High-Speed Multi-dimensional Relative Navigation for Uncooperative Space Objects

O. Kechagias-Stamatis [a,*], N. Aouf [b] and M. A. Richardson [a]

[a] Centre of Electronic Warfare, Cranfield University Defence and Security, Shrivenham, SN6 8LA, UK

[b] Department of Electrical and Electronic Engineering, City University of London, EC1V 0HB, UK

Abstract

This work proposes a high-speed Light Detection and Ranging (LIDAR) based navigation architecture that is appropriate for uncooperative relative space navigation applications. In contrast to current solutions that exploit 3D LIDAR data, our architecture transforms the odometry problem from the 3D space into multiple 2.5D ones and completes the odometry problem by utilizing a recursive filtering scheme. Trials evaluate several current state-of-the-art 2D keypoint detection and local feature description methods as well as recursive filtering techniques on a number of simulated but credible scenarios that involve a satellite model developed by Thales Alenia Space (France). Most appealing performance is attained by the 2D keypoint detector Good Features to Track (GFFT) combined with the feature descriptor KAZE, that are further combined with either the H∞ or the Kalman recursive filter. Experimental results demonstrate that compared to current algorithms, the GFTT/ KAZE combination is highly appealing affording one order of magnitude more accurate odometry and a very low processing burden, which depending on the competitor method, may exceed one order of magnitude faster computation.

Keywords: Multi-dimensional processing, Relative navigation, Spaceborne LIDAR, Uncooperative target

1. Introduction

Ego-motion estimation, i.e. odometry, for space applications is an active research domain due to the increasing number of spacecrafts deployed. Specifically, great research interest considers relative space navigation of a *Source* spacecraft platform in relation to a non-cooperative *Target* platform, i.e. with unknown attitude (pose). This is because relative space navigation will enable a *Source* spacecraft with the capability to perform autonomous close-

[*] Corresponding author
E-mail address: o.kechagiasstamatis @Cranfield .ac.uk

proximity manoeuvres and achieve uncooperative rendezvous with a non-cooperative *Target* platform, contributing towards autonomous active space debris removal, satellite inspection and docking. In any of these scenarios, the *Target* is likely to be non-cooperative and therefore unable to exchange with the *Source* its pose neither actively nor passively, i.e. via known markers placed on the *Target*. Therefore, the *Source* spacecraft must estimate its relative position and attitude with respect to the *Target* platform by utilizing only its onboard sensors. Current solutions involve 2D visual data in a monocular [1,2] or a stereo camera configuration [3–6], 2D Infrared (IR) thermal data [7], and 3D Light Detection and Ranging (LIDAR) data [8,9,18,10–17]. A thorough review of spacecraft pose determination techniques for close-proximity operations is presented in [19].

Despite each modality, i.e. visual, IR and LIDAR, having its own strengths and weaknesses, LIDAR is preferred either in a scanning or in a flash operating mode due to its proven robustness in the space environment [20]. Indeed, visual data can be an effective solution [21] but the use of IR data have several advantages over the visual data because they can operate during day and night under several harsh illumination conditions like eclipse and solar glare. Despite these advantages, the accuracy of IR thermal odometry relies on the *Target's* temperature that is affected by internal parameters, e.g. heat dissemination of the platform's components, and external parameters, e.g. reflection of sun's radiation. This temperature fluctuation can affect the robustness of the IR based local feature detection and matching process, which are the core procedures of the IR thermal odometry presented in [7]. On the contrary, 3D LIDAR based odometry outperforms its 2D counterparts (visual and IR) as it operates during day, night and under poor visibility conditions, is independent of the *Target's* thermal properties, is capable of revealing the underlying structure of an object and can provide both 3D position and intensity data. [19,22].

Despite the advantages of LIDAR, the associated hardware requirements for power, physical space and the corresponding computational cost of the algorithms used are higher compared to 2D based architectures exploiting visual or IR sensors. This is because LIDAR sensors are complex active devices involving 3D data manipulation, while visual and IR cameras are passive and less complex devices that in principle output 2D data. However, spurred by the advantages of LIDAR odometry, LIDAR sensors have already been placed on space platforms [23] trading off the amplified requirements of this type of sensors with their advantages over visual and IR cameras. An open case is the processing recourses onboard space platforms that are typically based on space-graded field programmable gate arrays (FPGA). However, recent work [24] demonstrated that FPGA boards are capable of performing 2D computer vision based navigation. Hence, there is the potential for FPGA boards to perform complex space navigation utilizing 3D LIDAR data. For further details on spaceborne sensors for spacecraft pose estimation the reader is referred to [19].

Spurred by the advantages of 3D LIDAR odometry for space applications, current literature suggests quite a few techniques that are summarized in Table 1. Specifically, [13] presents the capabilities of the *Argon* relative navigation system that uses a stereo optical camera and a flash LIDAR configuration. *Argon* application relies on edge detection and a custom Iterative Closest Point (ICP) scheme for 6-degrees of freedom pose estimation. Other solutions involve template matching for pose initialization and then exploit the typical ICP [25] for frame-to-frame pose estimation [9,16,17,26,27]. Variants of that methodology substitute the template matching scheme for pose initialization either with Principle Component Analysis (PCA) [9,28] or with global 3D feature matching using the Oriented Unique Repeatable Clustered Viewpoint Feature Histogram (OUR-CVFH) [14,29]. Other solutions available in the literature fuse pose estimation based on OUR-CVFH or on Spin Images [30] (a 3D local feature descriptor) and ICP, with gyroscopic data and then perform *Target* platform tracking using a Multiplicative Extended Kalman Filter (MEKF) [10,28,31]. Volpe *et al.* [11] suggest utilizing 2D features from the visual domain combined with LIDAR based distance estimation and Unscented Kalman Filtering (UKF) for performance improvement. Alternatives to pure ICP registration for pose estimation have also been proposed by substituting ICP with a UKF filter, an iterative least-squares (LS) scheme, or with an Extended Kalman Filter (EKF) [32], [33]. An additional alternative is suggested in [18] that combines 3D local feature matching based on the Histogram of Distances – Short (HoD-S) descriptor [34] and the $H\infty$ filter.

Driven by the advantages of 3D LIDAR odometry, the availability of affordable LIDAR technologies and considering the need for space odometry with increased accuracy and less computational burden, we suggest a novel LIDAR based architecture that transforms the odometry problem from the 3D space into multiple 2D ones that involve 2.5D imagery (range maps) and completes the odometry problem by utilizing a recursive filtering technique. Specifically, in the context of uncooperative space odometry the contributions of this work are:

a. A high-speed space odometry architecture that has a processing burden in the order of milliseconds and provides one order of magnitude more accurate relative odometry compared to current solutions.

b. A multi-dimensional solution that combines the advantages of the 2D and 3D data space. Indeed, our architecture reaches high odometry accuracy as it exploits 3D data and a very low processing time due to transforming the odometry problem from the 3D space into multiple 2D ones that involve 2.5D imagery to minimize information loss.

c. It evaluates state-of-the-art local keypoint detection, feature description and recursive filtering methods and analyses their performance.

The remainder of the article is organized as follows: Section 2 introduces the proposed LIDAR based space odometry architecture and extensively presents the evaluation of 2D keypoint detectors, feature descriptors and recursive filtering methodologies. Section 3 evaluates the suggested architecture against current LIDAR based odometry methods on several simulated but highly realistic scenarios and our conclusions are presented in Section 4.

Table 1.
Current 3D space odometry architectures

Nº	Reference	Year	Target	Hardware	Relative navigation method
1	Galante *et al.* [13]	2012	Real	Stereo optical camera and LIDAR	2D edge tracking and custom ICP for pose estimation
2	Sell *et al.* [14]	2014	Real	LIDAR	OUR-CVFH for pose initialization and ICP for point cloud registration and pose estimation
3	Opromolla *et al.* [16]	2014	Simulated	LIDAR	Optimized template matching for pose initialization and ICP for point cloud registration and pose estimation
4	Opromolla *et al.* [17,26]	2015	Simulated	LIDAR	Optimized template matching for pose initialization and ICP for point cloud registration and pose estimation
5	Rhodes *et al.* [28]	2016	Simulated	Gyroscope, star tracker, LIDAR	OUR-CVFH or Spin Images combined with ICP for pose estimation that is fused with sensor inputs via a MEKF module
6	Liu, Zhao and Bo [27]	2016	Simulated and real	LIDAR	Template based pose initialization and ICP object tracking
7	Woods and Christian [10]	2016	Simulated	Gyroscope, GPS, star tracker, LIDAR	OUR-CVFH for pose initialization and ICP for point cloud registration and pose estimation that is fused with sensor inputs via a MEKF module
8	Opromolla *et al.*[9]	2017	Real	LIDAR	Optimized template matching or PCA for pose initialization and ICP for point cloud registration and pose estimation
9	Volpe et al. [11]	2017	Simulated	Optical camera and LIDAR	2D feature based visual odometry with LIDAR based distance measurement combined with UKF
10	Rhodes, Christian and Evans [31]	2017	Simulated	LIDAR	OUR-CVFH or OUR-CVFH combined with MEKF for trajectory smoothing
11	Dietrich and McMahon [33]	2017	Simulated	LIDAR	point cloud registration using UKF
12	Dietrich and McMahon [32]	2018	Simulated	LIDAR	point cloud registration using UKF, LS and EKF
13	Kechagias-Stamatis and Aouf [18]	2019	Real	LIDAR	HoD-S local features with adaptive $H\infty$ recursive filtering

2. Proposed Architecture

2.1. LIDAR based Odometry

The suggested LIDAR based relative navigation architecture involves a *Source* platform that has a 3D LIDAR sensor and an uncooperative *Target* platform with an unknown structure. The aim of the proposed technique is to estimate the relative position between the *Source* platform and the *Target* platform, with equal priority given to position accuracy and computational requirements.

Given two consecutive point clouds $\boldsymbol{P}_k = \{p_k^1,..., p_k^a\}$ and $\boldsymbol{P}_{k+1} = \{p_{k+1}^1,..., p_{k+1}^b\}$ of the *Target* platform that are captured from the *Source's* LIDAR sensor, with each vertex being in the form $p = (x, y, z)$, the odometry process aims at calculating a rigid body transformation,

$$R^* = \left[\begin{array}{c|c} R & T \\ 0 & 1 \end{array}\right] \tag{1}$$

with R as the rotation and T the translation component that remap point cloud $\boldsymbol{P}_k$ to $\boldsymbol{P}_{k+1}$:

$$p_{k+1} = Rp_k + T \tag{2}$$

$$\begin{bmatrix} x_{k+1} \\ y_{k+1} \\ z_{k+1} \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \begin{bmatrix} x_k \\ y_k \\ z_k \end{bmatrix} + \begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix} \tag{3}$$

Then at instance u, the position of the *Source* platform relative to the unknown and uncooperative *Target* is given by:

$$R_u^* = \prod_{\mu=1}^{u} R_\mu^* \tag{4}$$

Current literature addresses LIDAR odometry for space applications mainly by calculating R^* via a two-staged process, i.e. coarse *Target* pose initialization using template matching or 3D feature matching (global or local features), and then perform *Target* pose estimation via an ICP process. However, as presented in Section 3, these solutions still exhibit certain challenges including low odometry accuracy and high processing burden.

2.2. Multi-Projection LIDAR Odometry

Driven by the need of achieving a high performance and efficient uncooperative relative navigation architecture, we suggest an appealing multi-discipline architecture, which accurately estimates the transformation R^* with a very low computational burden. An analysis of the developed approach is presented in the following paragraphs of this section.

2.2.1 Multi-2.5D local keypoint detection, description, and matching

Although 3D data have several advantages over their 2D counterpart (see Section 1), the computational burden to manipulate 3D data is substantially higher compared to exploiting 2D data [35]. Therefore, we take advantage of both data modalities by remapping $\boldsymbol{P}_k$ and $\boldsymbol{P}_{k+1}$ into several 2.5D images, i.e. 2D range maps/ images. Specifically, for $\boldsymbol{P}_k$ and accordingly for $\boldsymbol{P}_{k+1}$, we transfer the XYZ_{LIDAR} reference frame that is centred at the LIDAR sensor onboard the *Source* platform to $\boldsymbol{P}_k$ and create the XYZ_{Target} reference frame. Then, we quantize the floating-point vertex coordinates $\boldsymbol{P}_k = \{p_k^1, ..., p_k^a\}$ into $\boldsymbol{P}_{Q\_k} = \{p_{Q\_k}^1, ..., p_{Q\_k}^a\}$ with,

$$p_{Q\_k}\left(x_Q, y_Q, z_Q\right) = \left\lfloor q_f \cdot p_k\left(x, y, z\right)\right\rfloor \tag{5}$$

where q_f is a quantization factor and $\lfloor \cdot \rfloor$ the bottom-round process. Next, we multi-project $\boldsymbol{P}_{Q\_k}$ to every plane of the XYZ_{Target} reference frame by utilizing an orthographic projection process P_{ortho}. Depending on the projection plane, we substitute with zero the appropriate binary remapping coefficients $c_1, c_2, c_3 \in \{0,1\}$ of P_{ortho}, i.e. for $c_1 = c_2 = 1$ and $c_3 = 0$, the XY 2.5D image coordinates $\tilde{p}_{Q\_k}$ are created:

$$\tilde{p}_{Q\_k} = \begin{bmatrix} \tilde{x}_q \\ \tilde{y}_q \\ \tilde{z}_q \\ 1 \end{bmatrix} = P_{ortho} \cdot p_{Q\_k} = \begin{bmatrix} c_1 & 0 & 0 & 0 \\ 0 & c_2 & 0 & 0 \\ 0 & 0 & c_3 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_{Q\_k} \\ y_{Q\_k} \\ z_{Q\_k} \\ 1 \end{bmatrix} \tag{6}$$

The three orthographic projections $\tilde{p}_{Q\_k}^{XY}, \tilde{p}_{Q\_k}^{XZ}, \tilde{p}_{Q\_k}^{YZ}$ are 2.5D images, which are simplified versions of $\boldsymbol{P}_{Q\_k}$. The depth value of each $\tilde{p}_{Q\_k}$ is unique and represents the distance between the target and the LIDAR sensor. An example of the multi-2.5D process is presented in Fig. 1.

Next on each 2.5D image we apply current state-of-the-art 2D keypoint detection methods to analyse the structure around each pixel and classify as keypoints the ones that fulfil some specific criteria that depend on the detector. Ideally, keypoints are prominent among their surroundings, have unique features, and can be redetected even if the object they belong to is distorted or corrupted. Despite literature offering quite a few 2D keypoint detection methods, for better readability in this work we evaluate one representative of the two main keypoint detection categories, namely *blob* and *corner* detectors. Since in this work, keypoint detection performance and processing efficiency are of equal importance, for the former category we select the Fast Hessian (FH) [36] and the for the latter the Good Features To Track (GFTT) [37]. For completeness we present the operating principle of each keypoint detector evaluated.

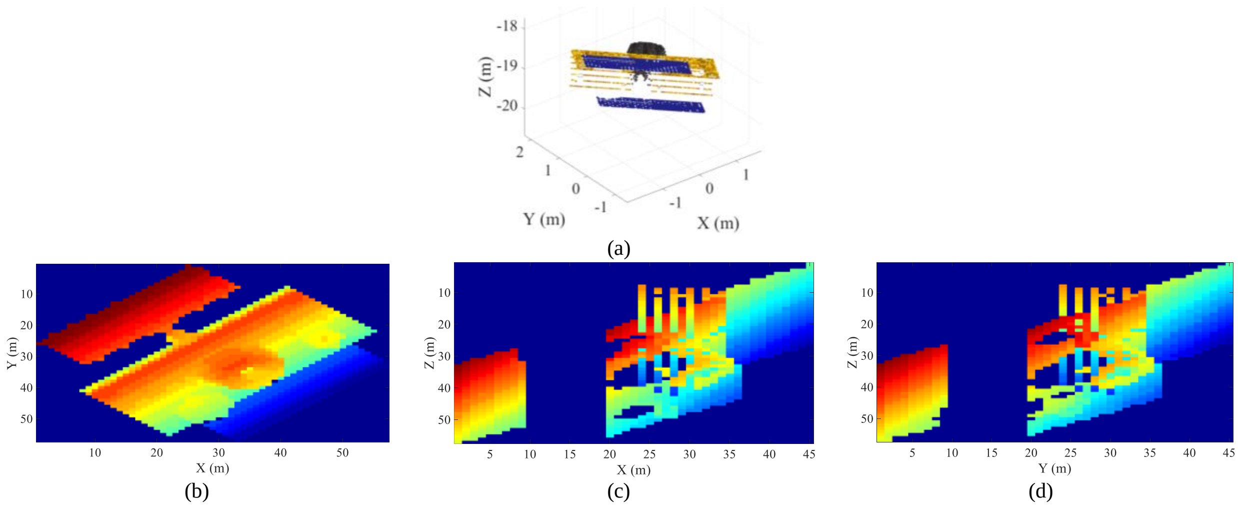

Fig. 1. 3D to multi 2.5D projection, a) 3D point cloud data in the XYZ_{Target}. reference frame, (b)-(d) multi-2.5D imagery (red corresponds to close and blue to far *Source* – *Target* platform distance)

Fast Hessian (FH) [36] neglects the processing burden of convolving the input image with second-order derivatives by approximating the Gaussian kernels with their discretized version (i.e. box filters) that are computed with a constant time cost by utilizing the integral image concept [39]. Candidate features are obtained after a 3 × 3 × 3 neighbourhood non-maximum suppression process and the ones with a response *Rp* exceeding a pre-defined threshold are preserved while the rest are discarded:

$$Rp(x,y,\sigma) = D_{xx}(\sigma)D_{yy}(\sigma) - \left(0.9Dxy(\sigma)\right)^2 \tag{7}$$

where $Dxx(\sigma)$, $Dyy(\sigma)$ and $Dxy(\sigma)$ are the outputs after convolving the corresponding box filters of standard deviation σ with each 2.5D image $I = \tilde{p}^{XY}_{Q\_k}, \tilde{p}^{XZ}_{Q\_k}, \tilde{p}^{YZ}_{Q\_k}$.

The Good Features To Track (GFTT) keypoint detector [37] relies on an autocorrelation function that captures the intensity variations of an image *I* in a neighbourhood window *Q* centred at pixel *p(x,y)*:

$$E(x,y) = \sum_{Q} w(u,v)\left[I(u+x,v+y) - I(u,v)\right]^2 \tag{8}$$

where *(x, y)* are the pixel coordinates in *I*, and *w(u, v)* is the window patch at position *(u, v)*. Using Taylor's approximation, Eq. (8) becomes:

$$E(x,y) = \begin{bmatrix} x & y \end{bmatrix} M \begin{bmatrix} x \\ y \end{bmatrix} \tag{9}$$

$$M = \begin{bmatrix} \sum_Q I_u^2 & \sum_Q I_u I_v \\ \sum_Q I_u I_v & \sum_Q I_v^2 \end{bmatrix} \tag{10}$$

where I_u, I_v represent the spatial gradients of the image.

The shape of Q is classified based on the eigenvalues λ_1 and λ_2 of M. Specifically, if both values are small, E also has a small value and Q has an approximately constant intensity. If both are large, E has a sharp peak indicating that Q includes a corner, if $\lambda_1 > \lambda_2$ then Q includes an edge and if $\min(\lambda_1, \lambda_2) > \lambda$, then Q encloses a corner, where λ is a predefined threshold. To measure the corner or edge quality, metric R_G is used:

$$R_G(x,y) = \det(M) - k \cdot tr(M) = \lambda_1 \lambda_2 - k(\lambda_1 + \lambda_2) \tag{11}$$

where $k \in [0.04,..,0.15]$.

After detecting keypoints on the 2.5D images I, each keypoint is encoded using a local feature description technique, which aims at encoding the properties of a local patch centred on each keypoint. Ideally, feature descriptors describe each keypoint in a unique manner and are robust to orientation variations and affine transformations. Given that odometry accuracy and processing efficiency are of equal importance, we evaluate the SURF [36], KAZE [41], Fast Retina Keypoint (FREAK) [42] and the Binary Robust Invariant Scalable Keypoints (BRISK) [43] feature descriptors. It should be noted that we carefully select the feature descriptor candidates such that both floating point (SURF and KAZE) and binary class (FREAK and BRISK) descriptors are included in order to evaluate not only each descriptor individually, but also the overall performance of each class. It is worth noting that despite the Scale Invariant Feature Transform (SIFT) [38] method being unarguably one of the most robust feature descriptors, its computational burden is higher compared to the floating point descriptors SURF and KAZE [41] and is therefore discarded. For completeness we present the operating principle of each descriptor evaluated.

SURF [36] initially performs an orientation assignment by computing Gaussian-weighted Haar wavelet responses over a circular region with a radius six times the scale where the keypoint is detected. Once an orientation is assigned, a square region (20 × scale) is centred on the keypoint, oriented accordingly and is then further divided into 4×4 sub-regions. For each sub-region vertical and horizontal Haar-wavelet responses weighted with a Gaussian kernel are computed. This process is performed at fixed sample points and is summed up in each sub-region. Finally, the polarity of intensity changes is also calculated by summing the absolute values of the horizontal and vertical responses. SURF features of opposing polarity are not matched. The keypoint description part of KAZE [41] is similar to SURF but is properly adapted to facilitate a non-linear scale-space framework, rather than a linear that is used in SURF.

The BRISK method [43] encodes keypoints using a handcrafted sampling pattern comprising of concentric circular patches centred at a keypoint. Aliasing effects during sampling are avoided by applying local Gaussian smoothing on the patch to be described, with a standard deviation proportional to the distance between the circle centre and the keypoint. There are two types of sampling pairs (short and long pairs) that depend on the distance between them. The long pairs have a distance greater than threshold d_{min} and are used to compute the local gradient (of the patch) that defines the orientation of the feature. The short pairs with a distance less than threshold d_{max} are then rotated accordingly to achieve rotation invariance and are used to compute the binary BRISK descriptor via intensity tests.

FREAK [42] is a biologically-inspired binary descriptor that applies a series of intensity tests on a patch that is centred at the keypoint. FREAK and BRISK share the same sampling pattern and use the same mechanism to estimate the keypoint orientation. However, FREAK is influenced by the human retinal system and uses a circular sampling grid with sampling points that are denser near the centre and become exponentially less dense further away from the centre. The advantage of this concept is that the test pairs naturally form a coarse-to-fine approach. Feature matching is accelerated by comparing the coarse part of the descriptor and if these exceed a threshold then the fine part is tested.

Once we describe all keypoints, we then employ a feature matching stage that cross-matches all features originating from every 2.5D image projection of both $\boldsymbol{P}_k$ and $\boldsymbol{P}_{k+1}$. This strategy involves cross-matching all nine 2.5D image projection combinations compensating a high-speed relative motion between the *Source* and the *Target* platform where a keypoint during the multi-projection process might shift from one 2.5D image to another. Let $\boldsymbol{F}_k = \{f_k^1, ..., f_k^{N_i}\}$ and $\boldsymbol{F}_{k+1} = \{f_{k+1}^1, ..., f_{k+1}^{N_j}\}$ be two sets of features belonging to the 2.5D images of point clouds $\boldsymbol{P}_k$ and $\boldsymbol{P}_{k+1}$, respectively. We match feature f_k^i from $\boldsymbol{F}_k$ with its nearest feature f_{k+1}^j from $\boldsymbol{F}_{k+1}$ based on an L_2-norm metric:

$$f_k^i \triangleq f_{k+1}^j \longleftarrow \arg \min_{n=1,2,...,N_j} \left(\left\| f_k^i - f_{k+1}^j \right\|_2 \right) < \tau \tag{12}$$

where ι, j are the feature indexes and the threshold τ is set to 1 to reduce the dependency between the threshold value and the metric used [44]. We speedup the process of Eq. (12) by employing the Fast Library for Approximate Nearest Neighbors (FLANN) [45]. FLANN is a library that is used for fast approximate nearest neighbour searches in high dimensional spaces. It either uses a hierarchical k-means trees search with a priority search order or a multiple randomized kd-trees scheme. The selection of the search scheme and the optimum parameters are automatically chosen from the FLANN library and depend on the data applied to FLANN. Feature matching is then performed by extending the geometric consistency checks of [46] in the 2.5D domain. Specifically, the correspondences obtained from FLANN (Eq. (12)) are clustered into hypotheses, using their true physical geometric (pixel distance) consistency. Geometric consistency aims at reducing mismatches by grouping correspondences into clusters that are geometrically consistent. For the latter, from the FLANN matching stage (Eq. (12)) a list of descriptor correspondences is created $H_u = \left\{ p^u_{Q\_k}, p^u_{Q\_k+1} \right\}$, where $p^u_{Q\_k}$ and $p^u_{Q\_k+1}$ are the *Target* correspondences in pixel coordinates at instance *k* and *k+1*:

$$H_u = \left\{ p^u_{Q\_k}, p^u_{Q\_k+1} \right\} \longleftarrow f^i_k \triangleq f^j_{k+1} \tag{13}$$

Given a seed correspondence from H_u, the first cluster is initialized and all correspondences $H_v = \left\{ p^v_{Q\_k}, p^v_{Q\_k+1} \right\}$, $v < u$ not yet grouped that are geometrically consistent with the cluster are added to it. The consistency check for a pair of correspondences H_u, H_v is valid if the following distance relation holds:

$$\left| \| p^u_{Q\_k} - p^m_{Q\_k} \|_2 - \| p^u_{Q\_k+1} - p^m_{Q\_k+1} \|_2 \right| < \varepsilon \tag{14}$$

ε being the threshold tolerance for their consensus set. The matched feature pairs $\{ f^i_k, f^j_{k+1} \}$ belonging to the cluster with the largest cardinality are considered as feature matches, while their associated vertices $\Omega_Q = \left\{ p^i_{Q\_k}, p^j_{Q\_k+1} \right\}$ are considered as point correspondences. Finally, we back-project Ω_Q to the initial 3D space and establish a set of 3D correspondences $\Omega = \left\{ p^i_k, p^j_{k+1} \right\}$. Due to the quantization process of Eq. (5), we create Ω by correlating each back-projected vertex pair of Ω_Q to its nearest neighbour vertex in $\boldsymbol{P}_k$ and $\boldsymbol{P}_{k+1}$ respectively.

2.2.2 Recursive filtering

We solve Eq. (2) utilizing a recursive filtering scheme where the state variable $x_k = [r_{11}\ r_{12}\ r_{13}\ r_{21}\ r_{22}\ r_{23}\ r_{31}\ r_{32}\ r_{33}\ t_x\ t_y\ t_z]^T$ encompasses the rigid transformation between ***P****k* and ***P****k+1* by exploiting the correspondences Ω. It should be noted that we intentionally do not apply the recursive filtering scheme in the 2D space by exploiting Ω_Q as this would increase the overall processing time due to the feature cross-matching approach. In this paper we evaluate the H∞ and the Kalman recursive filters.

H∞ filter [47] is a recursive optimal state estimator that is adapted to our formulated registration model with x_k the state variable vector and $\psi_k = [x_k, y_k, z_k]^{\mathrm{T}}$ the measurement vector that contains the 3D coordinates of the point correspondences p_{k+1}^j belonging to ***P****k+1*, which are included in Ω. The registration model is given then by:

$$x_k = \Phi\ x_{k-1} + w_{k-1} \tag{15}$$

$$\psi_{k+1} = H_k x_k + v_k \tag{16}$$

where Φ is the state transition matrix and H the measurement model matrix. We set $\Phi = R_0^* = [I \mid T_0]$ with *I* the identity matrix and $T_0 = [0\ 0\ 0]^T$, w and v are the model and the measurement noise factors respectively with covariance matrices $W \sim N(0, \sigma_w^2 J_{12})$ and $V \sim N(0, M\sigma_v^2 J_3)$ where σ_w and σ_v are small positive values and J is the unity matrix. H_k contains the actual measured 3D coordinates of p_k^i belonging to ***P****k* that are included in Ω:

$$H_k = \begin{bmatrix} x_{k+1} & y_{k+1} & z_{k+1} & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & x_{k+1} & y_{k+1} & z_{k+1} & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & x_{k+1} & y_{k+1} & z_{k+1} & 0 & 0 & 1 \end{bmatrix} \tag{17}$$

The problem that the *H*∞ filter is trying to solve is the $\min_x \max_{w,v} G$ where *G* is defined as:

$$G = \frac{average\left(\left\| x_k - x_k \right\|\right)_Q}{average\left(\left\| w_k \right\|\right)_W + average\left(\left\| v_k \right\|\right)_V} \tag{18}$$

subject to $G < 1/\gamma$, with Q being a weighting matrix and γ a small constant representing the required accuracy of the filter. The *H*∞ filter equations solving Eq. (18) are:

$$L_k = \left(I - gQP_{k-1} + H_k^T V^{-1} H_k P_{k-1}\right)^{-1} \tag{19}$$

$$K_k = \Phi P_{k-1} L_k H_k^T V^{-1} \tag{20}$$

$$P_k = \Phi P_{k-1} L_k \Phi^T + W \tag{21}$$

$$x_{k+1} = \Phi x_k + K_k \left(\psi_k - H x_k\right) \tag{22}$$

where $Q = Idt$ with $dt = 10^{-5}$ and $g = 0.1$ being regulating parameters. The number of iterations of the $H\infty$ filter is the cardinality of Ω and ultimately the final χ is transformed into R^* after all iterations, which is input to Eq. (4) in order to estimate the LIDAR based odometry. The parameters of the $H\infty$ filter as well as rest of the filters evaluated in this work are calibrated based on scenario 1.

We also evaluate the performance of the Kalman filter [48], which using the same notation as for the $H\infty$ filter, is given by:

$$x_k = \Phi x_{k-1} + \mathrm{B} q_k + w_{k-1} \tag{23}$$

$$\psi_k = H_k x_k + v_k \tag{24}$$

with B as the control input model matrix and q the control vector of the system. The *Kalman* filter equations are:

$$K_k = AP_k H_k^T \left(H_k P_k H_k^T + VM\right)^{-1} \tag{25}$$

$$x_{k+1} = \left(\Phi x_k + Bu_k\right) + K_k \left(\psi_k - H_k x_k\right) \tag{26}$$

$$P_k = \Phi P_{k-1} \Phi^T + W - \Phi P_k \mathrm{H}_k^T V^{-1} H_k P_k \Phi^T \tag{27}$$

where K is the Kalman gain and P the estimation error covariance, with $\sigma_v = 1$ and $\sigma_w = 5 \cdot 10^{-3}$ that are experimentally defined on scenario 1 to gain optimum odometry performance.

It should be noted that depending on the *Target's* pose at instance k and $k+1$, the multi-projection and feature cross-matching process may provide correspondences with a cardinality that is not adequate for the recursive filtering process to iterate properly and estimate R^* accurately. Thus, in case the correspondence cardinality is below a pre-defined threshold, we input the initialization value $R = R_0^*$ to the Eq. (1). The suggested architecture is presented in Fig. 2.

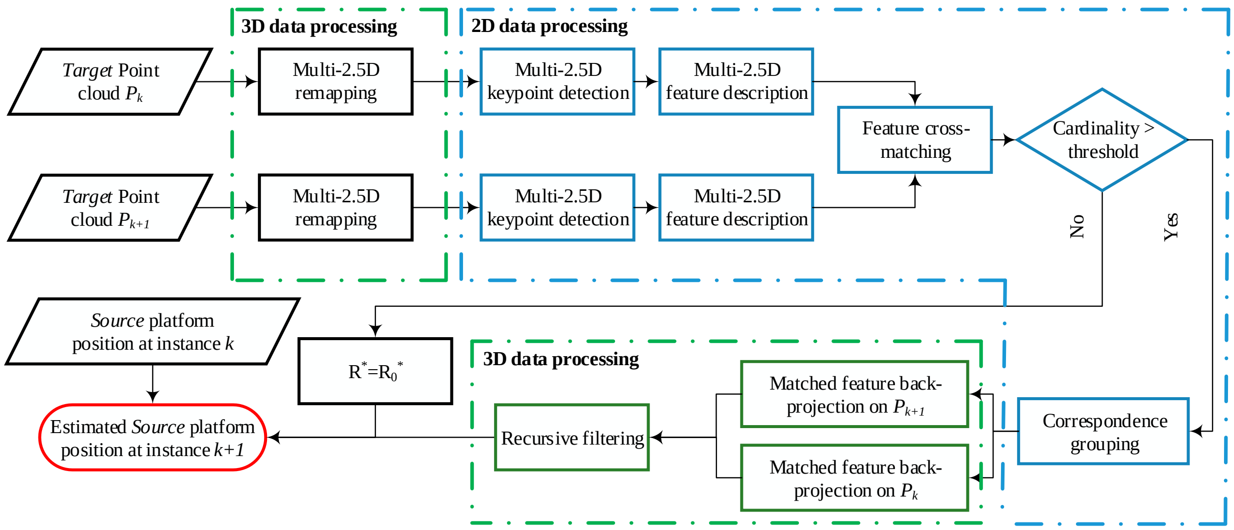

Fig. 2. Suggested recursive LIDAR odometry

3. Experiments

3.1 Experimental Setup

For our trials we use simulated trajectories of a space platform that is a customized version inspired from the Globalstar-2 and Iridium constellations, based on the Elite platform developed by Thales Alenia Space (France). In our trials we consider three scenarios, namely a straight-line approach (SLA), an ellipse of inspection (EOI) and a static station keeping (SSK). In order to increase the realistic nature of the trials, simulation considers the Earth's mass, the Sun's sunlight power with respect to each spectral band and the typical physical size of the *Source* and the *Target* platforms. An example of the *Target* platform along with the ground truth trajectory of the SLA and EOI scenarios and the corresponding cardinality of $\boldsymbol{P_k}$ are presented in Fig. 3. We intentionally do not present the SSK scenario plot as it involves a single position in the 3D space rather than a trajectory. In the SSK scenario $\boldsymbol{P_k}$ constantly comprises of 4556 vertices.

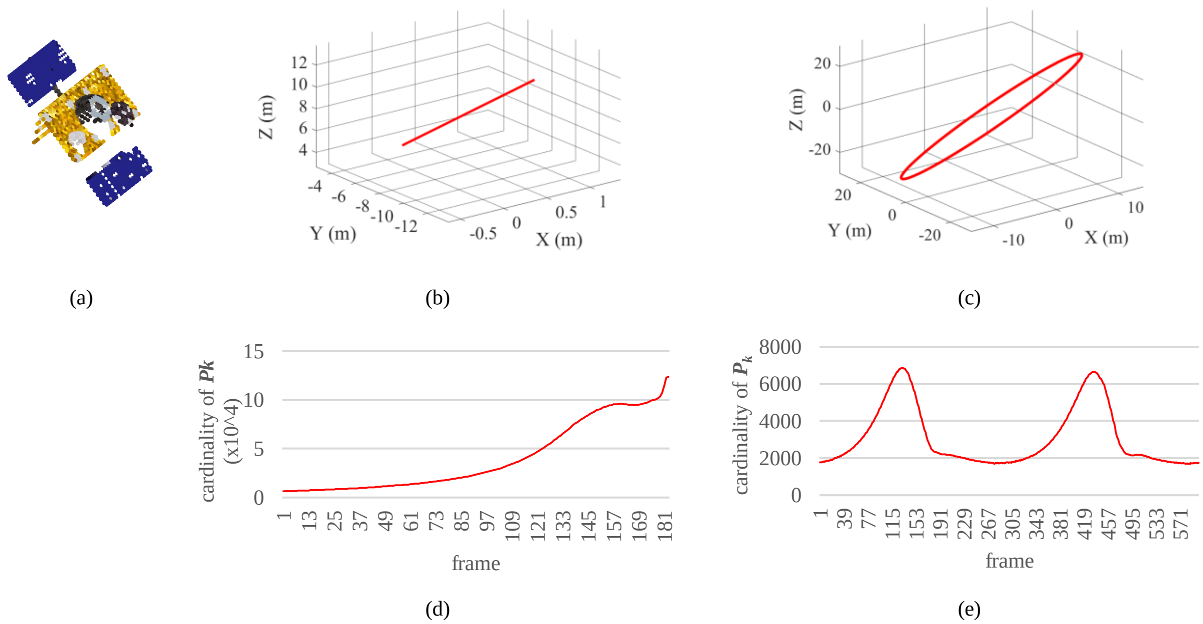

Fig. 3. (a) Target platform, trajectory plot of the (b) SLA trajectory (c) EOI trajectory and *Target* point cloud cardinality for the (d) SLA trajectory and (e) EOI trajectory

In the following trials, we compare the suggested architecture against current space oriented architectures and specifically against OUR-CVFH combined with ICP [28], Spin Images combined with ICP [28] and ICP only [9,16,17,26,27] with pose initialization considered as given. The parameters of the architecture and of the competing methods are tuned based on the SLA Scenario. Table 2 presents the tuned parameters, while the ones not tuned are fixed either to the ones originally proposed by their authors or for OUR-CVFH and Spin Images to their PCL implementation [34,49,50]. Odometry performance is evaluated based on drift, i.e. RMSE between the estimated end-point and the ground truth (GT) end-point, T_{error} presenting the overall translational error as a percentage over the GT distance travelled, average and maximum translational error per axis, rotational error, and processing time.

Table 2.
Tuned parameters

Module	Tuned parameters
q_f quantization factor	15
FH keypoint detector	6 scale levels / blob threshold 10^{-5}
GFTT keypoint detector	Min. corner quality 10^{-3} / Gaussian filter size 3x3
Correspondence grouping	ε =200 times the $\boldsymbol{P}_{k+1}$ resolution / minimum cluster size 10
Kalman filtering	$\sigma_v = 1$ / $\sigma_w = 5 \cdot 10^{-3}$ / number of iterations equal to the cardinality of Ω
H∞ filtering	$dt = 10^{-5}$ / $g = 0.1$ / number of iterations equal to the cardinality of Ω
OUR-CVFH	5° angular threshold / curvature threshold 1, axis ratio 0.8
Spin Images	description radius 0.02 / 8 resolution bins
ICP	point-to-point variant / 1% translational tolerance / max iterations 1000
Cardinality threshold	3

3.2 Odometry trials

3.2.1 SLA Scenario

This is a constant *Target* pose scenario where the *Source* – *Target* range is decreasing, simulating the approaching phase of the *Source* towards the *Target* platform. Most accurate odometry is provided by the GFTT keypoint detector combined with the KAZE feature descriptor regardless of the recursive filtering method used. Indeed, the GFTT / KAZE combined with Kalman attains 0.354m drift (1.598% translational error) and if combined with H∞ it provides 0.355m (1.602%). Lowest accuracy is delivered by the FH / FREAK and FH / BRISK combinations, regardless of the recursive filtering method used. This is because neither of the binary descriptors provide adequate feature matches and therefore at most *Target* pose instances our algorithm preserves the initialization value $R = R_0^*$, imposing the FH / FREAK and FH / BRISK combinations to be constrained close to the initial *X*,*Y*,*Z* coordinates of this trial. The low number of feature matches attained by both binary descriptors confirms [51]. Interestingly, both recursive filtering methods provide similar results when combined with the same keypoint detection and feature description method, highlighting the importance of selecting a robust keypoint detection and feature description combination. Table 3 presents the performance metrics on the SLA scenario, while Fig. 4 illustrates the corresponding odometry trajectories.

In terms of rotational accuracy, all combinations perform equally well attaining very low errors for Kalman and H∞ filtering, which are in the order of 10^{-8} and 10^{-3} °/m, respectively. One of the major contributions of the proposed architecture is the very low processing time required. Indeed, the computational burden of each method is in the order of milliseconds validating the capability of the suggested odometry architecture to fully exploit the low processing cost of the 2D keypoint detection and feature description methods. It should be noted that processing time includes not only the keypoint detection, feature description, geometric consistency checks and recursive filtering processes, but also the 3D to multi-2.5D remapping and the multi-2.5D to 3D back-projection processes.

Compared to the competitor odometry solutions evaluated in this paper, the proposed architecture in most combinations attains at least one order of magnitude better performance in all metrics. An exception is only the processing time of ICP, which is only 40 milliseconds faster compared to the fastest combination of the proposed architecture. However, the translational and the rotational error provided by ICP are much higher compared to any combinations of the architecture suggested. In fact, even though the LIDAR point cloud acquisition rate is large enough to provide a small frame-to-frame *Target* pose change, yet ICP still fails to properly register the two successive point clouds. Regarding OUR-CVFH / ICP, it lacks an appealing performance because it fails to cluster the *Target*'s surfaces and thus it considers the entire *Target* as a single cluster and automatically degrades to the less accurate VFH technique. Main reason for OUR-CVFH failing to cluster the *Target* platform is the relative pose of the latter as observed by the LIDAR sensor in combination with the varying *Source* – *Target* distance, forcing the *Target* platform to comprise of connected flat surfaces at most instances. Spin Images / ICP also lack of a high performance due to the symmetric and mostly flat surfaces of the *Target* platform affecting the descriptiveness and robustness of the Spin Image feature descriptor. The rotational error of all competitor methods is approximately 2.6 degrees per meter (°/m), indicating that ICP, which is the common module of all three techniques evaluated has a great impact on the rotational error. In terms of computational burden, ICP and OUR-CVFH/ ICP have a processing burden in the order of milliseconds. In contrast, Spin Images with ICP require the highest processing time among all methods evaluated including the suggested architecture. This is because, in current literature [28], Spin Images is not combined with a 3D keypoint detector and thus all *Target* vertices are encoded. Additionally, Spin Images is a 3D local description method which requires establishing a reference axis for each described keypoint, imposing an additional processing burden.

From Table 3 it is evident that the suggested architecture, apart from the FH / FREAK and FH / BRISK, is considerably more accurate than any competitor technique. The proposed architecture is both accurate and computationally efficient for the following reasons; first, it employs 2D keypoint detection and description methods that are unarguably robust to minor scale and rotational changes that are present in the point cloud projections of sequential *Target* pose instances, second, recursive filtering is designed for robustness against noise and outliers, and third, the 2D methods employed are considerably faster to execute compared to their 3D counterparts [35].

Table 3.
Performance metrics for the SLA scenario (Top performance is highlighted in bold)

	drift (m)	T_{error} (%)	Max error (m)			Average error (m)			Rotational error (°/m)	Processing time (s)
			X	Y	Z	X	Y	Z		
					***Kalman* recursive filtering**					
FH / SURF	1.151	5.183	0.954	1.248	**0.231**	0.285	0.420	**0.103**	**$3.99\ 10^{-8}$**	**0.281**
FH / FREAK	14.969	67.419	0.055	10.608	10.561	0.035	5.371	5.335	$2.36\ 10^{-8}$	0.374
FH / BRISK	15.504	69.828	**0.014**	10.960	10.966	**0.004**	5.511	5.513	$2.36\ 10^{-8}$	0.801
FH / KAZE	0.735	3.312	0.674	0.969	**0.231**	0.103	0.237	**0.103**	**$3.99\ 10^{-8}$**	0.286
GFTT / SURF	0.482	2.175	0.476	0.790	0.246	0.079	0.215	0.113	$1.77\ 10^{-5}$	0.355
GFTT / FREAK	1.636	7.372	1.106	1.400	0.643	0.384	0.532	0.112	$2.36\ 10^{-8}$	0.395
GFTT / BRISK	1.316	5.928	0.501	0.939	0.921	0.094	0.246	0.186	$2.36\ 10^{-8}$	0.741
GFTT / KAZE	**0.354**	**1.598**	0.431	**0.446**	0.246	0.244	**0.117**	0.112	$1.49\ 10^{-8}$	0.363
					***H∞* recursive filtering**					
FH / SURF	1.134	5.109	0.943	1.240	**0.219**	0.278	0.415	**0.095**	0.005	**0.281**
FH / FREAK	14.968	67.413	0.055	10.607	10.560	0.035	5.370	5.334	**0.001**	0.374
FH / BRISK	15.504	69.827	**0.014**	10.960	10.966	**0.004**	5.511	5.513	**0.001**	0.801
FH / KAZE	0.717	3.233	0.663	0.960	0.219	0.097	0.231	**0.095**	0.005	0.286
GFTT / SURF	0.472	2.127	0.466	0.790	0.236	0.075	0.215	0.106	0.005	0.357
GFTT / FREAK	1.616	7.279	1.095	1.388	0.631	0.377	0.524	0.104	0.004	0.395
GFTT / BRISK	1.299	5.851	0.490	0.927	0.909	0.095	0.239	0.179	0.004	0.741
GFTT / KAZE	**0.355**	**1.602**	0.441	**0.445**	0.236	0.251	**0.117**	0.105	0.005	0.364
					competitor schemes					
ICP	5.629	25.352	2.981	3.326	3.685	1.703	1.542	2.071	2.609	0.239
OUR-CVFH / ICP	5.631	25.361	10.541	10.857	36.903	2.699	2.789	6.618	2.620	0.953
Spin Images / ICP	5.631	25.361	16.402	16.537	69.554	6.839	6.976	24.229	2.620	391.312

3.2.2 EOI Scenario

This scenario considers a frame-to-frame varying *Target* pose at a fixed *Source* – *Target* distance, simulating the *Source* platform orbiting around the *Target* platform. For the proposed odometry technique, the hierarchy of the top performing combinations is maintained with the GFTT / KAZE achieving 0.947m drift (0.325%). Similar to scenario one, both recursive filtering methods provide an equally accurate odometry trajectory. Regarding the rotational error, all methods under both recursive filtering schemes perform equally well attaining a small error in the order of 10^{-3} °/m. In contrast to the SLA scenario, the majority of methods evaluated afford a considerably smaller processing burden and this is because of the point cloud cardinality of the *Target* platform which in this trial is much smaller compared to the SLA scenario. Table 4 presents the performance metrics on the EOI scenario, while Fig. 5 illustrates the corresponding odometry trajectories.

Also, in terms of translational and rotational error, the competitor methods attain an inferior performance compared to the suggested architecture. An exception is the processing requirement that is of the same order compared to the solution presented in this work.

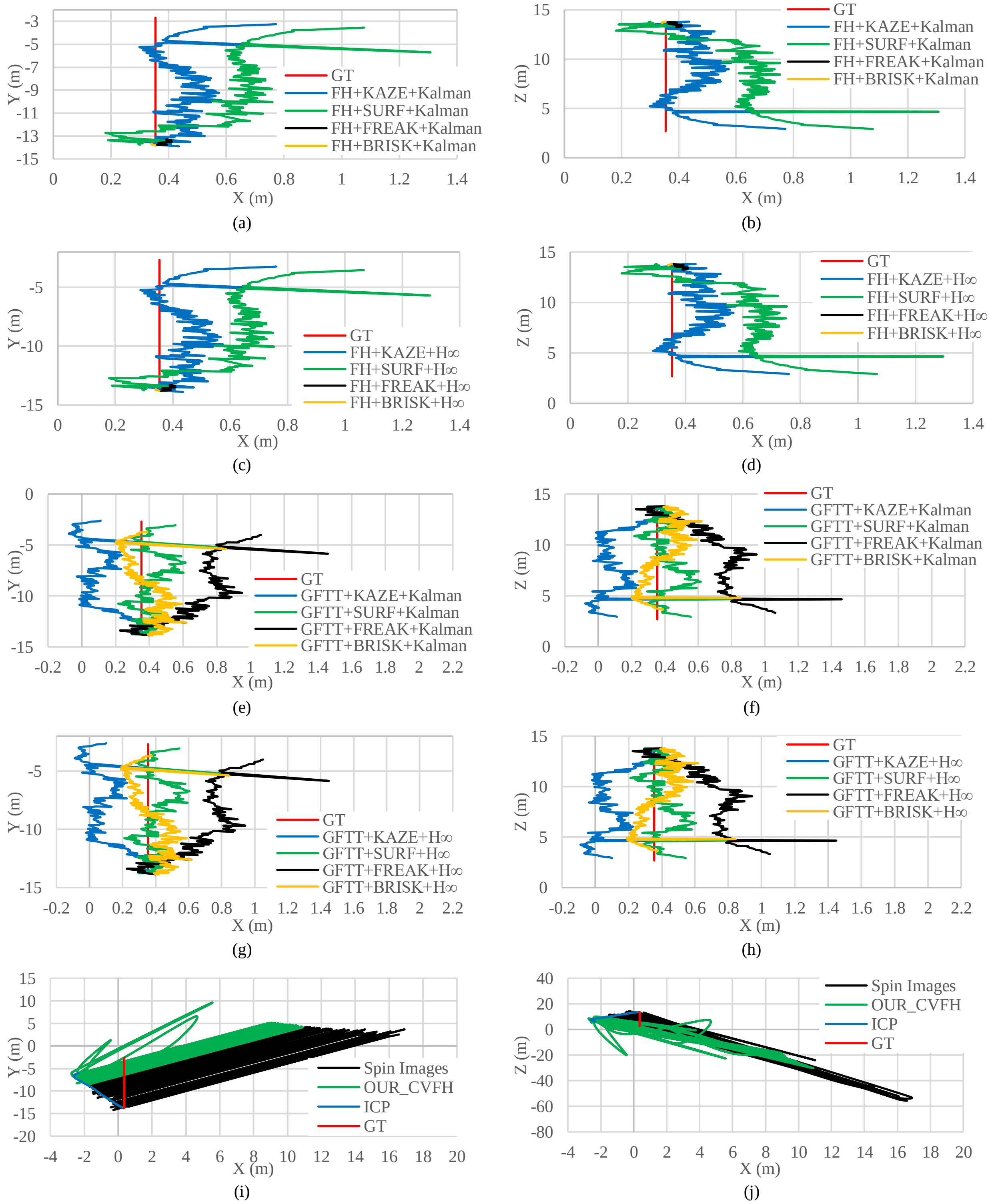

Fig. 4. 2D odometry plots for the SLA scenario (a)-(h) proposed architecture under various configurations, (i)-(j) competitor odometry methods

Table 4.
Performance metrics for the EOI scenario (Top performance is highlighted in bold)

	drift (m)	T_{error} (%)	Max error (m)			Average error (m)			Rotational error (°/m)	Processing time (s)
			X	Y	Z	X	Y	Z		
					***Kalman* recursive filtering**					
FH / SURF	2.601	0.892	3.013	4.414	1.067	1.164	1.026	0.496	**0.002**	**0.053**
FH / FREAK	9.241	3.171	15.521	47.039	51.061	9.0726	22.479	23.667	**0.002**	0.128
FH / BRISK	1.577	0.541	14.096	55.732	56.893	8.544	27.162	27.73	**0.002**	0.482
FH / KAZE	2.298	0.788	3.129	1.823	1.068	1.062	0.641	0.364	**0.002**	0.057
GFTT / SURF	1.486	0.509	2.462	1.649	1.229	0.740	0.696	0.505	**0.002**	0.096
GFTT / FREAK	3.079	1.056	3.440	4.412	2.964	0.8032	0.9022	0.797	**0.002**	0.122
GFTT / BRISK	4.607	1.581	6.694	6.242	3.501	1.983	1.718	1.990	**0.002**	0.481
GFTT / KAZE	**0.947**	**0.325**	2.500	2.454	0.918	0.566	1.085	0.390	**0.002**	0.094
					***H∞* recursive filtering**					
FH / SURF	2.639	0.905	2.995	4.455	1.025	1.161	1.033	0.484	0.003	**0.053**
FH / FREAK	9.254	3.175	15.512	47.030	51.053	9.070	22.475	23.660	**0.002**	0.128
FH / BRISK	1.581	0.542	14.097	55.731	56.891	8.543	27.161	27.729	**0.002**	0.481
FH / KAZE	2.306	0.791	3.111	1.780	1.060	1.057	0.630	0.360	0.003	0.058
GFTT / SURF	1.532	0.525	2.444	1.601	1.150	0.740	0.708	0.468	0.003	0.098
GFTT / FREAK	3.085	1.058	3.421	4.466	2.665	0.806	0.901	0.792	0.003	0.123
GFTT / BRISK	4.605	1.580	6.725	6.236	3.496	1.994	1.705	1.998	**0.002**	0.481
GFTT / KAZE	**0.909**	**0.312**	2.500	2.401	0.822	0.568	1.059	0.371	0.003	0.095
					competitor schemes					
ICP	50.320	17.266	35.619	9.370	59.270	12.575	3.752	29.812	0.199	0.030
OUR-CVFH / ICP	50.312	17.263	35.667	9.199	58.965	12.557	3.755	29.744	0.200	0.163
Spin Images / ICP	35.661	12.236	53.077	102.206	183.474	13.926	8.599	36.730	0.148	0.672

3.2.3 SSK Scenario

This scenario simulates the case where the *Source* platform is relatively stationary against the *Target* platform. Even though this can be considered as a low complexity scenario, it nevertheless is the last part of a complete space trajectory and therefore we investigate it. Table 5 presents the performance metrics on the SSK scenario. Regarding the proposed odometry technique, all evaluated combinations attain a very low drift. Even though both binary descriptors, i.e. FREAK and BRISK, combined with any of the keypoint detectors and recursive filtering methods evaluated achieve zero drift, the results for these two descriptors are ostensive. This is because for the *Source* – *Target* distance examined in this scenario, both binary descriptors do not manage to provide any feature matches. Therefore, the suggested pipeline (Fig. 2) inputs the initialization value $R = R_0^*$ to Eq. (1), and thus ultimately it remains at the initial X,Y,Z position. In terms of processing efficiency, all combinations attain a low execution time.

For this scenario, ICP also presents an appealing option providing only a small drift and a low computational burden. Although OUR-CVFH/ ICP provides a low drift, it imposes a quite high computational burden neglecting it from an appealing near-real-time solution. Finally, despite the combination of Spin Images with ICP being highly accurate, it has an extremely high processing requirement neglecting it from an optimum odometry solution.

It should be noted that, in any case, since the ground-truth translation between the initial and the end-point of the *Source* platform position coincide, T_{error} and rotational error per meter travelled metrics are not applicable.

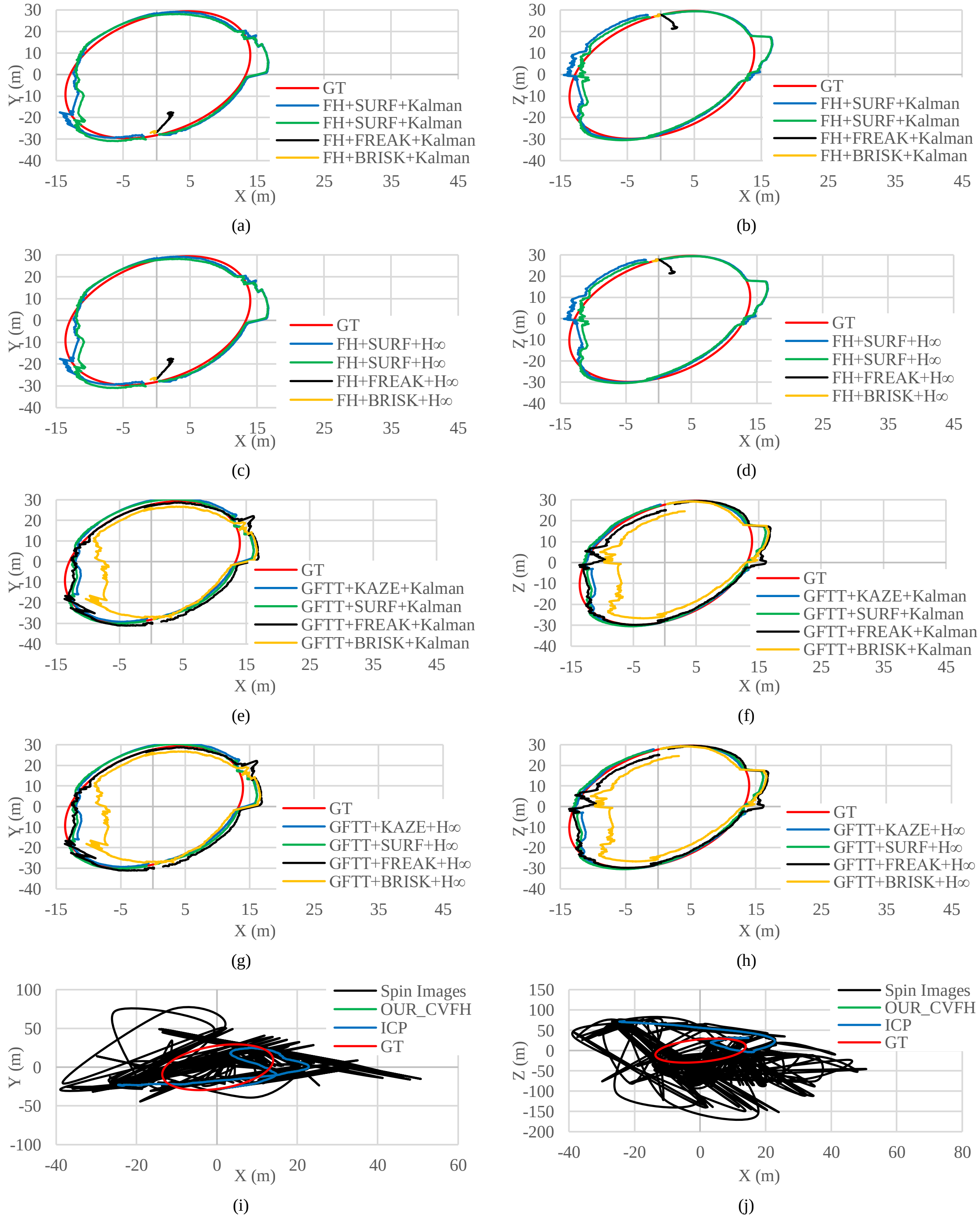

Fig. 5. 2D odometry plots for the EOI scenario (a)-(h) proposed architecture under various configurations, (i)-(j) competitor odometry methods

Table 5.
Performance metrics for the SSK scenario

	drift (m)	T_{error} (%)	Max error (m) X	Max error (m) Y	Max error (m) Z	Average error (m) X	Average error (m) Y	Average error (m) Z	Rotational error (°/m)	Processing time (s)
					***Kalman* recursive filtering**					
FH / SURF	6 10^{-3}	-	5 10^{-3}	3 10^{-3}	5 10^{-4}	10^{-3}	8 10^{-4}	10^{-4}	-	0.817
FH / FREAK	0	-	0	0	0	0	0	0	-	1.125
FH / BRISK	0	-	2 10^{-9}	10^{-8}	10^{-8}	2 10^{-9}	10^{-8}	10^{-8}	-	1.205
FH / KAZE	10^{-8}	-	2 10^{-9}	10^{-8}	10^{-8}	2 10^{-9}	10^{-8}	10^{-8}	-	0.893
GFTT / SURF	6 10^{-6}	-	10^{-6}	10^{-8}	6 10^{-6}	7 10^{-7}	10^{-8}	3 10^{-6}	-	0.855
GFTT / FREAK	10^{-8}	-	2 10^{-9}	10^{-8}	10^{-8}	2 10^{-9}	10^{-8}	10^{-8}	-	0.948
GFTT / BRISK	10^{-8}	-	0	0	0	0	0	0	-	1.274
GFTT / KAZE	6 10^{-6}	-	2 10^{-9}	10^{-8}	6 10^{-5}	2 10^{-9}	10^{-8}	3 10^{-6}	-	0.886
					***H∞* recursive filtering**					
FH / SURF	6 10^{-3}	-	5 10^{-3}	3 10^{-3}	5 10^{-4}	10^{-3}	9 10^{-4}	10^{-4}	-	0.817
FH / FREAK	0	-	0	0	0	0	0	0	-	1.120
FH / BRISK	0	-	4 10^{-5}	7 10^{-4}	2 10^{-5}	2 10^{-5}	3 10^{-4}	10^{-5}	-	1.201
FH / KAZE	7 10^{-4}	-	9 10^{-5}	7 10^{-4}	10^{-5}	2 10^{-5}	3 10^{-4}	6 10^{-6}	-	0.893
GFTT / SURF	7 10^{-4}	-	4 10^{-5}	7 10^{-4}	4 10^{-5}	2 10^{-5}	3 10^{-4}	2 10^{-5}	-	0.856
GFTT / FREAK	7 10^{-4}	-	4 10^{-5}	7 10^{-4}	2 10^{-5}	2 10^{-5}	3 10^{-4}	10^{-5}	-	0.948
GFTT / BRISK	7 10^{-4}	-	0	0	0	0	0	00	-	1.274
GFTT / KAZE	7 10^{-4}	-	4 10^{-5}	7 10^{-4}	10^{-5}	2 10^{-5}	3 10^{-4}	9 10^{-6}	-	0.887
					competitor schemes					
ICP	7 10^{-6}	-	4 10^{-6}	10^{-8}	6 10^{-6}	2 10^{-6}	10^{-8}	3 10^{-6}	-	0.283
OUR-CVFH / ICP	4 10^{-6}	-	4 10^{-6}	2 10^{-7}	4 10^{-6}	10^{-6}	2 10^{-7}	10^{-6}	-	33.242
Spin Images / ICP	0	-	0	0	0	0	0	0	-	2700.00

3.3 Discussion

In Section 3, for each scenario we present the overall performance of several keypoint detection, feature description, and recursive filtering combinations. Therefore, for a more comprehensive analysis, Table 6 presents the overall performance attained by each scheme on an individual basis, e.g. overall performance of each keypoint detection method independent of the feature description and recursive filtering method.

From the results presented in Table 6, it can be concluded that GFTT keypoints contribute to a more accurate odometry solution. This is because when the point cloud is remapped from the 3D to the 2D space, the number of corners detected by GFTT are more compared to the blob-type keypoints detected by FH. This performance is highly related to the quantization factor of Eq. (5) because it defines the level of details that each projection encloses. However, since in this work odometry accuracy and processing efficiency are of equal importance, we choose a relatively small q_f value that affords high-speed odometry but favours the corner type detectors. Increasing q_f creates sparse 2.5D projections negatively influencing the performance of the 2D keypoint detection and feature description methods employed. Reducing q_f on the other hand prohibits the 2D keypoint detectors from providing repeatable keypoints. Finally, in terms of rotational error and computational requirements, both keypoint detection methods attain similar results.

Considering the performance of the feature description methods, KAZE and SURF are the most appealing ones attaining lowest drift at a relatively low computational cost, which is one of the lowest presented in our experiments. Their performance is similar because both descriptors belong to the same category, i.e. floating-point, and share the same description method, i.e. Gaussian-weighted Haar wavelet responses, with the difference being that SURF has a linear and KAZE a non-linear scale-space description scheme. Due to this difference, KAZE affords a lower drift but also a mildly larger processing requirement. Similarly, BRISK and FREAK achieve similar results as both are binary and rely on a sampling pattern comprising of concentric circular patches centred at a keypoint. Their difference is that BRISK has a constant sampling point density, while FREAK has a variable one with the sampling points being denser near the centre becoming exponentially less dense further away from the centre. However, in the context of multi-projecting point clouds, this variable sampling point density does not provide any performance gain to FREAK. In fact, the fixed sampling pattern of both binary techniques does not encode the keypoints detected on the 2.5D projection images robustly, and thus it can be concluded that these descriptors are less suitable for 2.5D imagery. Finally, from Table 6 we conclude that given the keypoint detection and feature description method's capability to provide good matches, the selection of the recursive filtering method remains less important. Indeed, the overall performance of the two recursive methods evaluated is very similar for the scenarios of this work.

Table 6.
Performance analysis (Top performance is highlighted in bold)

module	method	drift (m)	T_{error} (%)	Rotational error (°/m)	Processing time (s)
Keypoint detection	FH	4.008	18.884	**$1.66\ 10^{-3}$**	**0.422**
	GFTT	**1.158**	**2.555**	$2.37\ 10^{-3}$	0.552
Feature description	SURF	0.960	2.179	$2.42\ 10^{-3}$	0.410
	FREAK	4.821	19.743	$1.64\ 10^{-3}$	**0.370**
	BRISK	3.833	19.460	**$1.59\ 10^{-3}$**	0.737
	KAZE	**0.719**	**1.495**	$2.42\ 10^{-3}$	0.431
Filtering method	Kalman	**2.583**	**10.730**	**$8.97\ 10^{-4}$**	**0.487**
	$H\infty$	**2.583**	**10.730**	$3.14\ 10^{-3}$	**0.487**

We also assess the interplay between the feature matching and the geometric consistency checks (GCC) module of our odometry architecture by discarding the latter and setting to eq. (12) a fixed threshold of 0.8 [38,52]. For better readability we only assess the SLA scenario, with the corresponding results presented in Table 7. Our findings demonstrate that the top performing combination utilizing the GFTT keypoint detector with the KAZE feature descriptor, indeed benefits from using the GCC, with the translational improvement being approximately 55% for both recursive filtering schemes, i.e. Kalman and $H\infty$. In terms of rotation, utilizing a GCC has a minor impact that is less than 1% and regarding computational efficiency, the GCC imposes an additional 26% processing time. However, as presented in Table 3, the total computational burden including the GCC module is only 363ms and thus the extra 95ms required by the GCC module are considered as minor drawback.

In Table 7 we also demonstrate that the keypoint detection and feature description methods have a great interplay with the GCC module. In fact, we show that when GFFT is combined with GCC, it attains a translational performance gain regardless of the feature descriptor and recursive filtering scheme used, while the impact on the rotational error is minor. Accordingly, SURF is the most affected descriptor with BRISK to follow. On the contrary, the FH keypoint detector is more robust and thus, depending on the feature descriptor that SURF is combined with, neglecting the GCC module may have a greater impact. This is because the *Target* has a frame-to-frame 3D rotation imposing some of the keypoints detected on the 2.5D images being transferred from the background to the foreground and vice versa leading to a local zooming effect [53]. Given that GFTT is prone to scale changes and to affine transformations, the frame-to-frame keypoints detected in all 2.5D projections include both true and false matching correspondences affecting accordingly the performance of the feature descriptor. However, the GCC module evaluates the geometric consistency of the correspondences discarding the majority of the false matches and ultimately provides an appealing odometry. On the contrary, FH is robust to scale changes and to out-of-plane *Target* rotations of up to 30° affording a great number of true matching and fewer false matching correspondences. Hence the strict threshold within the GCC module force true matching correspondences to be discarded, reducing the number of iterations of the recursive filter and thus imposing it not to properly settle. In simple terms, GFTT provides one order of magnitude more keypoints than FH, where only a few of these keypoints are true matching correspondences and GCC assists into discarding the false matching ones. However, it should be noted that FH/ SURF, which is the most accurate combination among the ones relying on FH when a GCC scheme is neglected, is still inferior to the GFTT/KAZE that uses a GCC module.

Table 7.
Performance assessment on the SLA scenario neglecting correspondence grouping (all metrics in %, positive refers to performance gain by using correspondence grouping and negative refers to loss)

	drift	T_{error}	Rotational error	Processing time	drift	T_{error}	Rotational error	Processing time
	***Kalman* recursive filtering**				***H∞* recursive filtering**			
FH / SURF	-5.26	-5.26	-40.76	-9.65	-4.73	-4.73	-0.99	-9.67
FH / FREAK	-0.18	-0.18	**0.00**	-36.22	-0.18	-0.18	0.16	-36.26
FH / BRISK	**0.05**	**0.05**	**0.00**	-8.05	**0.05**	**0.05**	**0.66**	-8.06
FH / KAZE	-14.56	-14.56	-40.76	-14.80	-15.03	-15.03	-0.12	-14.83
GFTT / SURF	**115.32**	**115.32**	-0.87	-23.92	**117.96**	**117.96**	**0.76**	-23.97
GFTT / FREAK	**50.53**	**50.53**	**0.00**	-32.78	**51.48**	**51.48**	-1.84	-32.79
GFTT / BRISK	**83.81**	**83.81**	**0.00**	-15.39	**85.40**	**85.40**	-1.89	-15.40
GFTT / KAZE	**54.89**	**54.89**	-0.84	-26.22	**55.70**	**55.70**	**0.08**	-26.28

For completeness, it is worth noting that although the suggested architecture presents an overall appealing odometry performance, it poses the following limitations:

a. The quantization factor q_f has to be tuned based on the *Target* point cloud resolution. Properly tuning q_f is important as it defines the 3D to multi-2D remapping and ultimately affects the performance of the 2D keypoint detectors and descriptors, and thus the accuracy of the proposed odometry architecture. However, tuning q_f is done offline neglecting any impact during the odometry process.

b. *Target* tumbling should not exceed the robustness of the 2D method's used affine transformation. This is the case where the *Target* undergoes a 3D rotation creating on at least one of the 2.5D projection planes a large out-of-plane projection. This is due to the XYZ_{LIDAR} reference frame and the translated XYZ_{Target} reference frame having axes that are fixed on the LIDAR sensor onboard the *Source*. However, this is only for the case where parts of the *Target* have not shifted yet from one 2.5D projection plane to another and thus remain on the same 2.5D plane but under a large affine transformation.

4. Conclusion

LIDAR based odometry for space relative navigation is a challenging task. Given the cost and the importance of space missions, highly accurate and processing efficient odometry becomes mandatory. Driven by these requirements and the performance of current methods, we propose a high-speed and robust LIDAR based odometry architecture appropriate for space odometry that combines the advantages of the 3D and 2D data domains along with the robustness of recursive filtering. Specifically, our architecture attains a high odometry accuracy by exploiting the advantages of 3D LIDAR data and recursive filtering, while in parallel it achieves a low computational burden by transforming the odometry problem from the 3D space into multiple 2D ones that involve 2.5D image projections of the 3D data.

Trials evaluate several current state-of-the-art 2D keypoint detection, local feature description and recursive filtering techniques on several simulated scenarios that involve a realistic *Target* space platform. Results demonstrate that the proposed architecture affords higher odometry accuracy and a lower processing burden compared to current methods. Specifically, highest performance is gained by the GFTT/ KAZE combination that manages one order of magnitude more accurate odometry and a very low processing burden, which depending on the competitor method, may exceed one order of magnitude faster odometry computation. Spurred by the appealing performance of the proposed architecture, future work shall include implementation on space-graded FPGA boards and extended to provide pose initialization.

Conflict of interest statement

None declared

Funding Sources

This research was supported by the European Union's Horizon 2020 research and innovation program, under project "Integrated 3D Sensors (I3DS)" with grant agreement No 730118.

Acknowledgements

The authors would like to thank Thales Alenia Space (France) for providing simulated data.

References

[1] M.S. Krämer, S. Hardt, K. Kuhnert, Image Features in Space - Evaluation of Feature Algorithms for Motion Estimation in Space Scenarios, in: Proc. 7th Int. Conf. Pattern Recognit. Appl. Methods, SCITEPRESS - Science and Technology Publications, Funchal, Madeira, Portugal, 2018: pp. 300–308. doi:10.5220/0006555303000308.

[2] D. Rondao, N. Aouf, Multi-View Monocular Pose Estimation for Spacecraft Relative Navigation, 2018 AIAA Guid. Navig. Control Conf. (2018). doi:10.2514/6.2018-2100.

[3] L. Li, J. Lian, L. Guo, R. Wang, Visual odometry for planetary exploration rovers in sandy terrains, Int. J. Adv. Robot. Syst. 10 (2013) 1–7. doi:10.5772/56342.

[4] T. Tykkala, A.I. Comport, A dense structure model for image based stereo SLAM, in: Robot. Autom. (ICRA), 2011 IEEE Int. Conf.,

Shangai, China, 2011: pp. 1758–1763. doi:10.1109/ICRA.2011.5979805.

[5] Yang Cheng, M. Maimone, L. Matthies, Visual Odometry on the Mars Exploration Rovers, in: 2005 IEEE Int. Conf. Syst. Man Cybern., Waikoloa, HI, USA, 2006: pp. 903–910. doi:10.1109/ICSMC.2005.1571261.

[6] M. Maimone, Y. Cheng, L. Matthies, Two years of Visual Odometry on the Mars Exploration Rovers, J. F. Robot. 24 (2007) 169–186. doi:10.1002/rob.20184.

[7] O. Yılmaz, N. Aouf, L. Majewski, M. Sanchez-Gestido, G. Ortega, Using infrared based relative navigation for active debris removal, in: 10th Int. ESA Conf. Guid. Navig. Control Syst., Salzburg, Austria, 2017: pp. 1–16.

[8] B. Naasz, M. Moreau, Autonomous RPOD technology challenges for the coming decade, Adv. Astronaut. Sci. 144 (2012) 403–425.

[9] R. Opromolla, M.Z. Di Fraia, G. Fasano, G. Rufino, M. Grassi, Laboratory test of pose determination algorithms for uncooperative spacecraft, in: 4th IEEE Int. Work. Metrol. AeroSpace, Metroaerosp. 2017 - Proc., Padua, Italy, 2017: pp. 169–174. doi:10.1109/MetroAeroSpace.2017.7999558.

[10] J.O. Woods, J.A. Christian, Lidar-based relative navigation with respect to non-cooperative objects, Acta Astronaut. 126 (2016) 298–311. doi:10.1016/j.actaastro.2016.05.007.

[11] R. Volpe, G. Palmerini, M. Sabatini, Monocular and Lidar Based Determination of Shape , Relative Attitude and Position of a Non-Cooperative , Unknown Satellite, in: Int. Astronaut. Congr. (IAC 2017), Adelaide, Australia, 2017: pp. 25–29.

[12] H. Gómez Martínez, G. Giorgi, B. Eissfeller, Pose estimation and tracking of non-cooperative rocket bodies using Time-of-Flight cameras, Acta Astronaut. 139 (2017) 165–175. doi:10.1016/j.actaastro.2017.07.002.

[13] J. Galante, J. Van Eepoel, M. Strube, N. Gill, M. Gonzalez, A. Hyslop, B. Patrick, Pose Measurement Performance of the Argon Relative Navigation Sensor Suite in Simulated-Flight Conditions, in: AIAA Guid. Navig. Control Conf., American Institute of Aeronautics and Astronautics, Reston, Virigina, 2012: pp. 1–26. doi:10.2514/6.2012-4927.

[14] J.L. Sell, A. Rhodes, J.O. Woods, J.A. Christian, T. Evans, Pose Performance of LIDAR-Based Navigation for Satellite Servicing, AIAA/AAS Astrodyn. Spec. Conf. (2014) 1–14. doi:10.2514/6.2014-4360.

[15] J. Song, Sliding window filter based unknown object pose estimation, in: 2017 IEEE Int. Conf. Image Process., IEEE, 2017: pp. 2642–2646. doi:10.1109/ICIP.2017.8296761.

[16] R. Opromolla, G. Fasano, G. Rufino, M. Grassi, Spaceborne LIDAR-based system for pose determination of uncooperative targets, in: 2014 IEEE Int. Work. Metrol. Aerospace, Metroaerosp. 2014 - Proc., Benevento, Italy, 2014: pp. 265–270. doi:10.1109/MetroAeroSpace.2014.6865932.

[17] R. Opromolla, G. Fasano, G. Rufino, M. Grassi, A model-based 3D template matching technique for pose acquisition of an uncooperative space object, Sensors (Switzerland). 15 (2015) 6360–6382. doi:10.3390/s150306360.

[18] O. Kechagias-stamatis, N. Aouf, H∞ LIDAR Odometry for Spacecraft Relative Navigation, IET Radar, Sonar Navig. (2019). doi:10.1049/iet-rsn.2018.5354.

[19] R. Opromolla, G. Fasano, G. Rufino, M. Grassi, A review of cooperative and uncooperative spacecraft pose determination techniques for close-proximity operations, Prog. Aerosp. Sci. 93 (2017) 53–72. doi:10.1016/j.paerosci.2017.07.001.

[20] M. Zarei-Jalalabadi, S.M.-B. Malaek, Motion estimation of uncooperative space objects: A case of multi-platform fusion, Adv. Sp. Res. 62 (2018) 2665–2678. doi:10.1016/j.asr.2018.07.031.

[21] C. Bonnal, J.M. Ruault, M.C. Desjean, Active debris removal: Recent progress and current trends, Acta Astronaut. 85 (2013) 51–60. doi:10.1016/j.actaastro.2012.11.009.

[22] Y. Guo, F. Sohel, M. Bennamoun, M. Lu, J. Wan, Rotational Projection Statistics for 3D Local Surface Description and Object Recognition, Int. J. Comput. Vis. 105 (2013) 63–86. doi:10.1007/s11263-013-0627-y.

[23] R.P. Kornfeld, R.L. Bunker, G.C. Cucullu, J.C. Essmiller, F.Y. Hadaegh, C. Christian Liebe, C.W. Padgett, E.C. Wong, New millennium ST6 autonomous rendezvous experiment (ARX), in: 2003 IEEE Aerosp. Conf. Proc. (Cat. No.03TH8652), IEEE, 2003: pp. 1–380. doi:10.1109/AERO.2003.1235067.

[24] M. Estébanez Camarena, L.M. Feetham, A. Scannapieco, N. Aouf, FPGA-based multi-sensor relative navigation in space: Preliminary analysis in the framework of the I3DS H2020 project, in: 69 Th Int. Astronaut. Congr. (IAC), International Astronautical Federation, Bremen, 2018: pp. 1–8.

[25] P.J. Besl, N.D. McKay, A method for registration of 3-D shapes, IEEE Trans. Pattern Anal. Mach. Intell. 14 (1992) 239–256. doi:10.1109/34.121791.

[26] R. Opromolla, G. Fasano, G. Rufino, M. Grassi, Uncooperative pose estimation with a LIDAR-based system, Acta Astronaut. 110 (2015) 287–297. doi:10.1016/j.actaastro.2014.11.003.

[27] L. Liu, G. Zhao, Y. Bo, Point cloud based relative pose estimation of a satellite in close range, Sensors (Switzerland). 16 (2016). doi:10.3390/s16060824.

[28] A. Rhodes, E. Kim, J.A. Christian, T. Evans, LIDAR-based Relative Navigation of Non-Cooperative Objects Using Point Cloud Descriptors, in: AIAA/AAS Astrodyn. Spec. Conf., American Institute of Aeronautics and Astronautics, Reston, Virginia, 2016. doi:10.2514/6.2016-5517.

[29] A. Aldoma, F. Tombari, R.B. Rusu, M. Vincze, OUR-CVFH – Oriented, Unique and Repeatable Clustered Viewpoint Feature Histogram for Object Recognition and 6DOF Pose Estimation, in: Lect. Notes Comput. Sci. (Including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), 2012: pp. 113–122. doi:10.1007/978-3-642-32717-9_12.

[30] A.E. Johnson, M. Hebert, Using spin images for efficient object recognition in cluttered 3D scenes, IEEE Trans. Pattern Anal. Mach. Intell. 21 (1999) 433–449. doi:10.1109/34.765655.

[31] A.P. Rhodes, J.A. Christian, T. Evans, A Concise Guide to Feature Histograms with Applications to LIDAR-Based Spacecraft Relative Navigation, J. Astronaut. Sci. 64 (2017) 414–445. doi:10.1007/s40295-016-0108-y.

[32] A.B. Dietrich, J.W. McMahon, Robust Orbit Determination with Flash Lidar Around Small Bodies, J. Guid. Control. Dyn. 41 (2018) 2163–2184. doi:10.2514/1.G003023.

[33] A. Dietrich, J.W. McMahon, Orbit Determination Using Flash Lidar Around Small Bodies, J. Guid. Control. Dyn. 40 (2017) 650–665. doi:10.2514/1.G000615.

[34] O. Kechagias-Stamatis, N. Aouf, D. Nam, 3D Automatic Target Recognition for UAV Platforms, in: 2017 Sens. Signal Process. Def. Conf., IEEE, London, UK, 2017: pp. 1–5. doi:10.1109/SSPD.2017.8233223.

[35] O. Kechagias-Stamatis, N. Aouf, M.A. Richardson, 3D automatic target recognition for future LIDAR missiles, IEEE Trans. Aerosp. Electron. Syst. 52 (2016) 2662–2675. doi:10.1109/TAES.2016.150300.

[36] H. Bay, A. Ess, T. Tuytelaars, L. Van Gool, Speeded-Up Robust Features (SURF), Comput. Vis. Image Underst. 110 (2008) 346–359.

doi:10.1016/j.cviu.2007.09.014.

[37] Jianbo Shi, Tomasi, Good features to track, in: Proc. IEEE Conf. Comput. Vis. Pattern Recognit. CVPR-94, IEEE Comput. Soc. Press, 1994: pp. 593–600. doi:10.1109/CVPR.1994.323794.

[38] D.G. Lowe, Distinctive image features from scale invariant keypoints, Int. J. Comput. Vis. 60 (2004) 91–110. doi:10.1023/B:VISI.0000029664.99615.94.

[39] P. Viola, M. Jones, Robust real-time face detection, Int. J. Comput. Vis. 57 (2004) 137–154.

[40] C. Harris, M. Stephens, A Combined Corner and Edge Detector, in: Procedings Alvey Vis. Conf. 1988, Alvey Vision Club, 1988: p. 23.1-23.6. doi:10.5244/C.2.23.

[41] P.F. Alcantarilla, A. Bartoli, A.J. Davison, KAZE features, in: Lect. Notes Comput. Sci. (Including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), 2012: pp. 214–227. doi:10.1007/978-3-642-33783-3_16.

[42] A. Alahi, R. Ortiz, P. Vandergheynst, FREAK: Fast Retina Keypoint, in: 2012 IEEE Conf. Comput. Vis. Pattern Recognit., IEEE, 2012: pp. 510–517. doi:10.1109/CVPR.2012.6247715.

[43] S. Leutenegger, M. Chli, R.Y. Siegwart, BRISK: Binary Robust invariant scalable keypoints, 2011 Int. Conf. Comput. Vis. (2011) 2548–2555. doi:10.1109/ICCV.2011.6126542.

[44] O. Kechagias-Stamatis, N. Aouf, G. Gray, L. Chermak, M. Richardson, F. Oudyi, Local feature based automatic target recognition for future 3D active homing seeker missiles, Aerosp. Sci. Technol. 73 (2018) 309–317. doi:10.1016/j.ast.2017.12.011.

[45] M. Muja, D.G. Lowe, Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration, in: Int. Conf. Comput. Vis. Theory Appl. (VISAPP ’09), Lisboa, Portugal, 2009: pp. 1–10. doi:10.1.1.160.1721.

[46] H. Chen, B. Bhanu, 3D free-form object recognition in range images using local surface patches, Pattern Recognit. Lett. 28 (2007) 1252–1262. doi:10.1016/j.patrec.2007.02.009.

[47] A. Amamra, N. Aouf, D. Stuart, M. Richardson, A recursive robust filtering approach for 3D registration, Signal, Image Video Process. 10 (2016) 835–842. doi:10.1007/s11760-015-0823-z.

[48] D. Simon, Kalman Filtering, Embed. Syst. Program. (2001) 72–79.

[49] Y. Guo, M. Bennamoun, F. Sohel, M. Lu, J. Wan, N.M. Kwok, A Comprehensive Performance Evaluation of 3D Local Feature Descriptors, Int. J. Comput. Vis. 116 (2016) 66–89. doi:10.1007/s11263-015-0824-y.

[50] O. Kechagias-Stamatis, N. Aouf, Histogram of distances for local surface description, in: 2016 IEEE Int. Conf. Robot. Autom., IEEE, Stockholm, Sweden, 2016: pp. 2487–2493. doi:10.1109/ICRA.2016.7487402.

[51] J. Krizaj, V. Struc, F. Mihelic, A feasibility study on the use of binary keypoint descriptors for 3D face recognition, in: Mex. Conf. Pattern Recognit., 2014: pp. 142–151. doi:10.1007/978-3-319-07491-7_15.

[52] J. Heinly, E. Dunn, J. Frahm, Comparative Evaluation of Binary Features, in: Comput. Vis. -- ECCV 2012, Springer Berlin Heidelberg, 2012: pp. 759–773. doi:10.1007/978-3-642-33709-3_54.

[53] O. Kechagias-Stamatis, N. Aouf, Fast 3D object matching with Projection Density Energy, in: 2015 23rd Mediterr. Conf. Control Autom. MED 2015 - Conf. Proc., IEEE, 2015: pp. 752–758. doi:10.1109/MED.2015.7158836.