 Rule-base intelligence for work planning

To investigate the use of machine intelligence in determining the order of building the construction task.

A rule-base expert system was developed that pre-processes the CAD design of a building project into a 'theoretical task', using 175 assembly rules (chapter 9.4.1). The project design was created using a CAD programme which allows designs to be surface modelled with the creation of a 'project description' file. These rules process each project design into a symbolic representation, using the 'project description' file, and determine the order in which a project is to be built. The design of the robot end-effector and the collision zones, which clearly affects the order of building the corners in the design project, were all taken into consideration. Knowledge elicitation for this rule-based expert system was provided by the author, through the investigation of block and robot geometry, as well as trial and error work.

The resulting rule-base to carry out the task of ordering the building of a designed masonry project was confirmed successful, as well as the C code to utilise it in the process of translating the CAD generated project description into a 'theoretical task'. The resulting programme was tested using the example of a generated CAD design. The result of this test was checked to confirm that the system produced the order of building without collision, which proved to be so.

A limitation to the system was the intelligence of the decision on which end of the block is to be buttered in the construction of corners. A bond parameter should be added to show which face to apply mortar on, and the building order should take into consideration the method of bond application chosen, i.e. if applied on two sides then mortar may have to be injected vertically. In conclusion, even though the system developed had the capability of ordering the construction of limited range of project designs, the use of rule-base intelligence was shown to be effective in tackling the task.

10.9 Rule-base intelligence for optimum move

To investigate the use of machine intelligence for motion control, and determining optimum moves.

A rule-base expert system, made up of thirteen rules, was developed to enable the programming of the robot in an efficient and effective way (chapter 9.4.3). This rule-base translates the ‘project description’ into the ‘theoretical task’. The rule-base detects impossible moves and advises the user against their use. Knowledge elicitation for this rule-based expert system was from two sources, both covered in section 10.4. The first was the theory of robot programming and the results of the experiments testing the system limits and capabilities. The second was the knowledge gathered from the investigation into the vibration of the robot end-effector, which resulted in the establishment of optimum moves.

The rule-base developed, succeeded in demonstrating the principle behind the use of rule-base intelligence for move optimisation. The rule-base was tested by entering different combinations of input values for the attributes, to simulate different move requests. Each time the results concluded from the rule-base were checked and found to be correct. This was done by comparing the results concluded by the rule-base expert system with those determined from the experiments on robot performance and end-effector vibration, both described in section 10.4.

10.10 Integration

To investigate the integration of the design stage of construction task with the process of realisation of the robot building task, using sensing and intelligent processing in the different construction tasks.

The different stages of the building task were each automated successfully (table 10.1), but not all fully integrated, mainly due to lack of time. Each of the tasks listed in table 10.1 was tested individually, thus constraining the variations in the different variables to test the accuracy of the methods developed. Tasks number 1 and 2, and 4, 5, and 6 were successfully integrated, as well as tasks 7 and 9. The end result of the integrated tasks was tested, and the results were discussed earlier in this chapter. In practise, it would be straightforward to integrate the various tasks by passing parameters between the various software modules. Full integration would have enabled further assessment of the effectiveness of the different stages of the real time decisions.

Table 10.1 *Automating task activities*

Task number	Task Activity	Results
1	Designing the masonry project, and determining the position, orientation, and dimension of each individual block	Task automated using a specialised AutoCAD programme.
2	Determining the order of building the designed masonry project	Task automated by developing a rule-base expert system covered in section 10.8.
3	Determining the optimised robot moves to carry out the building process	Task automated by experiments on robot performance covered in section 10.4, and a rule-based expert system covered in section 10.9.
4	Locating the block pick-up position on the conveyor	Task automated by the automatic location of a target on the conveyor, covered in section 10.5.
5	Determining the dimensions of the blocks	Task automated using conveyor sensing, covered in section 10.2
6	Picking-up a block from the conveyor	Task automated using real-time intelligence with sensing, covered in sections 10.2 and 10.7
7	Determining the way the block was picked, and adjusting for that in the operation of the application of mortar, and the position the block is to be laid at	Task automated using gripper sensing and a laser profiler, covered in section 10.6.
8	Application of mortar on individual blocks	Task automated using robot gripper and a computer driven peristaltic pump, covered in section 10.3
9	Laying the blocks in a pre-determined position	Task automated using various sensor information, covered in section 10.6

10.11 Outline system cost

To investigate an outline of the cost of the system

Demonstrations with research machinery have proved a cycle time of 1 min. However, in the research community it is generally thought that time of 40 sec/unit should be possible, improving substantially in the future with advances in sensor provisions and control technology.

The economic evaluation has been calculated using a rate of 40 sec/unit, considering a project of 5 units of masonry work representing small buildings. The cost of manual work for this is estimated at £10685.5, and robotic work at £7805.5.

A cycle time of 20 sec/unit, for example, would lead to a further £1000 advantage with robotic construction. Further savings might be possible if a design for low maintenance strategy could be adopted in the developed of masonry automation systems.

10.12 Future developments

To increase the efficiency and productivity of the process of automating the building task, the issues listed below need further consideration.

(i)- The software programs developed were all sequential. For future work, parallel processing would enable the execution and control of two or more processes, and thus add to the efficiency and productivity of the overall building process.

(ii)- Masonry unit supply to the conveyor should be automated using a robot that supplies the conveyor masonry units from a pallet.

(iii)- Equipment which allows dispensing of mortar on the conveyor, thus reducing the time of cycle of each block, is needed. For a typical work cycle, starting from picking the block, from the conveyor, the duration is 55-65 seconds. The cycle time can vary, depending on the real time adjustments needed, e.g. if the conveyor is out of position and the routine for locating it has to be performed.

(iv)- Integration of the image processing programmes that detect damaged masonry units, needs to be achieved. Where units may be rejected, this would have to communicate with the conveyor processor, to ensure suitable replacement material.

(v)- On the method developed to lay masonry units, the errors in the offsets found for the y-axis direction were not due to incorrect measurements or calculations. A closer examination of the way that blocks were gripped showed, that when the gripper was not central to the block, the block was caused to slant. This error could be corrected if the robot had a pitch axis. Alternatively, a more complex type of gripper could be provided which ensures a central grip of block material.

(vi)- Further studies are needed on mortar dispensing and building bonded assemblies, to understand and document accuracy in building larger assemblies. Realistic designs, which include corners, openings and other geometrical features, would need to be considered. Use of the laser beacon or other automatic survey aids should be considered in this work.

(vii)- For the process of laying the blocks, with good adhesion, a practical solution might be the addition of a small vibration device to the gripper. This would be applied when the block is laid after the gripper has released it.

(viii)- While examining the robot's vibration in a single plane, we have seen how a variety of factors influence the vibration and the settling time of the robot end-effector. The principles and methods used in this study to establish optimal settings for the robot in this single axis can equally well be applied to the other axes, and to different orientations of the end-effector. Further experimentation should be undertaken to give comprehensive multi-axis optimisation.

(ix)- Other methods of cell component detection and location need to be investigated, including image based object recognition. The speed, accuracy and cost of these should be determined in this.

(x)- To enable automatic generation of the necessary robot programme of moves, the result of the three rule-base systems needs to be integrated, along with the robot cell configuration.

(xi) Investigation of robot mobility with obstacle avoidance, using sensing and intelligence to allow the robot to navigate its way in the unstructured environment of the construction site.

Appendix A: Experimental Cell Equipment

A.1 Robot inverse kinematics (Chamberlain, 1989)

Figure 2.3 in chapter 2.2.1 shows the symbolic representation of the robot's degrees of freedom (DOF). The robot used has five degrees of freedom, three are linear motion and two are rotation. Given the required tool centre point (TCP) co-ordinates x,y,z and the orientation angles 'α' (roll) and 'σ' (yaw), we need to find the corresponding absolute values (figure A.1). Mathematically this is stated as:

$$\begin{Bmatrix} (c_1 - s_1)k_1 \\ (c_2 - s_2)k_2 \\ (c_3 - s_3)k_3 \\ (c_4 - s_4)k_4 \\ (c_5 - s_5)k_5 \end{Bmatrix} = \begin{Bmatrix} x_p \\ y_p \\ z_p \\ \sigma_p \\ \alpha_p \end{Bmatrix} - \begin{bmatrix} 3x5Transformation \\ matrix \end{bmatrix} \times \begin{Bmatrix} a_1 \\ a_2 \\ a_3 \end{Bmatrix} \qquad \text{eqn a.1}$$

Where $c_1 \rightarrow c_5$ are the motor counts corresponding to the required vector x_p, y_p ,z_p ,σ_p ,α_p, (i.e. required positions of the TCP). $k_1 \rightarrow k_5$ are the calibration constants which are applied to the required count values $c_1 \rightarrow c_5$, $s_1 \rightarrow s_5$ are the starting motor counts at $x_p=y_p=z_p=\sigma_p=\alpha_p=0$, and d_1, d_2 and d_3 are relative offsets on the x, y and z axes of the robot.

Relative to the joint 5, the TCP position is given by:

$$P_5 = \begin{Bmatrix} a_1 \\ a_2 \\ a_3 \end{Bmatrix} \qquad \text{eqn a.2}$$

where a_1 is +ve, a_2 is +ve and a_3 is -ve. Having defined the TCP in terms of the 5th axis set, we can now define the TCP relative to joint 4, using the position vector P_4 and the transformation matrix $T_{4/5}$.

$$P_4 = (T_{4/5}) \times P_5 \qquad \text{eqn a.3}$$

The next step is to define the TCP in terms of the 3rd axis set, using the position vector P_3 and the transformation matrix $T_{3/4}$:

$$P_3 = T_{(3/4)} P_4 \qquad \text{eqn a.4}$$

The next step is to define the TCP in terms of the 2rd axis set, using the position vector P_2 and the transformation matrix $T_{2/3}$.

$$P_2 = T_{(2/3)} P_3 \qquad \text{eqn a.5}$$

The next step is to define the TCP in terms of the 1st axis set, using the position vector P_1 and the transformation matrix $T_{2/1}$.

$$P_1 = T_{(2/1)} P_2 \qquad \text{eqn a.6}$$

Finally to define the TCP in terms of the origin, using the position vector P_0 and the transformation matrix $T_{1/0}$.

$$P_0 = T_{(1/0)} P_1 \qquad \text{eqn a.7}$$

where

$$T_{(4/5)} = \begin{bmatrix} \sin\alpha & 0 & \cos\alpha \\ 0 & 1 & 0 \\ \cos\alpha & 0 & -\sin\alpha \end{bmatrix} \qquad \text{eqn a.8}$$

and $$T_{(3/4)} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\sigma & -\sin\sigma \\ 0 & \sin\sigma & \cos\sigma \end{bmatrix} \qquad \text{eqn a.9}$$

and $$T_{(2/3)} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & -1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \qquad \text{eqn a.10}$$

and $$T_{(2/1)} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \text{eqn a.11}$$

and $$T_{(1/0)} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \text{eqn a.12}$$

Computing the transformations in equations a.8 $\rightarrow$ a.12 we get:

$$\begin{Bmatrix} d_1 \\ d_2 \\ d_3 \end{Bmatrix} = \begin{Bmatrix} x_p \\ y_p \\ z_p \end{Bmatrix} - \begin{bmatrix} -\sin\sigma\cos\alpha & \cos\sigma & -\sin\sigma\sin\alpha \\ \cos\sigma\cos\alpha & \sin\sigma & \cos\sigma\sin\alpha \\ \sin\alpha & 0 & -\cos\alpha \end{bmatrix} \begin{Bmatrix} a_1 \\ a_2 \\ a_3 \end{Bmatrix} = \begin{Bmatrix} (c_1 - s_1)k_1 \\ (c_2 - s_2)k_2 \\ (c_3 - s_3)k_3 \end{Bmatrix} \qquad \text{eqn a.13}$$

, where d_1, d_2, and d_3 are relative offsets on the x, y, and z axes of the robot.

Finally using equation a.1 and the transformation matrix in equation a.13, and assuming at $x_p=y_p=z_p=\sigma_p=\alpha_p=0$ $s_1=s_2=s_3=s_4=s_5=0$, we get equations a.14-a.17, the motor counts required for the given TCP position and orientation:

$$c_1 = (1/k_1)\times(x_p + a_1\sin\sigma\cos\alpha - a_2\cos\sigma - a_3\sin\sigma\sin\alpha) \qquad \text{eqn a.14}$$

$$c_2 = (1/k_2)\times(y_p - a_1\cos\sigma\cos\alpha - a_2\sin\sigma + a_3\cos\sigma\sin\alpha) \qquad \text{eqn a.15}$$

$$c_3 = (1/k_3)\times(z_p - a_1\sin\alpha - a_3\cos\alpha) \qquad \text{eqn a.16}$$

$$c_4 = (1/k_4)\times\sigma_p \qquad \text{eqn a.17}$$

$$c_5 = (1/k_5)\times\alpha_p \qquad \text{eqn a.18}$$

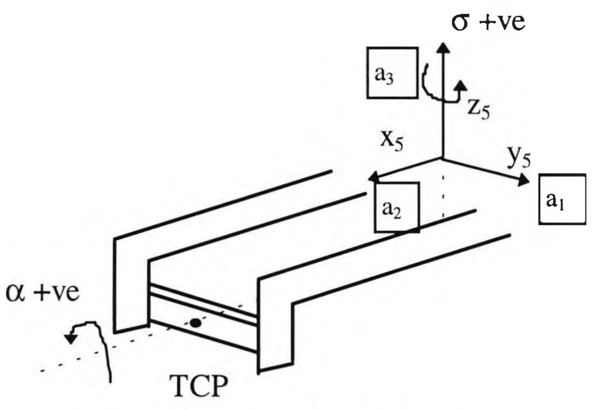
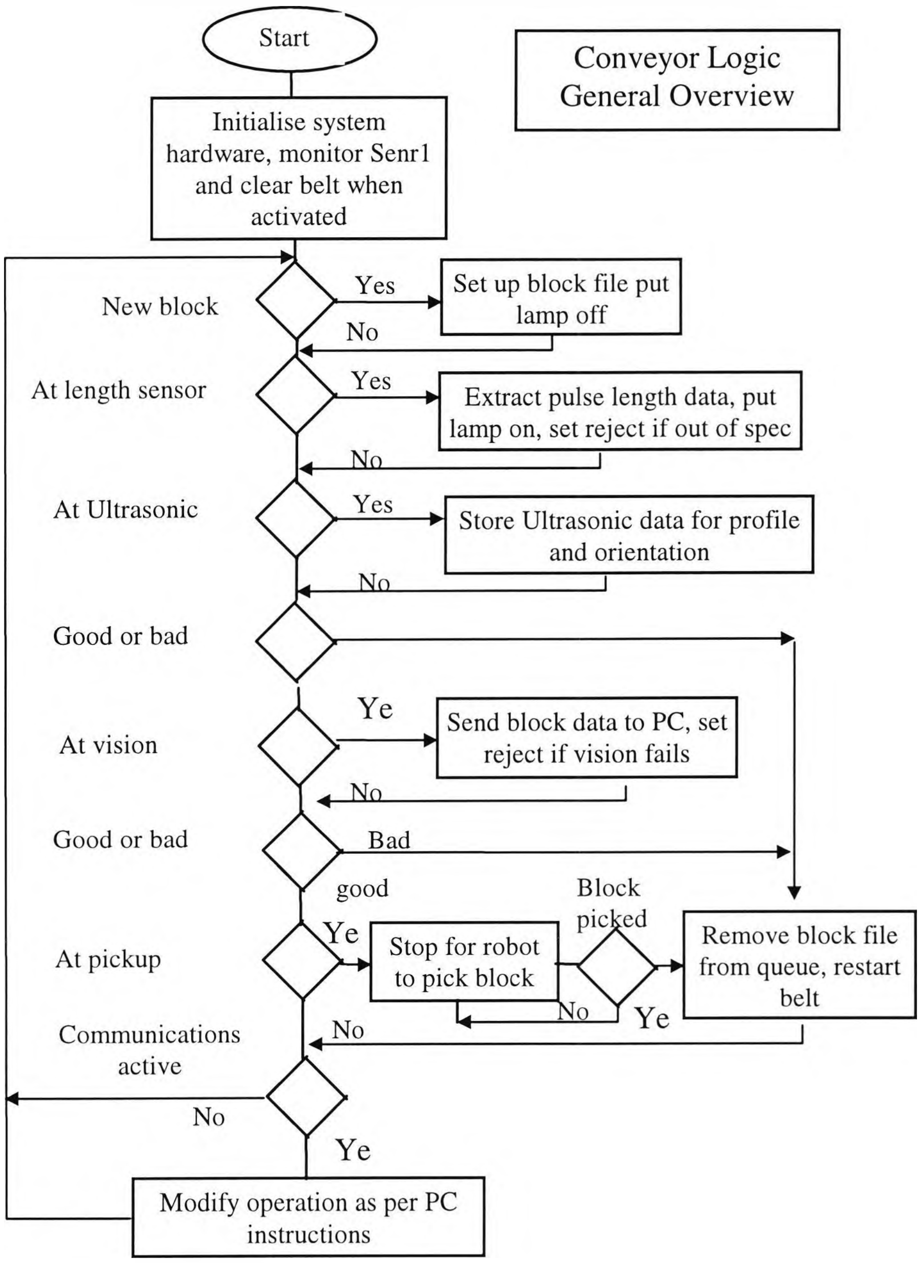


Figure A.1 *Robot end-effector*

A.2 Conveyor logic overview (Reilly, 1992)

Figure A.2 shows the top level logic of the conveyor described in chapter 2.3.1.1.



Start
Conveyor Logic General Overview
Initialise system hardware, monitor Senr1 and clear belt when activated
New block
Yes
No
Set up block file put lamp off
At length sensor
Yes
Extract pulse length data, put lamp on, set reject if out of spec
No
At Ultrasonic
Yes
Store Ultrasonic data for profile and orientation
No
Good or bad
At vision
Ye
Send block data to PC, set reject if vision fails
No
Good or bad
Bad
good
At pickup
Ye
Stop for robot to pick block
Block picked
No
Ye
Remove block file from queue, restart belt
No
Communications active
No
Ye
Modify operation as per PC instructions

Figure A.2 *Top level conveyor logic (Reilly, 1992)*

A.3 Ultrasonic and photoelectric raw data file

Example of a data file generated using the conveyor ultrasonic and photoelectric sensors. For this example a standard block of length 440 mm, height 220 mm, and width of 100 mm was used. The block profiles are used (chapter 2.3.1.2) in determining the block dimensions and orientation on the conveyor. The ASCII characters are the hexadecimal representation of block profile length in encoder pulse counts. The resolution is 2.93 pulses per mm.

The block data file is shown below:

```
GP
050C00
Data for Uson3, side scan. Resolution to 0.5 mm. Represents distance from sensor
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF5E8D9CBBBAC96949281807
F7E7E7E7D7D7D7D7D7C7C7C7C7C7C7C7B7B7B7B7A7A7A7A7A7979797878787
87878787777777777777767676767676757575757574747474737373737373372727
27272727171717070717070707070706F6F6F6F6F6F6E6E6E6E6D6D6D6D6D6D6D6D
6D6D6D6C6C6C6C6C6B6B6B6B6B6A6A6A6A6A6A69696969696969696868686
767676666666666666656665666667 6A6B6B6A6968646362626262616262626262626262
62626262616060 5F5F5F5F5F5F5F5E5E5E5E5E5E5E5E5D5D5D5D5D5C5C5C5C5B
5B5B5B5B5A5A5A5A595959595959595959585858585858575757575757565656565
6565656565555555545454545454545353535352525252525252515151505050505050504F4
F4F4F4F4F4F4F4F4F4F4F4E4E4E4E4D4D4D4D4C4C4C4C4C4B4B4B4B4A4A4A4
A4A4A4A4A4A4949494949494848484848484847484848484848484949 4A4B4C4D4E4F515
2545C6670859AAFD2DFEAFFFFFFFFFFFFFFFFFFFFFF000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000

Data for Uson2, overhead. Resolution to 1.0 mm. Represents height of block.
00000000000000000101010101010101000101011 9334D819BB4D1D2D2D3D3D4D4
D4D4D5D5D6D6D6D6D6D7D7D7D7D7D7D7D6D6D6D7D7D7D7D7D7D7D7D7D7
D7D6D7D6D6D6D6D6D6D6D6D7D7D7D7D7D7D7D6D6D6D6D6D6D6D6D6D6D6
D6D7D7D7D7D7D7D6D6D6D6D6D6D6D6D6D6D6D6D6D6D6D6D6D6D6D6D6
D6D6D6D6D6D6D6D6D6D6D6D6D6D7D7D7D7D7D6D6D7D6D6D6D7D6D6D6D6
D7D6D7D7D7D7D6D7D7D7D7D7D7D7D7D6D6D6D6D6D6D6D6D6D6D6D7D7D7
D7D7D7D7D7D6D6D6D6D6D6D6D6D7D7D7D7D7D7D7D7D7D7D6D6D5D5D5
D5D5D6D6D6D6D6D6D7D7D6D6D6D6D6D6D6D6D6D6D6D6D6D6D6D6D6D6
D6D7D6D6D6D6D6D6D6D6D6D6D6D6D6D7D7D6D7D6D6D6D6D6D6D6D6D6
D6D6D6D6D6D6D6D6D6D6D6D6D6D6D6D6D6D6D6D6D6D6D6D6D6D6D6D6
D6D6D6D6D6D6D6D6D6D6D6D6D6D6D6D6D6D6D6D5D5D6D6D6D6D6D6D6
D6D6D6D6D6D6D6D6D6D6D6D6D6D6D6D6D6D6D6D6D6D6D6D6D6D6D6D6
D6D6D6D6D5D5D5D4D4D3D1D1BA866C521E030100000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000
```

Data for Uson1, side scan. Resolution to 0.5 mm. Represents distance from sensor

FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF6EADBB19C8561574F
4B494846464544444343434242424242424242424242424242424242434343444444444
44444444444445454545464646464646464646464647474747474848484848484848494 9
49494949494A4A4A4A4A4A4A4B4B4B4B4B4C4C4C4C4C4C4D4D4D4D4D4E4E4
E4E4E4E4E4E4E4E4F4F4F50504F505050515151525252525252525253535353535
4545454555555555555555555556565657575757575757575858595A5A5B5B5B5A59595
95959595959595A5A5A5B5D5F5F60605F5E5D5D5D5D5E5E5E5E5E5E5E5E5E5F5
F5F5F5F5F6060606060606060616161616262626262626262626263636363636363646
4646464646565656666666767676767676767686868686868686969696969696969696A6A6
A6A6B6B6B6B6B6B6B6C6C6C6C6C6D6D6D6D6D6E6E6E6E6F6F6F6F6F6F6F6F6
F7070707070707071717172727272727273737373737373747474747575767676767778797
B7C7D8192A2C2D2E1FFFFFFFFF00000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
00000000

A.4 Survey of 'Celcon' blocks used

A survey of 12 blocks was carried out, and results are in table A.1. This was done to show the tolerance expected from the type of materials used for this research (chapter 2.4.2).

Table A.1 *Survey of blocks*

Block No.	length	Width	Height
1	438.34	101.48	212.80
2	438.80	99.52	213.04
3	440.20	100.42	212.16
4	438.64	101.60	212.50
5	438.86	100.20	213.06
6	438.84	100.60	214.30
7	440.00	100.52	214.80
8	441.30	101.38	214.30
9	440.00	101.50	212.36
10	438.72	100.50	213.34
11	438.44	99.40	214.30
12	440.60	98.86	212.36
Mean reading	439.37	100.5	213.28
Max. reading	441.30	101.6	214.80
Mean reading	438.34	98.86	212.16
Mean of absolute difference	0.96	0.87	0.75
Max. of absolute difference	1.66	1.6	1.8
Min. of absolute difference	0	0.2	0.04

A.5 Laser analog displacement sensor (LADS)

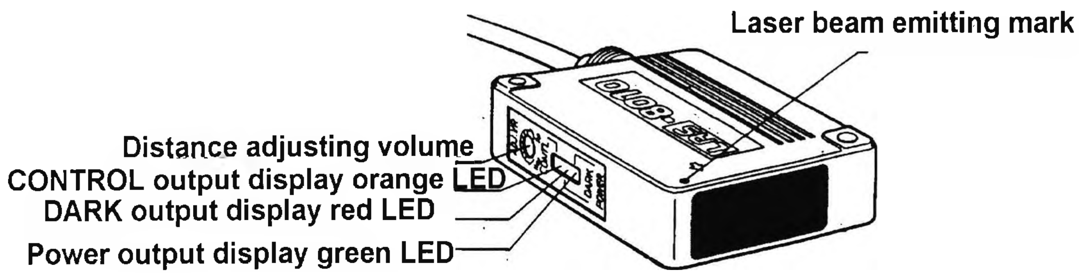

Laser beam emitting mark
Distance adjusting volume
CONTROL output display orange LED
DARK output display red LED
Power output display green LED

Figure A.3 *Identification of LADS parts*

Table A.2 *Specification of laser displacement sensor*

Model	LAS-8010V
ANALOG output	4~20 mA
Reference distance	100 ± 1mm
Measurable range	±40 mm from reference distance
Response	20mS
Variation due to temperature fluctuation	±0.1% F.S. /^{0}C
Spot diameter	Ellipsoid, less than 2mm in shorter diameter at reference distance
Operation environment temperature	0~+50 ^{0}C
Error of ideal linearity	±150μm ±(displaced length x 1%)
Operation environment humidity	35~85%RH
Net weight (include. cable)	310g

A.6 PICO A/D converter

The ADC-11 is an analog to digital converter which connects to the printer port of an IBM PC compatible computer. It has 11 analogue input channels, three test channels and a digital output. Figure A.4 shows this device. The sensor signals used in this research were converted from analog to digital using this A/D converter (PICO) as described in chapters 2.4.1 and 2.5.2.

Table A.3 *Specifications of A/D converter*

Resolution	10 bits
Number of analogy inputs	11
Analog input range	0-2.5V
Maximum sampling rate	18ksps (33Mhz 386/486
Linearity	+ LSB
Accuracy	+1%
Overload protection	+30V
Analog Input impedance	>200KΩ
Number of digital outputs	1 (TTL)
Digital output impedance	approx. 1 - 3Ω
Input connector	25 way female D-type
Connection to computer	25 way male D-type

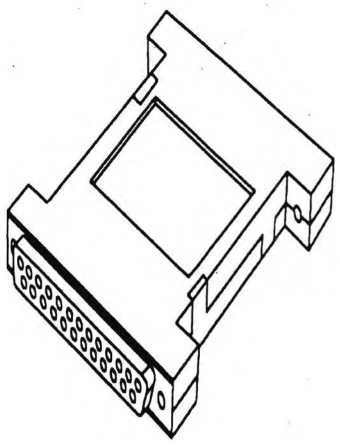


Figure A.4 *PICO A/D converter*

A.7 Accelerometer

This section gives a detailed description of the accelerometer used, in the experiments described in chapter 2.8. The method of calibrating the accelerometer, and filtering the accelerometer signal are also covered.

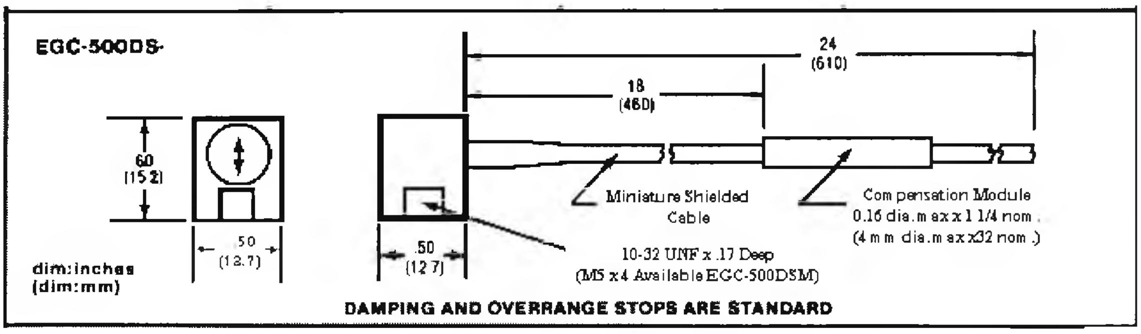


Figure A.5 *Accelerometer*

Table A.4 *Accelerometer specification*

RANGE	±5g	±10g	±20g	±50g	±100g	±200g	±500g	±1000g	±2000g	±5000g
[1] OVERRANGE	±50g	±100g	±200g	±500g	±1000g	±2000g	±5000g	±10000g	±20000g	±50000g
SENS. mV/g nom.	25	20	10	4	2	1	0.4	0.2	0.1	0.04
[2] RES. FREQ. nom.	370 Hz	500 Hz	800 Hz	1500 Hz	2300 Hz	2800 Hz	3500 Hz	4000 Hz	5000 Hz	7000 Hz

* "OFF-THE-SHELF" STOCK IN EGC-500DS-5, -50 AND -200.

NON-LINEARITY	±1%	INPUT IMPEDANCE nom.	1000Ω typ.: 2000Ω optional (500Ω min.) with 900Ω output
TRANS. SENS.	3% max.	OUTPUT IMPED. nom.	450Ω: 900Ω optional with 2000Ω input
THERMAL ZERO	[4,5] ±1%F.S/100°F	EXCITATION	[6] 15VDC
THERMAL SENS.	±2½%/100°F	COMPENSATED TEMP.	[6] 70° F to 170° F OPTION "Z": 32° F to 140° F
DAMPING	[3] 0.7cr nom.	OPERATING TEMP.	[6] -40°F to 250°F (-40°C to 121°C)
USEFUL FREQ. RANGE	[2] 30% to 50% Res. Freq	WEIGHT (without cable)	Steel 20gms Al 10gms

A.7.1 Calibrating the accelerometer

The accelerometer was calibrated using the following method. The value of x_1, which is the accelerometer's readings when its beam is deflected one way, i.e. reading the value of "g" as seen in figure A.6, was recorded. The value x_2 , which is when the beam is deflected the opposite way, i.e. reading the value of "-g" as shown in figure A.7, was also recorded. The absolute values for each was taken. They should equal twice "g". The calibration constant will be equal to the value of X calculated using equation a.19 shown below.

$$X = \frac{2\times "g"}{|x_1| + |x_2|} \quad m/s^2 \big/ volt \qquad \text{eqn a.19}$$

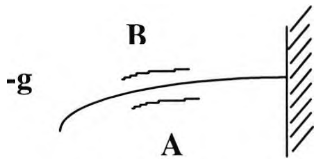


Figure A.6 Accelerometer reading '-g'

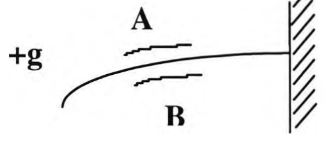
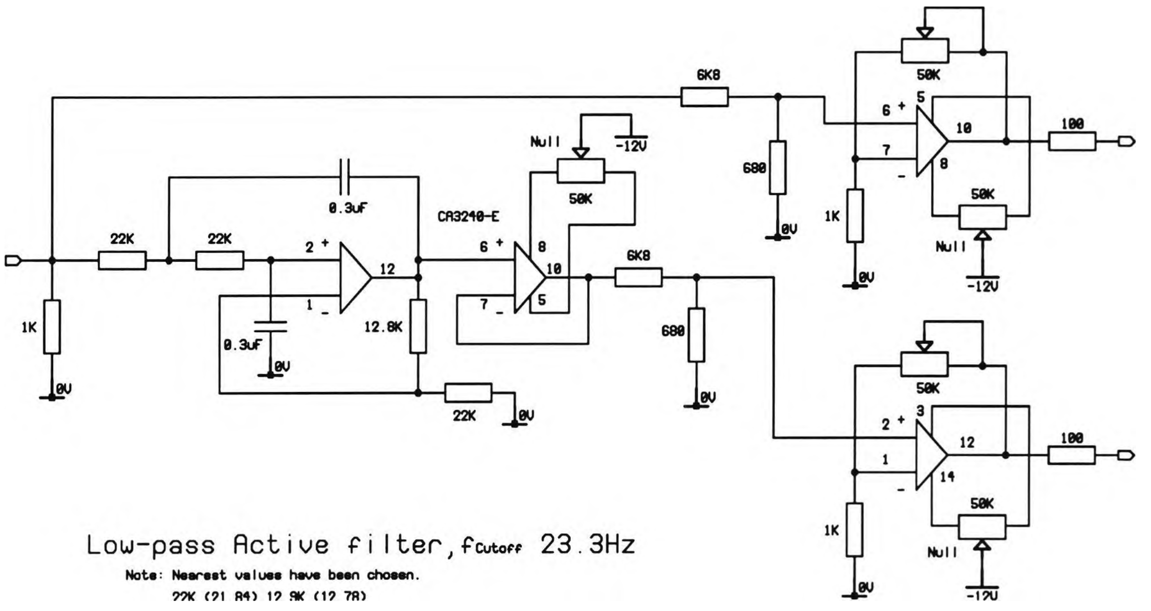


Figure A.7 Accelerometer reading '+g'

The calibration constant is used to convert voltage reading into accelerations as described in chapter 2.8.3. (Entran's sensor accelerometer instruction and selection manual).

A.7.2 Accelerometer signal filter

Figure A.8 shows a low pass active filter, used in filtering the accelerometer signal, captured by the *PC-30* board. This is done to reduce noise, as described in chapter 2.8.2.2.



22K
22K
1K
0.3uF
0.3uF
0V
12.8K
22K
CA3240-E
Null
-12V
50K
6K8
680
1K
50K
100
Null
-12V
Low-pass Active filter, fCutoff 23.3Hz
Note: Nearest values have been chosen.
22K (21.84) 12.9K (12.78)

Figure A.8 *Low-pass active filter for the accelerometer signal*

A.7.3 Accelerometer data capturing

A.7.3.1 *PC-30* board

The *PC-30* board is an analog to digital converter. It has a maximum throughput of 25KHz. There are 24 digital I/O lines, each of which may be programmed as an input or output. Part of the capabilities of *PC-30* are diagnostics, analog output on all four D/A outputs, obtain a series of A/D samples on either a single or multiple channel, interrupt operations, digital I/O, counter/timer operation (*PC-30* Driver software package user manual).

A.7.3.2 Status-30 software

Example of data captured using the status-30 software. The data captured is of 2 channels, read in *PC-30*, and translated into text file using status 30 is shown in table A.5. This is referred to in chapter 2.8.2.

Table A.5 *Example of data captured from accelerometer signal*

```
"Sample Taken at : ","12:16:7 Tue 13 Jun 1995"
"Channel 0 : Volts"
"Channel 2 : Volts"

"Channel :",        0,        2
  0.00000, 0.073242,  0.200195
  0.00500, 0.075684, -0.065918
  0.01000, 0.065918,  0.097656
  0.01500, 0.073242, -0.002441
  0.02000, 0.073242,  0.214844
  0.02500, 0.075684,  0.083008
  0.03000, 0.080566, -0.114746
  0.03500, 0.080566, -0.185547
     |          |          |
```

A.7.3.3 Sampling theorem

If the highest frequency component in a signal is f_{max} then the signal should be sampled at the rate of at least $2f_{max}$ for the samples to describe the signal completely:

$$F_s \geq 2f_{\max}$$

Where F_s is the sampling frequency or rate (Emmanuel *et al.*, 1993). Thus, if the maximum frequency component in an analogue signal is 4 kHz, ten to preserve or capture all the information in the signal it should be sampled at 8 kHz or more. An important point to remember is that a signal often has significant energy outside of the highest frequency of interest and/or contains noise, which invariably has a wide bandwidth. Thus, the sampling theorem will be violated if we do not remove the signal or noise outside of the band of interest. In practice, this is achieved by first passing the signal through an analogue anti-aliasing filter.

A.7.4 Results of end-effector manual rocking experiment

This section has a listing of the results for the manual rocking motion experiment described in section 2.8.3.5.4. The data for figure 2.8.33, which is the plot of the differences in the average amplitudes of the accelerometer and the laser, is shown in table A.6 below.

Table A.6 *Data for figure 2.8.33*

Time (secs)	Accelerometer d_{ave} (mm)	Time (secs)	laser d_{ave} (mm)	Time (secs)	d_{ave} difference (mm)
0.09	6.46	0.09	6.68	0.09	-0.22
0.47	11.10	0.57	11.11	0.47	0.00
0.84	11.92	0.85	11.73	0.84	0.19
1.21	11.64	1.21	11.23	1.21	0.41
1.59	12.05	1.58	11.73	1.59	0.32
1.95	11.85	1.95	11.54	1.95	0.31
2.32	10.60	2.32	10.34	2.32	0.26
2.68	10.03	2.68	9.74	2.68	0.29
3.05	11.91	3.05	11.66	3.05	0.26
3.42	11.02	3.42	10.87	3.42	0.16
3.78	8.32	3.78	8.23	3.78	0.09
4.11	6.50	4.11	6.49	4.11	0.01
4.47	4.60	4.47	4.31	4.47	0.29
4.83	2.67	4.83	2.54	4.83	0.13
5.16	1.58	5.16	1.56	5.16	0.02
5.45	0.93	5.46	0.96	5.45	-0.02
5.74	0.39	5.74	0.45	5.74	-0.06
5.86	0.35	5.86	0.15	5.86	0.20

A.7.5 Results of robot motion experiments

Three different types of moves were made by the robot in order to study the effectiveness of the accelerometer in displacement estimation (chapter 2.8.3.5.5).

A.7.5.1 Strong vibration move

Results of the experiments carried on a strong vibration move, described in chapter 2.8.3.5.5.1. The data for figure 2.8.36, which is the plot of the differences in the average amplitudes of the accelerometer and the laser, is listed in table A.7 below.

Table A.7 *Data for figure 2.8.36*

Time (secs)	Accelerometer d_{ave} (mm)	Time (secs)	laser d_{ave} (mm)	Time (secs)	d_{ave} difference (mm)
0.08	7.26	0.08	8.09	0.08	-0.83
0.40	8.62	0.40	9.24	0.40	-0.62
0.74	5.62	0.72	5.94	0.74	-0.31
1.24	3.87	1.24	3.90	1.24	-0.04
1.92	5.44	1.92	5.46	1.92	-0.02
2.30	2.80	2.30	2.99	2.30	-0.19
2.74	2.70	2.74	2.78	2.74	-0.07
3.12	2.03	3.12	2.20	3.12	-0.18
3.60	0.62	3.60	0.84	3.60	-0.22
4.06	0.75	3.96	0.88	4.06	-0.13
4.48	0.51	4.48	0.48	4.48	0.03
4.80	0.37	4.80	0.38	4.80	-0.02
5.06	0.45	5.06	0.29	5.06	0.16
5.52	0.17	5.28	0.06	5.52	0.10
5.70	0.13	5.78	0.17	5.70	-0.04
6.00	0.18	5.90	0.02	6.00	0.15
6.34	0.35	6.32	0.19	6.34	0.16

A.7.5.2 Medium vibration move

Results of the experiments carried on a medium vibration move, described in chapter 2.8.3.5.5.2. The data for figure 2.8.39, which is the plot of the differences in the average amplitudes of the accelerometer and the laser, are listed in table A.8.

Table A.8 *Data for figure 2.8.39*

Time (secs)	Accelerometer d_{ave} (mm)	Time (secs)	laser d_{ave} (mm)	Time (secs)	d_{ave} difference (mm)
0.10	2.48	0.10	2.99	0.10	-0.52
0.38	1.30	0.38	1.65	0.38	-0.35
0.64	1.09	0.64	0.32	0.64	0.77
0.90	0.98	0.90	1.03	0.90	-0.05
1.18	0.66	1.18	0.60	1.18	0.06
1.46	0.28	1.42	0.50	1.46	-0.22
1.70	0.22	1.68	0.24	1.70	-0.02
2.04	0.19	1.96	0.13	2.04	0.07
2.42	0.05	2.46	0.22	2.42	-0.17
2.68	0.09	2.72	0.21	2.68	-0.12
2.96	0.12	2.88	0.10	2.96	0.02

A.7.5.3 Low vibration move

Results of the experiments carried on a medium vibration move, described in chapter 2.8.3.5.5.3. The data for figure 2.8.42, which is the plot of the differences in the average amplitudes of the accelerometer and the laser, are listed in table A.9.

Table A.9 *Data for figure 2.8.42*

Time (secs)	Accelerometer d_{ave} (mm)	Time (secs)	laser d_{ave} (mm)	Time (secs)	d_{ave} difference (mm)
0.12	0.43	0.12	0.60	0.12	-0.17
0.38	0.34	0.38	0.50	0.38	-0.16
0.82	0.78	0.80	0.53	0.82	0.25
1.02	0.15	1.00	0.17	1.02	-0.02
1.36	0.13	1.34	0.19	1.36	-0.06
1.74	0.03	1.74	0.41	1.74	-0.38
1.92	0.04	1.96	0.14	1.92	-0.10
2.12	0.01	2.12	0.05	2.12	-0.04
2.26	0.03	2.28	0.00	2.26	0.02

Appendix B: Robot Move Profile

B.1 Profile of a Parabolic 'U' move

Data captured as a result of executing a parabolic U move, of ΔP =743 mm in Δt = 30 seconds, using program '*plot.c*' (chapter 3.8.1).

Time (Seconds)	Distance (mm)	Velocity (mm/s)
0.00	0.00	0.00
0.28	0.49	1.02
0.55	1.13	2.54
0.99	2.91	4.06
1.37	5.27	6.09
1.76	8.29	7.61
2.14	11.94	9.14
2.53	16.20	10.66
2.91	21.11	11.68
3.30	26.59	13.20
3.68	32.61	14.72
4.07	39.18	17.26
4.45	46.31	18.27
4.89	55.05	19.80
5.27	63.25	20.81
5.66	71.92	21.83
6.04	81.02	22.84
6.43	90.53	24.37
6.81	100.47	25.38
7.20	110.87	25.89
7.58	121.59	27.92
7.96	132.67	28.93
8.35	144.07	29.44
8.73	155.84	30.46
9.12	167.88	30.96
9.50	180.18	31.47
9.89	192.79	32.49
10.27	205.59	32.99
10.66	218.70	33.50
11.04	231.94	34.01
11.43	245.40	35.03
11.81	259.05	35.53
12.19	272.77	35.53
12.58	286.72	36.04
12.96	300.72	36.55
13.35	314.97	36.55
13.73	329.04	36.55
14.12	343.31	37.06
14.50	357.55	37.06
14.89	371.83	37.06
15.27	386.22	37.06
15.65	400.43	37.06
16.04	414.70	37.06
16.42	428.86	37.06
16.81	442.96	36.55
17.19	457.01	36.55
17.58	470.90	36.04
17.96	484.70	36.04
18.35	498.31	35.53
18.73	511.75	35.03
19.11	525.03	34.52
19.50	538.08	34.01
19.88	550.93	33.50
20.27	563.46	32.99
20.65	575.78	31.98
21.04	587.87	31.47
21.42	599.68	30.96
21.81	610.98	29.95
22.19	622.06	29.44
22.58	632.79	27.92
22.96	643.14	26.90
23.34	653.09	25.89
23.73	662.58	24.87
24.11	671.72	23.86
24.50	680.38	22.84
24.88	688.55	21.83
25.27	696.23	20.81
25.65	703.40	19.29
26.04	710.05	18.27
26.42	716.16	15.74
26.80	721.69	14.21
27.19	726.63	13.20
27.57	730.99	11.68
27.96	734.73	10.15
28.34	737.80	8.63
28.73	740.23	7.11
29.11	741.97	5.58
29.50	743.00	4.06
29.88	743.34	2.03

B.2 Profile of a S-curve move

Data captured as a result of executing a single part S-curve move, of distance ΔP =1744 mm and duration of Δt = 45 seconds, using program '*plot.c*' (chapter 3.8.2).

Time (Secs)	**Distance (mm)**	**Velocity (mm/s)**
0.00	0.00	0.00
1.54	0.32	0.51
1.65	0.35	0.51
1.93	0.59	1.02
2.2	0.88	1.02
2.48	1.27	1.52
2.75	1.75	2.03
3.02	2.33	2.54
3.3	3.03	3.05
3.57	3.86	3.05
3.85	4.82	3.55
4.12	5.96	4.57
4.4	7.21	5.08
4.67	8.66	5.08
4.95	10.29	6.09
5.22	12.12	6.60
5.5	14.13	7.61
5.77	16.37	8.63
6.05	18.83	8.63
6.32	21.51	9.64
6.59	24.45	11.17
6.87	27.64	12.18
7.25	32.56	13.71
7.53	36.40	14.72
7.8	40.56	14.72
8.08	45.00	16.24
8.35	49.77	17.77
8.63	54.83	19.29
8.9	60.27	20.81
9.18	66.04	20.81
9.45	72.21	22.34
9.73	78.64	24.37
10	85.52	25.89
10.27	92.79	25.89
10.55	100.44	27.92
10.82	108.49	29.95
11.1	116.97	31.98
11.37	125.98	34.01
11.65	135.29	34.01
11.92	145.09	36.04
12.2	155.35	38.07
12.47	166.13	40.10
12.75	177.35	42.64
13.02	189.12	42.64
13.3	201.34	44.67
13.57	214.13	47.21
13.84	227.39	49.75
14.12	241.25	49.75
14.39	255.55	52.28
14.67	270.58	54.82
14.94	286.05	57.36
15.22	302.24	60.41
15.49	318.87	60.41
15.77	335.98	62.94
16.04	353.75	64.97
16.32	371.97	67.51
16.59	390.66	69.54
16.87	409.95	69.54
17.14	429.56	71.57
17.41	449.63	73.60
17.69	470.12	75.13
18.13	490.84	77.16
18.35	520.66	78.68
18.62	542.35	78.68
18.9	564.25	80.20
19.17	586.45	81.22
19.45	608.98	82.23
19.72	631.68	83.25
20.00	654.74	83.25
20.27	677.85	84.26
20.55	701.14	85.28
20.82	724.57	85.79
21.09	748.23	85.79
21.37	771.81	86.29
21.64	795.69	86.80
21.92	819.51	86.80
22.19	843.57	86.80
22.47	867.65	86.80
22.74	891.39	87.31
23.02	915.29	86.80
23.29	939.24	86.80
23.57	963.11	86.29
23.84	986.83	86.29
24.12	1010.44	85.79

24.39	1034.10	85.28
24.66	1057.38	84.77
24.94	1080.55	84.77
25.21	1103.70	83.76
25.49	1126.40	82.74
25.76	1149.10	81.73
26.04	1171.33	80.20
26.31	1193.41	80.20
26.59	1215.23	79.19
26.86	1236.71	77.66
27.14	1257.79	76.14
27.41	1278.51	74.11
27.69	1298.83	74.11
27.96	1318.90	72.08
28.24	1338.32	70.05
28.51	1357.45	68.02
28.78	1376.05	68.02
29.06	1394.16	65.99
29.33	1411.89	63.45
29.61	1428.95	60.91
29.88	1445.47	58.38
30.16	1461.46	58.38
30.43	1476.83	55.84
30.71	1491.62	53.30
30.98	1506.01	50.76
31.26	1519.65	48.22
31.53	1532.80	48.22
31.81	1545.44	45.69
32.08	1557.55	43.15
32.35	1569.24	41.12
32.63	1580.34	41.12
32.9	1591.04	38.58
33.18	1601.17	36.55
33.45	1610.95	34.52
33.73	1620.17	32.49
34	1629.04	32.49
34.28	1637.42	28.43
34.72	1645.40	28.43
34.94	1655.88	26.40
35.21	1662.93	24.87
35.49	1669.53	22.84
35.76	1675.79	22.84
36.03	1681.72	21.32
36.31	1687.27	19.80
36.58	1692.48	18.27
36.86	1697.36	16.75
37.13	1701.94	16.75
37.41	1706.19	15.23
37.68	1710.17	13.71
37.96	1713.84	12.69
38.23	1717.23	11.68
38.51	1720.37	11.68
38.78	1723.25	10.15
39.06	1725.90	9.14
39.44	1729.19	8.12
39.71	1731.31	7.11
40.1	1733.90	6.09
40.37	1735.52	5.58
40.65	1736.95	5.58
40.92	1738.22	4.57
41.2	1739.33	4.06
41.47	1740.29	4.06
41.75	1741.10	3.05
42.02	1741.80	2.54
42.3	1742.38	2.54
42.57	1742.85	2.03
42.85	1743.23	1.52
43.12	1743.52	1.02
43.39	1743.73	1.02
43.67	1743.89	1.02
43.94	1743.99	0.51
44.22	1744.05	0.51
44.49	1744.09	0.00
44.77	1744.10	0.00

Appendix C: Robot Motion Experiments

C.1 Linear move experiments

Listed below are the linear moves tested (chapter 4.3.1).

Programming a one part linear move

T(xx)	;Where (xx) = time from start to start of deceleration
t (xx)	; Time to acceleration and deceleration.
U(yy)	;Where (yy) = Δp = Total move increment. This move start
END	;and ends with a zero velocity.

Δt = Time for the move + Deceleration time of move

Each move has a total move time of 3.32 seconds and a move distance ΔP of 1981 mm

File name	$t_{accel/decel}$ (secs)	T (secs)	V_{max} (mm/s)	A_{max} (mm/s^2)	Max. amplitude (mm/s^2)	Settling time (secs)
dy1	1.66	0.05	1194	719	6078	1.84
dy6	1.56	1.76	1127	722	6165	1.84
dy3	0.98	2.34	846	866	1727	0.66
dy4	0.39	2.93	676	1732	4301	2.4
dy5	0.15	3.17	624	4263	8150	4.88

Each move has a total move time of 5.0 seconds and a move distance ΔP of 1981 mm

File name	$t_{accel/decel}$ (secs)	T (secs)	V_{max} (mm/s)	A_{max} (mm/s^2)	Max. amplitude (mm/s^2)	Settling time (secs)
dy7	2.50	0.05	793	317	1013	0
dy8	2.27	2.73	725	320	915	0
dy9	0.68	4.32	459	672	1806	0.96
dy11	0.15	4.85	408	2787	9779	3.4

Each move has a total move time of 10.0 seconds and a move distance ΔP of 1981 mm

File name	$t_{accel/decel}$ (secs)	T (secs)	V_{max} (mm/s)	A_{max} (mm/s^2)	Max. amplitude (mm/s^2)	Settling time (secs)
dy12	5.00	0.05	396	79	639	0
dy13	4.82	5.18	383	79	542	0
dy14	0.98	9.02	220	225	697	0
dy15	0.24	9.76	203	832	2406	1.62
dy16	0.12	9.85	201	1614	4153	3.02

Each move has a total move time of 1.66 seconds and a move distance ΔP of 991 mm

File name	$t_{accel/decel}$ (secs)	T (secs)	V_{max} (mm/s)	A_{max} (mm/s^2)	Max. amplitude (mm/s^2)	Settling time (secs)
dy17	0.83	0.02	1194	1438	8435	4.7
dy23	0.98	0.68	1015	1039	4514	2.86
dy18	0.88	0.78	1127	1283	7224	1.84
dy19	0.59	1.07	922	1574	3994	1.56
dy20	0.39	1.27	780	1998	7125	2.4
dy22	0.20	1.46	676	3463	7955	4.24

Each move has a total move time of 3.52 seconds and a move distance ΔP of 991 mm

File name	$t_{accel/decel}$ (secs)	T (secs)	V_{max} (mm/s)	A_{max} (mm/s^2)	Max. amplitude (mm/s^2)	Settling time (secs)
dy24	1.76	0.02	564	321	923	0
dy25	1.27	2.25	441	347	758	0
dy26	0.49	3.03	327	670	1361	0.7
dy27	0.29	3.22	307	1049	3223	2.14
dy28	0.15	3.37	294	2008	6997	2.6

Each move has a total move time of 7 seconds and a move distance ΔP of 991 mm

File name	$t_{accel/decel}$ (secs)	T (secs)	V_{max} (mm/s)	A_{max} (mm/s^2)	Max. amplitude (mm/s^2)	Settling time (secs)
dy29	3.52	0.02	282	80	445	0
dy30	2.93	4.10	242	82	518	0
dy31	1.46	5.57	178	122	426	0
dy32	0.68	6.35	156	228	456	0
dy33	0.29	6.74	147	502	1119	0.62
dy34	0.15	6.88	144	982	2921	2.16

Each move has a total move time of 0.5 seconds and a move distance ΔP of 15.2 mm

File name	$t_{accel/decel}$ (secs)	T (secs)	V_{max} (mm/s)	A_{max} (mm/s^2)	Max. amplitude (mm/s^2)	Settling time (secs)
dy66	0.25	0.0010	61	244	375	0
dy67	0.23	0.27	57	245	716	0.64
dy68	0.20	0.30	50	256	503	0
dy69	0.15	0.35	43	294	437	0
dy70	0.05	0.45	34	692	368	0

Each move has a total move time of 1 (sec) and a move distance ΔP of 15.2 mm

File name	$t_{accel/decel}$ (secs)	T (secs)	V_{max} (mm/s)	A_{max} (mm/s^2)	Max. amplitude (mm/s^2)	Settling time (secs)
dy58	0.5	0.001	30	61	186	0
dy60	0.24	0.76	20	83	188	0
dy61	0.10	0.90	17	173	172	0
dy62	0.05	0.95	16	328	219	0

Each move has a total move time of 3 seconds and a move distance ΔP of 15.2 mm

File name	$t_{accel/decel}$ (secs)	T (secs)	V_{max} (mm/s)	A_{max} (mm/s^2)	Max. amplitude (mm/s^2)	Settling time (secs)
dy64	1.5	0.019	10	7	147	0

Each move has a total move time of 0.117 seconds and a move distance ΔP of 3.9 mm

File name	$t_{accel/decel}$ (secs)	T (secs)	V_{max} (mm/s)	A_{max} (mm/s^2)	Max. amplitude (mm/s^2)	Settling time (secs)
dy45	0.0586	0.0195	68	1154	597	0

Each move has a total move time of 0.195 seconds and a move distance ΔP of 3.9 mm

File name	$t_{accel/decel}$ (secs)	T (secs)	V_{max} (mm/s)	A_{max} (mm/s^2)	Max. amplitude (mm/s^2)	Settling time (secs)
dy48	0.098	0.020	41	416	433	0
dy49	0.088	0.107	37	420	412	0

Each move has a total move time of 0.78 seconds and a move distance ΔP of 3.9 mm

File name	$t_{accel/decel}$ (secs)	T (secs)	V_{max} (mm/s)	A_{max} (mm/s^2)	Max. amplitude (mm/s^2)	Settling time (secs)
dy52	0.391	0.020	10	26	138	0
dy53	0.381	0.400	10	26	129	0
dy54	0.195	0.586	7	35	170	0
dy55	0.049	0.732	5	111	134	0

C.2 Parabolic move experiments

C.2.1 Parabolic 'U' move

Listed below are the parabolic U moves tested (chapter 4.3.2.1).

Programming parabolic U move

t (xx)	;where (xx) = Δt = Total time for the move
U(yy):0	;where (yy) = Δp = Total move increment. This move starts
END	; and end with zero velocity.

Each move has a total distance ΔP of 1981 mm

File name	Δt (secs)	V_{max} (mm/s)	A_{max} (mm/s^2)	Max. amplitude (mm/s^2)	Settling time (secs)
p106	2.93	1015	1385	1695	0.80
p1	3.32	895	1079	1321	0.68
p2	5.00	595	476	691	0.00
p3	10.00	297	119	547	0.00

Each move has a total distance ΔP of 991 mm

File name	Δt (secs)	V_{max} (mm/s)	A_{max} (mm/s^2)	Max. amplitude (mm/s^2)	Settling time (secs)
p107	1.46	1015	2771	7922	1.8
p4	1.66	895	2157	2838	2.9
p108	1.95	761	1558	2352	1.9
p5	3.52	423	481	634	0
p6	7.03	211	120	481	0

Each move has a total distance ΔP of 15.2 mm

File name	Δt (secs)	V_{max} (mm/s)	A_{max} (mm/s^2)	Max. amplitude (mm/s^2)	Settling time (secs)
p154	1	23	91	149	0
p155	3	8	10	160	0
p156	5	5	4	127	0

Each move has a total distance ΔP of 3.9 mm

File name	Δt (secs)	V_{max} (mm/s)	A_{max} (mm/s^2)	Max. amplitude (mm/s^2)	Settling time (secs)
p10	0.17	36	862	1186	0.62
p11	0.29	20	277	310	0.00
p110	0.39	15	156	201	0.00
p109	0.78	8	39	147	0.00

Each move has a total distance ΔP of 1 mm

File name	Δt (secs)	V_{max} (mm/s)	A_{max} (mm/s^2)	Max. amplitude (mm/s^2)	Settling time (secs)
p112	0.04	38	3896	168	0.00

C.2.2 Segmented parabolic moves tested

Listed below are the segmented parabolic moves tested (chapter 4.3.2.2).

Each move has a total move time of 10 seconds and a move distance ΔP of 1981 mm

File name	V_{max} (mm/s)	$\Delta t_{accel/decel}$ (secs)	$\Delta p_{accel/decel}$ (mm)	Δt_{slew} (secs)	Δp_{slew} (mm)	A_{max} (mm/s^2)	Max. amplitude (mm/s^2)	Settling time (secs)
p135	255	3.33	566	3.33	849	153	472	0
p176	200	0.11	15.24	9.77	1951	3488	5826	1.98
p177	212	0.97	137.20	8.06	1707	436	537	0
p178	213	1.07	152.44	7.86	1677	398	500	0
p179	244	2.81	457.32	4.38	1067	173	524	0
p180	259	3.53	609.76	2.94	762	147	696	0
p181	282	4.46	838.41	1.08	305	126	659	0
p182	294	4.90	960.37	0.21	61	120	495	0
p183	296	4.97	983.23	0.05	15	119	557	0

Each move has a total move time of 5 seconds and a move distance ΔP of 1981 mm

File name	V_{max} (mm/s)	$\Delta t_{accel/decel}$ (secs)	$\Delta p_{accel/decel}$ (mm)	Δt_{slew} (secs)	Δp_{slew} (mm)	A_{max} (mm/s^2)	Max. amplitude (mm/s^2)	Settling time (secs)
p136	510	1.67	566	1.67	849	611	717	0
p184	409	0.24	65	4.53	1852	3448	9913	1.76
p185	427	0.54	152	3.93	1677	1593	2655	1
p186	457	1.00	305	3.00	1372	915	1250	0
p187	518	1.76	610	1.47	762	587	663	0
p188	549	2.08	762	0.83	457	527	782	0
p189	564	2.23	838	0.54	305	506	624	0

Each move has a total move time of 3.32 seconds and a move distance ΔP of 1981 mm

File name	V_{max} (mm/s)	$\Delta t_{accel/decel}$ (secs)	$\Delta p_{accel/decel}$ (mm)	Δt_{slew} (secs)	Δp_{slew} (mm)	A_{max} (mm/s^2)	Max. amplitude (mm/s^2)	Settling time (secs)
p137	767	1.11	566	1.11	849	1387	2132	0.16
p191	648	0.39	169	2.54	1.60	3306	7791	2.02
p192	732	0.92	448	1.49	1.06	1595	1792	0.58
p193	762	1.08	549	1.16	0.86	1411	2000	0.7
p194	793	1.23	650	0.86	0.67	1289	1950	0.38
p195	838	1.43	802	0.45	0.37	1169	1800	0.16
p196	884	1.62	954	0.08	0.07	1093	1815	0.18

Each move has a total move time of 2.92 seconds and a move distance ΔP of 1981 mm

File name	V_{max} (mm/s)	$\Delta t_{accel/decel}$ (secs)	$\Delta p_{accel/decel}$ (mm)	Δt_{slew} (secs)	Δp_{slew} (mm)	A_{max} (mm/s^2)	Max. amplitude (mm/s^2)	Settling time (secs)
p139	873	0.97	566	0.97	849	1793	2215	0.82

Each move has a total move time of 2.73 seconds and a move distance ΔP of 1981 mm

File name	V_{max} (mm/s)	$\Delta t_{accel/decel}$ (secs)	$\Delta p_{accel/decel}$ (mm)	Δt_{slew} (secs)	Δp_{slew} (mm)	A_{max} (mm/s^2)	Max. amplitude (mm/s^2)	Settling time (secs)
p138	932	0.91	566	0.91	849	2045	2324	1.16

Each move has a total move time of 10 seconds and a move distance ΔP of 991 mm

File name	V_{max} (mm/s)	$\Delta t_{accel/decel}$ (secs)	$\Delta p_{accel/decel}$ (mm)	Δt_{slew} (secs)	Δp_{slew} (mm)	A_{max} (mm/s^2)	Max. amplitude (mm/s^2)	Settling time (secs)
p140	127	3.33	283	3.33	425	76	467	0
p198	99.5	0.06	4.3	9.87	982	3093	2459	1.18
p199	99.8	0.11	7.6	9.77	976	1744	2275	1.32
p200	106.7	1.07	76.2	7.86	838	199	332	0
p201	122.0	2.81	228.7	4.38	534	87	327	0
p202	129.6	3.53	304.9	2.94	381	73	303	0
p203	143.3	4.63	442.1	0.74	107	62	323	0
p204	147.9	4.95	487.8	0.10	15	60	382	0

Each move has a total move time of 7.03 seconds and a move distance ΔP of 991 mm

File name	V_{max} (mm/s)	$\Delta t_{accel/decel}$ (secs)	$\Delta p_{accel/decel}$ (mm)	Δt_{slew} (secs)	Δp_{slew} (mm)	A_{max} (mm/s^2)	Max. amplitude (mm/s^2)	Settling time (secs)
p141	181	2.34	283	2.34	425	155	430	0

Each move has a total move time of 5 seconds and a move distance ΔP of 991 mm

File name	V_{max} (mm/s)	$\Delta t_{accel/decel}$ (secs)	$\Delta p_{accel/decel}$ (mm)	Δt_{slew} (secs)	Δp_{slew} (mm)	A_{max} (mm/s^2)	Max. amplitude (mm/s^2)	Settling time (secs)
p142	255	1.67	283	1.67	425	306	443	0
p205	202	0.12	16.8	4.75	957	3229	5089	21.6
p206	206	0.28	38.1	4.44	914	1482	2416	1.14
p207	213	0.54	76.2	3.93	838	797	863	0
p208	244	1.41	228.7	2.19	533	347	422	0
p209	259	1.76	304.9	1.47	381	294	453	0
p210	274	2.08	381.1	0.83	228.7	263	580	0
p211	290	2.37	457.3	0.26	76.2	245	412	0
p212	296	2.47	487.8	0.05	15.2	239	417	0

Each move has a total move time of 3.5 seconds and a move distance ΔP of 991 mm

File name	V_{max} (mm/s)	$\Delta t_{accel/decel}$ (secs)	$\Delta p_{accel/decel}$ (mm)	Δt_{slew} (secs)	Δp_{slew} (mm)	A_{max} (mm/s^2)	Max. amplitude (mm/s^2)	Settling time (secs)
p213	293	0.18	35.7	3.14	919.5	3215	6797	2.72
p214	300	0.29	57.5	2.92	875.8	2079	5123	1.16
p215	335	0.82	182.9	1.86	625.0	820	1348	0
p216	381	1.35	343.0	0.80	304.9	565	493	0
p217	396	1.50	396.3	0.50	198.2	528	447	0
p218	413	1.65	455.0	0.20	80.8	500	503	0
p219	422	1.73	487.0	0.04	16.8	488	581	0

Each move has a total move time of 5 seconds and a move distance ΔP of 991 mm

File name	V_{max} (mm/s)	$\Delta t_{accel/decel}$ (secs)	$\Delta p_{accel/decel}$ (mm)	Δt_{slew} (secs)	Δp_{slew} (mm)	A_{max} (mm/s^2)	Max. amplitude (mm/s^2)	Settling time (secs)
p143	364	1.17	283	1.17	425	624	624	0

Each move has a total move time of 1.66 seconds and a move distance ΔP of 991 mm

File name	V_{max} (mm/s)	$\Delta t_{accel/decel}$ (secs)	$\Delta p_{accel/decel}$ (mm)	Δt_{slew} (secs)	Δp_{slew} (mm)	A_{max} (mm/s^2)	Max. amplitude (mm/s^2)	Settling time (secs)
p220	724	0.44	211	0.79	569	3311	7683	2.26
p221	762	0.54	274	0.58	442	2823	5832	1.66
p222	793	0.62	325	0.43	341	2578	4298	2.56
p223	854	0.75	426	0.16	138	2279	2925	2.06
p224	877	0.79	464	0.07	63	2207	2659	2
p225	884	0.81	477	0.04	37	2186	2526	2.12

Each move has a total move time of 1.46 seconds and a move distance ΔP of 991 mm

File name	V_{max} (mm/s)	$\Delta t_{accel/decel}$ (secs)	$\Delta p_{accel/decel}$ (mm)	Δt_{slew} (secs)	Δp_{slew} (mm)	A_{max} (mm/s^2)	Max. amplitude (mm/s^2)	Settling time (secs)
p144	873	0.49	283	0.49	425	3586	5598	2.94

Each move has a total move time of 5 seconds and a move distance ΔP of 15.2 mm

File name	V_{max} (mm/s)	$\Delta t_{accel/decel}$ (secs)	$\Delta p_{accel/decel}$ (mm)	Δt_{slew} (secs)	Δp_{slew} (mm)	A_{max} (mm/s^2)	Max. amplitude (mm/s^2)	Settling time (secs)
p173	4	1.67	4.36	1.67	6.53	5	165	0
p227	3.08	0.07	0.15	4.85	14.94	83	175	0
p228	3.20	0.36	0.76	4.29	13.72	18	141	0
p229	3.81	1.50	3.81	2.00	7.62	5	173	0
p230	4.27	2.14	6.10	0.71	3.05	4	185	0
p231	4.50	2.42	7.24	0.17	0.76	4	178	0
p232	4.54	2.47	7.47	0.07	0.30	4	144	0

Each move has a total move time of 3 seconds and a move distance ΔP of 15.2 mm

File name	V_{max} (mm/s)	$\Delta t_{accel/decel}$ (secs)	$\Delta p_{accel/decel}$ (mm)	Δt_{slew} (secs)	Δp_{slew} (mm)	A_{max} (mm/s^2)	Max. amplitude (mm/s^2)	Settling time (secs)
p174	7	1.00	4.36	1.00	7	13	152	0
p234	5.5	0.33	1.22	2.33	12.80	33	145	0
p235	6.1	0.75	3.05	1.50	9.15	16	180	0
p236	6.9	1.20	5.56	0.59	4.12	12	153	0
p237	7.2	1.31	6.25	0.38	2.74	11	129	0
p238	7.5	1.43	7.12	0.13	1.01	10	168	0
p239	7.6	1.48	7.48	0.04	0.27	10	166	0

Each move has a total move time of 1 seconds and a move distance ΔP of 15.2 mm

File name	V_{max} (mm/s)	$\Delta t_{accel/decel}$ (secs)	$\Delta p_{accel/decel}$ (mm)	Δt_{slew} (secs)	Δp_{slew} (mm)	A_{max} (mm/s^2)	Max. amplitude (mm/s^2)	Settling time (secs)
p175	20	0.33	4.36	0.33	7	118	218	0
p242	15.9	0.06	0.61	0.88	14.02	550	251	0
p243	19.8	0.35	4.57	0.31	6.10	114	142	0
p244	21.3	0.43	6.10	0.14	3.05	100	140	0
p245	22.4	0.48	7.16	0.04	0.91	93	148	0

Each move has a total move time of 0.78 seconds and a move distance ΔP of 3.9 mm

File name	V_{max} (mm/s)	$\Delta t_{accel/decel}$ (secs)	$\Delta p_{accel/decel}$ (mm)	Δt_{slew} (secs)	Δp_{slew} (mm)	A_{max} (mm/s^2)	Max. amplitude (mm/s^2)	Settling time (secs)
p145	7	0.26	1.13	0.26	1.70	50	173	0

Each move has a total move time of 0.39 seconds and a move distance ΔP of 3.9 mm

File name	V_{max} (mm/s)	$\Delta t_{accel/decel}$ (secs)	$\Delta p_{accel/decel}$ (mm)	Δt_{slew} (secs)	Δp_{slew} (mm)	A_{max} (mm/s^2)	Max. amplitude (mm/s^2)	Settling time (secs)
p146	13	0.13	1.13	0.13	1.70	200	178	0
p257	11	0.06	0.44	0.27	3.09	387	193	0
p258	13	0.13	1.09	0.14	1.78	205	195	0

Each move has a total move time of 0.29 seconds and a move distance ΔP of 3.9 mm

File name	V_{max} (mm/s)	$\Delta t_{accel/decel}$ (secs)	$\Delta p_{accel/decel}$ (mm)	Δt_{slew} (secs)	Δp_{slew} (mm)	A_{max} (mm/s^2)	Max. amplitude (mm/s^2)	Settling time (secs)
p147	17	0.10	1	0.10	2	356	224	0

Each move has a total move time of 0.195 seconds and a move distance ΔP of 3.9 mm

File name	V_{max} (mm/s)	$\Delta t_{accel/decel}$ (secs)	$\Delta p_{accel/decel}$ (mm)	Δt_{slew} (secs)	Δp_{slew} (mm)	A_{max} (mm/s^2)	Max. amplitude (mm/s^2)	Settling time (secs)
p148	26	0.07	1	0.07	2	802	391	0
p253	24	0.05	0.80	0.10	2.36	991	453	0

C.3 S-curve move

C.3.1 Minimum time three part S-curve move

Listed below are the minimum time, three part S-curve moves tested (chapter 4.3.2.2). For details on minimum time moves refer to chapter 3.5.1.

Each move has a total move time of 2.63 seconds and a move distance ΔP of 1981 mm

File name	V_{max} (mm/s)	A_{max} (mm/s^2)	$\Delta t_{accel/decel}$ (secs)	$\Delta p_{accel/decel}$ (mm)	Δt_{slew} (secs)	Δp_{slew} (mm)	Max. amplitude (mm/s^2)	Settling time (secs)
p265	1067	2744	0.39	415	1.08	1152	5590	1.82

Each move has a total move time of 1.71seconds and a move distance ΔP of 991 mm

File name	V_{max} (mm/s)	A_{max} (mm/s^2)	$\Delta t_{accel/decel}$ (secs)	$\Delta p_{accel/decel}$ (mm)	Δt_{slew} (secs)	Δp_{slew} (mm)	Max. amplitude (mm/s^2)	Settling time (secs)
p266	1067	2744	0.39	415	0.15	161	8713	2.12

Each move has a total move time of 1.60seconds and a move distance ΔP of 881 mm

File name	V_{max} (mm/s)	A_{max} (mm/s^2)	$\Delta t_{accel/decel}$ (secs)	$\Delta p_{accel/decel}$ (mm)	Δt_{slew} (secs)	Δp_{slew} (mm)	Max. amplitude (mm/s^2)	Settling time (secs)
p267	1067	2744	0.39	415	0.05	51	1.86	9094

C.3.2 Minimum time single part move

Listed below are the minimum time single part S-curve moves tested (chapter 4.3.3.1). For details on minimum time moves refer to section 3.5.1.

Each move has a total move time of 0.22 seconds and a move distance ΔP of 15.24 mm

File name	V_{max} (mm/s)	A_{max} (mm/s^2)	Δt (secs)	Δp (mm)	Max. amplitude (mm/s^2)	Settling time (secs)
p268	1143	2744	0.07	15.24	5767	1.90

Each move has a total move time of 0.11 seconds and a move distance ΔP of 15.24 mm

File name	V_{max} (mm/s)	A_{max} (mm/s^2)	Δt (secs)	Δp (mm)	Max. amplitude (mm/s^2)	Settling time (secs)
p269	1143	2744	0.04	3.96	675	0.18

Each move has a total move time of 0.06 seconds and a move distance ΔP of 1 mm

File name	V_{max} (mm/s)	A_{max} (mm/s^2)	Δt (secs)	Δp (mm)	Max. amplitude (mm/s^2)	Settling time (secs)
p270	1143	2744	0.02	1	210	0.00

For a move that is too small for a minimum three-3-part and too large for a maximum-1-part, which will occurs when $V_{max} = V_{peak}$ in a Maximum-1-part move, this move will be contained by V_{max} but never reaches A_{max}.

Each move has a total move time of 1.76 seconds and a move distance ΔP of 869 mm

File name	V_{max} (mm/s)	Δt (secs)	Δp (mm)	Max. amplitude (mm/s^2)	Settling time (secs)
p271	1113	0.59	869	8910	2.62

C.3.3 Three part S-curve move

Listed below are the three part S-curve moves tested (Chapter 4.3.3.2).

Each move has a total move time of 10 seconds and a move distance ΔP of 1981 mm

File name	V_{max} (mm/s)	$\Delta t_{accel/decel}$ (secs)	$\Delta p_{accel/decel}$ (mm)	Δt_{slew} (secs)	Δp_{slew} (mm)	A_{max} (mm/s^2)	Max. amplitude (mm/s^2)	Settling time (secs)
p272	201	0.07	14	9.73	1954	2939	4955	2.80
p273	202	0.10	20	9.61	1942	2070	3735	2.18
p274	211	0.29	62	8.83	1858	719	816	0.00
p275	220	0.49	107	8.05	1767	450	610	0.00
p276	325	1.95	635	2.19	711	167	429	0.00
p277	387	2.44	945	0.23	91	159	494	0.00

Each move has a total move time of 5 seconds and a move distance ΔP of 1981 mm

File name	V_{max} (mm/s)	$\Delta t_{accel/decel}$ (secs)	$\Delta p_{accel/decel}$ (mm)	Δt_{slew} (secs)	Δp_{slew} (mm)	A_{max} (mm/s^2)	Max. amplitude (mm/s^2)	Settling time (secs)
p278	421	0.15	62	4.41	1858	2874	8548	1.92
p279	449	0.29	132	3.83	1719	1532	1533	0.00
p280	493	0.49	240	3.05	1501	1009	732	0.00
p281	576	0.78	450	1.88	1081	738	760	0.00

Each move has a total move time of 3.32 seconds and a move distance ΔP of 1981 mm

File name	V_{max} (mm/s)	$\Delta t_{accel/decel}$ (secs)	$\Delta p_{accel/decel}$ (mm)	Δt_{slew} (secs)	Δp_{slew} (mm)	A_{max} (mm/s^2)	Max. amplitude (mm/s^2)	Settling time (secs)
p283	690	0.22	155	2.42	1672	3073	6583	1.06
p284	725	0.29	212	2.15	1557	2474	3516	1.10
p285	846	0.49	413	1.37	1156	1732	857	0.00
p286	1128	0.78	881	0.20	220	1443	7844	0.70

Each move has a total move time of 10 seconds and a move distance ΔP of 991 mm

File name	V_{max} (mm/s)	$\Delta t_{accel/decel}$ (secs)	$\Delta p_{accel/decel}$ (mm)	Δt_{slew} (secs)	Δp_{slew} (mm)	A_{max} (mm/s^2)	Max. amplitude (mm/s^2)	Settling time (secs)
p287	100	0.03	3.41	9.86	984	2919	2096	0.96
p288	100	0.05	4.89	9.80	981	2049	2019	0.86
p289	101	0.10	9.87	9.61	971	1035	2210	1.22
p290	105	0.29	30.84	8.83	929	359	361	0.00
p291	115	0.68	78.46	7.27	834	168	260	0.00
p292	140	1.46	205.29	4.14	580	96	156	0.00
p293	197	2.49	489.65	0.06	12	79	426	0.00

Each move has a total move time of 5 seconds and a move distance ΔP of 991 mm

File name	V_{max} (mm/s)	$\Delta t_{accel/decel}$ (secs)	$\Delta p_{accel/decel}$ (mm)	Δt_{slew} (secs)	Δp_{slew} (mm)	A_{max} (mm/s^2)	Max. amplitude (mm/s^2)	Settling time (secs)
p294	204	0.07	14	4.73	963	2980	5178	1.66
p295	206	0.10	20	4.61	951	2112	4393	1.98
p296	246	0.49	120	3.05	750	504	441	0.00
p297	393	1.24	488	0.04	15	317	446	0.00

Each move has a total move time of 3.52 seconds and a move distance ΔP of 991 mm

File name	V_{max} (mm/s)	$\Delta t_{accel/decel}$ (secs)	$\Delta p_{accel/decel}$ (mm)	Δt_{slew} (secs)	Δp_{slew} (mm)	A_{max} (mm/s^2)	Max. amplitude (mm/s^2)	Settling time (secs)
p298	301	0.10	31	3.09	929	2933	6439	1.48
p299	309	0.15	45	2.91	900	2109	6350	2.40
p300	364	0.39	142	1.94	706	933	938	0.00
p301	559	0.86	483	0.04	24	647	539	0.00

Each move has a total move time of 1.66 seconds and a move distance ΔP of 991 mm

File name	V_{max} (mm/s)	$\Delta t_{accel/decel}$ (secs)	$\Delta p_{accel/decel}$ (mm)	Δt_{slew} (secs)	Δp_{slew} (mm)	A_{max} (mm/s^2)	Max. amplitude (mm/s^2)	Settling time (secs)
p303	1015	0.34	347	0.29	297	2969	9017	2.18

Each move has a total move time of 5 seconds and a move distance ΔP of 15.2 mm

File name	V_{max} (mm/s)	$\Delta t_{accel/decel}$ (secs)	$\Delta p_{accel/decel}$ (mm)	Δt_{slew} (secs)	Δp_{slew} (mm)	A_{max} (mm/s^2)	Max. amplitude (mm/s^2)	Settling time (secs)
p304	209	0.03	7.13	4.86	0.99	6100	157	0
p305	76	0.10	7.45	4.61	0.34	781	149	0
p306	6	1.23	7.62	0.10	0.00	5	148	0

Each move has a total move time of 3 seconds and a move distance ΔP of 15.2 mm

File name	V_{max} (mm/s)	$\Delta t_{accel/decel}$ (secs)	$\Delta p_{accel/decel}$ (mm)	Δt_{slew} (secs)	Δp_{slew} (mm)	A_{max} (mm/s^2)	Max. amplitude (mm/s^2)	Settling time (secs)
p307	248	0.03	7	2.88	0.70	8473	171	0
p308	77	0.10	8	2.61	0.20	789	154	0
p309	10	0.74	8	0.05	0.00	14	202	0

Each move has a total move time of 1 seconds and a move distance ΔP of 15.2 mm

File name	V_{max} (mm/s)	$\Delta t_{accel/decel}$ (secs)	$\Delta p_{accel/decel}$ (mm)	Δt_{slew} (secs)	Δp_{slew} (mm)	A_{max} (mm/s^2)	Max. amplitude (mm/s^2)	Settling time (secs)
p311	78	0.10	7.60	0.61	0	797	230	0
p312	33	0.23	7.62	0.06	0	139	210	0

Each move has a total move time of 0.39 seconds and a move distance ΔP of 3.96 mm

File name	V_{max} (mm/s)	$\Delta t_{accel/decel}$ (secs)	$\Delta p_{accel/decel}$ (mm)	Δt_{slew} (secs)	Δp_{slew} (mm)	A_{max} (mm/s^2)	Max. amplitude (mm/s^2)	Settling time (secs)
p314	29	0.07	1.98	0.12	0.00	424	272	0
p315	21	0.09	1.98	0.02	0.00	230	259	0

Each move has a total move time of 0.195 seconds and a move distance ΔP of 3.96 mm

File name	V_{max} (mm/s)	$\Delta t_{accel/decel}$ (secs)	$\Delta p_{accel/decel}$ (mm)	Δt_{slew} (secs)	Δp_{slew} (mm)	A_{max} (mm/s^2)	Max. amplitude (mm/s^2)	Settling time (secs)
p316	68	0.03	1.98	0.08	0.01	2306	410	0.44

C.3.4 Single part S-curve

Listed below are the single part S-curve moves tested (chapter 4.3.3.1).

Programming the S-curve move

```
CIR4        ; Put in parabolic S-curve mode
t(xx)       ; Move segment takes (xx) binary milliseconds
U(yy)       ; where (yy) = Total move increment ΔP. This move starts
END         ; and end with zero velocity.
```

Each move has a move distance ΔP of 1982 mm

File name	t (secs)	V_{max} (mm/s)	A_{max} (mm/s^2)	Δt (secs)	Max. amplitude (mm/s^2)	Settling time (secs)
p319	1	1121	1128	3.98	7153	0.4
p320	1.47	1010	915	4.42	988	0
p321	1.66	894	716	4.99	829	0
p322	3.36	442	175	10.09	551	0

Each move has a move distance ΔP of 991 mm

File name	t (secs)	V_{max} (mm/s)	A_{max} (mm/s^2)	Δt (secs)	Max. amplitude (mm/s^2)	Settling time (secs)
p323	0.66	1129	2287	1.97	8599	2.1
p324	0.81	922	1524	2.42	1082	0
p325	1.18	632	716	3.53	766	0
p326	1.80	412	305	5.41	691	0
p327	2.35	317	180	7.04	454	0

Each move has a move distance ΔP of 15.24 mm

File name	t (secs)	V_{max} (mm/s)	A_{max} (mm/s^2)	Δt (secs)	Max. amplitude (mm/s^2)	Settling time (secs)
p328	0.07	162	3049	0.21	6003	1.78
p329	0.12	96	1067	0.36	2763	1.22
p330	0.18	63	457	0.55	904	0.58
p331	0.35	32	122	1.06	181	0
p332	1.00	11	15	3.00	195	0
p333	1.67	7	5	5.00	161	0

Each move has a move distance ΔP of 3.96 mm

File name	t (secs)	V_{max} (mm/s)	A_{max} (mm/s^2)	Δt (secs)	Max. amplitude (mm/s^2)	Settling time (secs)
p335	0.04	82	3049	0.11	795	0.18
p336	0.04	71	2287	0.12	614	0
p337	0.05	58	1524	0.15	512	0
p338	0.07	46	930	0.20	404	0.04
p339	0.10	30	396	0.30	450	0
p340	0.13	23	229	0.39	257	0.02
p341	0.26	11	59	0.78	154	0

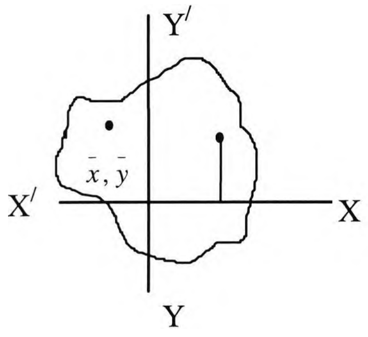
Each move has a move distance ΔP of 0.99 mm

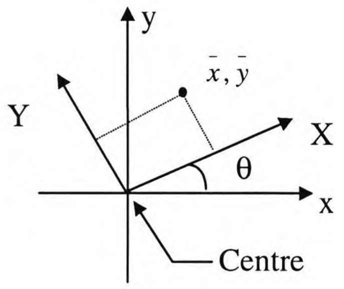
File name	t (secs)	V_{max} (mm/s)	A_{max} (mm/s^2)	Δt (secs)	Max. amplitude (mm/s^2)	Settling time (secs)
p343	0.02	46	3811	0.05	168	0
p344	0.02	38	2591	0.06	225	0

Appendix D: Target Location

D.1 Theory of moments

This section describes the method of object moments (Gonzalez, 1987). It was utilised in this research for the purpose of determining a boundary description for objects in a specified region (chapter 5.3.1). This technique made it possible to quantify the shape of the object for recognition. Generally, only the first few moments are required to differentiate between signatures of clearly distinct shapes.





The first step is to find $\bar{x}, \bar{y}$

$$\bar{x} = \sum x' / N, \qquad \text{eqn d.1}$$

$$\bar{y} = \sum y' / N \qquad \text{eqn d.2}$$

, where $\bar{x}, \bar{y}$ are the centroids of the object $i = 1 \ldots N$

The second step is to find the shape and direction of an object. The second moments, described below, quantify the spread of the data points about the mean value (i.e. the centroid).

$$\mathrm{I}x'x' = \sum\left[(y_i - \bar{y})^2\right] \qquad \text{eqn d.3}$$

$$\mathrm{I}y'y' = \sum\left[(x_i - \bar{x})^2\right] \qquad \text{eqn d.4}$$

$$\mathrm{I}x'y' = \sum\left[(y_i - \bar{y})(x_i - \bar{x})\right] \qquad \text{eqn d.5}$$

, where X,Y are the principle axis and θ the principle direction.

Giving Ixx MAX

and Iyy MIN

and Ixy=0

$X= x\text{Cos}\theta + y\text{Sin}\theta$ eqn d.6

$Y= -x\text{Sin}\theta + y\text{Cos}\theta$ eqn d.7

$I_pxx = \sum y^2 = \sum(x^2\text{Sin}^2\theta - 2xy\text{Sin}\theta\text{Cos}\theta + y^2\text{Cos}^2\theta)$ eqn d.8

$I_pxx = IyySin^2\theta - 2IxySin\theta Cos\theta + IxxCos^2\theta$ eqn d.9

$I_pxx = IyySin^2\theta - IxySin2\theta + IxxCos^2\theta$ eqn d.10

$dIxx/d\theta = 0$ for Max/Min

$0 = Iyy2\sin\theta Cos\theta - 2Ixy\cos 2\theta - 2Ixx\cos\theta\sin\theta$ eqn d.11

$0 = Iyy\sin 2\theta - 2Ixy\cos 2\theta - Ixx\sin 2\theta$ eqn d.12

from equation d.12 we get,

$\tan 2\theta = [2Ixy/(Iyy - Ixx)]$ eqn d.13

$\theta = (1/2)\tan^{-1}[2Ixy/(Iyy - Ixx)]$ eqn d.14

$I_pxy = \sum XY$

$I_pxy = \sum(-x^2\sin\theta\cos\theta + xy\cos^2\theta - xy\sin^2\theta + y^2\sin\theta\cos\theta)$

$I_pxy = -(Iyy/2)\sin 2\theta + (Ixy/2)(\cos 2\theta + 1) + (Ixy/2)(\cos 2\theta - 1) + (Ixx/2)\sin 2\theta$

$I_pxy = (-(Iyy - Ixx/2)\sin 2\theta + Ixy\cos 2\theta) = 0$ eqn d.15

The radious of gyration is calculated as follows,

$ry = \sqrt{I_pxx/N}$ eqn d.16

$rx = \sqrt{I_pyy/N}$ eqn d.17

, where N is the number of data points to be processed.

D.2 Target location experiments results

D.2.1 Target scanning co-ordinate file (stage 1)

Table D.1 shows an example of a co-ordinate in file '*scan_results*', captured while conducting a scanning detection of the conveyor target using the C program '*targ_fun.mak*'. This is considered as part of stage 1 of the target detection, described in chapter 5.3.1.2.2. From these data points, the target centroid and angle are estimated using the method of moments. This can be seen clearly in the experiments covered in chapter 5.3.1.2.2.1.

Table D.1 *Data file of stage1 target scanning*

Data point	X-axis co-ordinates (counts)	Y-axis co-ordinates (counts)	V_sens reading (mm)
1	135318	29788	392
2	137435	38299	394
3	139646	38299	389
4	137231	55321	393
5	139375	55321	388
6	141686	55321	396
7	137523	63832	391
8	135613	63832	389

D.2.2 Target allocation data file

The results below show an example of a data file, captured when conducting experiments to determine the accuracy of the complete target location procedure (i.e. stages 1, 2 and 3 of the target location process). The data file is generated as a result of running the C program '*Find_target.c*', and the C function '*correct_ang()*', both part of the 'C' program '*targ_fun.mak*', described in section 5.3.2. The data file shows the a listing of the actions taken by the robot, the sensor readings, and the calculations of results, in the order they occurred during the process of locating he target. The results of the experiments carried out to test the target location programmes are covered in chapter 5.4.

The data file is as follows:

```
Xcentre         2648.12 ,       Ycentre         1542.05
Ixx             3557599.36,     Iyy             846313.81
Ixy             1060872.50,     Ipxx            3923356.54
Ipyy            480556.63,      Ipxy            -0.05
rx              269.54,         ry              94.34
Estimated target angle = -19.02°
Angle is -ve and Ipyy < Ipxx result target angle 70.97°
X-move          2905.3,         Y-move          1542.1
Estimated Yaw Angle -19.02
Correct the YAW estimated angle
start search at TCP :
X 2527.18 mm  Y 1672.43 mm  Z-350.0 mm YAW-19.02°
H_sens out of range   642     move Yaw(-5°) & forward 15 mm
H_sens out of range   642     move Yaw(+5°) & forward 15 mm
move forward  -75 mm
Yaw Angle now is      -19.02°,        move forward  1 mm
Yaw Angle now is      -19.02°,        move forward  2 mm
Yaw Angle now is      -19.02°,        move forward  -1 mm
Yaw Angle now is      -19.02°,        move forward  -1 mm
Yaw Angle now is      -19.02°
Find the edge of the target
move side +30 mm      H_sens 356
move side +30 mm      H_sens 642
move side -10 mm      H_sens 642
move side -10 mm      H_sens 406
move side -300 mm     H_sens 406
Correct angle again
move Yaw(-13°) H_sens out of range            642
Yaw     -32.02°
H_sens still out of range move Yaw(+26°)      642     yaw     -6.02°
move Yaw(-1.5°) H_sens is   432                       yaw     -7.52°
move Yaw(-1.5°) H_sens is   433                       yaw     -9.02°
move Yaw(-1.5°) H_sens is   433                       yaw     -10.52°
move Yaw(-1.5°) H_sens is   434                       yaw     -12.02°
move Yaw(+12°)              yaw_ang is        -0.02°
H_sens  435     V_sens  430
Yaw angle correction  0.72°
corrected angle is        0.69°
H_sens 431      V_sens 430
Yaw angle correction  0.14°
Corrected angle is        0.84°
first reading from edge, at X  2639.64        Y         1640.97
TCP at X        2669.61  Y     1301.45
```

Appendix E: Block Manipulation Experiments

E.1 Results of block manipulation experiments

Results of the six experiments are covered below. Experiments 1-3 are of a block picked horizontally, and experiments 4-6 of a block picked at an inclination (chapter 7.4.1).

Experiment number 1

Table E.1 *Result of six scans from a laser profile of a block picked horizontally*

Scan no.	edge1 at (mm)	edge2 at (mm)	edge3 at (mm)	edge4 at (mm)	edge5 at (mm)	edge6 at (mm)	edge7 at (mm)	edge8 at (mm)
Scan1	123.6	168.7	179.2	212.5	290.3	322.5	331.8	371.2
Scan2	122.2	169.2	178.5	211.1	290.3	322.5	332.7	371.9
Scan3	120.9	166.4	175.2	208.3	288.4	319.7	328.5	369.4
Scan4	121.6	169.2	178.9	210.2	291.3	322.0	331.4	371.2
Scan5	123.9	170.6	180.3	213.0	292.2	323.4	332.5	371.8
Scan6	124.0	169.6	179.9	212.5	291.7	321.1	331.9	371.3

Scan no.	A2 (mm)	A (mm)	B (mm)	B2 (mm)	C (mm)	C2 (mm)	D (mm)
Scan1	45.1	39.4	10.3	10.7	33.3	32.2	77.8
Scan2	47.0	39.2	10.7	11.2	32.6	32.2	79.2
Scan3	45.5	40.9	10.3	9.8	33.1	31.3	80.1
Scan4	47.6	39.8	9.8	12.1	31.3	30.7	81.1
Scan5	46.7	39.3	10.7	10.7	32.7	31.2	79.2
Scan6	45.6	39.4	10.3	9.3	32.6	29.4	79.2
Average	**46.3**	**39.7**	**10.3**	**10.6**	**32.6**	**31.2**	**79.4**

Experiment number 2

Table E.2 *Result of six scans from a laser profile of a block picked horizontally*

Scan no.	edge1 at (mm)	edge2 at (mm)	edge3 at (mm)	edge4 at (mm)	edge5 at (mm)	edge6 at (mm)	edge7 at (mm)	edge8 at (mm)
Scan1	139.3	166.4	176.2	208.8	289.4	320.6	329.6	389.1
Scan2	138.9	165.4	176.2	208.8	288.9	318.3	328.7	386.8
Scan3	140.3	167.8	178.3	211.1	289.9	320.6	328.9	389.6
Scan4	140.3	168.3	176.6	208.8	289.4	320.1	328.5	389.1
Scan5	140.3	167.8	177.1	210.2	289.9	319.7	328.5	387.7
Scan6	140.7	168.2	177.1	210.2	290.3	320.0	329.6	388.2

Scan no.	A2 (mm)	A (mm)	B (mm)	B2 (mm)	C (mm)	C2 (mm)	D (mm)
Scan1	27.1	59.5	9.8	9.0	32.6	31.2	80.6
Scan2	26.6	58.1	10.7	10.4	32.6	29.4	80.1
Scan3	27.5	60.7	10.5	8.3	32.8	30.7	78.8
Scan4	28.0	60.6	8.3	8.4	32.2	30.7	80.6
Scan5	27.5	59.2	9.3	8.8	33.1	29.8	79.7
Scan6	27.5	58.6	8.9	9.6	33.1	29.7	80.1
Average	**27.4**	**59.4**	**9.6**	**9.1**	**32.7**	**30.3**	**80.0**

Experiment number 3

Table E.3 *Result of six scans from a laser profile of a block picked horizontally*

Scan no.	edge1 at (mm)	edge2 at (mm)	edge3 at (mm)	edge4 at (mm)	edge5 at (mm)	edge6 at (mm)	edge7 at (mm)	edge8 at (mm)
Scan1	124.4	165.2	175.7	210.2	287.1	318.3	328.5	373.3
Scan2	124.4	165.4	175.6	210.6	288.5	319.8	329.4	372.3
Scan3	123.4	165.9	176.2	210.6	287.5	320.1	329.5	373.3
Scan4	124.0	166.9	175.7	209.2	288.9	319.2	328.1	372.3
Scan5	123.4	166.8	177.0	210.6	288.0	319.1	329.3	373.3
Scan6	123.5	165.9	177.6	210.6	288.0	318.7	329.0	372.4

Scan no.	A2 (mm)	A (mm)	B (mm)	B2 (mm)	C (mm)	C2 (mm)	D (mm)
Scan1	40.8	44.7	10.5	10.2	34.5	31.2	76.9
Scan2	41.0	42.9	10.2	9.6	35.0	31.3	77.9
Scan3	42.5	43.8	10.3	9.3	34.4	32.6	76.9
Scan4	42.9	44.3	8.8	8.9	33.5	30.3	79.7
Scan5	43.4	43.9	10.1	10.2	33.6	31.1	77.4
Scan6	42.4	43.4	11.7	10.3	33.0	30.7	77.4
Average	**42.2**	**43.8**	**10.3**	**9.7**	**34.0**	**31.2**	**77.7**

Experiment number 4

Table E.4 *Result of six scans from a laser profile of a block picked at an angle*

Scan no.	edge1 at (mm)	edge2 at (mm)	edge3 at (mm)	edge4 at (mm)	edge5 at (mm)	edge6 at (mm)	edge7 at (mm)	edge8 at (mm)
Scan1	132.3	173.4	183.1	221.4	293.1	328.5	338.8	381.7
Scan2	131.4	173.4	182.7	220.4	292.2	329.0	334.6	380.7
Scan3	132.3	174.8	183.6	220.9	292.2	328.1	338.2	380.2
Scan4	131.4	174.8	183.1	219.5	292.2	328.1	336.0	380.7
Scan5	131.2	173.4	183.1	220.9	292.2	327.6	336.0	381.7
Scan6	131.4	173.8	183.1	219.5	292.2	328.1	336.0	380.3

Scan no.	A2 (mm)	A (mm)	B (mm)	B2 (mm)	C (mm)	C2 (mm)	D (mm)
Scan1	41.0	42.9	9.8	10.3	38.3	35.4	71.7
Scan2	41.9	46.1	9.3	5.6	37.7	36.8	71.8
Scan3	42.5	42.0	8.8	10.1	37.3	35.9	71.3
Scan4	43.4	44.7	8.3	7.9	36.4	35.9	72.7
Scan5	42.2	45.7	9.8	8.4	37.8	35.4	71.3
Scan6	42.4	44.3	9.3	7.9	36.4	35.9	72.7
Average	**42.2**	**44.3**	**9.2**	**8.4**	**37.3**	**35.9**	**71.9**

Experiment number 5

Table E.5 *Result of six scans from a laser profile of a block picked at an angle*

Scan no.	edge1 at (mm)	edge2 at (mm)	edge3 at (mm)	edge4 at (mm)	edge5 at (mm)	edge6 at (mm)	edge7 at (mm)	edge8 at (mm)
Scan1	118.8	167.3	177.6	210.6	291.7	322.8	332.8	369.1
Scan2	117.9	167.7	178.5	210.6	292.2	321.5	330.4	368.6
Scan3	118.4	168.2	178.5	210.2	291.3	323.4	331.8	367.7
Scan4	118.2	168.9	177.1	210.6	292.7	323.4	332.3	367.7
Scan5	119.3	168.2	178.5	212.5	292.2	322.7	332.7	369.5
Scan6	119.3	168.2	178.5	210.6	291.7	323.9	332.4	368.1

Scan no.	A2 (mm)	A (mm)	B (mm)	B2 (mm)	C (mm)	C2 (mm)	D (mm)
Scan1	48.4	36.3	10.3	10.0	33.0	31.1	81.1
Scan2	49.8	38.2	10.8	8.8	32.1	29.3	81.6
Scan3	49.9	35.9	10.3	8.4	31.7	32.1	81.1
Scan4	50.7	35.4	8.2	8.9	33.5	30.7	82.1
Scan5	48.9	36.8	10.2	10.0	34.0	30.5	79.7
Scan6	48.9	35.8	10.3	8.5	32.1	32.2	81.1
Average	**49.4**	**36.4**	**10.0**	**9.1**	**32.7**	**31.0**	**81.1**

Experiment number 6

Table E.6 *Result of six scans from a laser profile of a block picked at an angle*

Scan no.	edge1 at (mm)	edge2 at (mm)	edge3 at (mm)	edge4 at (mm)	edge5 at (mm)	edge6 at (mm)	edge7 at (mm)	edge8 at (mm)
Scan1	134.7	191.5	201.8	225.1	323.9	342.5	353.2	383.1
Scan2	133.3	191.5	202.2	221.4	323.9	342.5	353.7	382.1
Scan3	134.2	192.0	202.2	223.7	325.3	343.9	353.7	383.5
Scan4	131.9	191.5	201.3	221.4	323.9	341.6	353.7	383.1
Scan5	134.2	190.6	201.3	222.3	324.3	342.5	353.2	382.6
Scan6	133.3	191.5	201.8	221.4	324.3	343.0	352.3	383.1

Scan no.	A2 (mm)	A (mm)	B (mm)	B2 (mm)	C (mm)	C2 (mm)	D (mm)
Scan1	56.9	29.8	10.3	10.7	23.3	18.6	98.8
Scan2	58.3	28.4	10.7	11.2	19.2	18.6	102.5
Scan3	57.8	29.8	10.3	9.8	21.5	18.6	101.6
Scan4	59.7	29.4	9.8	12.1	20.1	17.7	102.5
Scan5	56.4	29.4	10.7	10.7	21.0	18.2	102.0
Scan6	58.3	30.8	10.3	9.3	19.6	18.7	102.9
Average	**57.9**	**29.6**	**10.3**	**10.6**	**20.8**	**18.4**	**101.7**

E.2 Results of block placing experiment

E.2.1 Results of experiments using calibration block

Listed below are the results of experiments carried out using the calibration block (chapter 7.4.2.1 and table 7.7).

Experiment number 1

Scan no.	A (mm)	A_2 (mm)
1	43.338	45.668
2	44.27	45.202
3	43.804	45.668
4	43.804	45.668
5	44.27	46.134
6	42.872	45.668
Average	43.73	45.668

Experiment number 2

Scan no.	A (mm)	A_2 (mm)
1	37.28	48.464
2	38.212	48.93
3	36.814	48.464
4	37.746	48.93
5	37.28	48.93
Average	37.47	48.74

Experiment number 3

Scan no.	A (mm)	A_2 (mm)
1	47.07	42.41
2	47.53	43.34
3	46.13	41.94
4	47.07	42.87
5	47.53	42.41
Average	47.07	42.59

Experiment number 4

Scan no.	A (mm)	A_2 (mm)
1	47.06	38.14
2	47.53	36.59
3	47.99	37.61
4	50.33	38.08
5	47.99	37.61
Average	48.18	37.61

Experiment number 5

Scan no.	A (mm)	A_2 (mm)
1	43.80	45.67
2	42.41	45.67
3	45.20	45.67
4	43.80	47.07
5	44.27	46.13
Average	43.9	46.04

Experiment number 6

Scan no.	A (mm)	A_2 (mm)
1	52.66	36.35
2	54.99	38.68
3	52.66	37.75
4	54.06	38.21
5	52.66	38.68
Average	53.4	37.93

Experiment number 7

Scan no.	A (mm)	A_2 (mm)
1	29.82	61.51
2	30.76	61.05
3	29.82	60.20
4	32.15	60.58
5	28.89	61.05
Average	30.29	60.20

Experiment number 8

Scan no.	A (mm)	A_2 (mm)
1	37.28	50.40
2	40.08	49.79
3	37.28	50.15
4	39.14	50.26
5	37.28	49.24
Average	38.21	49.97

Experiment number 9

Scan no.	A (mm)	A_2 (mm)
1	42.41	46.13
2	45.67	48.00
3	41.94	46.60
4	44.74	47.46
5	43.80	46.13
Average	43.71	46.07

Experiment number 10

Scan no.	A (mm)	A_2 (mm)
1	58.3	30.3
2	58.3	29.4
3	56.9	30.8
4	59.2	28.4
5	57.3	29.4
Average	57.97	29.64

E.2.2 Tests using 'Celcon' blocks

Listed below are the results of experiments carried out using a 'Celcon' block (chapter 7.4.2.2).

Experiment number 1

Scan no.	A (mm)	A_2 (mm)
1	32.62	57.32
2	34.48	58.25
3	32.62	57.32
4	35.42	56.85
5	32.62	57.78
Average	33.55	57.50

Experiment number 2

Scan no.	A (mm)	A_2 (mm)
1	31.69	57.32
2	33.55	57.78
3	31.22	57.78
4	33.09	59.18
5	34.48	57.32
Average	32.81	57.88

Experiment number 3

Scan no.	A (mm)	A_2 (mm)
1	59.65	28.43
2	61.51	28.89
3	60.58	29.82
4	62.44	27.96
5	60.11	28.89
Average	60.86	28.80

Experiment number 4

Scan no.	A (mm)	A_2 (mm)
1	34.95	54.06
2	37.75	54.52
3	34.95	53.12
4	36.35	54.06
5	34.48	54.06
Average	35.70	53.96

Appendix F: Mortar Dispensing Experiments

F.1 Results of Tilcon mix dispensing experiments

Figure F.1 and table F.1 show the results of experiments carried out (chapter 8.3.2) to investigate the pump and nozzle effects on the dispensing operation using, Tilcon mix.

Figure F.1 *Typical bead geometry for increasing offsets of 5 mm, 10 mm, 15 mm to 20 mm (top to bottom) using Tilcon mix. Dispensing at nozzle angel 45°; pump setting = 35 rpm; conveyor belt speed = 90 mm/s*

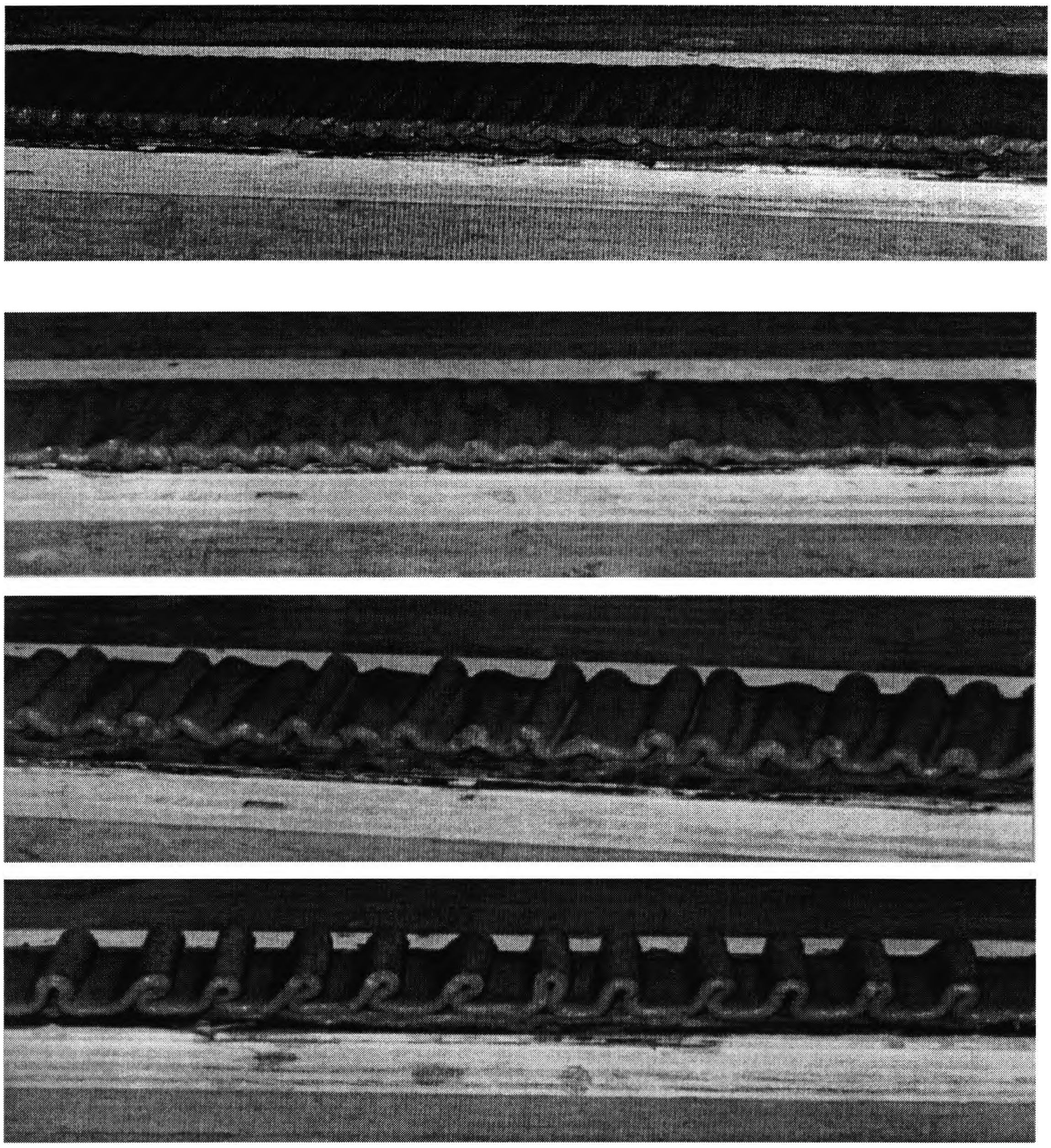
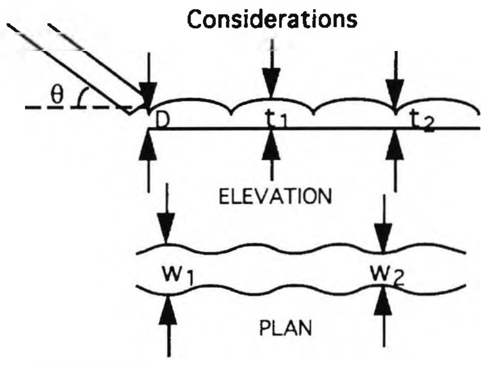


Table F.1 *Results of experiments using the recommended Tilcon mix*



Tilcon Mortar (mix no.9)
Pump setting = 30 revs/min
(0.037 lit/s)

: Conveyor belt speed = 90 mm/s

θ	D	No. of	t_1	t_2	W_1	W_2	Amplitude of ripples	
	(mm)	Ripples	(mm)	(mm)	(mm)	(mm)	T_a	W_a
15	3	11	10, 9, 9	8, 8, 8	72,72,71	65,66,65	1	7
	5	13	10,10,10	7, 7, 7	69,69,70	66,66,67	3	3
	10	14	12,12,12	5, 6, 5	68,68,69	63,63,64	7	5
	15	16	13,13,14	6, 7, 7	68,67,67	64,63,64	6	3
	20	16	17,17,17	6, 6, 6	64,64,65	59,60,60	11	4
30	3	15	9, 9, 9	8, 8, 8	69,69,69	65,65,65	1	4
	5	16	10,10,10	7, 7, 7	70,70,71	65,64,65	3	5
	10	16	10,11,11	7, 8, 8	70,70,69	68,68,68	3	2
	15	16	12,12,12	5, 5, 6	68,68,68	65,65,65	7	3
	20	16	13,13,13	5, 5, 6	66,66,67	64,64,65	8	2
45	3	11	9, 9, 9	8, 8, 9	76,76,77	69,69,69	1	7
	5	16	10,10, 9	9, 9, 9	75,75,74	70,70,70	1	5
	10	18	9,10,10	7, 8, 8	73,72,73	68,67,68	2	5
	15	18	11,11,11	7, 7, 7	71,71,70	67,67,67	4	4
	20	17	11,12,12	8, 8, 8	67,67,67	65,65,65	4	2
60	3	18	6, 6, 6	4, 4, 4	76,75,74	69,68,69	2	6
	5	18	7,7, 7	5, 5, 5	70,71,72	68,68,68	2	3
	10	18	8, 8, 8	7, 7, 7	70,70,71	66,66,67	1	4
	15	18	10,10,10	8, 8, 8	70,69,70	65,64,65	2	5
	20	18	12,12,12	7, 8, 7	70,69,69	68,67,67	5	2
75	3	18	10, 9, 9	10, 9, 8	80,81,83	67,69,70	–	12
	5	18	11,11,10	7, 7, 7	85,84,84	80,80,80	4	1
	10	18	9,10,11	8,9,9	75,74,74	70,70,70	1	4
	15	18	12,12,13	8,8,8	70,72,72	65,65,65	4	7
	20	17	15,15,15	10,10,10	72,71,71	68,67,67	5	4
90	3	18	7,10,6	6,9,6	85,85,84	87,87,87	1	2
	5	17	10,8,11	9,7,10	78,78,78	73,73,74	1	5
	10	18	8,10,9	9,9,8	78,77,77	74,74,74	1	3
	15	17	13,13,13	11,11,11	72,72,72	68,68,68	2	4
	20	18	20,18,17,	8,7,10	68,68,68	62,63,62	9	6

F.2 Results of Pozament mix dispensing experiments

Figure F.2 and table F.2 show the results of experiments carried out (chapter 8.3.2) to investigate the pump and nozzle effects on the dispensing operation using, Pozament mix.

Figure F.2 *Typical Pozament mortar bead geometry for increasing offsets of 10 mm, 15 mm to 20 mm (top to bottom). Dispensing at nozzle angel 30°; pump setting = 40 rpm; conveyor belt speed = 45 mm/s.*

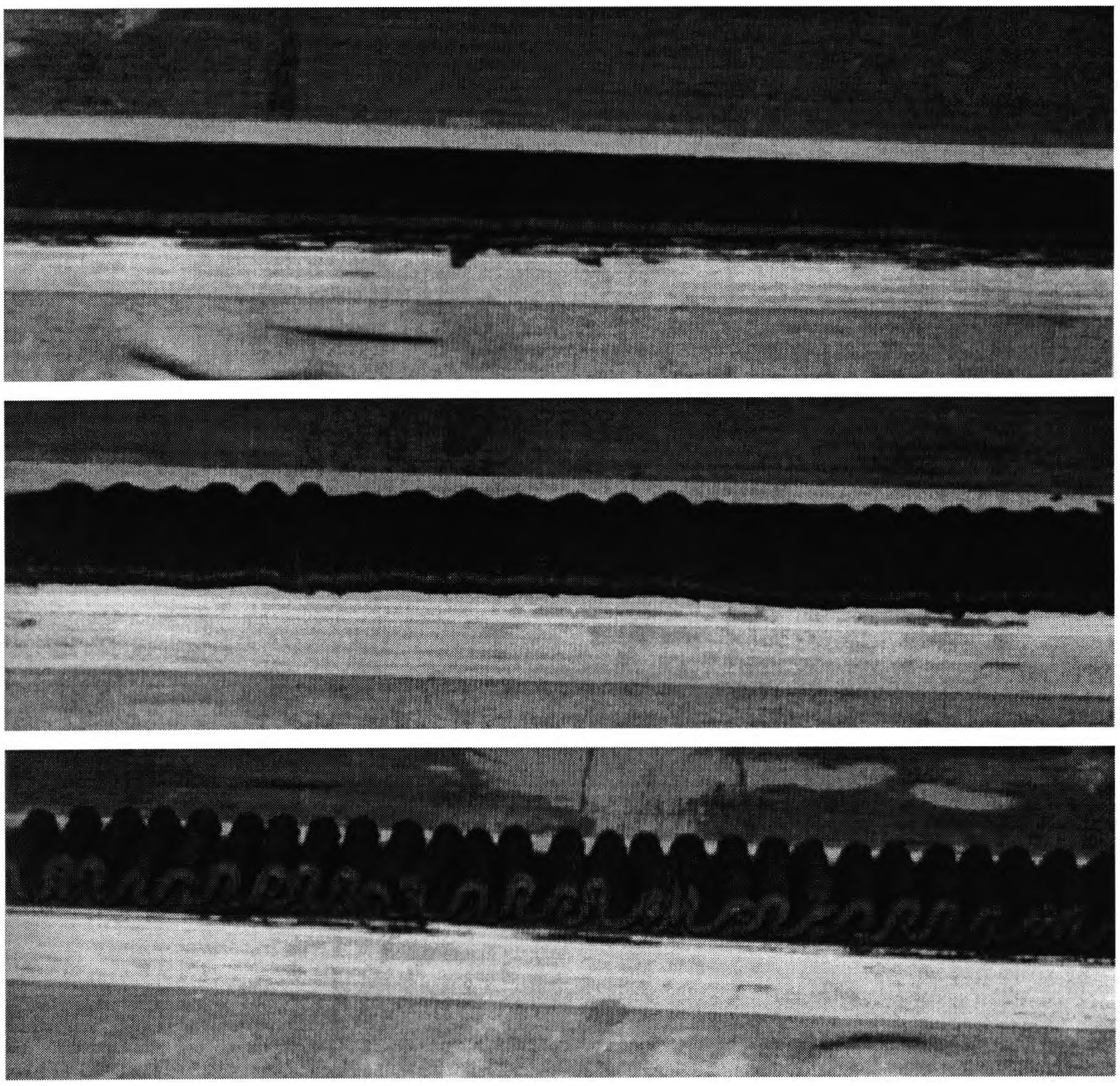
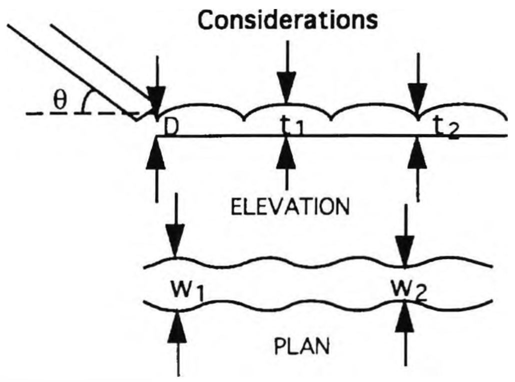


Table F.2 *Results of experiments using the recommended Pozament mix*



Pozament Mortar (mix no.1)
Pump setting = 40 revs/min
(0.0311 lit/s)

Conveyor belt speed = 45 mm/s

θ	D	No. of	t1	t2	w1	w2	Amplitudes	
	(mm)	Ripples	(mm)	(mm)	(mm)	(mm)	T	W
15	3		10,10,10	-	74,73,73	-	-	-
	5		10,10,12	-	73,73,73	-	-	-
	10		11,11,12	-	72,72,72	-	-	-
	15		15,14,15	11,11,10	71,71,71	-	4	-
	20		18,17,18	8,9,9	64,65,65	-	9	-
30	3		10,10,10	-	70,70,71	-	-	-
	5		10,11,12	-	71,70,71	-	-	-
	10		10,11,10	-	72,72,71	-	-	-
	15		13,12,13	11,10,12	71,71,71	69,69,69	2	2
	20		18,18,18	8, 9, 8	65,65,66	64,64,65	10	1
45	3		10,11,11	-	73,74,74	-	-	-
	5		13,11,12	-	72,73,73	-	-	-
	10		12,12,12	-	70,70,70	-	-	-
	15		13,13,12	11,11,10	69,69,69	-	2	-
	20		13,14,15	10,12,12	67,67,68	66,66,67	3	1
60	3		10,10,11	-	72,72,72	-	-	-
	5		11,11,11	-	71,71,72	-	-	-
	10		10,10,11	-	71,71,71	-	-	-
	15		10,10,10	10,10,9	68,68,68	-	-	-
	20		13,15,11	11,12,11	66,66,66	65,65,66	2	1
75	3		10,10,11	-	72,73,73	-	-	-
	5		11,10,11	-	70,72,72	-	-	-
	10		12,11,11	-	71,70,71	-	-	-
	15		12,12,13	11,11,12	71,70,70	-	1	-
	20		15,14,16	9,10,10	69,69,69	-	6	-

Appendix G: Rule Base Expert Systems

G.1 Expert system: Assembly order

Listing of the assembly order rule set (Rule_set_AO). The expert system rule base was described in details in chapter 9.4.1.

Attribute *Building_direction* Summary

Conclusions: start_shape, order_of_step_building, next_shape_ID, starting_block, block_build_order, current_wall_build, next_block_id
Premises: First_Block_in_wall
Possible Values: clockwise, anti_clockwise

Rules:
rule1: if First_Block_in_wall is end_to_face then Building_direction is clockwise
rule2: if First_Block_in_wall is face_to_end then Building_direction is anti_clockwise

Attribute *First_Block_in_wall* Summary

Conclusions: corner_placing_order, first_block_in_corner, Building_direction
Premises: which_level
Possible Values: face_to_end, end_to_face

Rules:
rule1: if which_level is !=0 then First_Block_in_wall is face_to_end
rule2: if which_level is ==0 then First_Block_in_wall is end_to_face

Attribute *LEVEL* Summary

Conclusions: level_state
Premises: BLOCK_Z_POS, NEXT_BLOCK_Z_POS, current_level

Rules:
```
rule1:  if BLOCK_Z_POS is number and  NEXT_BLOCK_Z_POS is
              >@BLOCK_Z_POS and  current_level is number
        then LEVEL is @current_level+1

rule2:  if BLOCK_Z_POS is number and NEXT_BLOCK_Z_POS is
              <@BLOCK_Z_POS and  current_level is number
        then LEVEL is @current_level-1

rule3:  if BLOCK_Z_POS is number and NEXT_BLOCK_Z_POS is
              ==@BLOCK_Z_POS and  current_level is number
        then LEVEL is @current_level
```

Attribute *block* Summary

Conclusions: first_block_in_corner
Premises: level_state, which_level, prev_direction, direction, prev_level_state, no_blocks, block_ID
Possible Values: end_to_face, face_to_end, not_corner

Rules:
rule1: if level_state is ==0 and prev_direction is ==0 and direction is ==0 and
no_blocks is number and block_ID is !=@no_blocks
then block is not_corner

rule2: if level_state is ==0 and which_level is !=0 and prev_direction is ==0 and
direction is !=0 and prev_level_state is ==0 and no_blocks is number and
block_ID is !=@no_blocks
then block is end_to_face

rule3: if level_state is ==0 and which_level is ==0 and prev_direction is ==0 and
direction is !=0 and prev_level_state is ==0 and no_blocks is number
and block_ID is !=@no_blocks
then block is face_to_end

rule4: if level_state is ==0 and which_level is !=0 and prev_direction is !=0 and
direction is ==0 and prev_level_state is ==0 and no_blocks is number
and block_ID is !=@no_blocks
then block is face_to_end

rule5: if level_state is ==0 and which_level is ==0 and prev_direction is !=0 and
direction is ==0 and prev_level_state is ==0 and no_blocks is number and
block_ID is !=@no_blocks
then block is end_to_face

rule6: if level_state is !=0 and which_level is !=0 and prev_direction is ==0 and
direction is !=0 and prev_level_state is ==0 and no_blocks is number
and block_ID is !=@no_blocks
then block is end_to_face

rule7: if level_state is !=0 and which_level is ==0 and prev_direction is ==0 and
direction is !=0 and prev_level_state is ==0 and no_blocks is number and
block_ID is !=@no_blocks
then block is face_to_end

rule8: if level_state is ==0 and which_level is ==0 and prev_direction is !=0 and
direction is ==0 and prev_level_state is !=0 and no_blocks is number and
block_ID is !=@no_blocks
then block is end_to_face

```
rule9:  if level_state is ==0 and which_level is !=0 and prev_direction is !=0 and
               direction is ==0 and prev_level_state is !=0 and no_blocks is number and
               block_ID is !=@no_blocks
        then block is face_to_end

rule10: if no_blocks is number and  block_ID is 1
        then block is face_to_end

rule11: if which_level is !=0 and  no_blocks is number and  block_ID is ==@no_blocks
       then block is end_to_face

rule12: if which_level is ==0 and  no_blocks is number and block_ID is ==@no_blocks
       then block is face_to_end
```

Attribute *block_build_order* Summary

Conclusions: none
Premises: Building_direction, wall_id, wall_build, starting_block, No_walls,
walls_processed, current_level, blocks_per_level, block_ID

Rules:

```
rule1:    if Building_direction is clockwise and wall_id is number and
               wall_build is ==@wall_id and starting_block is number
               and  No_walls is number and  walls_processed is <@No_walls and
               current_level is number and  blocks_per_level is number and
               block_ID is >=1+((@current_level-1)*@blocks_per_level)   and
               blocks_per_level*@current_level
          then block_build_order is @block_ID+1 and @block_ID

rule2:  if Building_direction is anti_clockwise and  wall_id is number
               and  wall_build is ==@wall_id  and starting_block is number
               and  No_walls is number and  walls_processed is <@No_walls
               and current_level is number and  blocks_per_level is number and
               block_ID is >1+((@current_level-1)*@blocks_per_level)   and
               =@blocks_per_level*@current_level
        then block_build_order is @block_ID-1 and @block_ID

rule3:  if Building_direction is clockwise and  wall_id is number
               and  wall_build is !=@wall_id and starting_block is number
               and  No_walls is number and walls_processed is <@No_walls
               and  current_level is number and  blocks_per_level is number
               and  block_ID is >=1+((@current_level-1)*@blocks_per_level)
              and  <@blocks_per_level*@current_level
        then block_build_order is @block_ID+1
```

```
rule4:  if Building_direction is anti_clockwise and  wall_id is number
             and  wall_build is !=@wall_id and starting_block is number
             and  No_walls is number and  walls_processed is <@No_walls
             and current_level is number and  blocks_per_level is number
             and  block_ID is <=@blocks_per_level*@current_level
             and  >1+((@current_level-1)*@blocks_per_level)
        then block_build_order is @block_ID-1

rule5:   if Building_direction is clockwise and  wall_id is number
             and  wall_build is ==@wall_id and starting_block is number
             and No_walls is number and  walls_processed is <@No_walls
             and current_level is number and  blocks_per_level is number
             and block_ID is >=@blocks_per_level*@current_level
        then block_build_order is @starting_block and @block_ID

rule6:   if Building_direction is anti_clockwise and  wall_id is number
             and wall_build is ==@wall_id or !=@wall_id and starting_block is number
             and No_walls is number and  walls_processed is <@No_walls
             and current_level is number and  blocks_per_level is number
             and block_ID is <=1+((@current_level-1)*@blocks_per_level)
        then block_build_order is @starting_block and @block_ID

rule7:   if Building_direction is clockwise and  wall_id is number
             and wall_build is !=@wall_id and starting_block is number
             and No_walls is number and  walls_processed is <@No_walls
             and current_level is number and  blocks_per_level is number
             and block_ID is >=@blocks_per_level*@current_level
        then block_build_order is @starting_block

rule8:   if Building_direction is clockwise or anti_clockwise
             and wall_id is number and wall_build is ==@wall_id or !=@wall_id
             and  starting_block is number and No_walls is number
             and walls_processed is =@No_walls and current_level is number
             and blocks_per_level is number
        then block_build_order is @starting_block

rule9:  if Building_direction is anti_clockwise and  wall_id is number
             and  wall_build is !=@wall_id and starting_block is number
             and  No_walls is number and  walls_processed is <@No_walls
             and  current_level is number and   blocks_per_level is number and
             block_ID is <=1+((@current_level-1)*@blocks_per_level)
        then block_build_order is @starting_block
```

Attribute *block_direction* Summary

Conclusions: none
Premises: block_XYSET_direction, block_ID, PREV_BLOCK_X_POS, BLOCK_X_POS, PREV_BLOCK_Y_POS, BLOCK_Y_POS

Rules:
rule1: if block_ID is 1 then block_direction is Y_pos_dirct

rule2: if block_XYSET_direction is 1 and block_ID is !=1
and PREV_BLOCK_X_POS is number and BLOCK_X_POS is
<@PREV_BLOCK_X_POS
then block_direction is X_neg_dirct

rule3: if block_XYSET_direction is 1 and block_ID is !=1
and PREV_BLOCK_X_POS is number and BOCK_X_POS is
>@PREV_BLOCK_X_POS
then block_direction is X_pos_dirct

rule4: if block_XYSET_direction is 2 and block_ID is !=1
and PREV_BLOCK_Y_POS is number and BLOCK_Y_POS is
<@PREV_BLOCK_Y_POS
then block_direction is Y_neg_dirct

rule5: if block_XYSET_direction is 2 and block_ID is !=1
and PREV_BLOCK_Y_POS is number and BLOCK_Y_POS is
>@PREV_BLOCK_Y_POS
then block_direction is Y_pos_dirct

Attribute *block_size* Summary

Conclusions: corner_placing_order
Premises: size

Rules:
rule1: if size is =>410 then block_size is large

rule2: if size is <410 then block_size is small

Attribute *block_to_wall_pos* Summary

Conclusions: none
Premises: prev_XYSET_direction, block_XYSET_direction, next_XYSET_direction, block_ID

Rules:
rule1: if prev_XYSET_direction is number and block_XYSET_direction is
!=@prev_XYSET_direction and block_ID is !=1
then block_to_wall_pos is first_block

```
rule2:  if block_ID is 1  then block_to_wall_pos is first_block

rule3:  if prev_XYSET_direction is number and  block_XYSET_direction is
            ==@prev_XYSET_direction      and  next_XYSET_direction is
            ==@block_XYSET_direction and  block_ID is !=1
        then block_to_wall_pos is middle_block

rule4:  if block_XYSET_direction is number and next_XYSET_direction is
            !=@block_XYSET_direction and  block_ID is !=1
        then block_to_wall_pos is last_block
```

Attribute *blocks_per_level* Summary

Conclusions: block_build_order, other_block_in_corner_id, next_block_id
Premises: no_blocks, no_levels

Rules:

```
rule1:  if no_blocks is number and  no_levels is number
        then blocks_per_level is (@no_blocks/@no_levels)
```

Attribute *corn_first_place* Summary

Conclusions: none
Premises: corner_placing_order, level_state, level_size,   block_ID
Rules:

```
rule1:  if corner_placing_order is first and level_size is number and block_ID is number
        then corn_first_place is @block_ID

rule2:  if corner_placing_order is second and  level_state is ==0  and  level_size is
                number and block_ID is number
        then corn_first_place is @block_ID+1

rule3:  if corner_placing_order is second and  level_state is !=0  and  level_size is
             number and  block_ID is number
        then corn_first_place is (@block_ID+1)-@level_size
```

Attribute *corn_second_place* Summary

Conclusions: none
Premises: corner_placing_order, level_state, level_size, block_ID
Rules:

```
rule1:if corner_placing_order is second and  level_size is number
           and  block_ID is number
      then corn_second_place is @block_ID

rule2:  if corner_placing_order is first and  level_state is !=0 and  level_size is number
             and  block_ID is number
```

then corn_second_place is (@block_ID+1)-@level_size

rule3: if corner_placing_order is first and level_state is ==0 and level_size is number
and block_ID is number
then corn_second_place is @block_ID+1

Attribute *corner_ids_in_shape* Summary

Conclusions: none
Premises: three_wall_shape, No_walls, current_level
Rules:
rule1: if three_wall_shape is number and No_walls is number
and current_level is number
then corner_ids_in_shape is @three_wall_shape +
(@No_walls*(@current_level-1))
and @three_wall_shape +(@No_walls*(@current_level-1))+1

Attribute *corner_placing_order* Summary

Conclusions: corn_first_place, corn_second_place
Premises: next_block_size, block_size, first_block_in_corner,
First_Block_in_wall
Rules:
rule1: if next_block_size is small and block_size is large and first_block_in_corner is
face_to_end and First_Block_in_wall is face_to_end or end_to_face
then corner_placing_order is first

rule2: if next_block_size is large and block_size is large and first_block_in_corner is
face_to_end and First_Block_in_wall is end_to_face
then corner_placing_order is first

rule3: if next_block_size is large and block_size is small and first_block_in_corner is
face_to_end and First_Block_in_wall is face_to_end or end_to_face
then corner_placing_order is first

rule4: if next_block_size is small and block_size is small and first_block_in_corner is
face_to_end and First_Block_in_wall is face_to_end or end_to_face
then corner_placing_order is first

rule5: if next_block_size is large and block_size is large and first_block_in_corner is
end_to_face and First_Block_in_wall is face_to_end
then corner_placing_order is second

rule6: if next_block_size is large and block_size is small and first_block_in_corner is
end_to_face and First_Block_in_wall is face_to_end or end_to_face

then corner_placing_order is first

rule7: if next_block_size is small and block_size is large and
first_block_in_corner is end_to_face
and First_Block_in_wall is face_to_end or end_to_face
then corner_placing_order is second

rule8: if next_block_size is small and block_size is small and first_block_in_corner is
end_to_face and First_Block_in_wall is face_to_end or end_to_face
then corner_placing_order is second

rule9: if first_block_in_corner is not_corner then corner_placing_order is not_corner

Attribute *current_wall_build* Summary

Conclusions: none
Premises: block_wall_id, Building_direction, wall_id, wall_build

Rules:
rule1: if block_wall_id is last_block and Building_direction is clockwise and
wall_id is number and wall_build is ==@wall_id
then current_wall_build is next

rule2: if block_wall_id is first_block and Building_direction is anti_clockwise and
wall_id is number and wall_build is ==@wall_id
then current_wall_build is next

rule3: if wall_id is number and wall_build is !=@wall_id
then current_wall_build is same

rule4: if block_wall_id is first_block or middle_block and Building_direction is
clockwise and wall_id is number and wall_build is ==@wall_id
then current_wall_build is same

rule5: if block_wall_id is last_block or middle_block and Building_direction is
anti_clockwise and wall_id is number and wall_build is ==@wall_id
then current_wall_build is same

Attribute *direction* Summary

Conclusions: block, yaw_placing_direction
Premises: block_XYSET_direction, next_XYSET_direction

Rules:
rule1: if block_XYSET_direction is number and next_XYSET_direction is number
then direction is @next_XYSET_direction-@block_XYSET_direction

Attribute *first_block_in_corner* Summary

Conclusions: corner_placing_order
Premises: First_Block_in_wall, block, next_block
Rules:
rule1: if First_Block_in_wall is face_to_end and block is end_to_face and
next_block is face_to_end
then first_block_in_corner is end_to_face

rule2: if First_Block_in_wall is face_to_end and block is face_to_end and
next_block is face_to_end or end_to_face or not_corner
then first_block_in_corner is not_corner

rule3: if First_Block_in_wall is end_to_face and block is face_to_end and
next_block is end_to_face
then first_block_in_corner is face_to_end

rule4: if First_Block_in_wall is end_to_face and block is end_to_face and
next_block is face_to_end or end_to_face or not_corner
then first_block_in_corner is not_corner

rule5: if First_Block_in_wall is face_to_end or end_to_face and block is not_corner
and next_block is face_to_end or end_to_face or not_corner
then first_block_in_corner is not_corner

Attribute *level_size* Summary

Conclusions: corn_first_place, corn_second_place
Premises: no_levels, no_blocks

Rules:
rule1: if no_levels is number and no_blocks is number
then level_size is @no_blocks/@no_levels

Attribute *level_state* Summary

Conclusions: block, corn_first_place, corn_second_place
Premises: LEVEL, current_level

Rules:
rule1: if LEVEL is number and current_level is number
then level_state is @LEVEL-@current_level

Attribute *next_block_id* Summary

Conclusions: none
Premises: starting_block, block_ID, Building_direction, blocks_per_level, current_level, blocks_processed

Rules:

rule1: if starting_block is number and block_ID is number and Building_direction is clockwise or anti_clockwise and blocks_per_level is number and current_level is number and blocks_processed is =@blocks_per_level*@current_level
then next_block_id is @starting_block

rule2: if starting_block is number and block_ID is number and Building_direction is clockwise and blocks_per_level is number and current_level is number and blocks_processed is !=@blocks_per_level*@current_level
then next_block_id is @block_ID+1

rule3: if starting_block is number and block_ID is number and Building_direction is anti_clockwise and blocks_per_level is number and current_level is number and blocks_processed is !=@blocks_per_level*@current_level
then next_block_id is @block_ID-1

rule4: if starting_block is number and block_ID is 1 and Building_direction is clockwise or anti_clockwise and blocks_per_level is number and current_level is number
then next_block_id is @starting_block

Attribute *next_block_size* Summary

Conclusions: corner_placing_order
Premises: size_next

Rules:

rule1: if size_next is =>410 then next_block_size is large

rule2: if size_next is <410 then next_block_size is small

Attribute *next_shape_ID* Summary

Conclusions: none
Premises: Building_direction, shape_ID, start_shape, prev_level, current_level, next_level

Rules:

rule1: if Building_direction is clockwise or anti_clockwise and shape_ID is ==0 and start_shape is number and prev_level is number
then next_shape_ID is @start_shape

rule2: if Building_direction is clockwise or anti_clockwise and start_shape is number
    and prev_level is number and current_level is number and
    next_level is !=@current_level
  then next_shape_ID is @start_shape

rule3: if Building_direction is anti_clockwise and shape_ID is !=0 and start_shape is
    number and prev_level is number and current_level is number and
    next_level is ==@current_level
  then next_shape_ID is @shape_ID-1

rule4: if Building_direction is clockwise and shape_ID is !=0 and start_shape is
number and prev_level is number and current_level is number and
    next_level is ==@current_level
  then next_shape_ID is @shape_ID+1

Attribute *no_shapes* Summary

Conclusions: start_shape
Premises: No_walls
Rules:
rule1: if No_walls is number then no_shapes is @No_walls-1

Attribute *order_of_step_building* Summary

Conclusions: none
Premises: Building_direction, shape_wall1, shape_wall2, shape_wall3, three_wall_shape
Rules:
rule1: if Building_direction is clockwise and shape_wall1 is number and shape_wall2
is number and shape_wall3 is number and three_wall_shape is string
  then order_of_step_building is @three_wall_shape and @shape_wall1 and
    @shape_wall2 and @shape_wall3 and @three_wall_shape

rule2: if Building_direction is anti_clockwise and shape_wall1 is number and
    shape_wall2 is number and shape_wall3 is number
    and three_wall_shape is string
  then order_of_step_building is @three_wall_shape and
    @shape_wall3 and @shape_wall2 and @shape_wall1

Attribute *other_block_in_corner_id* Summary

Conclusions: none
Premises: block_wall_id, blocks_per_level, current_level, block_ID
Rules:

```
rule2:  if block_wall_id is first_block and  blocks_per_level is number
                and current_level is number
                and block_ID is >1+((@current_level 1)*@blocks_per_level)
                and  <@blocks_per_level*@current_level
        then other_block_in_corner_id is @block_ID-1

rule3:  if block_wall_id is last_block and blocks_per_level is number and current_level
                is number and block_ID is <@blocks_per_level*@current_level  and
                >1+((@current_level-1)*@blocks_per_level)
          then other_block_in_corner_id is @block_ID+1

rule4:  if block_wall_id is first_block and blocks_per_level is number current_level is
                number and block_ID is =1+((@current_level-1)*@blocks_per_level)
          then other_block_in_corner_id is @blocks_per_level*@current_level

rule5:  if block_wall_id is last_block and blocks_per_level is number and current_level
                is number and block_ID is =@blocks_per_level*@current_level
          then other_block_in_corner_id is 1+((@current_level-1)*@blocks_per_level)
```

Attribute *prev_direction* Summary

Conclusions: block
Premises: prev_XYSET_direction, block_XYSET_direction

Rules:

```
rule1:  if prev_XYSET_direction is number and block_XYSET_direction is number
          then prev_direction is  @prev_XYSET_direction-@block_XYSET_direction
```

Attribute *prev_level_state* Summary

Conclusions: block
Premises: prev_level, current_level

Rules:

```
rule1:  if prev_level is number and current_level is number
        then prev_level_state is @prev_level-@current_level
```

Attribute *start_shape* Summary

Conclusions: next_shape_ID
Premises: Building_direction, prev_level, current_level, no_shapes, shapes_processed

Rules:

```
rule1:  if Building_direction is clockwise and  prev_level is number and
                current_level is ==@prev_level and  no_shapes is number
                and shapes_processed is <@no_shapes
```

```
          current_level is ==@prev_level and  no_shapes is number
          and shapes_processed is <@no_shapes
      then start_shape is 1

rule2:  if Building_direction is anti_clockwise and prev_level is number and
          current_level is ==@prev_level and  no_shapes is number and
          shapes_processed is <@no_shapes
      then start_shape is @no_shapes

rule3:  if Building_direction is clockwise and prev_level is number and
          no_shapes is number and  shapes_processed is >=@no_shapes
      then start_shape is (@no_shapes*@prev_level)+1

rule4:  if Building_direction is anti_clockwise and  prev_level is number and
          current_level is number and  no_shapes is number and
          shapes_processed is >=@no_shapes
     then start_shape is @no_shapes*(@current_level+1)
```

<u>Attribute *starting_block* Summary</u>

Conclusions: block_build_order, next_block_id

Premises: current_level, Building_direction, no_levels, no_blocks

Rules:

```
rule1:  if current_level is number and Building_direction is clockwise and
          no_levels is number and  no_blocks is number
     then starting_block is  1+(@current_level-1)*(@no_blocks/@no_levels)

rule2:  if current_level is number and  Building_direction is anti_clockwise and
          no_levels is number and  no_blocks is number
     then starting_block is  @current_level*(@no_blocks/@no_levels)
```

<u>Attribute *three_wall_shape* Summary</u>

Conclusions: corner_ids_in_shape, order_of_step_building

Premises: prev_wall_dirct, wall_direction, next_wall_dirct, No_walls, wall_id

Rules:

```
rule1:  if prev_wall_dirct is Y_pos_dirct and  wall_direction is X_pos_dirct and
          next_wall_dirct is Y_neg_dirct and  No_walls is number and
          wall_id is !=@No_walls
      then three_wall_shape is n_shape and @wall_id-1 and @wall_id
          and @wall_id+1

rule2:  if prev_wall_dirct is X_pos_dirct and  wall_direction is Y_neg_dirct and
          next_wall_dirct is X_neg_dirct and  No_walls is number and
          wall_id is !=@No_walls
      then three_wall_shape is c_mirr_shape and @wall_id-1
          and  @wall_id and @wall_id+1
```

```
rule3:  if prev_wall_dirct is Y_neg_dirct and  wall_direction is X_neg_dirct and
              next_wall_dirct is Y_pos_dirct and  No_walls is number and
              wall_id is !=@No_walls
        then three_wall_shape is u_shape and @wall_id-1 and @wall_id and
              @wall_id+1

rule4:  if prev_wall_dirct is X_neg_dirct and  wall_direction is Y_pos_dirct and
              next_wall_dirct is X_pos_dirct and   No_walls is number and
              wall_id is !=@No_walls
        then three_wall_shape is c_shape and @wall_id-1 and  @wall_id and
              @wall_id+1

rule5:  if prev_wall_dirct is X_pos_dirct and  wall_direction is Y_neg_dirct and
              next_wall_dirct is X_pos_dirct and    No_walls is number and
              wall_id is !=@No_walls
        then three_wall_shape is step and @wall_id-1 and @wall_id  and @wall_id+1

rule6:  if prev_wall_dirct is Y_neg_dirct and  wall_direction is X_pos_dirct and
              next_wall_dirct is Y_neg_dirct and  No_walls is number and
              wall_id is !=@No_walls
        then three_wall_shape is step and @wall_id-1 and @wall_id  and @wall_id+1

rule7:  if prev_wall_dirct is X_pos_dirct and  wall_direction is Y_pos_dirct and
              next_wall_dirct is X_pos_dirct and  No_walls is number and
              wall_id is !=@No_walls
        then three_wall_shape is step and @wall_id-1 and @wall_id  and @wall_id+1

rule8:  if prev_wall_dirct is X_neg_dirct and wall_direction is Y_neg_dirct and
              next_wall_dirct is X_neg_dirct and  No_walls is number
              and  wall_id is !=@No_walls
      then three_wall_shape is step and @wall_id-1 and @wall_id and @wall_id+1

rule9:  if prev_wall_dirct is Y_neg_dirct and  wall_direction is X_neg_dirct and
              next_wall_dirct is Y_neg_dirct and  No_walls is number
              and wall_id is !=@No_walls
        then three_wall_shape is step and @wall_id-1
              and @wall_id  and @wall_id+1

rule10: if prev_wall_dirct is X_neg_dirct and wall_direction is Y_pos_dirct and
              next_wall_dirct is X_neg_dirct and  No_walls is number and
              wall_id is !=@No_walls
        then three_wall_shape is step and @wall_id-1
              and @wall_id  and @wall_id+1
```

```
rule11: if prev_wall_dirct is Y_pos_dirct and  wall_direction is X_pos_dirct and
            next_wall_dirct is Y_neg_dirct and  No_walls is number and
            wall_id is ==@No_walls
        then three_wall_shape is n_shape and @wall_id-1
            and  @wall_id and @wall_id-@No_walls+1

rule12: if prev_wall_dirct is X_pos_dirct and wall_direction is Y_neg_dirct and
            next_wall_dirct is X_neg_dirct and  No_walls is number and
            wall_id is ==@No_walls
        then three_wall_shape is c_mirr_shape and @wall_id-1
            and  @wall_id and @wall_id-@No_walls+1

rule13: if prev_wall_dirct is Y_neg_dirct and  wall_direction is X_neg_dirct and
            next_wall_dirct is Y_pos_dirct and  No_walls is number and
            wall_id is ==@No_walls
        then three_wall_shape is u_shape and @wall_id-1
            and @wall_id and @wall_id-@No_walls+1

rule14: if prev_wall_dirct is X_neg_dirct and  wall_direction is Y_pos_dirct and
            next_wall_dirct is X_pos_dirct and  No_walls is number and
            wall_id is ==@No_walls
        then three_wall_shape is c_shape and @wall_id-1  and @wall_id

rule15: if prev_wall_dirct is X_pos_dirct and  wall_direction is Y_neg_dirct and
            next_wall_dirct is X_pos_dirct and  No_walls is number and
            wall_id is ==@No_walls
        then three_wall_shape is step and @wall_id-1
            and @wall_id  and @wall_id-@No_walls+1

rule16: if prev_wall_dirct is Y_neg_dirct and  wall_direction is X_pos_dirct and
            next_wall_dirct is Y_neg_dirct and No_walls is number and
            wall_id is ==@No_walls
        then three_wall_shape is step and @wall_id-1
            and @wall_id  and @wall_id-@No_walls+1

rule17: if prev_wall_dirct is X_pos_dirct and  wall_direction is Y_pos_dirct and
            next_wall_dirct is X_pos_dirct and  No_walls is number and
            wall_id is ==@No_walls
        then three_wall_shape is step and @wall_id-1
             and @wall_id  and @wall_id-@No_walls+1

rule18: if prev_wall_dirct is Y_neg_dirct and wall_direction is X_neg_dirct and
            next_wall_dirct is Y_neg_dirct and No_walls is number and
            wall_id is ==@No_walls
        then three_wall_shape is step and @wall_id-1
            and @wall_id  and @wall_id-@No_walls+1
```

```
rule19: if prev_wall_dirct is X_neg_dirct and  wall_direction is Y_neg_dirct and
            next_wall_dirct is X_neg_dirct and  No_walls is number and
            wall_id is ==@No_walls
        then three_wall_shape is step and @wall_id-1
            and @wall_id  and @wall_id-@No_walls+1

rule20: if prev_wall_dirct is X_neg_dirct and  wall_direction is Y_pos_dirct and
            next_wall_dirct is X_neg_dirct and No_walls is number
            and  wall_id is ==@No_walls
        then three_wall_shape is step and @wall_id-1
            and @wall_id  and @wall_id-@No_walls+1
```

Attribute *which_level* Summary
Conclusions: block, First_Block_in_wall
Premises: current_level
Rules:

```
rule1:if current_level is number  then which_level is fmod(@current_level,2)
```

Attribute *yaw_placing_direction* Summary
Conclusions: none
Premises: direction, block_wall_id, other_corner_block_direction, other_corner_block_wall_id, prev_yaw_direction
Rules:

```
rule1:if direction is Y_pos_dirct and  block_wall_id is first_block and
            other_corner_block_direction is X_neg_dirct and
            other_corner_block_wall_id is last_block
        then yaw_placing_direction is -90

rule2:  if direction is X_neg_dirct and  block_wall_id is last_block and
            other_corner_block_direction is Y_pos_dirct and
            other_corner_block_wall_id is first_block
        then yaw_placing_direction is 0

rule3:  if direction is Y_neg_dirct and block_wall_id is last_block and
          other_corner_block_direction is X_neg_dirct and
          other_corner_block_wall_id is first_block
        then yaw_placing_direction is 90

rule4:  if direction is X_neg_dirct and  block_wall_id is first_block and
            other_corner_block_direction is Y_neg_dirct and
            other_corner_block_wall_id is last_block
        then yaw_placing_direction is 0
```

```
rule5:  if direction is Y_pos_dirct and  block_wall_id is first_block and
              other_corner_block_direction is X_pos_dirct and
              other_corner_block_wall_id is last_block
        then yaw_placing_direction is 90

rule6:  if direction is X_pos_dirct and  block_wall_id is last_block and
          other_corner_block_direction is Y_pos_dirct and
          other_corner_block_wall_id is first_block
        then yaw_placing_direction is 0

rule7:  if direction is Y_neg_dirct and  block_wall_id is last_block and
          other_corner_block_direction is X_pos_dirct and
          other_corner_block_wall_id is first_block
        then yaw_placing_direction is -90

rule8:  if direction is X_neg_dirct and block_wall_id is first_block and
              other_corner_block_direction is Y_neg_dirct and
              other_corner_block_wall_id  is last_block
        then yaw_placing_direction is 0

rule9:  if direction is Y_pos_dirct and  block_wall_id is last_block and
              other_corner_block_direction is X_pos_dirct and
              other_corner_block_wall_id is first_block
        then yaw_placing_direction is -90

rule10: if direction is X_pos_dirct and  block_wall_id is first_block and
              other_corner_block_direction is Y_pos_dirct and
              other_corner_block_wall_id is last_block
        then yaw_placing_direction is 180

rule11: if direction is Y_neg_dirct and  block_wall_id is first_block and
              other_corner_block_direction is X_neg_dirct and
              other_corner_block_wall_id is last_block
        then yaw_placing_direction is -90

rule12: if direction is X_neg_dirct and  block_wall_id is last_block and
              other_corner_block_direction is Y_neg_dirct and
              other_corner_block_wall_id is first_block
        then yaw_placing_direction is 180

rule13: if direction is Y_pos_dirct and  block_wall_id is last_block and
              other_corner_block_direction is X_neg_dirct and
              other_corner_block_wall_id is first_block
        then yaw_placing_direction is 90
```

```
rule14: if direction is X_neg_dirct and  block_wall_id is first_block and
            other_corner_block_direction is Y_pos_dirct and
            other_corner_block_wall_id is last_block
        then yaw_placing_direction is 180

rule15: if direction is Y_neg_dirct and  block_wall_id is first_block and
            other_corner_block_direction is X_pos_dirct and
            other_corner_block_wall_id  is last_block
        then yaw_placing_direction is 90

rule16: if direction is X_pos_dirct and block_wall_id is last_block and
            other_corner_block_direction is Y_neg_dirct and
            other_corner_block_wall_id is first_block
        then yaw_placing_direction is 180

rule16: if block_wall_id is middle_block and Prev_yaw_direction is number
        then yaw_placing_direction is @prev_yaw_direction
```

The input attributes for the assembly order expert system

Attribute Name	Conclusion attribute
NEXT_BLOCK_Z_POS	LEVEL
No_walls	three_wall_shape, corner_ids_in_shape, no_shapes, block_build_order
PREV_BLOCK_X_POS	block_direction
PREV_BLOCK_Y_POS	block_direction
BLOCK_X_POS	block_direction
BLOCK_Y_POS	block_direction
BLOCK_Z_POS	LEVEL
block_ID	block, block_direction, block_to_wall_pos, corn_first_place, corn_second_place, block_build_order, other_block_in_corner_id, next_block_id
block_XYSET_direction	block_direction, direction, prev_direction, block_to_wall_pos
block_wall_id	current_wall_build, yaw_placing_direction, other_block_in_corner_id
blocks_processed	next_block_id
next_XYSET_direction	direction, block_to_wall_pos
wall_build	block_build_order, current_wall_build
wall_direction	three_wall_shape
wall_id	three_wall_shape, block_build_order, current_wall_build
walls_processed	block_build_order
prev_wall_dirct	three_wall_shape
prev_yaw_direction	yaw_placing_direction
shape_ID	next_shape_ID
shape_wall1	order_of_step_building

shape_wall2	order_of_step_building
shape_wall3	order_of_step_building
shapes_processed	start_shape
size	block_size
size_next	next_block_size
next_block	first_block_in_corner
next_level	next_shape_ID
next_wall_dirct	three_wall_shape
no_blocks	block, level_size, starting_block, blocks_per_level
no_levels	level_size, starting_block, blocks_per_level
other_corner_block_direction	yaw_placing_direction
other_corner_block_wall_id	yaw_placing_direction
prev_XYSET_direction	prev_direction, block_to_wall_pos
prev_level	prev_level_state, start_shape, next_shape_ID
current_level	LEVEL, level_state, prev_level_state, which_level, corner_ids_in_shape, start_shape, starting_block, next_shape_ID, block_build_order, other_block_in_corner_id, next_block_id

G.1.2 Example of ordering a building project

Listed is the project description file of the masonry project shown in figure 9.7, generated using the AutoCAD programme described in section 9.4.1.2. Note that type = 7 is a special block, type = 3 is half a block, and type = 4 is a whole block.

100 50 66

$(X, Y, Z, XYSET, ID, TYPE, SPLENG)$

698.205	701.253	107.5	2	**1**	7	410.0
698.205	1131.25	107.5	2	**2**	3	0
698.205	1525.0	107.5	2	**3**	4	0
698.205	1787.5	107.5	2	**4**	7	187.5
848.205	1936.25	107.5	1	**5**	7	410.0
1278.21	1936.25	107.5	1	**6**	3	0
1728.21	1936.25	107.5	1	**7**	3	0
2178.21	1936.25	107.5	1	**8**	3	0
2628.21	1936.25	107.5	1	**9**	3	0
3021.96	1936.25	107.5	1	**10**	4	0
3284.46	1936.25	107.5	1	**11**	7	187.5
3433.21	1786.25	107.5	2	**12**	7	410.0
3433.21	1487.5	107.5	2	**13**	7	187.5
3300.71	1338.75	107.5	1	**14**	7	375.0
2888.21	1338.75	107.5	1	**15**	3	0
2456.96	1338.75	107.5	1	**16**	7	412.5
2195.71	1206.25	107.5	2	**17**	7	375.0
2195.71	812.503	107.5	2	**18**	7	412.5
2045.71	551.253	107.5	1	**19**	7	410.0
1615.71	551.253	107.5	1	**20**	3	0
1165.71	551.253	107.5	1	**21**	3	0
846.955	551.253	107.5	1	**22**	7	187.5
698.205	700.003	322.5	2	**23**	7	187.5
698.205	962.503	322.5	2	**24**	4	0
698.205	1356.25	322.5	2	**25**	3	0
698.205	1786.25	322.5	2	**26**	7	410.0
846.955	1936.25	322.5	1	**27**	7	187.5
1109.46	1936.25	322.5	1	**28**	4	0
1503.21	1936.25	322.5	1	**29**	3	0
1953.21	1936.25	322.5	1	**30**	3	0
2403.21	1936.25	322.5	1	**31**	3	0
2853.21	1936.25	322.5	1	**32**	3	0
3283.21	1936.25	322.5	1	**33**	7	410.0
3433.21	1787.5	322.5	2	**34**	7	187.5
3433.21	1488.75	322.5	2	**35**	7	410.0
3189.46	1338.75	322.5	1	**36**	7	377.5
2775.71	1338.75	322.5	1	**37**	3	0
2345.71	1338.75	322.5	1	**38**	7	410.0
2195.71	1095.0	322.5	2	**39**	7	377.5
2195.71	701.253	322.5	2	**40**	7	410.0
2046.96	551.253	322.5	1	**41**	7	187.5
1728.21	551.253	322.5	1	**42**	3	0
1278.21	551.253	322.5	1	**43**	3	0

848.205	551.253	322.5	1	**44**	7	410.0
698.205	701.253	537.5	2	**45**	7	410.0
698.205	1131.25	537.5	2	**46**	3	0
698.205	1525.0	537.5	2	**47**	4	0
698.205	1787.5	537.5	2	**48**	7	187.5
848.205	1936.25	537.5	1	**49**	7	410.0
1278.21	1936.25	537.5	1	**50**	3	0
1728.21	1936.25	537.5	1	**51**	3	0
2178.21	1936.25	537.5	1	**52**	3	0
2628.21	1936.25	537.5	1	**53**	3	0
3021.96	1936.25	537.5	1	**54**	4	0
3284.46	1936.25	537.5	1	**55**	7	187.5

3433.21	1786.25	537.5	2	**56**	7	410.0
3433.21	1487.5	537.5	2	**57**	7	187.5
3300.71	1338.75	537.5	1	**58**	7	375.0
2888.21	1338.75	537.5	1	**59**	3	0
2456.96	1338.75	537.5	1	**60**	7	412.5
2195.71	1206.25	537.5	2	**61**	7	375.0
2195.71	812.503	537.5	2	**62**	7	412.5
2045.71	551.253	537.5	1	**63**	7	410.0
1615.71	551.253	537.5	1	**64**	3	0
1165.71	551.253	537.5	1	**65**	3	0
846.955	551.253	537.5	1	**66**	7	187.5

Ordered block file

ID	X	Y	Z	YAW
1	698.21	701.25	107.50	-90
22	846.96	551.25	107.50	0
21	1165.71	551.25	107.50	0
20	1615.71	551.25	107.50	0
19	2045.71	551.25	107.50	0
18	2195.71	812.50	107.50	90
17	2195.71	1206.25	107.50	-90
16	2456.96	1338.75	107.50	180
15	2888.21	1338.75	107.50	180
14	3300.71	1338.75	107.50	0
13	3433.21	1487.50	107.50	90
12	3433.21	1786.25	107.50	90
5	848.21	1936.25	107.50	180
4	698.21	1787.50	107.50	-90
3	698.21	1525.00	107.50	-90
2	698.21	1131.25	107.50	-90
11	3284.46	1936.25	107.50	180
10	3021.96	1936.25	107.50	180
9	2628.21	1936.25	107.50	180
8	2178.21	1936.25	107.50	180
7	1728.21	1936.25	107.50	180
6	1278.21	1936.25	107.50	180
34	3433.21	1787.50	322.50	90
35	3433.21	1488.75	322.50	90
36	3189.46	1338.75	322.50	0
37	2775.71	1338.75	322.50	0
38	2345.71	1338.75	322.50	180
39	2195.71	1095.00	322.50	-90
40	2195.71	701.25	322.50	90
41	2046.96	551.25	322.50	0
42	1728.21	551.25	322.50	0
43	1278.21	551.25	322.50	0
44	848.21	551.25	322.50	0
23	698.21	700.00	322.50	-90

24	698.21	962.50	322.50	-90
25	698.21	1356.25	322.50	-90
26	698.21	1786.25	322.50	-90
27	846.96	1936.25	322.50	180
28	1109.46	1936.25	322.50	180
29	1503.21	1936.25	322.50	180
30	1953.21	1936.25	322.50	180
31	2403.21	1936.25	322.50	180
32	2853.21	1936.25	322.50	180
33	3283.21	1936.25	322.50	180
66	846.96	551.25	537.50	0
65	1165.71	551.25	537.50	0
64	1615.71	551.25	537.50	0
63	2045.71	551.25	537.50	0
62	2195.71	812.50	537.50	90
61	2195.71	1206.25	537.50	-90
60	2456.96	1338.75	537.50	180
59	2888.21	1338.75	537.50	180
58	3300.71	1338.75	537.50	0
57	3433.21	1487.50	537.50	90
56	3433.21	1786.25	537.50	90
48	698.21	1787.50	537.50	-90
47	698.21	1525.00	537.50	-90
46	698.21	1131.25	537.50	-90
45	698.21	701.25	537.50	0
55	3284.46	1936.25	537.50	180
54	3021.96	1936.25	537.50	180
53	2628.21	1936.25	537.50	180
52	2178.21	1936.25	537.50	180
51	1728.21	1936.25	537.50	180
50	1278.21	1936.25	537.50	180
49	848.21	1936.25	537.50	180

G.2 Expert system: block picking

G.2.1 Rule set BP

Listing for block picking rule base (Rule_set_BP) described in chapter 9.4.1.

Attribute *Conveyor_status* Summary

Conclusions: block_location
Premises: conveyor_data
Possible Values: block_in_pick_pos, ready_for_block, block_file_created, block_file_downloaded

Rules:
rule1: if conveyor_data is X then Conveyor_status is block_in_pick_pos

rule2: if conveyor_data is W then Conveyor_status is ready_for_block

rule3: if conveyor_data is R then Conveyor_status is block_file_created

rule4: if conveyor_data is Z then Conveyor_status is block_file_downloaded

Attribute *Gripp_to_block_angle_check* Summary

Conclusions: block_location
Premises: no_checks, Hsens_mid, Hsens_side2, Hsens_side1, Gripp_to_block_angle
Possible Values: not_checked, redo_check, check_success, check_failed

Rules:
```
rule1:  if no_checks is <1   then Gripp_to_block_angle_check is not_checked
rule2:  if no_checks is 1 and  Hsens_mid is 100..400 and Hsens_side2 is 100..400 and
            Hsens_side1 is 100..400 and Gripp_to_block_angle is <10  and  >-10
            and >0.5  and  <-0.5
        then Gripp_to_block_angle_check is redo_check

rule3:  if no_checks is >=1 and  Hsens_mid is 100..400 and Hsens_side2 is 100..400
            and Hsens_side1 is 100..400 and Gripp_to_block_angle is <=0.5
            and >=-0.5
        then Gripp_to_block_angle_check is check_success

rule4:  if no_checks is 1:.3 and  Hsens_mid is 100..400 and
          Hsens_side2 is 100..400 and  Hsens_side1 is 100..400 and
            Gripp_to_block_angle is <10  and  >-10  and  >3  and  >-3
        then Gripp_to_block_angle_check is check_failed

rule5:  if no_checks is 1:.3 and  Hsens_mid is 100..400 and Hsens_side2 is 100..400
            and Hsens_side1 is 100..400 and Gripp_to_block_angle is <3  and  >-3
        then Gripp_to_block_angle_check is redo_check
```

rule6: if no_checks is >3 and Hsens_mid is 100..400 and Hsens_side2 is 100..400 and
Hsens_side1 is 100..400 and Gripp_to_block_angle is >0.5 and <-0.5
then Gripp_to_block_angle_check is check_failed

rule7: if no_checks is >=1 and Hsens_mid is <100 or >227
then Gripp_to_block_angle_check is check_failed

rule8: if no_checks is >=1 and Hsens_side2 is <100 or >227
then Gripp_to_block_angle_check is check_failed

rule9: if no_checks is >=1 and Hsens_side1 is <100 or >227
then Gripp_to_block_angle_check is check_failed

Attribute *H_sensor* Summary

Conclusions: block_location, pick_block
Premises: H_sensor_data
Possible Values: not_safe_to_pick, out_of_range, at_safe_to_pick

Rules:
rule1: if H_sensor_data is 100.:123 or 125:.500
then H_sensor is not_safe_to_pick

rule2: if H_sensor_data is =>123 and <=125
then H_sensor is at_safe_to_pick

rule3: if H_sensor_data is <100 or >500
then H_sensor is out_of_range

Attribute *L_Disp_Transducer* Summary

Conclusions: pick_block
Premises: L_Disp_transducer_data
Possible Values: inactive, at_safe_height_to_pick, at_block_close_to_gripper_top,
touching_block_but_not_safe_to_gripp, error_in_reading
Rules:
rule1: if L_Disp_transducer_data is 0.6..14.2
then L_Disp_Transducer is at_safe_height_to_pick

rule2: if L_Disp_transducer_data is 15.5..16.9
then L_Disp_Transducer is inactive

rule3: if L_Disp_transducer_data is 0.:0.6
then L_Disp_Transducer is at_block_close_to_gripper_top

rule4: if L_Disp_transducer_data is 14.2::15.5
then L_Disp_Transducer is touching_block_but_not_safe_to_gripp

rule5: if L_Disp_transducer_data is <0 or >=17
then L_Disp_Transducer is error_in_reading

Attribute *R_Disp_Transducer* Summary

Conclusions: pick_block
Premises: R_Disp_transducer_data
Possible Values: inactive, touching_block_but_not_safe_to_gripp, at_block_close_to_gripper_top, at_safe_height_to_pick, error_in_reading

Rules:
rule1: if R_Disp_transducer_data is 0.6..14.2
then R_Disp_Transducer is at_safe_height_to_pick

rule2: if R_Disp_transducer_data is 15.5..16.9
then R_Disp_Transducer is inactive

rule3: if R_Disp_transducer_data is 0.:0.6
then R_Disp_Transducer is at_block_close_to_gripper_top

rule4: if R_Disp_transducer_data is 14.2::15.5
then R_Disp_Transducer is touching_block_but_not_safe_to_gripp

rule5: if R_Disp_transducer_data is <0 or =>17
then R_Disp_Transducer is error_in_reading

Attribute *V_sens_at_check_pick_pos* Summary

Conclusions: block_location
Premises: V_senosr_adjusted
Possible Values: out_of_range, conveyor_level, block_on_conveyor_level, block_edge_on_conv_level, object_on_conveyor_level, below_conveyor_level,

Rules:
rule1: if V_senosr_adjusted is <100 or >600
then V_sens_at_check_pick_pos is out_of_range

rule2: if V_senosr_adjusted is 434:.437
then V_sens_at_check_pick_pos is conveyor_level

rule3: if V_senosr_adjusted is 217.:225
then V_sens_at_check_pick_pos is block_on_conveyor_level

rule4: if V_senosr_adjusted is 225.:235
then V_sens_at_check_pick_pos is block_edge_on_conv_level

rule5: if V_senosr_adjusted is 235..434 or 100.:217
then V_sens_at_check_pick_pos is object_on_conveyor_level

rule6: if V_senosr_adjusted is <600 and >437
then V_sens_at_check_pick_pos is below_conveyor_level

Attribute *V_sens_at_pick_pos* Summary

Conclusions: pick_block
Premises: V_senosr_adjusted
Possible Values: at_pick_pos_level, out_of_range, conveyor_level, object_on_conveyor_level, below_conveyor_level, number
Rules:
rule1: if V_senosr_adjusted is 138..142
then V_sens_at_pick_pos is at_pick_pos_level

rule2: if V_senosr_adjusted is <100 or >600
then V_sens_at_pick_pos is out_of_range

rule3: if V_senosr_adjusted is 352..354
then V_sens_at_pick_pos is conveyor_level

rule4: if V_senosr_adjusted is <600 and >360
then V_sens_at_pick_pos is below_conveyor_level

rule5: if V_senosr_adjusted is >142 and <354
then V_sens_at_pick_pos is object_on_conveyor_level

Attribute *Z_axis_pos* Summary

Conclusions: block_location, pick_block
Premises: Z_pos_data
Possible Values: at_check_pick_pos, at_pick_pos, not_at_check_pick_pos

Rules:
rule1: if Z_pos_data is -911..-909 then Z_axis_pos is at_check_pick_pos

rule2: if Z_pos_data is -993..-991 then Z_axis_pos is at_pick_pos

rule3: if Z_pos_data is >-909 or -911::-993 or <-993
then Z_axis_pos is not_at_check_pick_pos

Attribute *at_X_BPP* Summary

Conclusions: robot_XY_position
Premises: X_BLOCK_PICK_POS, X_axis_pos
Possible Values: abs(@X_BLOCK_PICK_POS-@X_axis_pos),

Rules:
rule1: if X_BLOCK_PICK_POS is number and X_axis_pos is number
then at_X_BPP is abs(@X_BLOCK_PICK_POS-@X_axis_pos)

Attribute *at_Y_BPP* Summary

Conclusions: robot_XY_position
Premises: Y_BLOCK_PICK_POS, Y_axis_pos
Possible Values: abs(@Y_BLOCK_PICK_POS-@Y_axis_pos), 0, >15, =<15

Rules:
rule1: if Y_BLOCK_PICK_POS is number and Y_axis_pos is number
then at_Y_BPP is abs(@Y_BLOCK_PICK_POS-@Y_axis_pos)

Attribute *block_height_difference* Summary

Conclusions: V_senosr_adjusted
Premises: ideal_height, block_height
Possible Values: number, @ideal_height-@block_height,

Rules:
rule1: if ideal_height is number and block_height is number
then block_height_difference is @ideal_height-@block_height

Attribute *block_location* Summary

Conclusions: pick_block
Premises: Conveyor_status, V_sens_at_check_pick_pos, H_sensor, Z_axis_pos, Gripp_to_block_angle_check, robot_XY_position, no_H_checks
Possible Values: in_position, check_block_pos, not_in_position, load_block_on_conveyor, block_file_not_processed, block_file_being_processed, check_robot_position,

relocate_conveyor_position,check_robot_to_block_horizontal_distance, object_on_conveyor, check_Vertical_sensor_readings, cannot_configure_robot, check_Rob_to_Blk_Hdistance, Rob_to_Block_distance_check_failed

Rules:
rule1: if Conveyor_status is block_in_pick_pos and V_sens_at_check_pick_pos is block_on_conveyor_level and H_sensor is at_safe_to_pick and Z_axis_pos is at_check_pick_pos and Gripp_to_block_angle_check is check_success and robot_XY_position is at_Block_Pick_position
then block_location is in_position

rule2: if Conveyor_status is block_in_pick_pos and V_sens_at_check_pick_pos is conveyor_level or block_on_conveyor_level or block_edge_on_conv_level and H_sensor is not_safe_to_pick or at_safe_to_pick and Z_axis_pos is at_check_pick_pos and Gripp_to_block_angle_check is not_checked and robot_XY_position is at_Block_Pick_position
then block_location is check_block_pos

rule3: if Conveyor_status is block_in_pick_pos and V_sens_at_check_pick_pos is out_of_range or below_conveyor_level and H_sensor is not_safe_to_pick and Z_axis_pos is at_check_pick_pos and Gripp_to_block_angle_check is not_checked and robot_XY_position is at_Block_Pick_position
then block_location is check_block_pos

rule4: if Conveyor_status is block_in_pick_pos and V_sens_at_check_pick_pos is conveyor_level or block_on_conveyor_level or block_edge_on_conv_level and H_sensor is not_safe_to_pick and Z_axis_pos is at_check_pick_pos and Gripp_to_block_angle_check is check_success and robot_XY_position is at_Block_Pick_position and no_H_checks is <3
then block_location is check_Rob_to_Blk_Hdistance

rule5: if Conveyor_status is block_in_pick_pos and V_sens_at_check_pick_pos is out_of_range or conveyor_level or block_edge_on_conv_level or below_conveyor_level and H_sensor is not_safe_to_pick and Z_axis_pos is at_check_pick_pos and Gripp_to_block_angle_check is check_success and robot_XY_position is at_Block_Pick_position and no_H_checks is <3
then block_location is check_Rob_to_Blk_Hdistance

rule6: if Gripp_to_block_angle_check is check_success and no_H_checks is >=3
then block_location is Rob_to_Block_distance_check_failed

rule7: if Z_axis_pos is at_check_pick_pos or at_pick_pos or not_at_check_pick_pos and robot_XY_position is not_at_block_pick_pos
then block_location is check_robot_position

rule8: if Z_axis_pos is at_pick_pos or not_at_check_pick_pos and robot_XY_position is at_Block_Pick_position
then block_location is check_robot_position

rule9: if Conveyor_status is block_file_created
then block_location is block_file_not_processed

rule10: if Conveyor_status is block_file_downloaded
then block_location is block_file_being_processed

```
rule11: if Conveyor_status is ready_for_block
        then block_location is load_block_on_conveyor

rule12: if Conveyor_status is block_in_pick_pos and V_sens_at_check_pick_pos is
                conveyor_level or  block_edge_on_conv_level and H_sensor is
                at_safe_to_pick and Z_axis_pos is at_check_pick_pos and
            Gripp_to_block_angle_check is check_success and  robot_XY_position is
            at_Block_Pick_position and no_H_checks is <3
        then block_location is not_in_position

rule13: if Conveyor_status is block_in_pick_pos and
            V_sens_at_check_pick_pos is block_on_conveyor_level or
            block_edge_on_conv_level and H_sensor is out_of_range and
            Z_axis_pos is at_check_pick_pos and Gripp_to_block_angle_check is
                not_checked and robot_XY_position is at_Block_Pick_position
        then block_location is not_in_position

rule14: if Conveyor_status is block_in_pick_pos and  V_sens_at_check_pick_pos is
            conveyor_level or block_on_conveyor_level or block_edge_on_conv_level
            and H_sensor is not_safe_to_pick or at_safe_to_pick and
            Z_axis_pos is at_check_pick_pos and Gripp_to_block_angle_check is
                check_failed and  robot_XY_position is at_Block_Pick_position
      then block_location is not_in_position

rule15: if Conveyor_status is block_in_pick_pos and V_sens_at_check_pick_pos is
                block_edge_on_conv_level  and H_sensor is out_of_range and
Z_axis_pos is           at_check_pick_pos and Gripp_to_block_angle_check is
check_success and                   robot_XY_position is at_Block_Pick_position
        then block_location is not_in_position

rule16: if Conveyor_status is block_in_pick_pos and V_sens_at_check_pick_pos is
                        block_on_conveyor_level  and H_sensor is out_of_range and
                        Z_axis_pos is at_check_pick_pos and Gripp_to_block_angle_check
is              check_success and robot_XY_position is at_Block_Pick_position
        then block_location is not_in_position and check_H_sensor_readings

rule17: if Conveyor_status is block_in_pick_pos and V_sens_at_check_pick_pos is
                    block_on_conveyor_level or block_edge_on_conv_level and
            H_sensor is out_of_range and  Z_axis_pos is at_check_pick_pos and
            Gripp_to_block_angle_check is check_failed and  robot_XY_position is
            at_Block_Pick_position
        then block_location is not_in_position
```

rule18: if Conveyor_status is block_in_pick_pos and V_sens_at_check_pick_pos is out_of_range and H_sensor is at_safe_to_pick and Z_axis_pos is at_check_pick_pos and Gripp_to_block_angle_check is check_success and robot_XY_position is at_Block_Pick_position
then block_location is check_Vertical_sensor_readings

rule19: if Conveyor_status is block_in_pick_pos and V_sens_at_check_pick_pos is out_of_range and H_sensor is not_safe_to_pick and Z_axis_pos is at_check_pick_pos and Gripp_to_block_angle_check is check_failed and robot_XY_position is at_Block_Pick_position
then block_location is relocate_conveyor_position

rule20: if Conveyor_status is block_in_pick_pos and V_sens_at_check_pick_pos is out_of_range or conveyor_level or below_conveyor_level and H_sensor is out_of_range and Z_axis_pos is at_check_pick_pos and Gripp_to_block_angle_check is not_checked or check_failed and robot_XY_position is at_Block_Pick_position
then block_location is relocate_conveyor_position

rule21: if Conveyor_status is block_in_pick_pos and V_sens_at_check_pick_pos is below_conveyor_level and H_sensor is not_safe_to_pick or at_safe_to_pick and Z_axis_pos is at_check_pick_pos and Gripp_to_block_angle_check is check_failed and robot_XY_position is at_Block_Pick_position
then block_location is relocate_conveyor_position

rule22: if Conveyor_status is block_in_pick_pos and V_sens_at_check_pick_pos is conveyor_level or below_conveyor_level and H_sensor is out_of_range and Z_axis_pos is at_check_pick_pos and Gripp_to_block_angle_check is check_failed and robot_XY_position is at_Block_Pick_position
then block_location is relocate_conveyor_position

rule23: if Conveyor_status is block_in_pick_pos and V_sens_at_check_pick_pos is below_conveyor_level and H_sensor is at_safe_to_pick and Z_axis_pos is at_check_pick_pos and Gripp_to_block_angle_check is check_success and robot_XY_position is at_Block_Pick_position
then block_location is relocate_conveyor_position and check_Vertical_sensor_readings

rule24: if Conveyor_status is block_in_pick_pos and V_sens_at_check_pick_pos is below_conveyor_level and H_sensor is at_safe_to_pick and Z_axis_pos is at_check_pick_pos and Gripp_to_block_angle_check is not_checked and robot_XY_position is at_Block_Pick_position
then block_location is relocate_conveyor_position

rule25: if Conveyor_status is block_in_pick_pos and V_sens_at_check_pick_pos is out_of_range and H_sensor is at_safe_to_pick and Z_axis_pos is at_check_pick_pos and Gripp_to_block_angle_check is not_checked or check_failed and robot_XY_position is at_Block_Pick_position
then block_location is relocate_conveyor_position

rule26: if Conveyor_status is block_in_pick_pos and V_sens_at_check_pick_pos is out_of_range or conveyor_level or below_conveyor_level and H_sensor is out_of_range and Z_axis_pos is at_check_pick_pos and Gripp_to_block_angle_check is check_success and robot_XY_position is at_Block_Pick_position
then block_location is relocate_conveyor_position check_Vertical_sensor_readings and check_H_sensor_readings

rule27: if Conveyor_status is block_in_pick_pos and V_sens_at_check_pick_pos is object_on_conveyor_level and Z_axis_pos is at_check_pick_pos and Gripp_to_block_angle_check is check_success and robot_XY_position is at_Block_Pick_position
then block_location is object_on_conveyor

rule28: if Conveyor_status is block_in_pick_pos and V_sens_at_check_pick_pos is object_on_conveyor_level and Z_axis_pos is at_check_pick_pos and Gripp_to_block_angle_check is not_checked or check_failed and robot_XY_position is at_Block_Pick_position
then block_location is object_on_conveyor

Attribute *gripper_status* Summary

Conclusions:	pick_block
Premises:	gripper_data
Possible Values:	open, closed, unknown

Rules:
rule1: if gripper_data is =1 then gripper_status is open

rule2: if gripper_data is =0 then gripper_status is closed

rule3: if gripper_data is >1 then gripper_status is unknown

Attribute *pick_block* Summary

Conclusions:	none
Premises:	block_location, gripper_status, Z_axis_pos, H_sensor, V_sens_at_pick_pos, R_Disp_Transducer, L_Disp_Transducer, no_roll_adjustments

Possible Values: open_gripper, object_but_not_block_detected_on_conveyor, @block_location, close_gripper, block_pick_successful, lower_Z_axis, lower_Z_axis_by_4mm, raise_Z_axis, block_picked_but_not_safely, error_in_Vsens_reading, error_in_Hsens_reading, relocate_conveyor, check_gripper_status, check_robot_Z_pos, not_safe_abort, error_in_L_transducer_reading, error_in_R_transducer_reading, adjust_roll, block_not_in_position, not_in_position, Rob_to_Block_distance_check_failed

Rules:

rule1: if block_location is in_position and gripper_status is open and Z_axis_pos is at_check_pick_pos and V_sens_at_pick_pos is out_of_range or conveyor_level or object_on_conveyor_level or below_conveyor_level and R_Disp_Transducer is inactive and L_Disp_Transducer is inactive
then pick_block is lower_Z_axis

rule2: if block_location is in_position and gripper_status is open or closed and Z_axis_pos is at_check_pick_pos and V_sens_at_pick_pos is at_pick_pos_level and R_Disp_Transducer is inactive and L_Disp_Transducer is inactive
then pick_block is object_but_not_block_detected_on_conveyor and not_safe_abort

Rule3: if block_location is in_position and gripper_status is closed and Z_axis_pos is at_check_pick_pos and R_Disp_Transducer is inactive and L_Disp_Transducer is inactive
then pick_block is open_gripper

rule4: if block_location is in_position and gripper_status is open and Z_axis_pos is at_pick_pos and H_sensor is at_safe_to_pick and V_sens_at_pick_pos is at_pick_pos_level and R_Disp_Transducer is at_safe_height_to_pick and L_Disp_Transducer is at_safe_height_to_pick
then pick_block is close_gripper

rule5: if block_location is in_position and gripper_status is closed and Z_axis_pos is at_pick_pos and H_sensor is at_safe_to_pick and V_sens_at_pick_pos is at_pick_pos_level and R_Disp_Transducer is at_safe_height_to_pick and L_Disp_Transducer is at_safe_height_to_pick
then pick_block is block_pick_successful

rule6: if block_location is in_position and gripper_status is open and Z_axis_pos is at_pick_pos and H_sensor is at_safe_to_pick and V_sens_at_pick_pos is at_pick_pos_level and R_Disp_Transducer is touching_block_but_not_safe_to_gripp and L_Disp_Transducer is touching_block_but_not_safe_to_gripp
then pick_block is lower_Z_axis_by_4mm

```
rule7:  if gripper_status is open and R_Disp_Transducer is
            at_block_close_to_gripper_top and L_Disp_Transducer is
            at_block_close_to_gripper_top
        then pick_block is raise_Z_axis

rule8:  if block_location is not_in_position or load_block_on_conveyor or
            block_file_not_processed or block_file_being_processed or
            check_robot_position or relocate_conveyor_position
            or object_on_conveyor or check_Vertical_sensor_readings or
            Rob_to_Block_distance_check_failed
        then pick_block is @block_location

rule9:  if block_location is in_position and  gripper_status is open or closed and
            Z_axis_pos is at_pick_pos and H_sensor is at_safe_to_pick and
            V_sens_at_pick_pos is out_of_range or conveyor_level
            or object_on_conveyor_level or below_conveyor_level  and
            R_Disp_Transducer is touching_block_but_not_safe_to_gripp or
            at_safe_height_to_pick and L_Disp_Transducer is at_safe_height_to_pick
            or touching_block_but_not_safe_to_gripp
        then pick_block is error_in_Vsens_reading

rule10: if block_location is in_position and gripper_status is open or closed and
            Z_axis_pos is at_pick_pos and H_sensor is not_safe_to_pick or
            out_of_range  and V_sens_at_pick_pos is at_pick_pos_level and
            R_Disp_Transducer is touching_block_but_not_safe_to_gripp or
            at_safe_height_to_pick and  L_Disp_Transducer is at_safe_height_to_pick
            or touching_block_but_not_safe_to_gripp
        then pick_block is error_in_Hsens_reading

rule11: if block_location is in_position and  gripper_status is closed and
            Z_axis_pos is at_pick_pos and  H_sensor is not_safe_to_pick or
            at_safe_to_pick and V_sens_at_pick_pos is at_pick_pos_level and
            R_Disp_Transducer is touching_block_but_not_safe_to_gripp and
            L_Disp_Transducer is touching_block_but_not_safe_to_gripp
        then pick_block is block_picked_but_not_safely

rule12: if block_location is in_position and gripper_status is open or closed and
            Z_axis_pos is at_pick_pos and  V_sens_at_pick_pos is conveyor_level and
            R_Disp_Transducer is inactive and  L_Disp_Transducer is inactive
        then pick_block is block_not_in_position

rule13: if block_location is in_position and gripper_status is open or closed and
            Z_axis_pos is at_pick_pos and V_sens_at_pick_pos is out_of_range or
                conveyor_level or below_conveyor_level and
            R_Disp_Transducer is inactive and  L_Disp_Transducer is inactive
        then pick_block is relocate_conveyor
```

```
rule14: if block_location is in_position and  gripper_status is open or closed and
            Z_axis_pos is at_pick_pos and  V_sens_at_pick_pos is
            object_on_conveyor_level and  R_Disp_Transducer is inactive and
            L_Disp_Transducer is inactive
        then pick_block is object_but_not_block_detected_on_conveyor
            and not_safe_abort
rule15: if gripper_status is unknown
        then pick_block is check_gripper_status and not_safe_abort

rule16: if block_location is in_position and  gripper_status is open and
            Z_axis_pos is at_pick_pos and H_sensor is at_safe_to_pick and
            V_sens_at_pick_pos is at_pick_pos_level and R_Disp_Transducer is
            at_safe_height_to_pick and L_Disp_Transducer is inactive or
            touching_block_but_not_safe_to_gripp and no_roll_adjustments is <5
       then pick_block is adjust_roll

rule17: if block_location is in_position and gripper_status is open and
            Z_axis_pos is at_pick_pos and H_sensor is at_safe_to_pick and
            V_sens_at_pick_pos is at_pick_pos_level and R_Disp_Transducer is
            inactive or  touching_block_but_not_safe_to_gripp and
               L_Disp_Transducer is at_safe_height_to_pick
            and no_roll_adjustments is <5
       then pick_block is adjust_roll

rule18: if block_location is in_position and  gripper_status is open and
            Z_axis_pos is at_pick_pos and  H_sensor is at_safe_to_pick and
            V_sens_at_pick_pos is at_pick_pos_level and R_Disp_Transducer is
            touching_block_but_not_safe_to_gripp or at_safe_height_to_pick and
            L_Disp_Transducer is inactive and no_roll_adjustments is >=5
        then pick_block is not_safe_abort and error_in_L_transducer_reading

rule19: if block_location is in_position and gripper_status is open and
            Z_axis_pos is at_pick_pos and H_sensor is at_safe_to_pick and
            V_sens_at_pick_pos is at_pick_pos_level and R_Disp_Transducer is
inactive       and L_Disp_Transducer is at_safe_height_to_pick or
            touching_block_but_not_safe_to_gripp and no_roll_adjustments is >=5
        then pick_block is not_safe_abort and error_in_R_transducer_reading

rule20: if block_location is in_position and  gripper_status is closed and
            Z_axis_pos is at_pick_pos and  H_sensor is at_safe_to_pick and
            V_sens_at_pick_pos is at_pick_pos_level and R_Disp_Transducer is
            touching_block_but_not_safe_to_gripp or at_safe_height_to_pick and
            L_Disp_Transducer is inactive and no_roll_adjustments is >=5
        then pick_block is block_picked_but_not_safely and not_safe_abort
```

rule21: if block_location is in_position and gripper_status is closed and
Z_axis_pos is at_pick_pos and H_sensor is at_safe_to_pick and
V_sens_at_pick_pos is at_pick_pos_level and R_Disp_Transducer is inactive
or touching_block_but_not_safe_to_gripp and L_Disp_Transducer is
at_safe_height_to_pick or touching_block_but_not_safe_to_gripp and
no_roll_adjustments is >=5
then pick_block is block_picked_but_not_safely and not_safe_abort

rule22: if block_location is in_position and Z_axis_pos is not_at_check_pick_pos
then pick_block is check_robot_Z_pos

rule23: if block_location is in_position and gripper_status is open and
Z_axis_pos is at_pick_pos and H_sensor is not_safe_to_pick or
out_of_range and V_sens_at_pick_pos is at_pick_pos_level or
conveyor_level or object_on_conveyor_level and
R_Disp_Transducer is inactive and L_Disp_Transducer is inactive
then pick_block is block_not_in_position and not_safe_abort

rule24: if block_location is in_position and gripper_status is open or closed and
Z_axis_pos is at_check_pick_pos and R_Disp_Transducer is
touching_block_but_not_safe_to_gripp or at_block_close_to_gripper_top or
at_safe_height_to_pick and L_Disp_Transducer is at_safe_height_to_pick
or at_block_close_to_gripper_top or touching_block_but_not_safe_to_gripp
then pick_block is not_safe_abort

rule25: if block_location is in_position and gripper_status is closed and
Z_axis_pos is at_pick_pos and H_sensor is out_of_range and
V_sens_at_pick_pos is at_pick_pos_level and R_Disp_Transducer is
touching_block_but_not_safe_to_gripp or at_safe_height_to_pick and
L_Disp_Transducer is inactive
then pick_block is block_picked_but_not_safely and not_safe_abort

rule26: if block_location is in_position and gripper_status is closed and
Z_axis_pos is at_pick_pos and H_sensor is not_safe_to_pick or
out_of_range and V_sens_at_pick_pos is at_pick_pos_level and
R_Disp_Transducer is inactive and L_Disp_Transducer is
touching_block_but_not_safe_to_gripp
then pick_block is block_picked_but_not_safely and not_safe_abort

rule27: if block_location is in_position and gripper_status is closed and
Z_axis_pos is at_pick_pos and H_sensor is out_of_range and
V_sens_at_pick_pos is at_pick_pos_level and
R_Disp_Transducer is inactive and L_Disp_Transducer is
at_safe_height_to_pick or touching_block_but_not_safe_to_gripp
then pick_block is block_picked_but_not_safely and not_safe_abort

rule28: if block_location is in_position and gripper_status is closed and
Z_axis_pos is at_pick_pos and V_sens_at_pick_pos is at_pick_pos_level and
R_Disp_Transducer is inactive and L_Disp_Transducer is inactive
then pick_block is not_safe_abort

rule29: if block_location is in_position and gripper_status is open and
Z_axis_pos is at_pick_pos and H_sensor is not_safe_to_pick or out_of_range and V_sens_at_pick_pos is at_pick_pos_level and R_Disp_Transducer is at_safe_height_to_pick or touching_block_but_not_safe_to_gripp and L_Disp_Transducer is inactive
then pick_block is not_safe_abort

rule30: if block_location is in_position and gripper_status is open and
Z_axis_pos is at_pick_pos and H_sensor is not_safe_to_pick or out_of_range and V_sens_at_pick_pos is at_pick_pos_level and R_Disp_Transducer is inactive and L_Disp_Transducer is at_safe_height_to_pick or touching_block_but_not_safe_to_gripp
then pick_block is not_safe_abort

rule31: if block_location is in_position and gripper_status is open and
Z_axis_pos is at_pick_pos and H_sensor is at_safe_to_pick and V_sens_at_pick_pos is at_pick_pos_level and R_Disp_Transducer is inactive and L_Disp_Transducer is inactive
then pick_block is not_safe_abort and error_in_L_transducer_reading and error_in_R_transducer_reading

rule32: if block_location is in_position and gripper_status is closed or open and
Z_axis_pos is at_check_pick_pos and V_sens_at_pick_pos is at_pick_pos_level and R_Disp_Transducer is at_safe_height_to_pick or touching_block_but_not_safe_to_gripp and L_Disp_Transducer is inactive
then pick_block is not_safe_abort

rule33: if block_location is in_position and gripper_status is closed or open and
Z_axis_pos is at_check_pick_pos and V_sens_at_pick_pos is at_pick_pos_level and R_Disp_Transducer is inactive and L_Disp_Transducer is at_safe_height_to_pick or touching_block_but_not_safe_to_gripp
then pick_block is not_safe_abort

rule34: if block_location is in_position and gripper_status is closed or open and
Z_axis_pos is at_check_pick_pos or at_pick_pos and V_sens_at_pick_pos is at_pick_pos_level and R_Disp_Transducer is inactive and L_Disp_Transducer is at_block_close_to_gripper_top
then pick_block is not_safe_abort

rule35: if block_location is in_position and gripper_status is closed or open and
Z_axis_pos is at_pick_pos and H_sensor is not_safe_to_pick or
out_of_range and V_sens_at_pick_pos is below_conveyor_level or
conveyor_level or object_on_conveyor_level or out_of_range and
R_Disp_Transducer is at_safe_height_to_pick or
touching_block_but_not_safe_to_gripp and L_Disp_Transducer is
at_safe_height_to_pick or touching_block_but_not_safe_to_gripp
then pick_block is not_safe_abort

rule36: if block_location is in_position and gripper_status is closed or open and
Z_axis_pos is at_check_pick_pos or at_pick_pos and
V_sens_at_pick_pos is below_conveyor_level or conveyor_level or
object_on_conveyor_level or out_of_range or number and
R_Disp_Transducer is at_safe_height_to_pick or
touching_block_but_not_safe_to_gripp and L_Disp_Transducer is inactive
then pick_block is not_safe_abort

rule37: if block_location is in_position and gripper_status is closed or open and
Z_axis_pos is at_check_pick_pos or at_pick_pos and
V_sens_at_pick_pos is below_conveyor_level or conveyor_level or
object_on_conveyor_level or out_of_range or number and
R_Disp_Transducer is inactive and L_Disp_Transducer is
at_safe_height_to_pick or at_block_close_to_gripper_top or
touching_block_but_not_safe_to_gripp
then pick_block is not_safe_abort

rule38: if block_location is in_position and gripper_status is closed or open and
Z_axis_pos is at_pick_pos and R_Disp_Transducer is
at_block_close_to_gripper_top and L_Disp_Transducer is
at_safe_height_to_pick or touching_block_but_not_safe_to_gripp
then pick_block is not_safe_abort

rule39: if block_location is in_position and gripper_status is closed or open and
Z_axis_pos is at_check_pick_pos or at_pick_pos and
R_Disp_Transducer is at_block_close_to_gripper_top and
L_Disp_Transducer is inactive
then pick_block is not_safe_abort

rule40: if R_Disp_Transducer is touching_block_but_not_safe_to_gripp or
at_block_close_to_gripper_top or at_safe_height_to_pick and
L_Disp_Transducer is error_in_reading
then pick_block is not_safe_abort and error_in_L_transducer_reading

rule41: if R_Disp_Transducer is error_in_reading and L_Disp_Transducer is
at_safe_height_to_pick or at_block_close_to_gripper_top or
touching_block_but_not_safe_to_gripp
then pick_block is not_safe_abort and error_in_R_transducer_reading

rule42: if R_Disp_Transducer is error_in_reading and L_Disp_Transducer is error_in_reading
then pick_block is not_safe_abort and error_in_L_transducer_reading and error_in_R_transducer_reading

rule43: if block_location is in_position and gripper_status is closed and Z_axis_pos is at_pick_pos and H_sensor is at_safe_to_pick and V_sens_at_pick_pos is at_pick_pos_level and R_Disp_Transducer is inactive or touching_block_but_not_safe_to_gripp and L_Disp_Transducer is at_safe_height_to_pick and no_roll_adjustments is <5
then pick_block is open_gripper and adjust_roll

rule44: if block_location is in_position and gripper_status is closed and Z_axis_pos is at_pick_pos and H_sensor is at_safe_to_pick and V_sens_at_pick_pos is at_pick_pos_level and R_Disp_Transducer is at_safe_height_to_pick and L_Disp_Transducer is inactive or touching_block_but_not_safe_to_gripp and no_roll_adjustments is <5
then pick_block is open_gripper and adjust_roll

Attribute *robot_XY_position* Summary

Conclusions: block_location
Premises: at_X_BPP, at_Y_BPP
Possible Values: at_Block_Pick_position, not_at_block_pick_pos
Rules:
rule1:if at_X_BPP is =<15 and at_Y_BPP is =<15
then robot_XY_position is at_Block_Pick_position

rule2:if at_X_BPP is >15 and at_Y_BPP is >15
then robot_XY_position is not_at_block_pick_pos

rule3:if at_X_BPP is >15 and at_Y_BPP is =<15
then robot_XY_position is not_at_block_pick_pos

rule4:if at_X_BPP is =<15 and at_Y_BPP is >15
then robot_XY_position is not_at_block_pick_pos

Attribute *V_senosr_adjusted* Summary

Conclusions: V_sens_at_check_pick_pos, V_sens_at_pick_pos
Premises: V_sensor_data, block_height_difference
Rules:
rule1:if V_sensor_data is number and block_height_difference is number
then V_senosr_adjusted is @V_sensor_data+@block_height_difference

The rest of these attributes have no rules as they are input attributes.

Attribute Name	Conclusion attribute
V_sensor_data	V_senosr_adjusted
H_sensor_data	H_sensor
Gripp_to_block_angle	Gripp_to_block_angle_check
Hsens_side1, Hsens_side2	Gripp_to_block_angle_check
L_Disp_transducer_data	L_Disp_Transducer
R_Disp_transducer_data	R_Disp_Transducer
X_BLOCK_PICK_POS	at_X_BPP
X_axis_pos	at_X_BPP
Y_BLOCK_PICK_POS	at_Y_BPP
Y_axis_pos	at_Y_BPP
ideal_height	block_height_difference
no_H_checks	block_location
block_height	block_height_difference
Z_pos_data	Z_axis_pos
conveyor_data	Conveyor_status
gripper_data	gripper_status
no_checks	Gripp_to_block_angle_check
no_roll_adjustments	pick_block

G.2.2 Results of testing the block_pick expert system

Results of the experiments carried out to test the calling program that the block pick expert system was embedded in, described in section 9.4.2.3. The programme was tested three times.

<u>Results of the first run</u>

Test 1,part 1

Attribute name	After function 'adjust_pick_pos' all new attribute values are entered and Bchain on attribute *block_location*	Conclusion was 'check_block_pos' which calls a function that aligns the gripper to the block 'check_pick_pos'	After the conclusion of check_pos robot position attributes are updated and Bchain on *block_location*
H_sensor_data	129	128	129
horizontal_sensor	not_safe _to_pick	not_safe _to_pick	not_at_safe _to_pick
v_sens_at_check_pick_pos	block_on_conv_level	block_on_ conv_level	block_on_ conv_level
V_sensor_data	220	220	220
V_sensor_adjusted	222.13	222.13	222.13
ideal_height	215	215	215
block_height	212.87	212.87	212.87

block_height_difference	213	213	213
conveyor_data	X	X	X
conveyor_status	block_in_pick_pos	block_in_pick_pos	block_in_pick_pos
gripper_data	1	1	1
gripper_status	open	open	open
gripp_to_block_angle	//	-0.3819	
gripp_to_block_angle_check	not_checked	check_success	check_success
hsens_mid	//	128	//
hsens_side1	//	130	//
hsens_side2	//	128	//
y_axis_pos	310.89	527.341	310.89
y_block_pick_pos	310.89	310.89	310.89
at_y_bpp	0	216.45	0
z_pos_data	-909.99	90.4	-909.99
z_axis_pos	at_check_pick_pos	not_at_check pick_pos	at_check_pick_pos
x_block_pick_pos	3290.65	3290.65	3290.65
x_axis_pos	3290.67	3549.57	3290.64
at_x_bpp	0	258.9	0
robot_xy_position	at_block_pick_pos	not_at_block_pick_pos	at_block_pick_pos
no_h_checks	0	0	0
no_checks	0	1	1
no_roll_adjustments	0	0	0
block_location	check_block_pos	check_robot_pos	check_robot_to_Blk_H_dist

Test 1,part 2

Attribute name	After function "check_rob_to_H_dist" attributes are updated and this will result in the conclusion in_position which automatically takes us out of the backward chaining for the ***block_location*** and starts to backward chaining on the ***pick_block*** attribute	After the conclusion lower_z_pos is called the function "lower_z_pos that will lower z to pick position, and update the attributes relevant and it will automatically take the next step depending on the conclusion.	After lowering the gripper and checking the LVDTs it is concluded that next the gripper is to be close the gripper. Call function to close the gripper and update attributes again and the conclusion of ***pick_block*** is picked successfully
h_sensor_data (mm)	124	125	
horizontal_sensor	at_safe_to_pick_pos	at_safe_to_pick_pos	
v_sens_at_check_pick_pos	block_on_conv_level	not relevant	
v_sens_at_pick_pos	//	at_pick_pos_level	at_pick_pos_level
v_sensor_data (mm)	220	139	
v_sensor_adjusted (mm)	222.13	141.13	
conveyor_data	X		
conveyor_status	block_in_pick_pos		block_in_pick_pos
gripper_data	1	1	0
gripper_status	open	open	close
gripp_to_block_angle_check	check_success		

l_disp_transducer (mm)	16.53mm		
l_disp_transducer_data	inactive	at_safe_to_pick	
r_disp_transducer (mm)	16.51		
r_disp_transducer_data	inactive	at_safe_to_pick	
y_axis_pos (mm)	305.88	305.87	
y_block_pick_pos (mm)	310.89	310.89	
at_y_bpp (mm)	5	5	
z_pos_data (mm)	-909.99	-991.99	
z_axis_pos	at_check_pick_pos	at_pick_pos	at_pick_pos
x_block_pick_pos (mm)	3290.65	3290.65	
x_axis_pos (mm)	3290.64	3290.61	
at_x_bpp	0	0	
robot_xy_position	at_block_pick_pos	at_block_pick_pos	at_block_pick_pos
no_h_checks	1		1
no_checks	1		1
no_roll_adjustments	0		0
block_location	in_position	in_position	in_position
pick_block	lower_Z_axis	close_gripper	block_pick_successful

Results of the second run

Test 2,part 1

Attribute name	After function "adjust_pick_pos", we enter all known values and backward chain on ***block_location***	Call the function "check_pick_pos", and update attributes, backward chain again on new values. Result is check_rob_to_blk_Hdist	After calling function "check_rob_Hdist" update relevant attributes and Backward chain on ***block_location***.
H_sensor_data (mm)	129	128	124
Horizontal_sensor	not_safe_to_pick	not_safe_to_pick	at_safe_to_pick
V_sens_at_check_pick_pos	block_on_conv_level	block_on_conv_level	block_on_conv_level
V_sens_at_pick_pos	//	//	//
V_sensor_data (mm)	220	220	220
V_sensor_adjusted (mm)	222.06	222.06	222.06
ideal_height (mm)	215	215	215
block_height (mm)	212.94	212.94	212.94
block_height_difference (mm)	2.06	2.06	2.06
conveyor_data	X	X	X
conveyor_status	block_in_pick_pos	block_in_pick_pos	block_in_pick_pos
gripper_data	1	1	1
gripper_status	open	open	open
gripp_to_block_angle (degrees)	//	-0.3819	
gripp_to_block_angle_check	not_checked	check_success	check_success
Hsens_mid (mm)	//	129	//
Hsens_side1 (mm)	//	130	//
Hsens_side2 (mm)	//	128	//
Y_axis_pos	311.60	311.62	306.61
Y_block_pick_pos	311.62	311.62	310.89

at_Y_bpp	0	0	4.28
Z_pos_data	-909.99	-909.99	-909.99
Z_axis_pos	at_check_ pick_pos	at_check_ pick_pos	at_check_ pick_pos
X_block_pick_pos	3290.68	3290.68	3290.65
X_axis_pos	3290.69	3290.68	3290.65
at_X_bpp	0	0	0
robot_XY_position	at_block_ pick_pos	at_block_ pick_pos	at_block_ pick_pos
no_H_checks	0	0	0
no_checks	0	1	1
no_roll_adjustments	0	0	0
block_location	check_block_pos	check_robot_ to_Blk_H_dist	in_position

Test 2,part 2

Attribute name	After function check_rob_to_H_dist the results are updated and this will result in the conclusion in_position which automatically takes us out of the backward chain for the ***block_location*** and starts to backward chain on the ***pick_block*** attribute	After the conclusion lower_z_pos we call the function lower_z_pos that will lower z axis to pick position, and update the attributes relevant then automatically take the next step depending on the conclusion.	After lowering the gripper and checking the LVDTs the conclude that we need to close the gripper so, after the function to close the gripper is called, update the attributes again and the conclusion of pick_block is that it was picked successfully
H_sensor_data (mm)	124	125	
horizontal_sensor	at_safe_to_pick_pos	at_safe_to_pick_pos	
V_sens_at_check_pick_pos	block_on_conv_level	not relevant	
V_sens_at_pick_pos	//	at_pick_pos_level	at_pick_pos_level
V_sensor_data (mm)	220	139	
V_sensor_adjusted (mm)	222.06	141.06	
conveyor_data	X		
conveyor_status	block_in_pick_pos		block_in_pick_pos
gripper_data	1	1	0
gripper_status	open	open	close
gripp_to_block_angle_check	check_success		
L_disp_transducer (mm)	16.53mm	6.23	
L_disp_transducer_data	inactive	at_safe_height_to_pick	
R_disp_transducer (mm)	16.51	6.00	
R_disp_transducer_data	inactive	at_safe_height_to_pick	
Y_axis_pos (mm)		306.61	
Y_block_pick_pos (mm)			
at_Y_bpp (mm)			
Z_pos_data (mm)	-909.99	-991	
Z_axis_pos	at_check_pick_pos	at_pick_pos	at_pick_pos
X_block_pick_pos (mm)	3290.65	3290.65	
X_axis_pos (mm)	3290.64	3290.61	
at_X_bpp	0	0	
robot_XY_position	at_block_pick_pos	at_block_pick_pos	at_block_pick_pos

no_H_checks	1	1	1
no_checks	1	1	1
no_roll_adjustments	0	0	0
block_location	in_position	in_position	in_position
pick_block	lower_Z_axis	close_gripper	block_pick_ successful

Results of the third run

Test 3,part 1

Attribute name	After function adjust_pick_pos, update attributes and backward chain on attribute ***block_location***	Call function check_pick_pos, update attributes and backward chain on attribute ***block_location*** Result is check_rob _to_blk_Hdist	After calling function check_rob_Hdist update relevant attributes and backward chain on ***block_location***
H_sensor_data (mm)	129	128	124
horizontal_sensor	not_safe_to_pick	not_safe_to_pick	at_safe_to_pick
V_sens_at_check_pick_pos	block_on _conv_level	block_on _conv_level	block_on _conv_level
V_sensor_data (mm)	220	220	220
V_sensor_adjusted (mm)	221.87	221.87	221.87
ideal_height (mm)	215	215	215
block_height (mm)	213.127	213.127	213.127
block_height_difference (mm)	1.87	1.87	1.87
conveyor_data	X	X	X
conveyor_status	block_in _pick_pos	block_in _pick_pos	block_in _pick_pos
gripper_data	1	1	1
gripper_status	open	open	open
gripp_to_block_angle (degrees)	//	-0.19	
gripp_to_block_angle_check	not_checked	check_success	check_success
Hsens_mid (mm)	//	129	//
Hsens_side1 (mm)	//	130	//
Hsens_side2 (mm)	//	129	//
L_disp_transducer (mm)	//		16.51
L_disp_transducer_data	//	//	inactive
R_disp_transducer	//	//	16.53
R_disp_transducer_data	//	//	inactive
Y_axis_pos (mm)	312.0	312.1	306.1
Y_block_pick_pos (mm)	312.1	312.1	310.89
at_Y_bpp (mm)	0	0	4.79
Z_pos_data (mm)	-909.99	-909.99	-909.99
Z_axis_pos	at_check _pick_pos	at_check _pick_pos	at_check _pick_pos
X_block_pick_pos (mm)	3290.74	3290.74	3290.65
X_axis_pos (mm)	3290.74	3290.74	3290.65
at_X_bpp (mm)	0	0	0
robot_XY_position	at_block	at_block	at_block

	_pick_pos	_pick_pos	_pick_pos
no_H_checks	0	0	0
no_checks	0	1	1
no_roll_adjustments	0	0	0
block_location	check_block_pos	check_robot _to_Blk_H_dist	in_position

Test 3,part 2

Attribute name	After lowering the Z axis to pick position, backward chain attribute ***pick_block*** on new attributes values. Result was to check the Z axis position of the robot	After the conclusion check_z_axis-pos call update attributes which automatically take the next step depending on the conclusion	After checking the position of the robot and checking and LVDTs, conclusion is to close the gripper. Call function to close gripper. Update the attributes again and the conclusion of ***pick_block*** is picked successfully
h_sensor_data (mm)	124	125	
horizontal_sensor	at_safe_to_pick	at_safe_to_pick	at_safe_to_pick
v_sens_at_pick_pos	at_pick _pos_level	at_pick _pos_level	at_pick _pos_level
v_sensor_data (mm)	139	139	
v_sensor_adjusted (mm)	140.87	140.87	
gripper_data	1	1	0
gripper_status	open	open	close
gripp_to_block_angle_check	check_success	check_success	
L_disp_transducer (mm)			6.54
L_disp_transducer_data	at_safe _height_to_pick	at_safe _height_to_pick	at_safe _height_to_pick
R_disp_transducer (mm)			6.23
R_disp_transducer_data	at_safe _height_to_pick	at_safe _height_to_pick	at_safe _height_to_pick
Y_axis_pos (mm)	522.7	306.1	
X_block_pick_pos	not_at_check _pick_pos	at_pick_pos	
X_axis_pos (mm)	3548.36	3290.7	
robot_XY_position	not_at _block_pick	at_block _pick_position	
no_h_checks		1	1
no_checks		1	1
no_roll_adjustments		0	0
block_location	in_position	in_position	in_position
pick_block	chech_robot _Z_pos	close_gripper	block_pick _successful

G.3 Expert system: move optimisation

Listing of the move optimisation rule set (Rule_set_MO). The expert system rule base was described in details in chapter 9.4.3.

Attribute *Apeak* Summary

Conclusions: Move_type, move_result
Premises: Move_distance, Move_time

Rules:
rule1: if Move_distance is number and Move_time is number
then Apeak is (6*@Move_distance)/(@Move_time*@Move_time)

Attribute *Move_type* Summary

Conclusions: move_result
Premises: Move_distance, Move_time, Axis_limit, Amax, Apeak, Vmax, Vpeak
Possible Values: out_of_range, parabolic_single, parabolic_segmented, SCurve_3_part, move_accedes_max_velocity, move_accedes_max_Acceleration,

Rules:
rule1: if Move_distance is >@Axis_limit and Axis_limit is 3969
then Move_type is out_of_range
rule2: if Move_distance is 2000 and Move_time is optimum
then Move_type is parabolic_single and 3.32

rule3: if Move_distance is 1000 and Move_time is optimum
then Move_type is parabolic_single and 2

rule4: if Move_distance is 4 and Move_time is optimum
then Move_type is parabolic_segmented and 0.35

rule5: if Move_distance is 2000 and Move_time is >=7
then Move_type is @Move_time and SCurve_3_part

rule6: if Move_distance is 1000 and Move_time is >=5
then Move_type

Attribute *Vpeak* Summary

Conclusions: Move_type, move_result
Premises: Move_distance, Move_time
Possible Values: (3*@Move_distance)/(2*Move_time), number

Rules:
```
rule1:  if Move_distance is number and  Move_time is number
        then Vpeak is (3*@Move_distance)/(2*Move_time)
```

Attribute *move_result* Summary

Conclusions: none
Premises: Move_type, Move_distance, Move_time, Amax, Apeak, Vmax, Vpeak
Possible Values: move_exceeds_max_Acceleration, move_exceeds_max_Velocity, @Move_type, parabolic_single, 3.32

Rules:
```
rule1:  if Move_type is parabolic_single and  Move_distance is number and
             Move_time is number and  Amax is <@Apeak and
             Apeak is number
        then move_result is move_exceeds_max_Acceleration

rule2:  if Move_type is parabolic_single and  Move_distance is number and
             Move_time is number and Vmax is <@Vpeak and
             Vpeak is number
        then move_result is move_exceeds_max_Velocity

rule3:  if Move_type is number or string
        then move_result is @Move_type
```

The input attributes for the move optimisation expert system

Attribute Name	Conclusion attribute
Move_time	Move_type, Apeak, Vpeak, move_result
Axis_limit	Move_type
Vmax	Move_type, move_result
Move_distance	Move_type, Apeak, Vpeak, move_result

Appendix H: Accelerometer signal processing

H.1 Verification of software

A 'C' program, *Sinwav.c,* developed by the author was written to perform an FFT on a signal of a sin wave with a fixed amplitude and frequency. The program creates a data file of 4096 samples for the sin wave according to the specifications entered and then does the FFT for it, using 'C' functions *fft()* , *offset_fft(),* and *fft_file().* These produce the power spectrum (FFT) and the amplitude power spectrum. The test data of the sin wave shown in figure H.1 is obtained from:

$$A\sin(wt) \qquad \text{eqn H.1}$$

, where w is the angular frequency and is expressed as follows:

$$w=2\Pi f(i/4096) \qquad \text{eqn H.2}$$

, where i is in the range i=0→ i=4095, A is the amplitude arbitrarily set to equal 2, and the frequency f of the sin wave expressed as:

$$f=2/T \qquad \text{eqn H.3}$$

The result of the FFT program on the data set is a power spectrum that spans the 4096 samples, as seen in figure H.2. The power spectrum is made up of a mirror image of the result that is reflected at the mid point i= 2048. At point i = 2 the frequency is f = 2/T, and the amplitude is half that of the sin wave. That spectrum is processed by adding the two mirror images of the spectrum together, apart from the value corresponding to i=0, which is the mean value. The result is an amplitude spectrum of the FFT shown in figure H.3. This indicates a frequency f=2/T at i=2, with an amplitude equivalent to that of the sin wave tested.

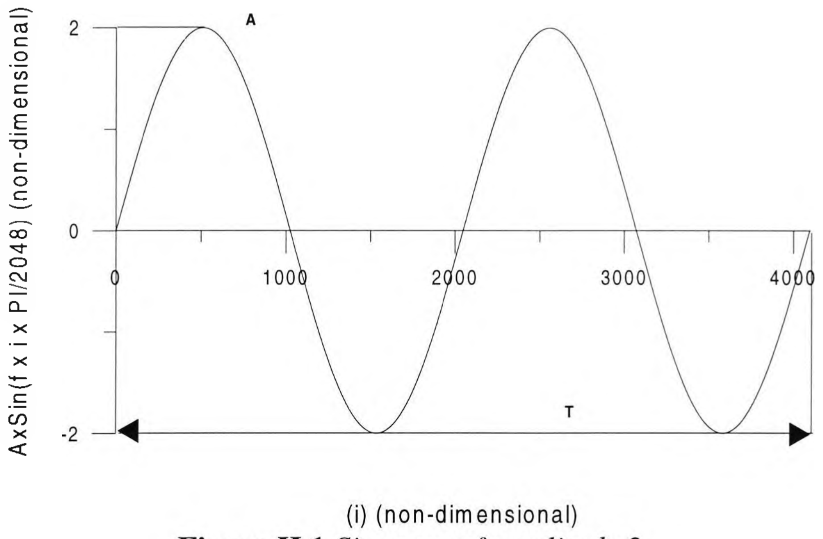


Figure H.1 *Sin wave of amplitude 2*

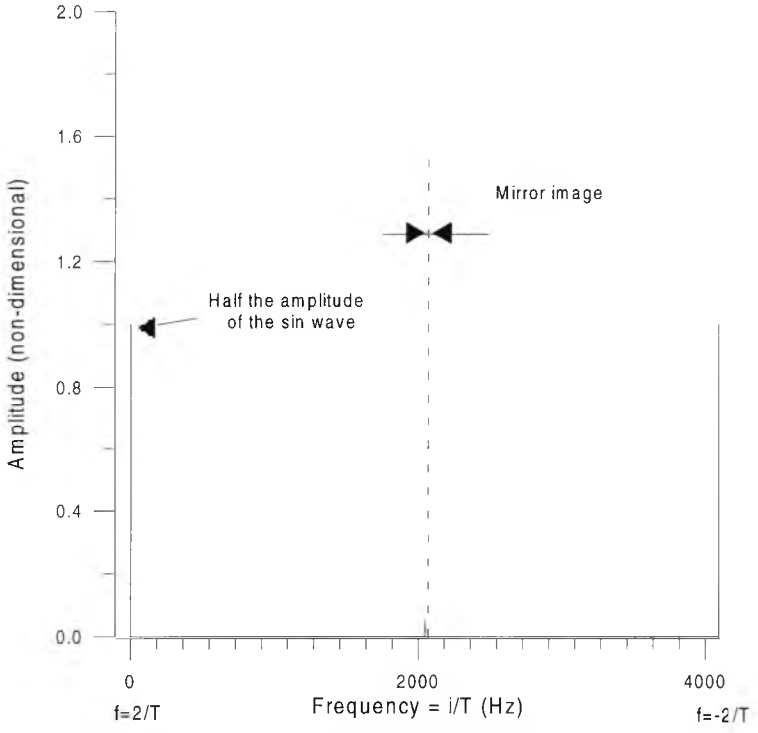


Figure H.2 *Power spectrum of the sin wave*

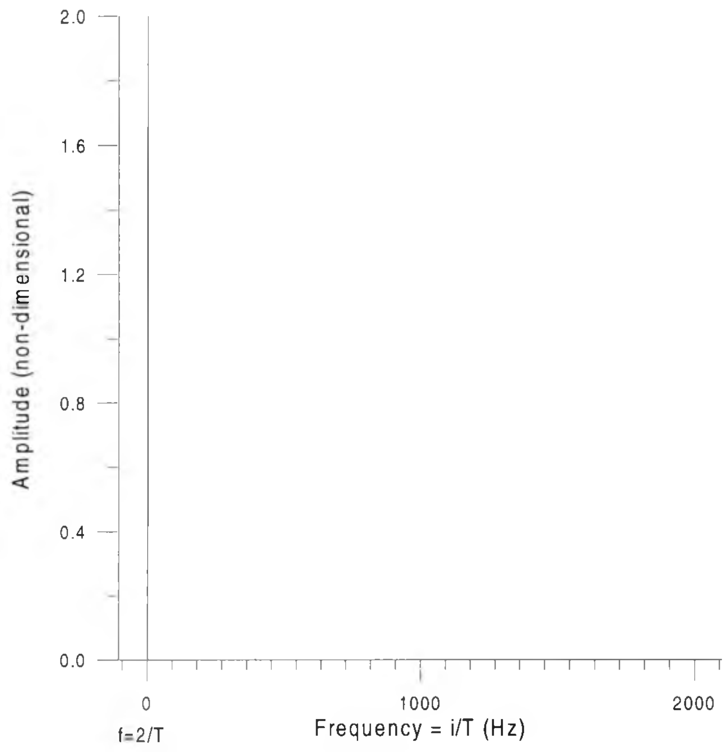


Figure H.3 *Amplitude spectrum of the sin wave*

The results of the power spectrum FFT and amplitude spectrum of the sin wave, shown in figure H.2 and figure H.3, are as expected. This demonstrates the validity of the functions developed.

H.2 Frequency selective filtering

The data captured by the *PC-30* board is pre-filtered to reduce noise. The relevant frequencies from the signal are the low frequencies, these corresponding to the vibration of the end-effector. A low pass filter with a 23 Hz cut-off is used (appendix A.7.2). To check the filtering does not cut out any vital frequencies, the original signal is kept and a modified one filtered, and then both signals are scaled to be at the same amplitude. Accelerometer data from the two signals, the raw as plotted in figure H.4 and the filtered as plotted in figure H.5, are captured simultaneously as a result of a move of the robot.

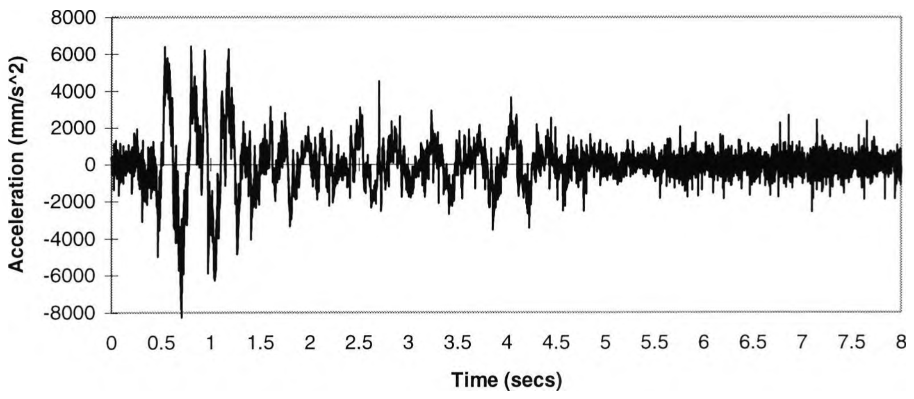


Figure H.4 *Accelerometer raw signal*

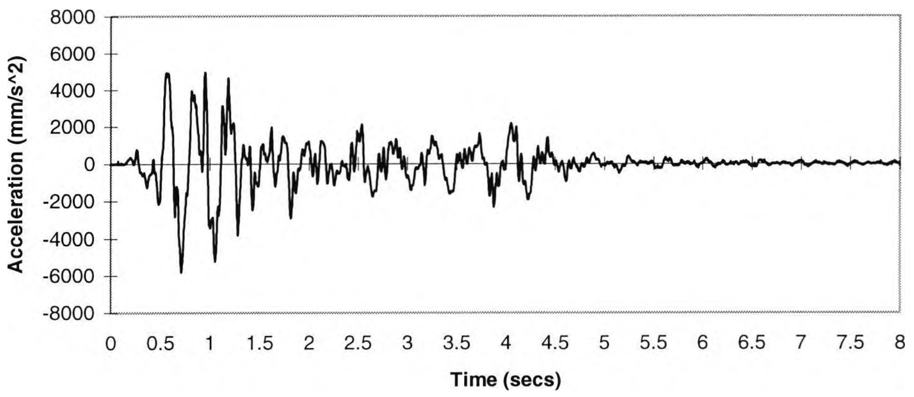


Figure H.5 *Accelerometer filtered signal*

The amplitude power spectrum of both signals are produced using the method described in section H.1. The result of the unfiltered signal in figure H.6 shows that there are significant low frequencies (i.e. around 4-6 Hz), which are in the range we are interested in, as they imply vibration of the end effector. Frequencies of around 50 Hz are also present, from the mains. High frequencies of around 180-190 Hz are also present, due to noise from the robot's motors.

Putting the same signal through the filter gives the result shown in figure H.7. The low frequencies are clearly preserved in the filtering process, as required.

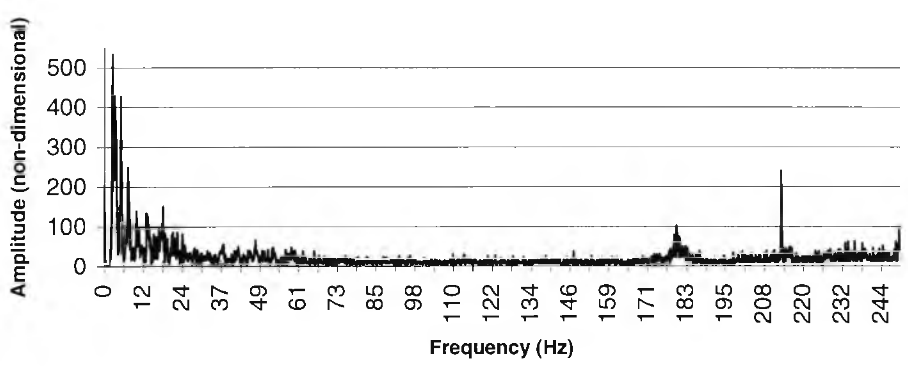


Figure H.6 *Amplitude spectrum of unfiltered signal*

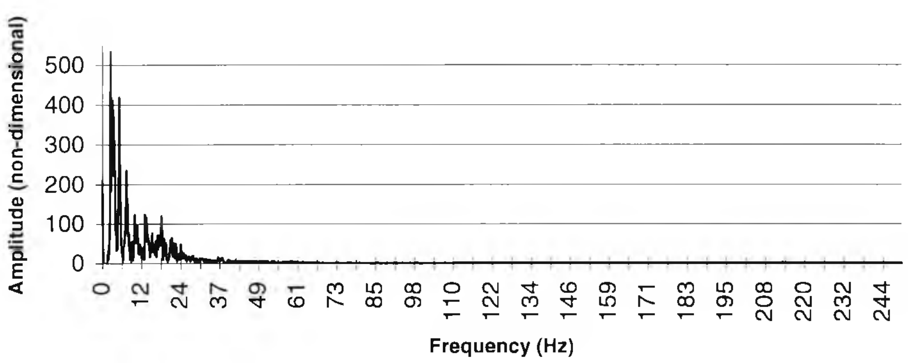


Figure H.7 *Amplitude spectrum of filtered signal*

H.2.1 Setting the sampling rate

As previously stated, each data set has 4096 values when using the *Status-30* facility. The rate of 200 Hz was found to be a reasonable sampling rate for our signal. It allowed the capturing of signals with duration of 20.48 seconds, which is enough time to capture signals for long duration moves. This sampling rate captured all the frequencies that where of interest. This is verified by comparing signals sampled at various frequencies. The filtered signals were of a move which induced high levels of vibration. The FFT of the signals were extracted using the method explained in section H.1. The first signal, sampled at a rate of 1000 Hz, is shown in figure H.8, and it's amplitude power spectrum shown in figure H.9. The second, sampled at 500 Hz, is shown in figure H.10, and it's amplitude power spectrum shown in figure H.11. And the third, sampled at 200 Hz, is shown in figure H.12, and it's amplitude power spectrum shown in figure H.13.

Normalising the amplitude spectrum of the Fourier transforms for the different scanning rates, to give figures H.14-H.16, shows the spectrum to be independent of the scanning rate. This means that there are no losses of any significant frequencies when sampling at the rate of 200 Hz (appendix A.7.3.3).

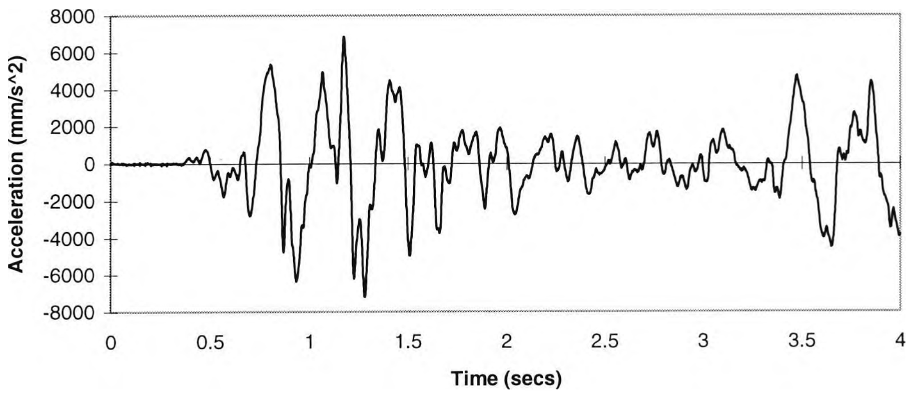


Figure H.8 *Signal sampled at 1000 Hz*

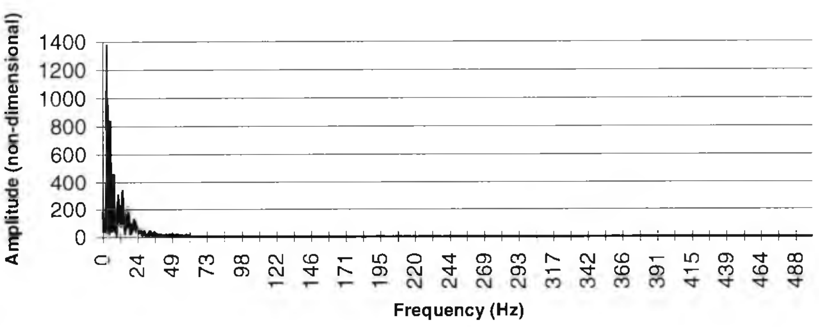


Figure H.9 *Amplitude spectrum (FFT) of signal sampled at 1000 Hz*

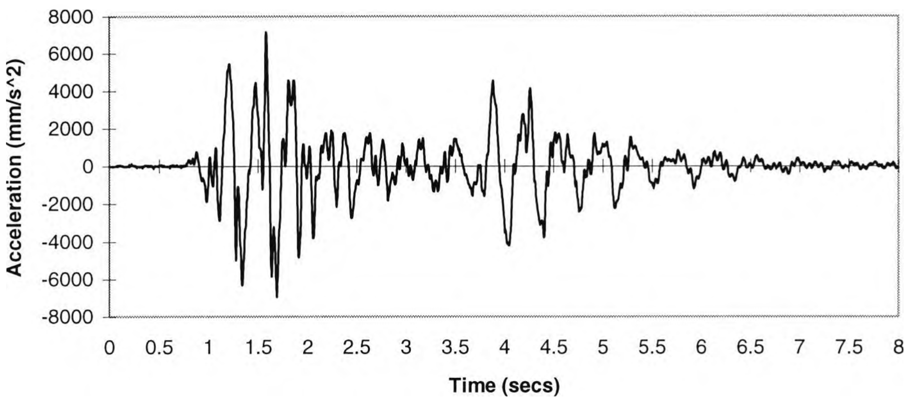


Figure H.10 *Signal sampled at 500 Hz*

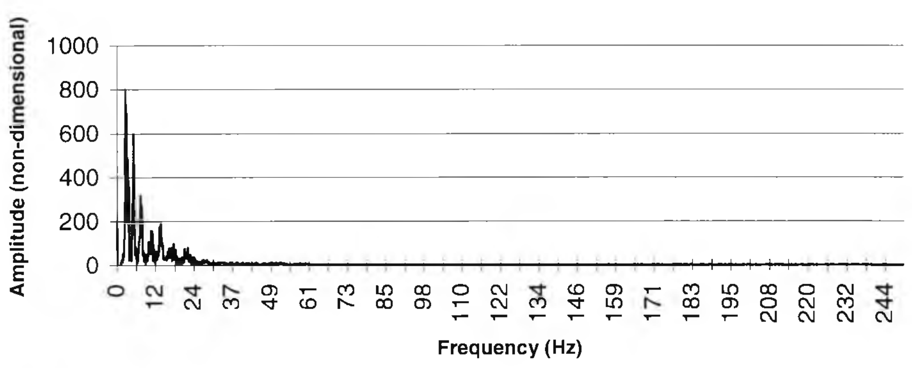


Figure H.11 *Amplitude spectrum (FFT) of signal sampled at 500 Hz*

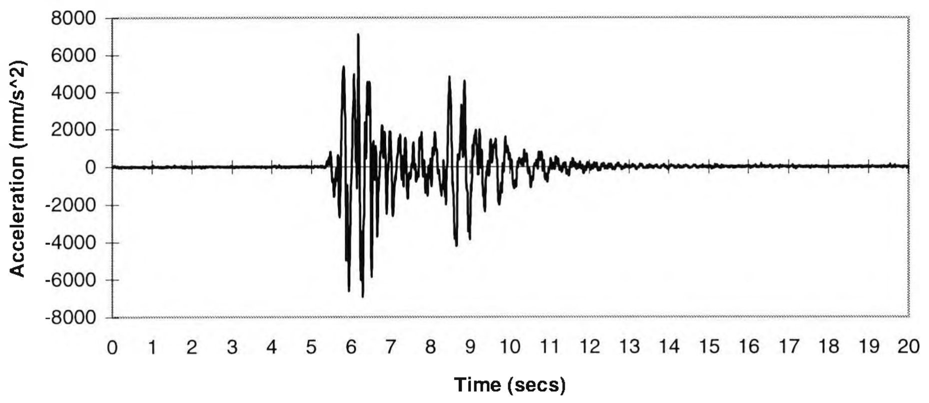


Figure H.12 *Signal sampled at 200 Hz*

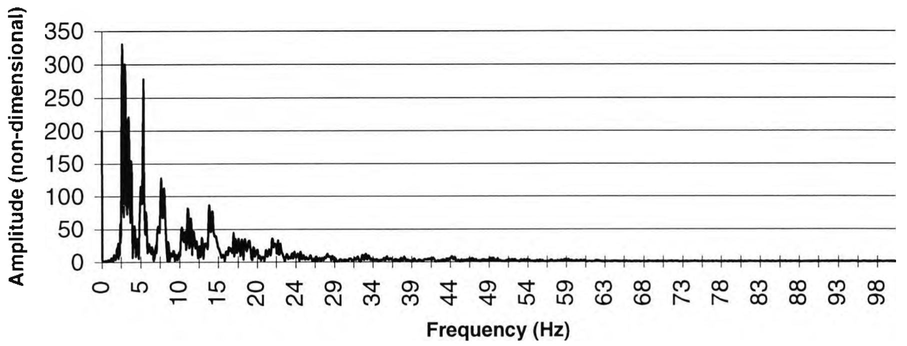


Figure H.13 *Amplitude spectrum (FFT) of signal sampled at 200 Hz*

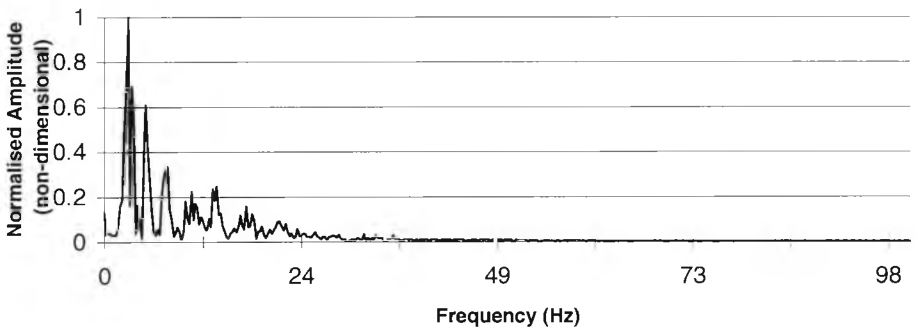


Figure H.14 *Normalised amplitude spectrum (FFT) of signal sampled at 1000 Hz*

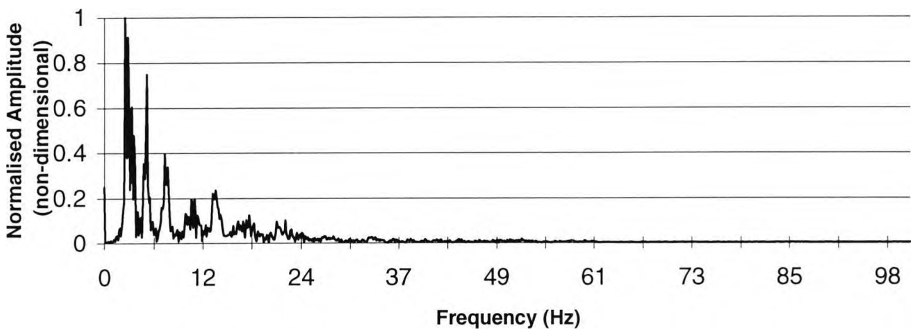


Figure H.15 *Normalised amplitude spectrum (FFT) of signal sampled at 500 Hz*

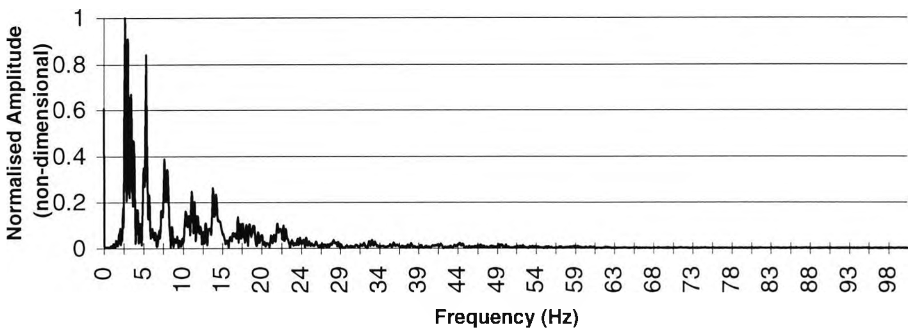


Figure H.16 *Normalised amplitude spectrum (FFT) of signal sampled at 200 Hz*

H.3 Accelerometer signal processing

The accelerometer output is in volts, and this is converted using the calibration constant determined in appendix A.7.1. Apart from acceleration of the end-effector, the displacement amplitude is of interested. To obtain approximate displacement values, a double integration process was applied to the acceleration data. A set of '*C*' code programs have been prepared and verified for this.

H.3.1 Numerical integration method

For the purpose of experimentation, the acceleration-time domain data is processed into the velocity-time domain and, subsequently, the displacement-time domain. These are approximate outcomes. They are achieved by single and double numerical integration of the acceleration-time domain data using Simpsons first rule. This approach has been adopted on account of its simplicity and accuracy (Philipson *et al.*, 1974). The two stage integration process is represented by the following:

$$V(x)_{i+2} = V(x)_i + \frac{h}{3}(f(x)_i + 4f(x)_{i+1} + f(x)_{i+2}) \qquad \text{eqn H.4}$$

Equation H.4 represents the first stage, where the data points $i = 0, 2, 4 \rightarrow N$, and $f(x)$ is acceleration, converted to $V(x)$ velocity.

$$D(x)_{j+2} = D(x)_j + \frac{h}{3}(V(x)_j + 4V(x)_{j+1} + V(x)_{j+2}) \qquad \text{eqn H.5}$$

Equation H.5 represents the second stage, where the data points $j = 0, 2, 4 \rightarrow N/2$ (as the number of data points are halved, due to the first stage integral approximation), and $V(x)$ is velocity, converted to $D(x)$ displacement.

H.3.2 Treatment of constants in numerical integration

The end-effector vibrates mainly according to a form of damped harmonic motion, with a dominant frequency. The objective is to determine the approximate velocities and displacements from the accelerometer data using numerical integration. Ideally, to achieve this, it would be necessary to identify the position of maximum or minimum acceleration, which correspond to zero velocity. by the first integration, therefore, the velocities would be the true values (i.e. a positive and negative variation about a mean position). Likewise, in order to achieve true displacement relative to a mean position, it is necessary to commence the second numerical integration from a position of maximum or minimum velocity. Where the integration process is not necessarily set at the correct starting points to give the true (approximate) velocity and displacement, an offset can occur in the resulting velocity variation, with an apparent drift in the displacements.

This is not a serious problem as these offsets and apparent drifts can be separated out, leaving the required oscillation responses. Allowing for this, gives a simple approach to the numerical integration method. This is demonstrated in the verification process.

H.3.3 Verification of integration method

Assuming the oscillation to be simple harmonic, for the purpose of clarification and verification, the integration method is tested using the following:

$$y = A sin(x) \qquad \text{eqn H.6}$$

,where A is the amplitude of the acceleration variant and x is time. By integration of equation H.6 therefore:

$$\int_{x_1}^{x_2} y \quad dx = -A\cos(x) + C \qquad \text{eqn H.7}$$

,where C is the constant of integration i.e. the correction necessary where the integration does not start from x_1 corresponding to maximum or minimum acceleration. By integration of equation H.7 therefore:

$$\int_{x_3}^{x_4} \int_{x_1}^{x_2} y \quad dx = -A\sin(x) + Cx + D \qquad \text{eqn H.8}$$

where $x_3 \rightarrow x_4$ is in the range $x_1 \rightarrow x_2$, and D likewise a constant of integration i.e. the correction for x_3 not corresponding to a position of maximum or minimum velocity. The frequency of the acceleration, velocity and displacement variations are clearly all equal and thus preserved in the integration process. In the processing of the results, the $Cx+D$ term, (in equation H.8) is evaluated and separated out from the true (approximate) displacement variation about the mean position, as demonstrated in section H.3.3.1. Where an overall drift (true signal drift) is apparent, this is eliminated using a low order polynomial fit to the means of oscillations. An example of this is shown in section H.3.5. The example of the sin wave integration demonstrated in this section, is closely related to the type of wave forms later processed.

H.3.3.1 Results of verification

A '*C*' code program '*INTSIN.C*' was developed to produce a test data file, derived using equation H.6, where A was chosen to equal 1. This is then double integrated using the numerical integration method described previously. The data file generated is shown in figure H.17. The result of the first integral shown in figure H.18, is as expected and corresponds to equation H.7, where *C* is equal to unity.

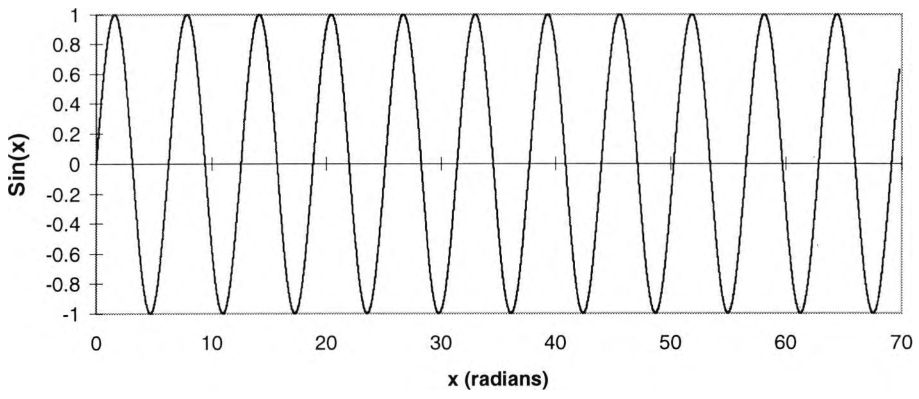


Figure H.17 *Sin wave for x*

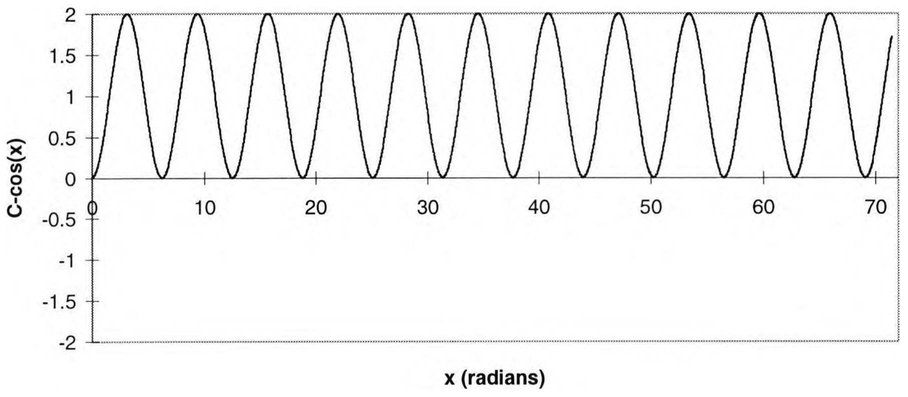


Figure H.18 *Integral of the sin wave*

The result of the double integral, shown in figure H.19, is as expected and corresponds to equation H.8. Fitting a line to the data, using a software tool called GRAPHER™, will result in equation H.9. This corresponds to the results of the integration constants *Cx+D* expected, where *C* in this case corresponds to 1 and *D* to 0.07:

$$y = 1.0x + 0.07 \qquad \text{eqn H.9}$$

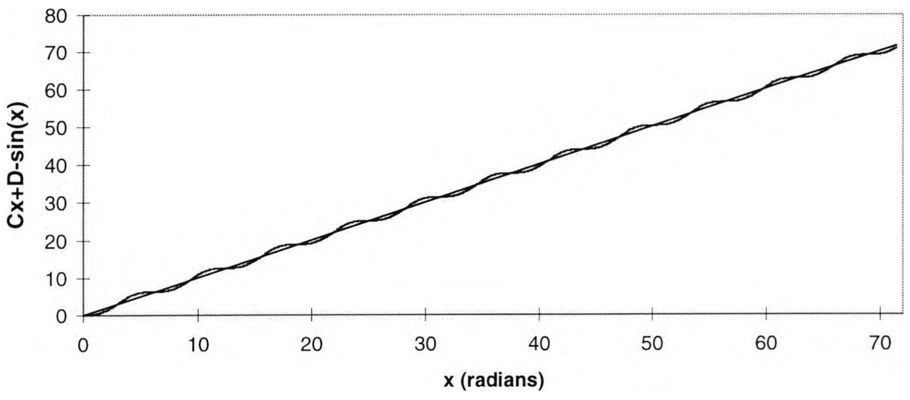


Figure H.19 *Double integral of sin wave*

Separating the *Cx+D* term determined, from the data in figure H.19, results in figure H.20. This corresponds to the double integral of equation H.6, where the oscillations are about a mean. The frequency of oscillation is preserved, as well as it's amplitude, thus verifying our integration method adopted and the '*C*' code program developed to execute it.

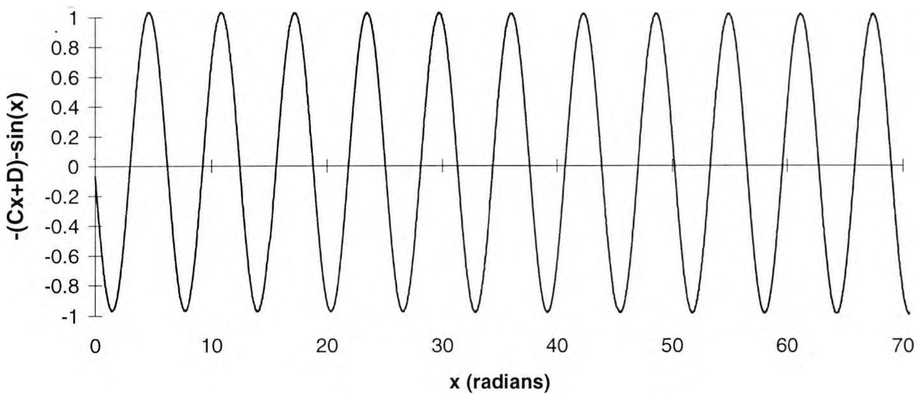


Figure H.20 *Corrected double integral of sin wave*

H.3.4 Procedure and trials with accelerometer

Using the method of integration described in section H.3.1, an algorithm was developed to process the accelerometer signal, the procedure described in this section.

Using this, and the previously discussed FFT '*C*' functions, a '*C*' code program (*INTEGRAT.C*) was developed. The procedure for examining the end-effector vibration in terms of settling time to a vibration amplitude of ±0.5 mm, is given in this section. The value of ±0.5 mm has been chosen in the settling time definition. This is demonstrated for a case where the end-effector is manually rocked, the resulting output being approximately a sin wave form, as shown in figure H.17. The actual signal is shown in figure H.21. Reference is made to the use of a laser device. This is included because the laser gave independent and direct measurements of distance, and thus provided a means of investigating the accuracy of the displacement found from the approximate numerical integration method. A description of the laser device and the experiments carried out using it are covered in section H.3.5.

Application procedure:

(i)- Read accelerometer signal from file generated using *Status-30*

(ii)- Convert signal (Volts-Time → Acceleration-Time)

(iii)- Calculate start and stop of end-effector vibration for the signal

- Threshold = |2.0 x Max_value {accel_sig[0]...accel_sig[200]}|
- Get first |accel_sig(x)| > Threshold
- Start of vibration time = Time(x)
- Start from x = no_values, decrease x until |accel_sig[x]| >Threshold
- End of vibration time = Time(x)

(iv)- Calculate duration of settling

If move is manual rocking of end-effector

- Start of settling time = Start time
- Duration of settling = End of vibration time -Start of settling time

or if move is a robot move

- Start of settling time = Start time + Move time
- Duration of settling = End of vibration time - Start of settling time

(v)- Maximum signal amplitude = Maximum { accel_sig[0]|....|accel_sig[4095]|}

(vi)- Store values from start of settling time to end of vibration

(vii)- Integrate new signal using Simpson's rule, result is (Acceleration-Time $\rightarrow$ Velocity-Time).

(viii)- Repeat step 7, result is (Velocity-Time $\rightarrow$Displacement-Time)

(ix)- If laser sensor data is captured

- Convert signal (Volts-Time $\rightarrow$ Displacement-Time)
- Store laser data corresponding to accelerometer data (as in step 6)

(x)- Actual settling time = from Start of settling time $\rightarrow$ time when displacement amplitude $\leq$0.5 mm.

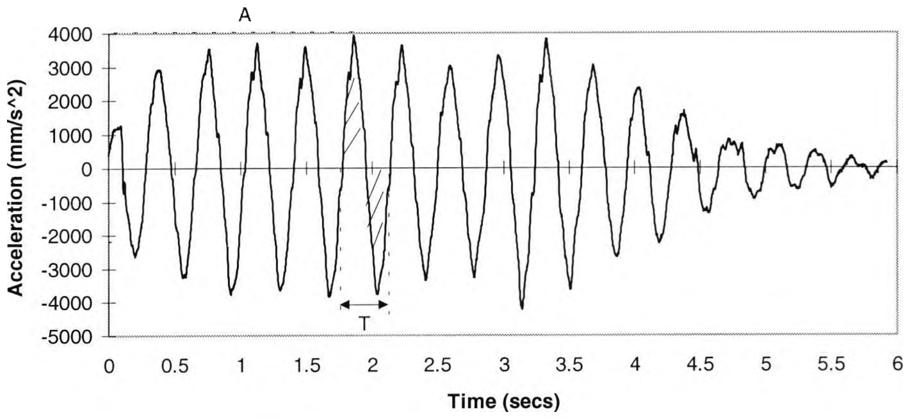


Figure H.21 *Transit acceleration of robot end-effector*

The data captured is always at a slight offset from the origin. This is adjusted using the mean of the signal derived from the FFT functions mentioned in section H.2.1.1. The result of the first integral of the signal is shown if figure H.22. This presents the velocity-time domain of the signal, with a constant of integration C (refer to equation H.7).

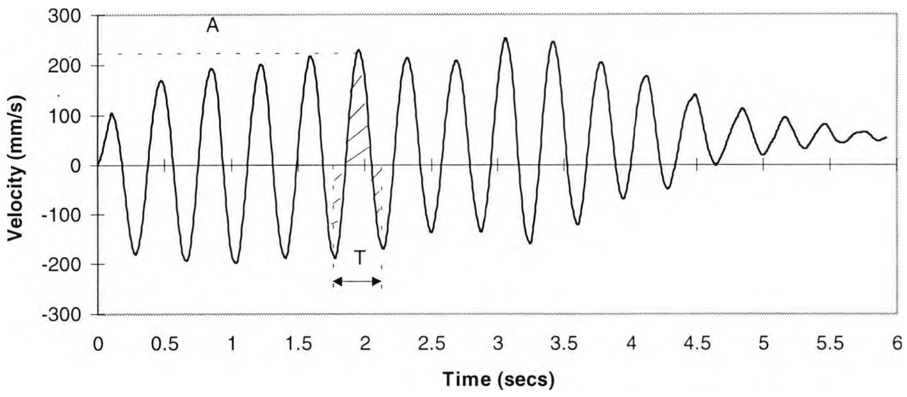


Figure H.22 *Integral of acceleration*

The result of the double integral of the sensor output signal is shown in figure H.23. This represents the estimated displacement-time domain of the signal. Constants *Ct+D* (refer to equation H.10) are neutralised, enabling the determination of the displacement and the settling time of the robot's end-effector. This is done, as shown in figure H.23, by fitting a low order polynomial curve to the position-time domain curve, using a software tool called GRAPHER™. This allows determination of the equation of a fitted polynomial curve. In this example, equation H.10 was given. The 'C' code program '*POLYCORR.C*' calculates the data points of the polynomial curve from the equation. These points are then subtracted from the position data to give the effective displacement oscillations about the mean position. The use of the low order polynomial allows the signal drift to be removed. This results in figure H.24.

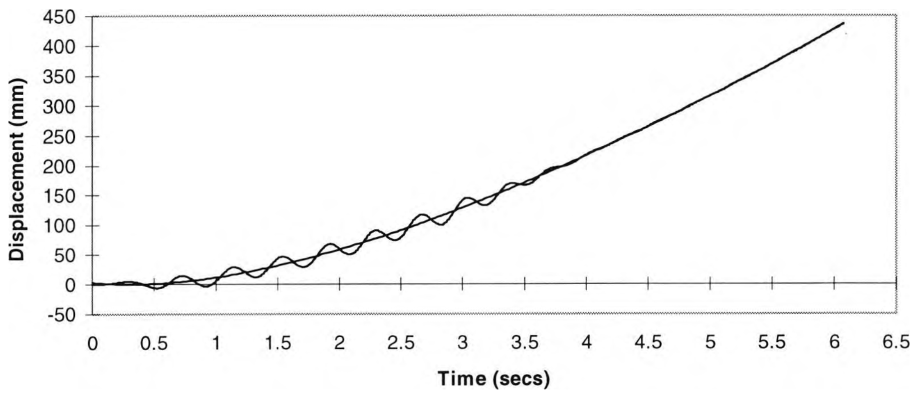


Figure H.23 *Integral of velocity*

$$f(x) = 0.002x^6 + 0.008x^5 - 0.573x^4 + 3.279x^3 + 4.935x^2 - 15.769x + 4.676 \qquad \text{eqn H.10}$$

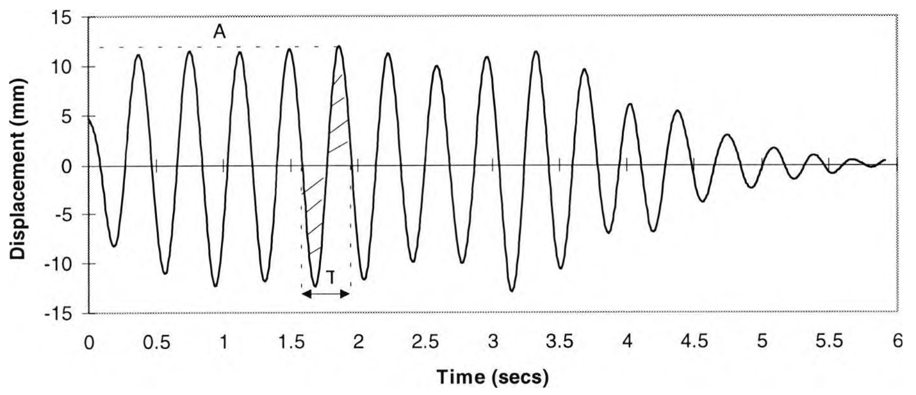


Figure H.24 *The derived displacement of the end-effector*

H.3.4.1 Accuracy analysis

The acceleration curve is predominantly of a sin wave, therefore, it is reasonable that the velocity and displacement variations are also a sin wave form. To check the accuracy of the integration result, the explicate integrals, shown in figure H.25, were checked against the velocity and displacement results using the amplitude A of the sin wave. The sections of the curve highlighted in figure H.21 is observed through the different stages of the process, highlighted in figures H.22 and H.24. The different sections are plotted separately in figure H.25 to allow them to be examined more closely. The result of each stage is compared with the theoretical result calculated using equations H.11-H.13, derived from equations H.6-8, and the percentage error is calculated:

$$d^2y/dt^2 = [-A]\sin(wt) \qquad \text{eqn H.11}$$

$$dy/dt = [A/w]\cos(wt) + C, \qquad \text{eqn H.12}$$

$$y = [A/w^2]\sin(wt) + Ct + D \qquad \text{eqn H.13}$$

From figure H.25 it can be seen that the maximum amplitude and minimum acceleration are 3946 mm/s^2 and -3776 mm/s^2 respectively, which results in an average amplitude of 3861 mm/s^2. The period T is found to be 0.362 sec. Using these values and equations H.11-13 the theoretical results of the integration were calculated. Using the angular velocity:

$$w = 2\pi/T = 17.367 \quad \text{radians/sec} \qquad \text{eqn H.14}$$

(i.e. θ=180°), and equation H.13, the theoretical amplitude of the displacement curve is calculated to be equal :

$$A/w^2 = 12.81 \text{ mm} \qquad \text{eqn H.15}$$

Determining the average amplitude from the result of the integration program, shown in figure H.25, this is 11.85 mm, resulted in a percentage discrepancy of 8%. Therefore, use of the accelerometer for displacement estimation appears to be reasonably reliable.

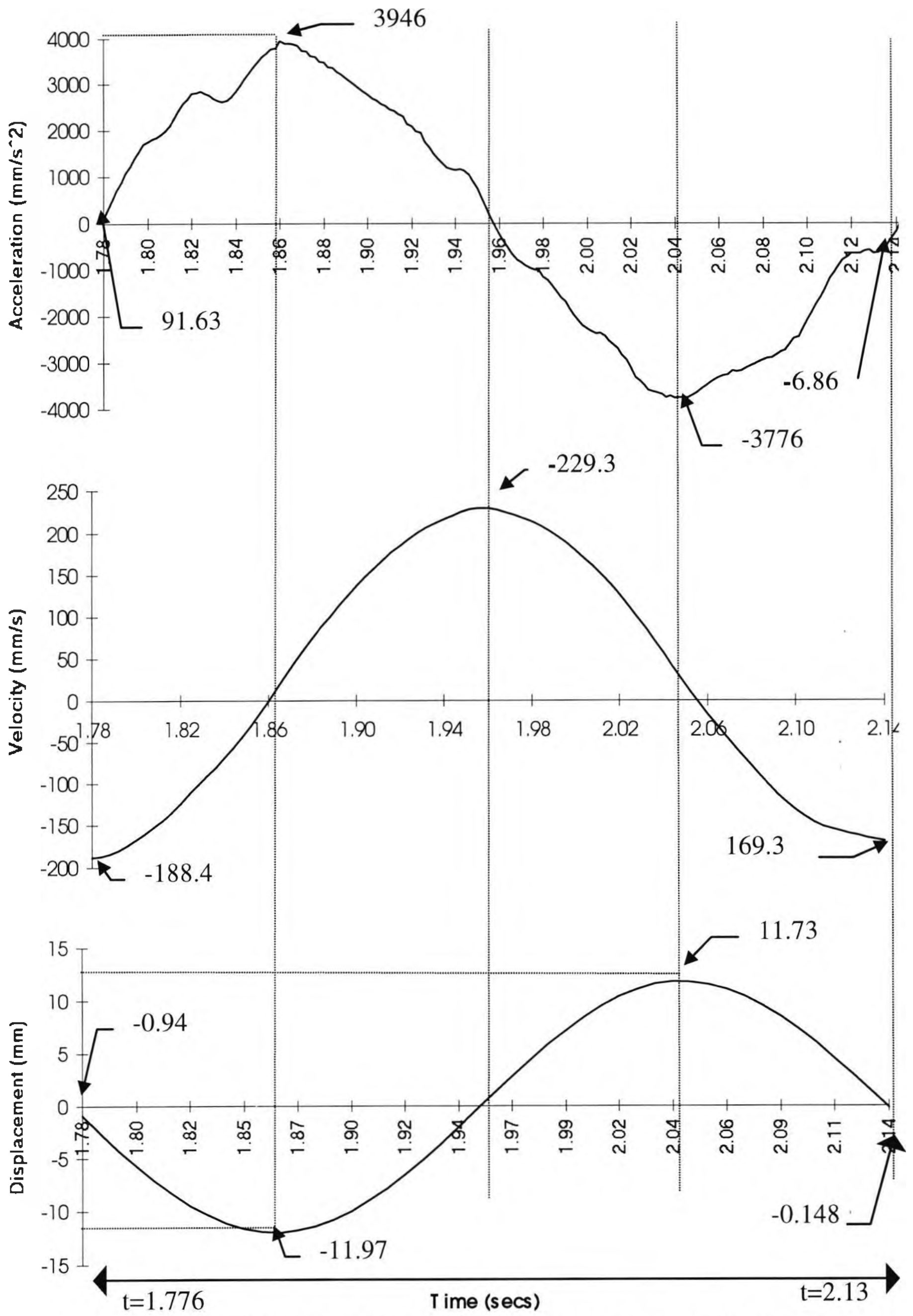


Figure H.25 *Integration process examined*

H.3.5 Verification of displacements

In this section, an independent check on the accuracy of the displacement results, described earlier, is made. This will be accomplished by comparing results of the displacement of the end-effector achieved from processing the accelerometer data with that using a laser analogue displacement sensor (LADS). Refer to section 3.7.1 an appendix A5. for a detailed description of the LADS. The signal is captured using the *PC-30* card and the application S*tatus-30*, both referred to in section H.2.

For each experiment, the signals from the accelerometer and the LADS are captured simultaneously. Two types of experiments were carried out, the first is using a signal resulting from manually rocking the end-effector, and the second using signals resulting from moves made by the robot.

H.3.5.1 Experiment set-up

The LADS was set up to point to a target, placed as shown in figure H.26. The accuracy of the laser device is ±0.01 mm. A target for the laser was mounted vertically below the accelerometer, at the furthest radial point of the end effector. This point is at an offset from the centre of rotation of the robot. The laser device and the target were parallel to each other, making sure that the laser beam hit the target at a right angle. The laser's working range is 60 mm to 140 mm. Placing the target at a distance of 100 mm from the laser device, shown in figure H.26, gives a working range of ±40 mm.

H.3.5.2 Processing the laser signal

To ensure that the effective output limits of the laser are less than the maximum input of the *PC-30* card, the laser output is scaled. For noise immunity, the laser uses a current-driven output. Even at 60 mm (or zero distance) there is a current present, as 0.496 volts. The calibration constant of the laser is 0.05 mm distance per volt of signal. (i.e. 1 mm range would generate 20 V output). So, 0.05 V/mm, gives an output, at 140 mm, of

$$((140\text{mm}-60\text{mm}) \times 0.05)+0.496 = 4.496\text{V} \qquad \text{eqn H.16}$$

The laser could not be set up at exactly 100 mm each time. However, this error is accommodated, by subtracting an *offset* from each data point in the signal output. The *offset* is calculated from the mean of the data making up part of the signal when the target is static. Thus, using equation H.17, the output '*Y*' of the signal is converted from volts to millimetres (i.e. displacement) '*X*' as follows:

$$X = 60mm + ((Y - 0.496) / 0.05) - offset \qquad \text{eqn H.17}$$

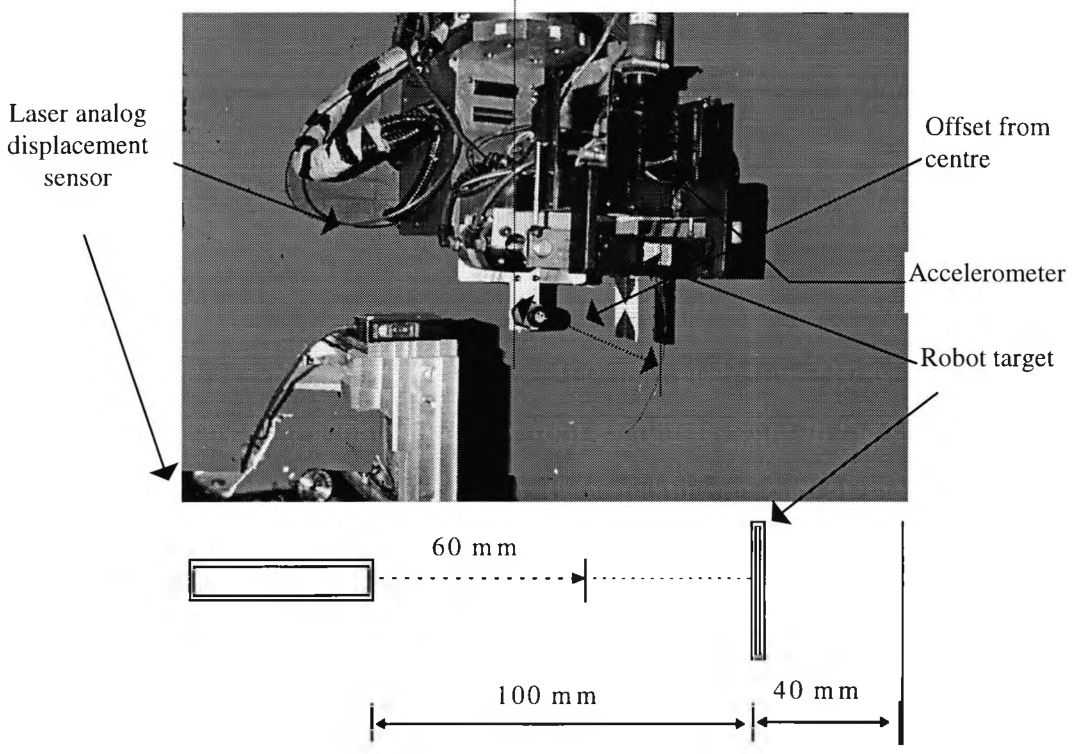


Figure H.26 *Analog displacement experiment set-up*

H.3.5.3 Analysing and comparing the signals

The signals from the LADS and the accelerometer are shown in figure H.27. They are analysed, and the difference in the results calculated, as shown in figure H.28. This is done by comparing the average displacement amplitudes d_{ave} of each signal, where:

$$d_{ave} = (|A| + |B|)/2 \qquad \text{eqn H.18}$$

at time

$$T(d_{ave}) = (T + t / 2), \qquad \text{eqn H.19}$$

as shown in figure H.28. The difference in the resulting d_{aves} are then plotted, as shown in figure H.29.

H.3.5.4 Manual rocking motion experiment

The robot end-effector was rocked manually, the resulting signal shown in figure H.27. Figure H.28 shows the displacement results from the accelerometer signal and the laser signal, plotted together. Figure H.29 shows the differences between the average amplitudes of the two signals. A maximum difference of 0.41 mm was found, and an average difference of 0.18 mm, which shows that it is satisfactory to use the accelerometer output to estimate displacements.

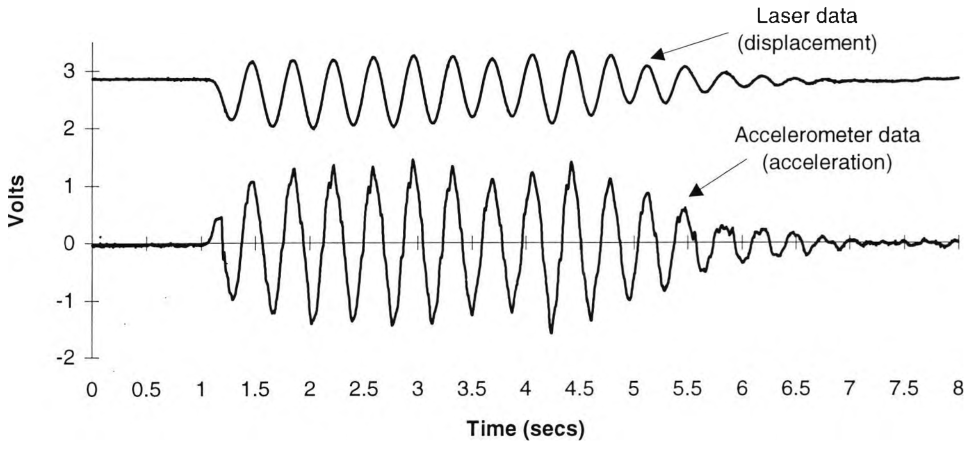


Figure H.27 *Raw signals : accelerometer and laser beam*

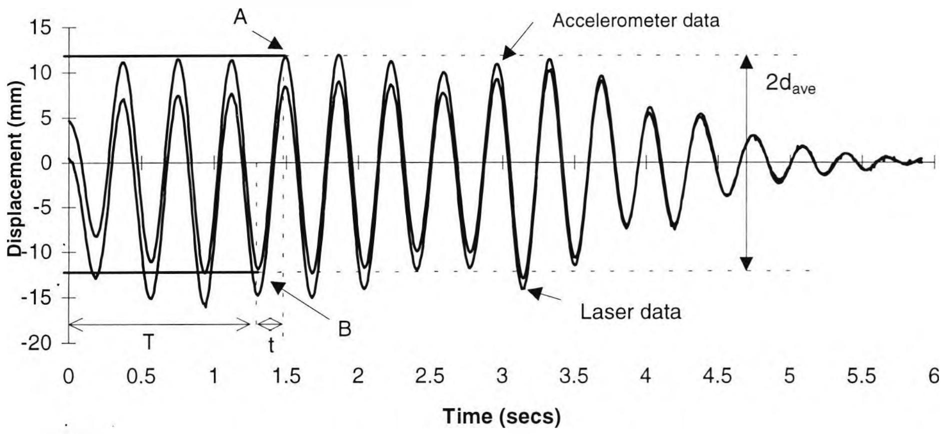


Figure H.28 *Laser and accelerometer displacements*

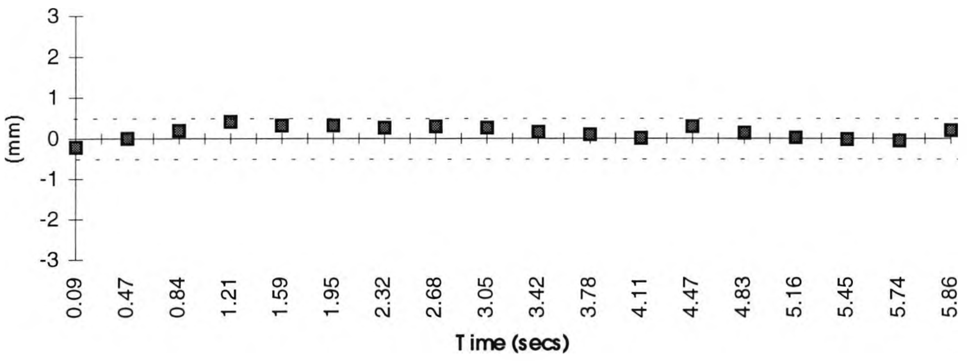


Figure H.29 *Difference in average displacement amplitude of accelerometer and laser results*

H.3.5.5 Move experiments

Three different types of moves were made by the robot in order to study the effectiveness of the accelerometer in displacement estimation. Each time, the LADS was set up to enable an independent check on the settling time and displacement results, both concluded from accelerometer data. Figure H.26 shows the arrangement for this.

H.3.5.5.1 Strong vibration move

A linear move covering a distance of 1 meter, in 4.053 secs, with $t_{accel/decel}$ of 0.146 secs, was performed on the x-axis of the robot. The Yaw-axis was at 90° angle. The maximum velocity V_{max} of the move, calculated using equation 4.1, was 246.7 mm/s, and the maximum acceleration A_{max}, calculated using equation 4.2, 1689 mm/s^2. This is an example of an extremely strong vibration inducing move, largely due to the high acceleration value reached during the move. The data captured, as a result of the move, is shown in figure H.30. Processing the data, results in the end-effector displacements shown in figure H.31, with a settling time of around 5 secs. Comparing the displacement results using the LADS, with those from the accelerometer, it is apparent that they are extremely close, with a maximum difference of 0.83 mm and an average difference of 0.19 mm, as shown in figure H.34.

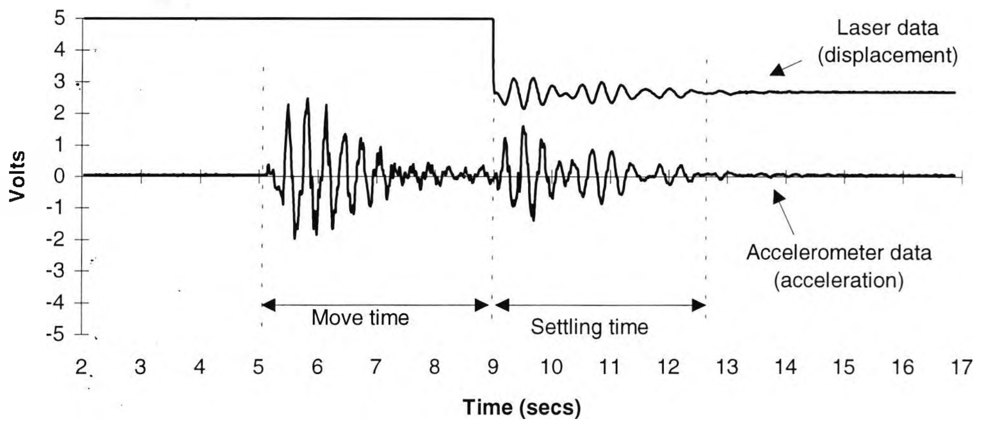
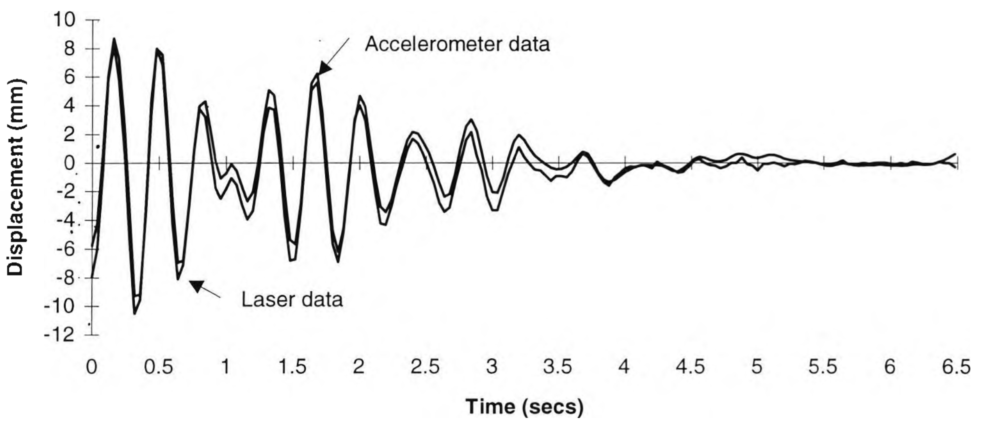


Figure H.30 *Raw signals : accelerometer and laser beam*



H.31 *Laser and accelerometer displacements*

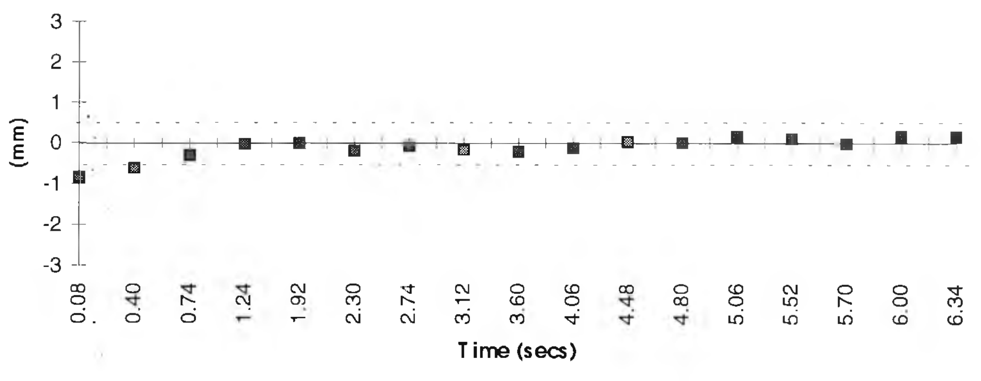


Figure H.32 *Difference in average displacement amplitude of accelerometer and laser results*

H.3.5.5.2 Medium vibration move

A linear move covering a distance of 1 meter, in 5.175 seconds, with $t_{accel/decel}$ of 0.29 seconds, was performed on the x-axis of the Robot. The yaw-axis was at a 90° angle. The maximum velocity of the move, calculated using equation 4.1, was V_{max} = 193 mm/s, and the maximum acceleration A_{max}, calculated using equation 4.2, was 665.5 mm/s^2. This is an example of a medium vibration inducing move, due to the average acceleration and velocity values reached during the move. The data captured, as a result of the move, is shown in figure H.33. Processing the data, results in the end-effector displacement shown in figure H.34, with a settling time of approximately 1.8 secs. Comparing the displacement results from the LADS with those from the accelerometer, it is apparent that they are extremely close, with a maximum difference of 0.77 mm and an average difference of 0.21 mm, as shown in figure H.35.

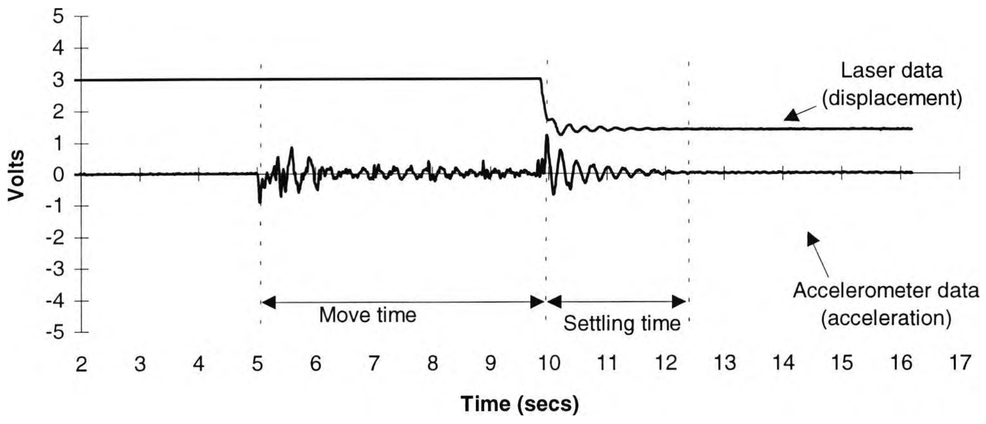


Figure H.33 *Raw signals : accelerometer and laser beam*

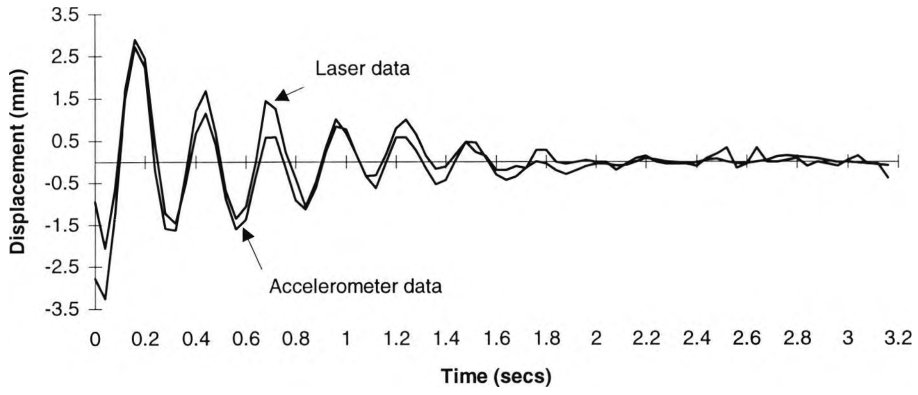


Figure H.34 *Laser and accelerometer displacements*

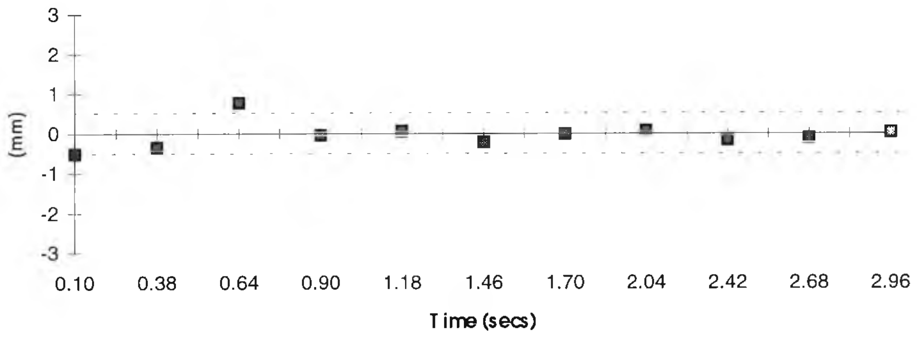


Figure H.35 *Difference in average displacement amplitude of accelerometer and laser results*

H.3.5.5.3 Low vibration move

A linear move covering a distance of 1 meter, in 7.52 seconds, with $t_{accel/decel}$ of 0.68 seconds, was performed on the x-axis of the Robot. The yaw-axis was at a 90° angle. The maximum velocity of the move, calculated using equation 4.1, was V_{max} = 133 mm/s, and the maximum acceleration A_{max}, calculated using equation 4.2 was 195 mm/s^2. This is an example of a low vibration inducing move, due to the low acceleration value reached during the move and the large move time, which resulted in a low V_{max}. The data captured, as a result of the move, is shown in figure H.36. Processing the data, results in the end-effector displacement shown in figure H.37, with a settling time of less than 1.8 secs. Comparing the displacement results from the LADS, with those from the accelerometer, it can be seen that they are extremely close, with a maximum difference of 0.8 mm and an average difference of 0.13 mm, as shown in figure H.38.

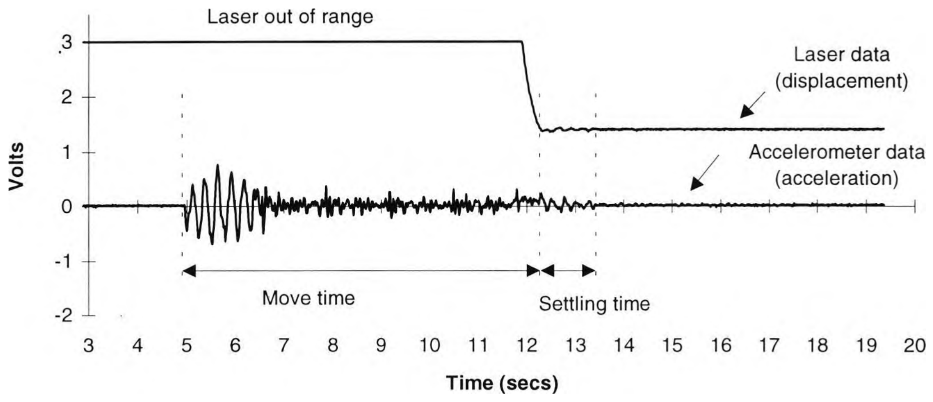


Figure H.36 *Raw signals : accelerometer and laser beam*

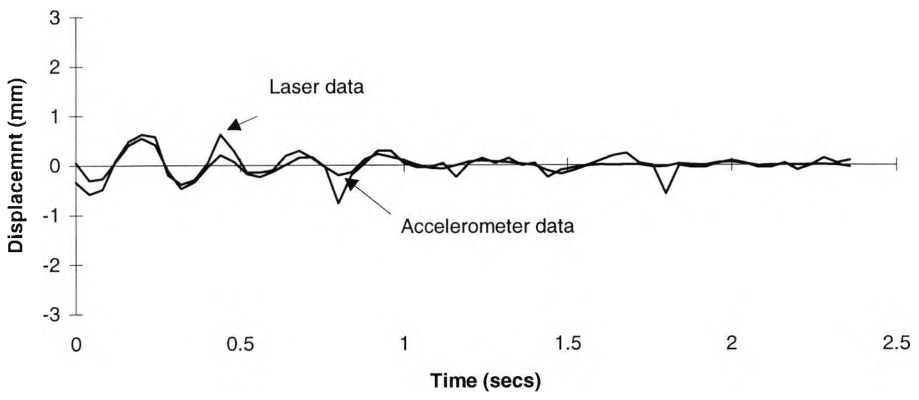


Figure H.37 *Laser and accelerometer displacement*

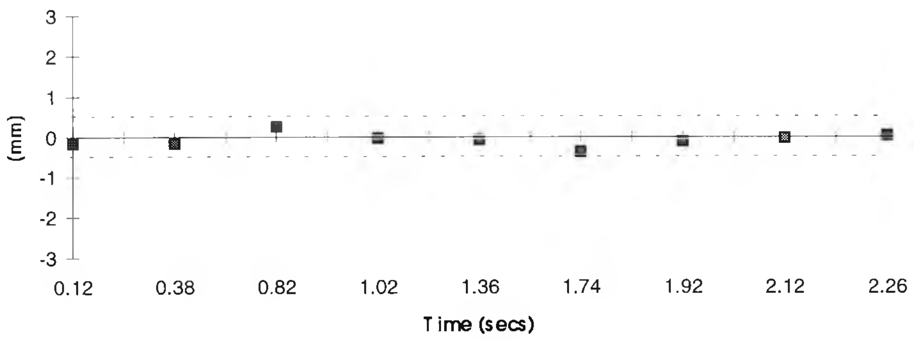


Figure H.38 *Difference in average displacement amplitude of accelerometer and laser results*

Appendix I: Robot Path Planning

I.1 Move description theory

The mathematical basis of each move type is given below. Generally, we will be working with velocity-vs-time graphs to describe these moves. On these graphs, the change in position is the area under the curve, and the acceleration is the slope of the curve at any one point in time.

I.2 Linear velocity moves

Linear velocity variations and associated constant acceleration are employed in the operation of many industrial robots, particularly older ones. This is, however, undesirable for the reasons previously stated in chapter 3.9.

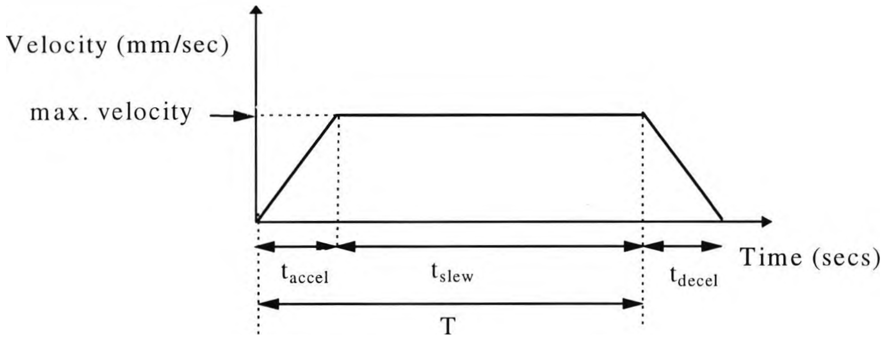


Figure I.1 *Linear move*

Referring to graph I.1, a standard linear point-to-point move of time 'T', with acceleration and deceleration times 't', is broken down into three concatenated parts: accel, slew, decel. For the move shown in figure I.1 we have:

$$\Delta P = 1/2 \times (2t_{accel\,/\,decel} \times V_{\max}) + V_{\max}(T - t_{accel\,/\,decel}) \qquad \text{eqn. I.1}$$

$$A_{\max} = V_{\max} / t_{accle\,/\,decel} \qquad \text{eqn. I.2}$$

were $t_{accel} = t_{decel}$.

I.3 Parabolic velocity moves

The parabolic velocity type move has greater flexibility for blending moves, smoothing starting and stopping.

Referring to figure I.2, for a single axis, the variation of distance P with time t is given

by: $S = At^3 + Bt^2 + Ct + D$ eqn. I.3

, where A, B, C and D are constants. Differentiating equation 3.3 with respect to t yields the parabolic velocity function given by:

$ds/dt = 3At^2 + 2Bt + C$ eqn. I.4

Differentiating again with respect to t yields the anticipated linear acceleration function given by:

$d^2s/dt^2 = 6At + 2B$ enq. I.5

The constant A, B, C and D in equations I.3-I.5 are determined according to the position (P), velocity (V) and acceleration (a) boundary conditions at the start and finish of a move as illustrated in figure I.2. In the case of the so called 'S' type moves described in section 3.5, blending is achieved by matching the velocities at the blending point. The minimum time moves derived below are on the basis of zero start and finish velocities. The means for achieving more elaborate paths which allow 'on the fly' activity, are discussed later.

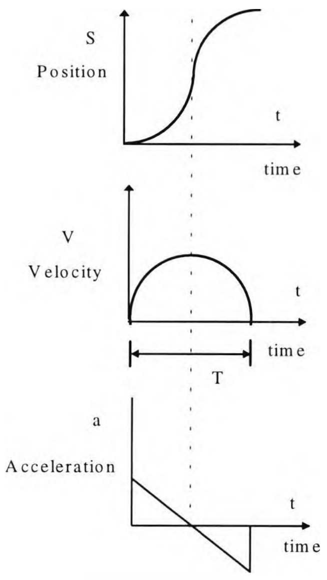


Figure I.2 *Parabolic move profiles*

I.3.1 Parabolic 'U' move

Figure I.2 shows the P, V and a variations for a so called 'U' type move which is, completed in time T. For such moves there are two distinct cases of boundary conditions, these corresponding to peak acceleration moves and peak velocity moves in respect to the system capability. A move which achieves both peaks is also of interest and will be considered later.

For the first case, we have at t = 0: S = 0, V = 0 and a = A_{peak} with the distance P covered, in time T. Substituting these values in equations I.3-I.5, the constants are determined as follows:

$$A = -A_{peak}/3\times T, \quad B = A_{peak}/2, \quad \text{and} \quad C = D = 0$$

Putting these values into equation I.3, setting t = T, S = P and rearranging we arrive at the expression:

$$T = \sqrt{(6\times P)/A_{peak}} \qquad \text{eqn. I.6}$$

The maximum velocity reached at t = $T/2$ must be less than V_{peak}, thus equation I.6 holds for the condition:

$$T < (4\times V_{peak})/A_{peak} \qquad \text{eqn. I.7}$$

In the second case we have at t = 0: V = 0 and t = $T/2$: V = V_{peak} and a = 0 with the distance covered P in time T. Substituting these values in equations I.3-I.5 the constants are determined as:

$$A = (4\times V_{peak})/(3\times T^2),$$

$$B = (2\times V_{peak})/T \quad \text{and}$$

$$C = D = 0$$

Putting these values into equation I.3 and setting t = T, S = P and rearranging we arrive at the expression:

$$T = (3\times P)/(2\times V_{peak}) \qquad \text{eqn. I.8}$$

Considering that this move should not require the system to produce peak acceleration at t = 0, equation I.7 is subject to the condition:

$$T > (4\times V_{peak})/A_{peak} \qquad \text{eqn. I.9}$$

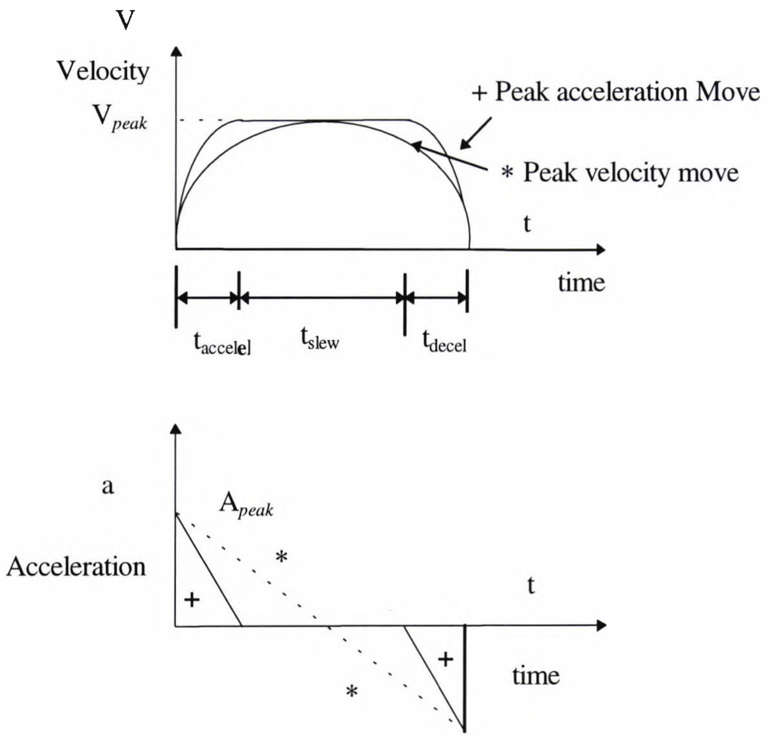


Figure I.3 V_{peak} *and* A_{peak} *parabolic moves*

Taking the two peak values to be numerically of the same order, equation I.6 and I.8 indicate that peak acceleration is appropriate for short moves lasting up to a few seconds and peak velocity for longer moves. However, the peak velocity move does not provide the shortest move time for large moves as can be seen in figure I.3. Recalling that the area under the velocity-time curve represents the distance covered, there are clear advantages in operating long moves with three time segments. Over time t_{accel} a peak acceleration move is made to peak velocity with the move continuing at peak velocity over time t_{slew} where after the velocity is reduced to zero over a further time t_{decel}. For distance move of P, the move time is determined as follows:

For a move of type shown in figure I.4, where V_{peak} and A_{peak} are achieved, equation I.6 and I.8 have the same T and P values, we have:

P1 (half distance travelled) = $P/2 = (4 \times V_{peak}^2)/(3 \times A_{peak})$ eqn. I.10

or $P = (2/3) \times V_{peak} \times T$ eqn. I.11

and $A_{peak} = (6 \times P)/T^2$ eqn. I.12

and substituting P from equation I.10 in equation I.6

$$t_{accel/decel} = T/2 = (2 \times V_{peak})/A_{peak} \qquad \text{eqn. I.13}$$

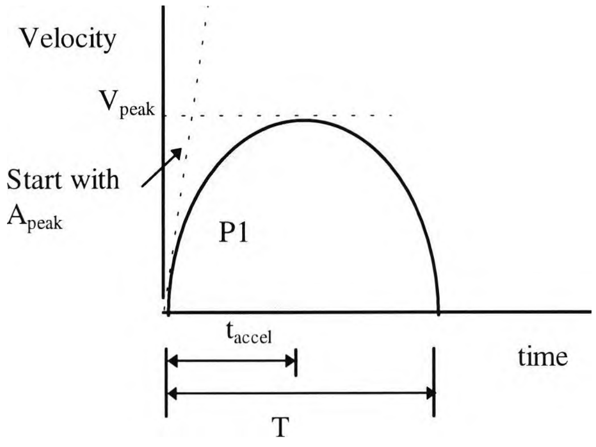


Figure I.4 *Parabolic 'U' move*

Equations I.10-I.13 can be used to solve for whichever term is the unknown when calculating a minimum-time move of a certain length, taking into account the system constraints of maximum velocity and/or acceleration. In this case P, V_{max}, and A_{max} would be known, and the equation solved for T. If both V_{peak} and A_{peak} were constraints, the larger of the two values calculated for T would have to be used, so as not to violate the other constraint.

Considering a move P therefore, the time for this is given by:

$$t_{tot} = ((P - 2 \times P1)/V_{peak}) + (2 \times t_{accel}) \qquad \text{eqn. I.14}$$

, where P1 is the distance travelled in the time interval t=0 to t=t_{accel} , as seen in figure I.4.

This type of move is important because it causes minimum energy dissipation in the motor (for covering a given distance in a given time). The power dissipated in the motor at any moment is proportional to the square of the current, and thus to the square of the acceleration. The parabolic profile, by providing the maximum acceleration at the lowest velocities, enables the bulk of the distance to be covered at low values of acceleration and therefore, low power dissipation.

I.3.2 Segmented parabolic move

In general, any arbitrary move profile can be described using a series of parabolic commands. The only restriction is that a ‘t’ time of at least 40 ms be used. This is due to the fact that SMCC requires approximately 20 ms per axis (two axes per card) to calculate the profile parameters for the next move segments. If ‘t’ is close to the minimum value, communications can take up enough time to cause the SMCC card to ‘hang up’. However, if ‘t’ is sufficiently large, this is not a concern.

The way to calculate these type of moves, is to decide on the best type of curves to be used for starting and stopping. For this, the situation to be solved is that of acceleration/deceleration to/from peak velocity in a parabolic move segment. A whole family of curves can perform this task, depending on what value should be maximised or minimised. The three most useful of these curves are shown in figure I.5, I.6, and I.7 below.

I.3.2.1 Parabolic velocity with maximum negative inflection

For the parabolic case with the maximum negative inflection, as shown in figure I.5, Δp and A_{max} are as follows:

$$\Delta p = (1/3) \times V_1 \times \Delta t \qquad \text{eqn. I.15}$$

or, $$\Delta t = 3 \times \Delta p / V_1 \qquad \text{eqn. I.16}$$

and $$A_{max} = a(\Delta t) = 2 \times V_1 / \Delta t \qquad \text{eqn. I.17}$$

or, $$\Delta t = 2 \times V_1 / A_{max} \qquad \text{eqn. I.18}$$

The curve in figure 3.5 shows the most gentle start-up possible. The equations above show the relations between distance Δp, end velocity (V_1), maximum acceleration, and time Δt. Typically, the time would be calculated from the desired end velocity and the constraint of maximum acceleration, from that the distnace would be calculated.

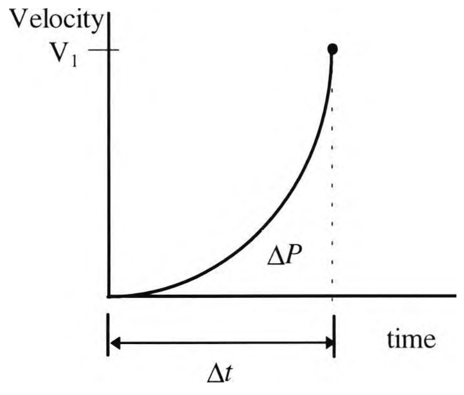


Figure I.5 *Linear acceleration: maximum negative inflection*

I.3.2.2 Linear velocity

For constant acceleration, as shown in figure I.6, Δp and A_{max} are as follows:

$$\Delta p = 1/2 \times V_1 \times \Delta t \qquad \text{eqn. I.19}$$

or $$\Delta t = 2 \times \Delta p / V_1 \qquad \text{eqn. I.20}$$

and $$A_{\max} = A_{avg} = V_1 / \Delta t \qquad \text{eqn. I.21}$$

or $$\Delta t = V_1 / A_{\max} \qquad \text{eqn. I.22}$$

Figure I.6 shows the case of constant acceleration, which is the minimum-time curve if maximum acceleration is the only constraint. This move puts a lot of strain on the equipment because of the instantaneous change to maximum acceleration.

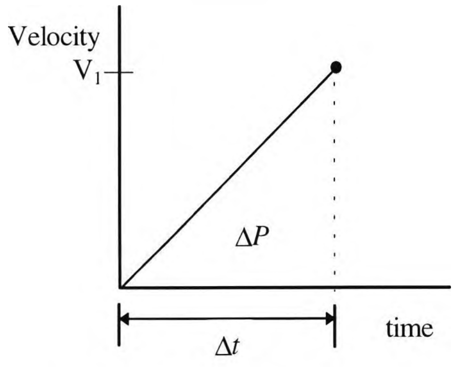


Figure I.6 *Constant acceleration*

I.3.2.3 Parabolic velocity with maximum positive inflection

For the parabolic case with the maximum positive inflection, as shown in figure I.7, Δp and A_{peak} are as follows:

$$\Delta p = (2/3) \times V_1 \times \Delta t \qquad \text{eqn. I.23}$$

or, $$\Delta t = (3/2) \times \Delta p / V_1 \qquad \text{eqn. I.24}$$

and $$A_{max} = a(0) = 2 \times V_1 / \Delta t \qquad \text{eqn. I.25}$$

or, $$\Delta t = 2 \times V_1 / A_{max} \qquad \text{eqn. I.26}$$

I.3.2.4 Blending the segments

The curve in figure I.7 shows the case of the parabolic velocity that blends most smoothly into a steady velocity move segment. The accompanying equations are identical in form to those in figure I.5 and I.6.

The next move, after the starting move segment will have a starting velocity of V_1 . The end velocity will be varied according to the distance of the move and the time required for the move. If we wish to add to the above acceleration segments a constant speed segment of Δp at a velocity of V_1, as shown in figure I.7, then the time for that segment will be calculated as follows:

$$\Delta t = \Delta p / V_1 \qquad \text{eqn. I.27}$$

The end velocity for that segment will be equal to V_1. If we wish to make another move that blends with that move (i.e. have a starting velocity equal to V_1), but have a different end velocity, then the time for that segment will be calculated as follows:

if $V_{final} < V_1$ or $V_{final} > V_1$

then, $$\Delta t = (3 \times \Delta p) / (V_{final} + 2V_1) \qquad \text{eqn. I.28}$$

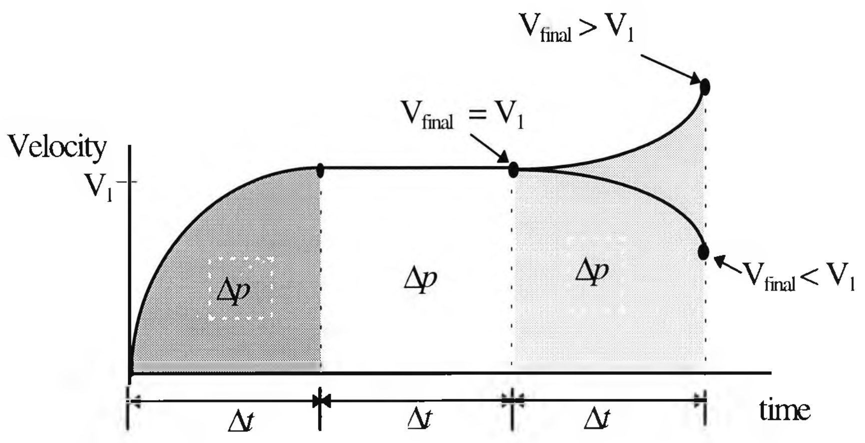


Figure I.7 *Parabolic velocity: maximum positive inflection at start, with two blended moves*

I.3.2.5 Velocity check during segments

With the general case of a parabolic move segment, it is important to check that at any point during the segment that the system capability is not exceeded. The following analysis can be used to determine whether the velocity profile, during a parabolic move, will generate a minimum or a maximum velocity which is outside the range of beginning and end velocities, and if so, the magnitude for the extreme velocity.

Figure I.8 shows a general parabolic move segment where V_s is the starting velocity of the segment, V_f the final velocity, V_m the mean velocity and $V_{extreme}$ is the extreme velocity reached in that segment.

First X, the time of extreme velocity is calculated as a fraction of the time for the move:

$$X = \frac{3V_m - 2V_s - V_f}{6V_m - 3V_s - 3V_f} \qquad \text{eqn. I.29}$$

If $0 \leq X \leq +1$, then extreme velocity occurs during the segment. If X is outside of this range, the extreme would not occur on a section of the parabolic that is actually executed, and one of the end velocities would be the extreme velocity for the segment.

If X is found to be $0 \leq X \leq +1$, the next step is to calculate the extreme velocity which is given as:

$$V_{extreme} = V_s + \frac{(3V_m - 2V_s - V_f)^2}{6V_m - 3V_s - 3V_f} = V_s + (3V_m - 2V_s - V_f) \times X \qquad \text{eqn I.30}$$

This velocity would then be compared to system capabilities.

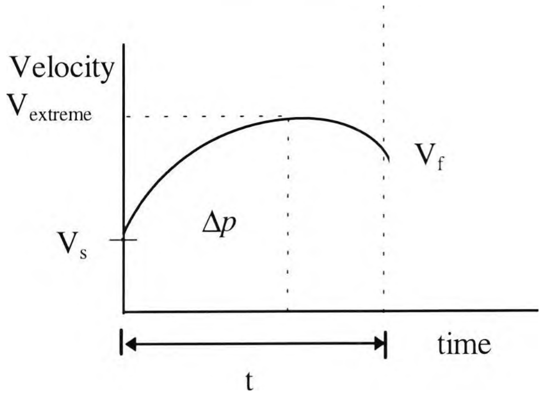


Figure I.8 *General parabolic move segment*

I.4 The S-curve move

Whilst the 'U' parabolic move is superior to the liner move, which is available with most robots, it does involve the application of peak acceleration at the start and finish of moves. A further improvement can be achieved by which motor power is built up at the start of the move and reduced to zero at the finish of the move. Such a move is the previously denoted 'S' move. The general form of a zero start to zero finish velocity move, incorporating 'S' move segments, is shown in figure I.9. Five distinct segments are apparent with this move, these being an increasing acceleration move to peak acceleration over t_1, a decreasing acceleration move to peak velocity over t_2, a slew move at peak velocity over t_3 and symmetrical moves towards the final zero velocity over time steps t_1 and t_2.

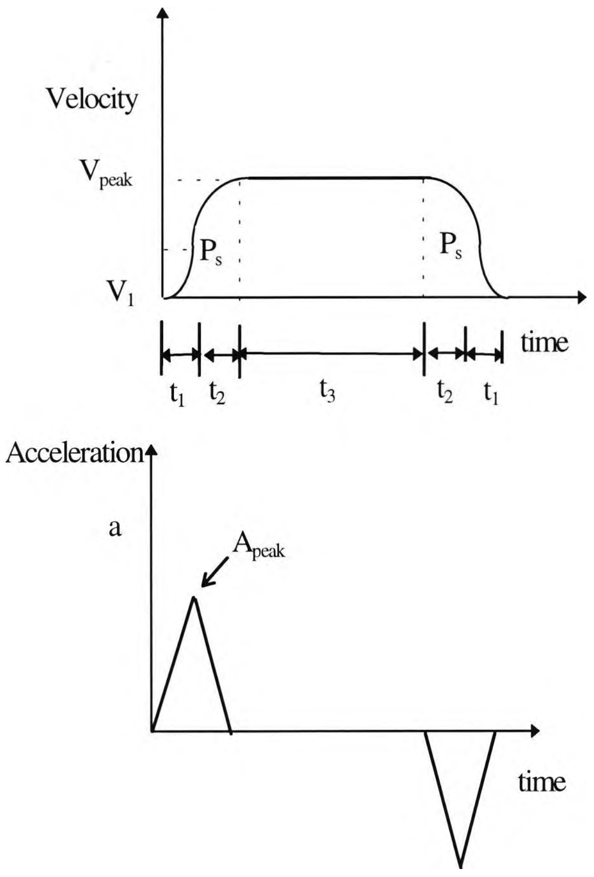


Figure I.9 *Three part S-curve move*

In what follows, the time for the rise to peak velocity is held constant. However, the distance covered varies with the selected time at which the peak acceleration is achieved. Figure I.11 shows an example of choosing the time of reaching peak acceleration, to be half of the total time of the rising portion of the s-curve. The selection of this point is a matter of the required performance in respect to possible motion induce vibration. A short move not requiring the t_3 segment is dealt with and then the longer move which requires it. The former is illustrated in figure I.10.

Considering the t_l segment we have at t = 0: S = V = a = 0 and at t = t_l: V = V_l and a = A_{peak}, where V_l is the blend velocity. Substituting these values in equations I.3-I.5 the constants are determined as:

$$A = A_{peak}/(6\times t_1) \quad \text{and} \quad B = C = D = 0$$

From equations I.4 and I.3 we have therefore:

$$t_l = 2\times V_l / A_{peak} \qquad \text{eqn. I.31}$$

and $P_1 = (1 \times V_1 \times t_1)/3$ eqn. I.32

where P_1 is the distance covered in t_1.

Considering the t_2 segment we have at t = 0 (start end t_1): S = 0, V = V_1 and a = A_{peak}

t = t_2: V = V_{peak} and a = 0

Substituting these values in equations I.3-I.5 the constants are determines as:

$$A = -A_{peak}/(6 \times t_2)$$

$$B = A_{peak}/2, C = V_1 \text{ and } D = 0$$

putting these values into equation I.4 and using V = V_{peak} at t = t_2 :

$$t_2 = 2 \times (V_{peak} - V_1)/A_{peak} \qquad \text{eqn. I.33}$$

and using the area under the velocity curve over t_2 :

$$P_2 = t_2 \times ((2 \times V_{peak} + V_1)/3) \qquad \text{eqn. I.34}$$

where P_2 is the distance covered in time t_2.

Adding equations I.31 and I.33 we arrive at the time constant T_k for the rising portion of the 'S' move:

$$T_k = 2 \times V_{peak}/A_{peak} \qquad \text{eqn. I.35}$$

and the total distance covered during T_k :

$$P_s = 2 \times V_{peak}^2 \times ((2 - N)/(3 \times A_{peak})) \qquad \text{eqn. I.36}$$

where $N = V1/V_{peak}$ and $0 < N < 1.0$ eqn. I.37

N can be varied to influence the end of move vibration

For a move greater than 2 x P_s as shows in figure I.9, the time can be found as follows:

$$t_{tot} = \frac{P_{tot} - 2 \times P_s}{V_{peak}} + 2 \times \frac{2 \times V_{peak}}{A_{peak}} \qquad \text{eqn. I.38}$$

For a move of less than 2 x P_s the V_{peak} value will not be reached. Such moves can be determines by using N = $V1/V_{max}$ in equation 3.36, together with V_{max} rather than V_{peak}.

Using N = 1/2, equation I.36 gives:

$$V_{max} = \sqrt{P_s \times A_{peak}} \qquad \text{eqn. I.39}$$

leading to:

$$T = 2 \times \sqrt{P_s / A_{peak}} \qquad \text{eqn. I.40}$$

the move time.

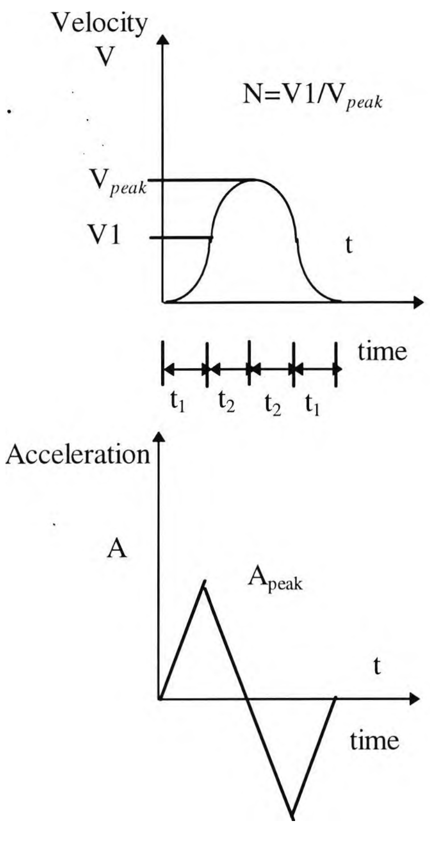


Figure I.10 *Single part S-curve move*

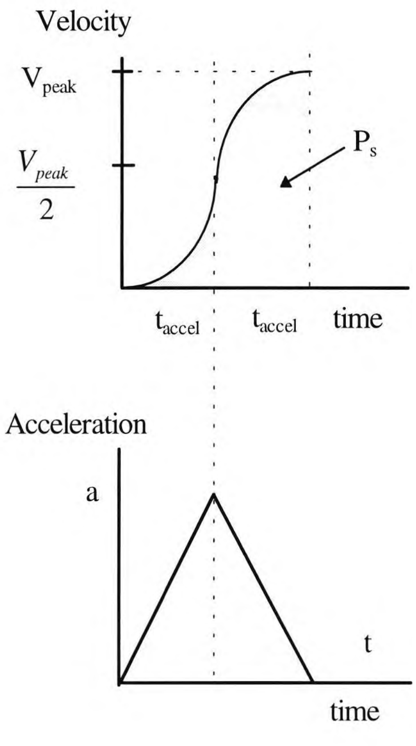


Figure I.11 *S-curve move acceleration*

For moves of the type expressed by equation I.36 the duration of the slew segment must be sufficient to allow computation of the first deceleration move.

I.4.1 Minimum time S-curve moves

A typical application for this provision is one that requires a simple point-to-point moves with S-curve acceleration for gentle starts and stops. For good productivity, however, it is desired to complete these moves in minimum time under the constraints of maximum allowed velocity and acceleration. Below is a general procedure for computing such moves.

First, using equations I.35 and I.36, and substituting T_k with 2t, and N with 1/2, we have for the S-curve acceleration:

$$\Delta P = V \times t \qquad \text{eqn I.41}$$

and $$A_{peak} = V/t \qquad \text{eqn I.42}$$

where t is the time for each of the two parabolic segments, and V is the non-zero starting or ending velocity. The time value and distance required for the fastest acceleration to the maximum allowed velocity, can be computed:

$$t = V/A_{\max} = (V_{peak}/A_{peak}) \qquad \text{eqn I.43}$$

$$\Delta P = V \times t = V_{peak} \times t \qquad \text{eqn I.44}$$

These values can be used for both the acceleration and deceleration portions of the move. For the constant speed (slew) portion, therefore:

$$\Delta P_{slew} = \text{Total Distance - } 2 \times \Delta P_{accel} \qquad \text{eqn I.45}$$

$$t_{slew} = \Delta P_{slew}/V_{peak} \qquad \text{eqn I.46}$$

In order to achieve a three part S-curve move in the manner described above, the total distance covered must be enough to allow this acceleration, deceleration, and a short slew. The minimum distance for which this is possible is:

$$\begin{aligned}\Delta P &= (2 \times V_{peak} \times t) + 0.04 \times V_{peak} \\ &= (2 \times V_{peak}^2/A_{peak}) + 0.04 \times V_{peak}\end{aligned} \qquad \text{eqn I.47}$$

were 0.04 is the least time of a segment (refer to section I.3.2). If the distance is shorter than this, it will be necessary to perform a single-part S-curve move instead.

For a very short move, the system will never attain the maximum allowed velocity, so the maximum allowed acceleration is our only constraint for calculating a minimum-time move. Figure I.12 shows a single-part S-curve move. Since there is only one position command in the move, the SMCC-P adds zero segments both before and after it. The move takes 3 x *t* total time to execute and the centre segments blends towards zero velocity on both sides.

The max. acceleration in this move occurs right at the boundaries between segments. It is calculated by:

$$A_{\max} = \Delta P / t^2 \qquad \text{eqn I.48}$$

Setting A_{max} to the maximum allowed acceleration, A_{peak}, we can solve for the minimum-time t:

$$t = \sqrt{\Delta P / A_{peak}} \qquad \text{eqn I.49}$$

The maximum distance move allowed that can be done with this calculation is the one in which $V_{max} = V_{peak}$. The equation for peak velocity is :

$$V_{peak} = \frac{3}{4} \times \sqrt{A_{peak} \times \Delta P} \qquad \text{eqn I.50}$$

and a maximum distance of a single part S-curve move will be,

$$\Delta P = (16/9) \times V_{peak}^2 / A_{peak} \qquad \text{eqn I.51}$$

Note that this is somewhat less than the minimum allowed distance of a 3 part S-curve move,

$$\text{minimum distance of a 3-part move} > 2 \times (V_{peak}^2 / A_{peak}) \qquad \text{eqn I.52}$$

For a move that is between these two limits above, a one-part move that is constrained by V_{peak}, and never reaches A_{peak} must be used. The equations for this type of move are:

$$\Delta P = (4/3) \times V_{\max} \times t$$

$$= (4/3) \times V_{peak} \times t \qquad \text{eqn I.53}$$

$$t = (3/4) \times (\Delta P / V_{peak}) \qquad \text{eqn I.54}$$

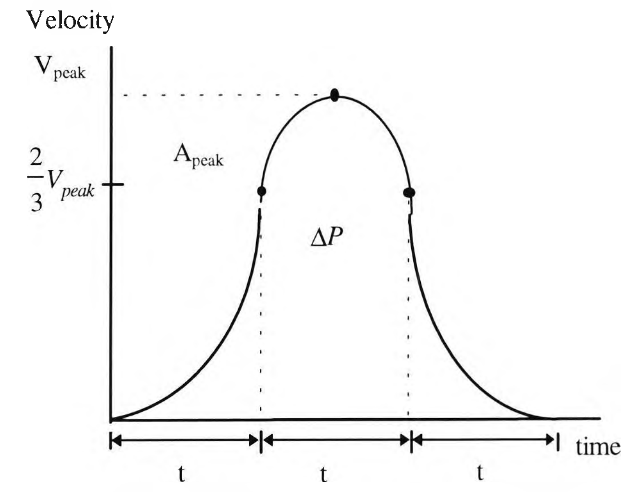


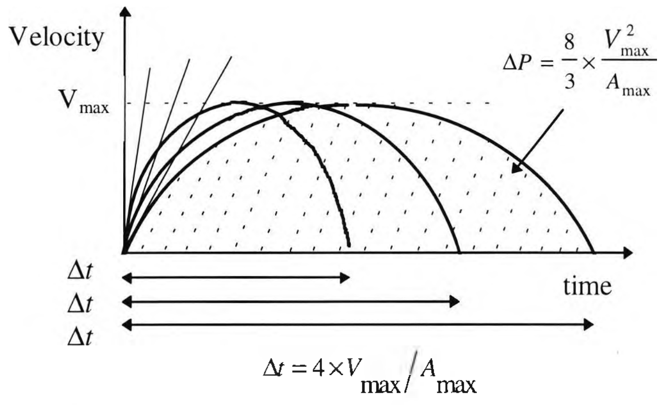
Figure I.12 *Single part S-curve move*

I.5 Determining system limits

Experiments have been undertaken to find the values of the maximum acceleration and maximum velocity of the system, with the gripper attached as described in chapter 3. On account of the partial compliance of the Yaw-axis and the long Z-axis stem, the major problems for vibration are associated with motion on the X and Y (horizontal translation) axes. Motion on the X-axis was investigated for the vibration of the end-effector (refer to chapter 4 for the end-effector vibration experiments). For the purpose of our work the limits used for the other axes are the ones given to us by the robot supplier.

I.5.1 Determining system maximum acceleration

Using equations I.10 and I.13 for the 'U' type move shown in figure I.13 and described in section I.1.2.1, and setting the value of V_{peak} to be provisionally 1000 mm/sec, different values of A_{max} were chosen in calculating different moves, starting with high values of acceleration. Each move was tested and depending on its success or failure the value of A_{max} was adjusted, until the highest, most reliable value of acceleration was reached (i.e. where the move command was successful).



Velocity
V_{max}
$\Delta P = \frac{8}{3} \times \frac{V_{max}^2}{A_{max}}$
time
Δt
Δt
Δt
$\Delta t = 4 \times V_{max} / A_{max}$

Figure I.13 *Determining maximum acceleration*

I.5.1.1 Peak acceleration results

The following parabolic moves were calculated and tested on the X-axis using V_{peak} =1000 mm/sec and $V_s = V_f = 0$. Table 3.1 below shows the moves made using the different maximum acceleration values. ΔP and Δt are the corresponding move distance and time. The value of A_{peak} was found to be 3232 mm/sec^2.

Table I.1 *Maximum acceleration using Vpeak = 1000 mm/sec*

A_{max} (mm/sec^2)	ΔP (mm)	Δt (secs)	Move Failed/Succeede d
2363	1129	1.69	S
2372	1124	1.69	S
2439	1093	1.64	S
3049	875	1.31	S
3201	833	1.25	S
3277	814	1.22	F
3247	821	1.23	F
3232	**825**	**1.24**	**S**
3239	823	1.23	F

Through experimentation, the value of V_{max} used for the moves was reduced. That had the effect of increasing the value of maximum acceleration reached up to 4116 mm/sec^2, as can be seen in tables I.2 and I.3.

Table I.2 *Maximum acceleration using Vpeak = 914 mm/sec*

A_{max} (mm/sec^2)	ΔP (mm)	Δt (secs)	Move Failed/Succeeded
2287	976	1.60	S
2439	915	1.50	S
2591	861	1.41	S
2744	813	1.33	S
2896	770	1.26	S
3049	732	1.20	S
3506	636	1.04	S
3659	610	1.00	S
3963	563	0.92	F
3811	**585**	**0.96**	**S**

Table I.3 *Maximum acceleration using Vpeak = 759 mm/sec*

A_{max} (mm/sec^2)	ΔP (mm)	Δt (secs)	Move Failed/Succeeded
4116	**376**	**0.74**	**S**

I.5.2 Determining system maximum velocity

Using the equations I.10 and I.13 for the 'U' type move shown in figure I.14 and described in section I.1.2.1, and setting the value of A_{max} to be less than A_{peak} found in section I.5.1, different values of V_{max} were chosen in calculating different moves, starting with high values. Each move was tested and, depending on its success or failure, the value of V_{max} was adjusted, until the highest, most reliable velocity was reached where the move was successful.

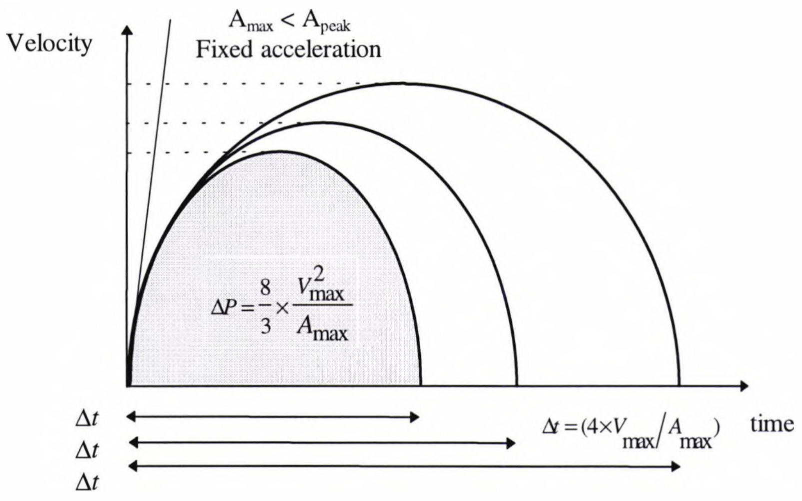

Velocity
$A_{max} < A_{peak}$
Fixed acceleration
$\Delta P = \frac{8}{3} \times \frac{V_{max}^2}{A_{max}}$
Δt
Δt
Δt
$\Delta t = (4 \times V_{max} / A_{max})$
time

Figure I.14 *Determining maximum velocity*

I.5.2.1 Peak velocity results

The following parabolic moves where calculated and tested on the X-axis using A_{max} = 1524 mm/sec^2 and $V_s = V_f = 0$. Table 3.4 shows the outcome of moves made using the different maximum velocity values. ΔP and Δt are the corresponding move distance and time. The value of V_{peak} was found to be 1143 mm/sec.

Table I.4 *Maximum velocity using A_{max} = 1524 mm/sec^2*

V_{max} (mm/sec)	ΔP (mm)	Δt (secs)	Move Failed/Succeeded
1067	1992	2.80	S
1220	2602	3.20	F
1143	**2287**	**3.00**	S
1159	2348	3.04	F

Through experimentation the value of A_{max} used for the moves was increased. This had the effect of reducing the value of maximum acceleration reached to 915 mm/sec, as can be seen in table I.5.

Table I.5 *Maximum velocity using A_{max} = 3506 mm/sec^2*

V_{max} (mm/sec)	ΔP (mm)	Δt (secs)	Move Failed/Succeeded
762	442	0.87	S
915	**915**	**1.04**	S
991	747	1.13	F

I.5.3 Setting maximum velocity and acceleration

Both values of V_{peak} and A_{peak} are related as shown in table I.6 below. The maximum velocity of the system varies with the maximum acceleration set, and vice-versa. The table below relates the value of maximum velocity reached in relation to the maximum acceleration of a move.

Table I.6 *Maximum velocity and acceleration*

Max. Velocity mm/sec	Max. Acceleration mm/sec^2
1173	2370
1143	2287
1000	3232
915	3506
762	4116

The extreme velocities used for the moves in these experiments were found to be unreliable and resulted in violent vibration of the robot. The use of extreme acceleration had the same effect. To achieve reliable results when programming the robot moves, the maximum velocity (V_{peak}) was decided at 1000 mm/sec, with the corresponding maximum acceleration (A_{peak}) at 3232 mm/sec^2. These results are used when programming the different moves in chapter 4 and when developing the rule base system to determine the best moves in chapter 9.

I.6 Verification of move

The theory of the different types of moves has been described in this chapter. However a check to compare the theoretical commanded move to the move delivered by the robot system was needed. A ‘C’ program *plot.c* was developed to capture the position, velocity and timing of the system during move execution. The results were plotted and the data compared to the theoretical results. The raw data for these tested can be found in appendix B.

Two moves were tested, parabolic U move and a single part S-curve move. Both moves were chosen to cover long distances and have reasonable move times. Short moves were not possible to accurately check as there is a limit to the speed of communication with the motion control cards when requesting position or velocity data i.e. the short move duration is inaccurate in respect to duration.

I.6.1 Profile of a parabolic "U' move

A parabolic U move of ΔP= 744 mm in time Δt= 30 seconds was made (refer to section I.3.1 for move theory). Results of position against time and velocity against time are shown in figure I.15 and figure I.16 respectively. The theoretical maximum velocity of the move was calculated using equation I.11 and resulted in V_{max} = 37.2 mm/sec. The experiment results gave us a value of 37.06 mm/sec, as shown in figure I.16. This makes the percentage error of the velocity reading to be 0.37 %.

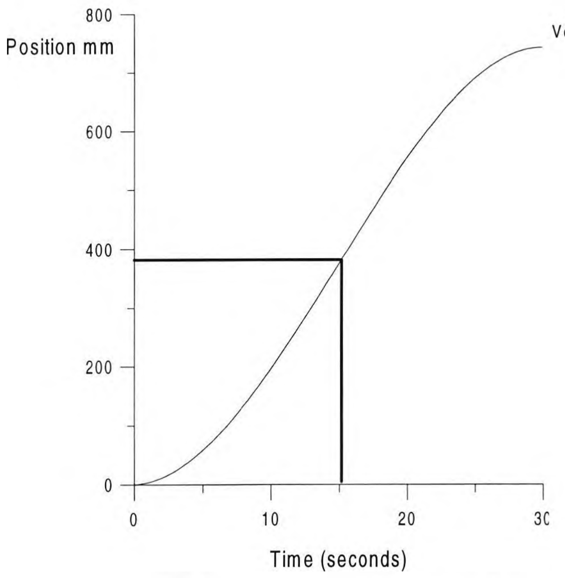


Figure I.15 *Position Vs time of a parabolic 'U' move*

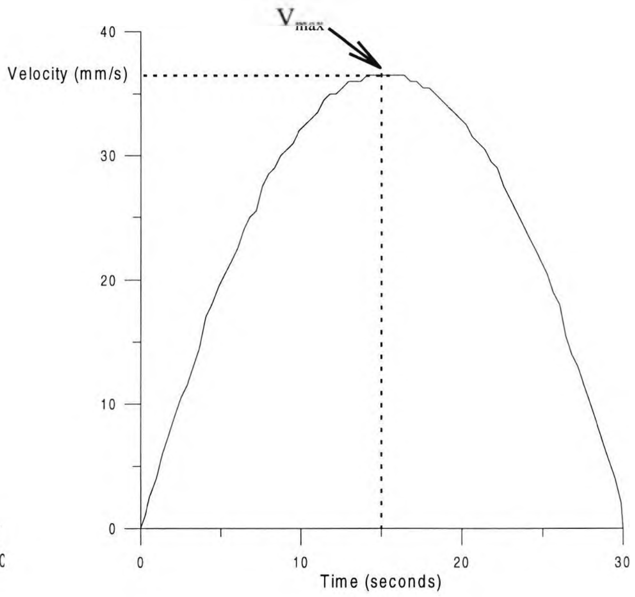


Figure I.16 *Velocity Vs time of a parabolic 'U' move*

I.6.2 Profile of a single part S-curve move

A single part S-curve move of ΔP= 1744 mm with time t = 15 seconds, which results in the total time of the move Δt= 45 seconds, was made (refer to section I.4 for move theory). Results of position against time and velocity against time were plotted and are shown in figure I.17 and figure I.18 below. The sampling rate of the velocity and position of the robot is on average 0.28 seconds.

Calculating the theoretical maximum velocity of the move using equation I.54 resulted in V_{max} = 87.25 mm/sec. The experiment results gave a value of 87.31 mm/sec as shown in figure I.18. This gives a percentage error of 0.07%. At the point where it maximum acceleration the theoretical velocity V_l was calculated to be 58.16 mm/sec (i.e. 2/3 V_{max}). The experiment results were estimated to be 58.03 mm/sec. This will results in a percentage error of 0.22%.

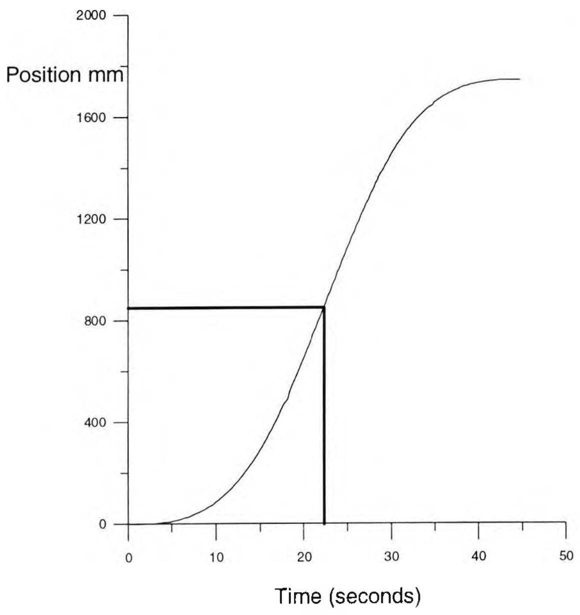


Figure I.17 Position *Vs time of a S-curve move*

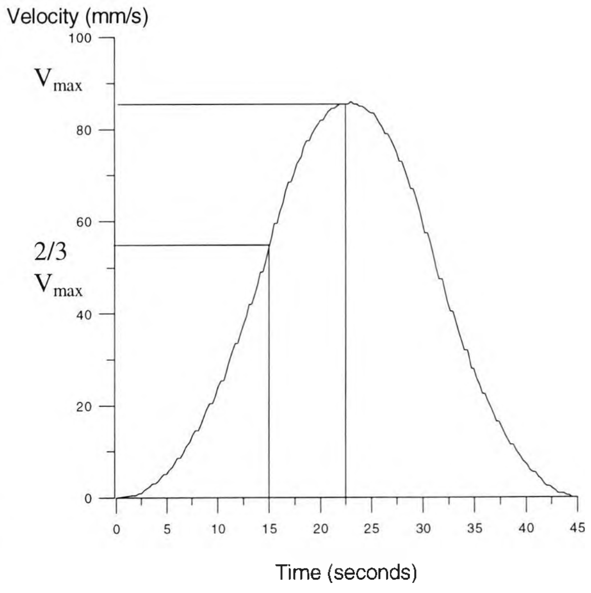


Figure I.18 *Velocity Vs time of a S-curve move*

I.6.3 Accuracy of the profile

Comparing the actual velocities of the captured move to the theoretical ones, an error of less that 0.4% was found. The inaccuracies are partly due to the inability to request the velocity and position information fast enough, from the motion control card on the robot. At the steepest part of the curve in a move profile there will have the highest following error. These errors were tolerated, as we do not require the robot to follow an exact path at these velocities.

References

Abu-Hamdan, M. and El-Gizawy, A. (1992). “An Error Diagnosis Expert System for Flexible Assembly Systems”. Conference on Information Control Problems in Manufacturing Technology, Proceedings. pp.307-312.

Ahmed, E. (1993). “Mortars for Automated Masonry Construction”. M.Sc. Thesis, City University, Department of Civil Engineering.

Ala, S.R. (1991). "Design Methodology of Boundary Data Structures". International Journal of Computational Geometry and Application 1 (3): pp.207-225.

Ala, S.R., Chamberlain, D. and Ellis, T. (1992). “Real-time inspection of masonry units”. ASCE Conference on Computing in Civil Engineering, Dallas, USA 1992. Proceeding.

Ala, S.R. (1994). “Effective computer vision for a construction robot”. Ph.D. thesis, City University, Department of Civil Engineering.

Altobelli, F., Taylor, H.F. and Bernold, L.E. (1993). “Prototype Robotics Masonry System”. Journal of Aerospace Engineering 6 (1): pp.19-33.

Andres, J., Bock, T., Gebhart, F. and Steck, W. (1994). “First Results of the Development of the Masonry Robot System ROCCO: a Fault Tolerant Assembly Tool”. International Symposium on Automation and Robotics, 11th, Brighton 1994. Proceedings. pp.87-93.

Anliker, F. (1988). "Needs for Robots and Advanced Machines at Construction Site". International Symposium on Automation and Robotics, 5th, Tokyo 1988. Proceedings. pp.145-150.

Anliker, F. and Anliker, M. (1991). "Automation of Design and Construction of Single Family Houses Made of Brickwork". International Symposium on Automation and Robotics, 8th, Stuttgart 1991. Proceedings. pp.843-847.

Askew, W., Mawdesley, M. and Booth, J. (1989). “An A.I. approach to automating the recognition and evaluation of logic classes in construction scheduling”. International Symposium on Automation and Robotics, 6th, San Francisco 1989. Proceedings. pp.197-203.

Balagure, C., Gambao, E., Barrientos, A., Puente, E. and Aracil, R. (1996). “Site Assembly in Construction Industry By Means of a Large Range Advanced Robot”. International Symposium on Automation and Robotics, 13th, Tokyo 1996. Proceedings. pp.65-72.

BALDOR ASR GmbH. (1989). Smart motion control card hardware manual.

Belohoubek, P., Kolibal, Z., Kadlec, Z. and Vystrcil, O. (1995). "The development of the intelligent multi-axis ALR robots". International Symposium on Automation and Robotics, 12th, Warsaw 1995. Proceedings. pp.307-311.

Belohoubek, P., Kolibal, Z. and Vystrcil, O. (1996). "Research on the field of the intelligent robot system control for construction". International Symposium on Automation and Robotics, 13th, Tokyo 1996. Proceedings. pp.81-86.

Bernold, L., Altobelli F. and Taylor H. (1992). "Computer-controlled brick masonry". Journal of Computing in Civil Engineering 6, (2): pp.147-160.

Bley, B. and Anliker, F. (1994). "PC-Controlled Flexible Production of Brickwork". International Symposium on Automation and Robotics, 11th, Brighton 1994. Proceedings. pp.111-116.

Bock, T., Gebhart, F. and Steck, W. (1993). "Functional Profile of a Wall Assembly Robot System and Different Solution Approach". International Symposium on Automation and Robotics, 10th, Houston 1993. Proceedings. pp.23-30.

Bock, T. and Leyh, W. (1995). "The Robotic Building Construction System". International Symposium on Automation and Robotics, 12th, Warsaw 1995. Proceedings. pp.267-280.

Bock, T., Fliedner, H. and Weingartner, H. (1996a). "The Current Status and Key Issues in the Future on Automation and Robotics in Construction on Germany". Proceedings of the 13th International Symposium on Automation and Robotics, pp.23-32.

Bock, T. (1996b). "Robotic Assembly System for Computer Integrated Construction". International Symposium on Automation and Robotics, 13th, Tokyo 1996. Proceedings. pp.169-178.

Bohm, D. (1991). "The Mason's Elevator Handling Machine". International Symposium on Automation and Robotics, 8th, Stuttgart 1991. Proceedings. pp.819-832.

Booth, J., Askew, W and Mawdesley, M. (1991). "Automated budgeting for construction". International Symposium on Automation and Robotics, 8th, Stuttgart 1991. Proceedings. pp.529-538.

Boussabaine, A. (1996). "An intelligent virtual reality model for site layout planning". International Symposium on Automation and Robotics, 13th, Tokyo 1996. Proceedings. pp.493-499.

Bradley, D. A., Seward, D. W., Heikkilab, T. and Vahab, P. (1994). "Control Architecture and Operational Strategies for Intelligent, High-powered Manipulators". International Symposium on Automation and Robotics, 11th, Brighton 1994. Proceedings. pp.367-372.

Bradley, D. A. and Seward, D. W. (1995). “Developing Real-time Autonomous Excavation - the LUCIE”. Control and Decision Conference, 34th, New Orleans 1995. Proceedings. pp. 3028-3033.

Breen, W. (1993). “Interactive CAD for Masonry Construction”. B.Sc. project report, City University, Department of Civil Engineering.

Chamberlain, D. (1989). Notes on Crocus robot, CRU.

Chamberlain, D., Speare, P. and West, G. (1990). “A Masonry Tasking Robot”. Journal of Mechatronics System Engineering **1**: pp.139-147

Chamberlain, D., Spears, P. and Ala S. (1991a). “Progress in A Masonry Tasking Robot International Symposium on Automation and Robotics, 8th, Stuttgart 1991. Proceedings. pp.909-918.

Chamberlain, D. (1991b). “Developments in Construction Robotics At City University”. Symposium on Construction Automation, Reading University 1991.

Chamberlain, D., Ala, S., Watson, J., Reilly, J. and Speare, P. (1992). “Masonry Construction by an Experimental Robot”. International Symposium on Automation and Robotics, 9th, Tokyo 1992. Proceedings. pp.573-582.

Chamberlain, D., Akrawi, S. and Watson, J.(1993). “Path Planning and sensing for an experimental masonry building robot”. International Symposium on Automation and Robotics, 10th, Houston 1993. Proceedings. pp.229-236.

Chamberlain, D.(1994). Chapter 43: A Robot for Masonry Construction, Building the Future, ed. by F. Garas. (E and FN Spoon). pp.439-449.

Chamberlain, D. (1997). “Working at height, meeting the requirements of the regulations”. HSE, Institution of Civil Engineers. (London: Thomas Telford)

Charles, R. (1993). “The Development of Pumpable Mortar Mixes Suitable for Robotics Applications”. M.Sc. Thesis. City University, Department of Civil Engineering.

Dawood, N. (1995). “An intelligent approach to automate production budget planning in the building products industry”. International Symposium on Automation and Robotics, 12th, Warsaw 1995. Proceedings. pp.93-101.

Drees, G., Laukemper, J., Pritschow, G. and Dalacker M. (1991). “Limits to Profitability of Automated Masonry”. International Symposium on Automation and Robotics, 8th, Stuttgard 1991. Proceedings. pp.833-842.

Emmanuel, C. Ifeachor, Barrie, W. Jervis. (1993). “Digital signal processing: A practical approach”. (Addison-Wesley). pp.15-16.

Everett, J.G. and Slocum, A.H. (1994). “Automation and Robotics Opportunities: Construction Versus Manufacturing”. Journal of Construction Engineering and Management **120** (2): pp. 443-451.

Gambao, E., Barrientos, A., Balaguer, C., Saltarn, R. and Aracil, R. (1996). “Fuzzy-Gain Scheduling Control Architecture for Large Range Robots”. IEEE Mediterranean Symposium on New Directions in Control & Automation, 4th, Crete 1996.

Garas, F. (1991). “State of the art in the United Kingdom”. International Symposium on Automation and Robotics, 8th, Stuttgard 1991. Proceedings. pp.45-49.

Gonzalez, R. and Wintz, P. (1987). “Digital Image Processing”, 2nd edition. (Addison-Wesley).

Green, P., Seward, D. W. and Bradley, D. A. (1990). “Knowledge Acquisition for a Robot Excavator”. International Symposium on Automation and Robotics, 7th, Bristol 1990. Proceedings. pp. 351-357.

Kimble, I. (1991). “The development of a masonry mortar facility for assisted manual and robotised construction”. B.Sc. project report, City University, Department of Civil Engineering.

Knowledge Shop User’s Guide. (1991). (USA: Descion system software).

Koskela, L., Hynynen, R., Kallavuo, M., Kahkonen, K. and Salokivi, J. (1986). “Expert System in Construction: Initial Experiences”. Joint International Conference CAD and Robotics in Architecture and Construction, Marseilles. Proceedings. pp.83-92.

Lehtinen, H., Salo, E. and Aalto, H. (1989). “Outlines of Two Masonry Robot Systems”. International Symposium on Automation and Robotics, 6th, San Francisco 1989. Proceedings. pp.143-150.

Lowe, J. and Campus, R. (1990). “Intelligent robots for construction International Symposium on Automation and Robotics, 7th, Bristol 1990. Proceedings. pp.399-405.

Malinovsky, E., Borshchevsky, A., Eler, E. and Pogodin, V. (1990). “A Robotic Complex for Brick-Laying Applications”. International Symposium on Automation and Robotics, 7th, Bristol 1990. Proceedings. pp.32-38.

Muspratt, M. (1988). "Robot Ensembles for Building Construction". Robotica **6**: pp.275-284.

Philipson, W. and Oates. (1974). “Numerical Analysis- The mathematics of computing 2”. (Edward Arnold).

Pritschow, G., Dalacker, M. and Kurz, J. (1993). “Configurable Control System of a Mobile Robot for On-Site Construction of Masonry”. International Symposium on Automation and Robotics, 10th, Houston 1993. Proceedings. pp.85-92.

Pritschow, G., Dalacker, M., Kurz, J., Gaenssle, M. and Haller, J. (1994). “Application Specific Realization of a Mobile Robot for On-Site Construction of Masonry”. International Symposium on Automation and Robotics, 11th, Brighton 1994. Proceedings. pp.95-102.

Pritschow, G., Dalacker, M., Kurz, J. and Gaenssle, M. (1995). “Technological Aspects in the Development of a Mobile Bricklaying Robot”. International Symposium on Automation and Robotics, 12th, Warsaw 1995. Proceedings. pp.281-290.

Ramirez, R. (1985). “The FFT: Fundamentals and Concepts”. (Prentice-Hall).

Reilly, J. (1992). “Autonomous Material Evaluation Supply System Version C1”. Technical Manual on the Conveyor System, Construction robotics unit, City university.

Rihani, R. and Bernold, L. (1994). "Computer Integration for Robotic Masonry". Microcomputers in Civil Engineering **9**: pp.61-67.

Rosenfeld, Y., Warszawski, A. and Zajicek, U. (1990). “Robotic Performance of Interior Finishing Works: Development of Full-size Applications”. International Symposium on Automation and Robotics, 7th, Bristol 1990. Proceedings. pp.63-70.

Rosenfeld, Y., Warszawski, A. and Zajicek, U. (1991). “Interior Finishing Building Robot 'TAMIR' International Symposium on Automation and Robotics, 8th, Stuttgart 1991. Proceedings. pp.345-354.

Seward, D., Bradley, D., Mann, J. and Goodwin, M. (1992). “Controlling and Intelligent excavator for autonomous digging in difficult ground”. International Symposium on Automation and Robotics, 9th, Tokyo 1992. Proceedings. pp. 743-750.

Seward, D.W. and Quayle, S.D. (1996). “System Architectures and Safety for Mobile Construction Robots”. International Symposium on Automation and Robotics, 7th, Tokyo 1996. Proceedings. pp.223-230.

Shohet, I. and Laufer, A. (1991). “Development of an expert system for the determination of the span of control of the construction foreman”. International Symposium on Automation and Robotics, 8th, Stuttgart 1991. Proceedings. pp. 507-517.

Shohet, I., Rosenfeld, Y. and Warszawski, A. (1994) "An intelligent task-planning system for autonomous interior-robots". International Symposium on Automation and Robotics, 11th, Brighton 1994. Proceedings. pp. 305-312.

Slocum, A.H. and Schena B. (1988). "Blockbot: a robot to automate construction of cement block walls". Robotics **4**: pp.111-129.

Spee, D. (1989). "Mobile Robots-A New Generation of Production Tasks for Robots". International Symposium on Automation and Robotics, 6th, San Francisco 1989. Proceedings. pp.356-363.

Stein, R. (1991). "Real Artificial Life". Byte Magazine **1**: pp.289-298.

Stouffs, R., Krishnamurti, R., Lee, S. and Oppenheim, I. (1993). "Construction process simulation with rule-based robot path planning". International Symposium on Automation and Robotics, 10th, Houston 1993. Proceedings. pp.495-502.

Thien, R. and Hill, S. (1991). "Sensor Fusion for Automated Assembly Using an Expert System Shell". International Conference on Advanced Robotics, 5th, 1991. Proceeding. Robots in unstructured environments **2**: pp.1270-1274.

Tommelein, I., Levitt, R. and Hayes-Roth, B. (1989). "Sightplan: An artificial Intelligence tool to assist construction managers with site layout". International Symposium on Automation and Robotics, 6th, San Francisco 1989. Proceedings. pp.340-347.

Walczewski, R. (1995). "Brawal 1611 and Brawal 4011 energy-economical hydraulic excavator with the artificial intelligence element". International Symposium on Automation and Robotics, 12th, Warsaw 1995. Proceedings. pp.167-177.

Warszawski, A. and Navon, R. (1996). "Survey of Building Robots". International Symposium on Automation and Robotics, 13th, Tokyo 1996. Proceedings. pp.205-211.

Wen, X., and Rovetta, A. (1991). "Remote Control and Robotics in Construction Engineering". International Conference on Advanced Robotics, 5th 2: pp.1429-1432.

Whittaker, W. (1986). "Construction Robotics: A Perspective". Joint International Conference CAD and Robotics in Architecture and Construction, Marseilles. Proceedings. pp.105-112.

Wurth, P. (1992) EU 377-FAMOS BRICK, Highly Flexible Automated and Integrated Brick Laying System, Luxembourg, Eureka project.