City Research Online

City, University of London Institutional Repository

Degraded Reality: Using VR/AR to simulate visual impairments

Pete R. Jones*
UCL Institute of Ophthalmology

Giovanni Ometto†
City, University of London

ABSTRACT

The effects of eye disease cannot be depicted accurately using traditional media. Consequently, public understanding of eye disease is often poor. We present a VR/AR system for simulating common visual impairments, including disability glare, spatial distortions (Metamorphopsia), the selective blurring and filling-in of information across the visual field, and color vision deficits. Unlike most existing simulators, the simulations are informed by patients' self-reported symptoms, can be quantitatively manipulated to provide custom disease profiles, and support gaze-contingent presentation (i.e., when using a VR/AR headset that contains eye-tracking technology, such as the Fove0). Such a simulator could be used as a teaching/empathy aid, or as a tool for evaluating the accessibility of new products and environments.

Index Terms: Computing methodologies—Graphics systems and interfaces—Mixed / augmented reality; Computing methodologies—Computer graphics—Image manipulation—Image processing; Modeling and simulation—Simulation types and techniques—Real-time simulation;

1 INTRODUCTION

Vision loss due to refractive error (e.g., short-sightedness, or astigmatism) can be depicted by static images, and can be simulated using lenses. Consequently, optical impairments tend to be well understood and catered for. However, in developed countries, the leading causes of blindness are not optical, but are neurological disorders of the retina and brain, such as glaucoma, age-related macular degeneration [AMD], and diabetic retinopathy [6].

The symptoms of these 'posterior' visual impairments are often complex, and can vary between conditions, across individuals, and progressively over time. Also, unlike optical defocus, posterior impairments tend to selectively affect different parts of the visual field: for example, affecting primarily peripheral vision in the case of glaucoma and diabetic retinopathy, or primarily central vision in the case of AMD. These disorders cannot be accurately depicted by a static image, nor can they be simulated using physical prisms or lenses. Consequently, public understanding of these diseases tends to be relatively poor. This is likely part of the reason why, for example, around 50% of glaucoma cases (1 in 80 adults over 40) currently go undiagnosed in developed countries [7].

Virtual and Augmented Reality (VR/AR) could provide a novel and effective solution. By using digital image processing techniques, disorders of arbitrary complexity can be simulated. And by combining these with near real-time eye-tracking, these disorders can be localized to specific regions of the user's retina. This would allow healthy individual to experience sight-loss, in an immersive, interactive, and intuitive way. In doing so, VR/AR simulators could provide a powerful tool for improving public understanding of eye disease, and be used, for example, to educate new health-care professionals, or to evaluate the accessibility of new products or environments.

---

*e-mail: p.r.jones@ucl.ac.uk
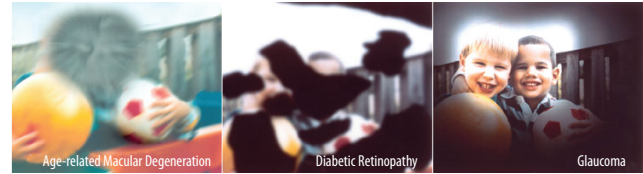†e-mail: giovanni.ometto@city.ac.uk

Figure 1: Existing simulations, provided by the US National Eye Institute (NEI). Impairments are primarily generated by superimposition and are static on the page, allowing the viewer to 'look past' them.

2 SYSTEM REQUIREMENTS AND DESIGN PRINCIPLES

For the simulator to be a useful and effective tool, the following requirements were identified:

1. **Near real-time**. The ability to update the display rapidly, with no loss or tearing of frames is crucial for user comfort (lagged or choppy displays were found to cause dizziness or nausea), and is necessary to support gaze-contingent presentation (see next).

2. **Gaze contingent**. To localize simulated-impairments on the retina, any image processing effects must be applied relative to the user's current point of gaze. This requires the ability to measure gaze, and to update the simulations in near-real-time.

3. **Portable**. A key potential application is the assessment of real-world environments, such as a patient's home, classroom or transportation networks. For this, the system must be capable of running on mobile architectures such as a smartphone.

4. **Support for AR**. To support Augmented Reality, it must be possible to apply the simulations to live content from one or more pass-through camera. Together with (1–3), this constrains the computational complexity of any algorithms used, and precludes any techniques which requires image statistics to be precomputed.

5. **Quantitative / Data-driven**. Due to the high heterogeneity in eye-disease (both between patients, and within a patient over time), any simulations must be dynamically manipulable. This would allow, for example, personalized profiles for individuals, or the use of statistical forecasting techniques to show how a given pathology may change over time or in response to treatment.

3 CURRENT APPROACHES

Visual impairments are most commonly simulated by superimposing the source image with an overlay: typically, one or more 'black blobs' of varying size, shape, and opacity (see Fig. 1). This approach is expedient, both computationally and programmatically, and it may be appropriate for a minority of visual disturbances, such as vitreous floaters and other media opacities. However, as a general solution, superimposition is incorrect: most patients rate such depictions as "poor", and instead tend to describe their vision as being blurred, warped, cloudy, discolored, jumbled, or elided [1]. At best, superimposition alone is gross simplification. At worst, such depictions are misleading, and may cause genuine symptoms to go unrecognized.

4 SYSTEM DESIGN

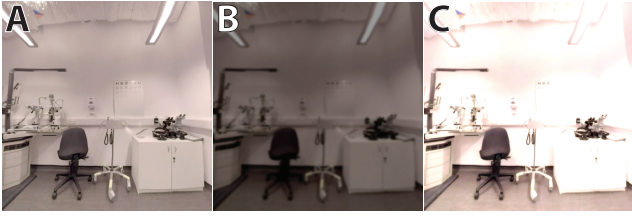Here, we detail how the most prevalent impairments were simulated.

Figure 2: Disability Glare. (**A**) The source image; (**B**) a blurred and contrast-gain modulated duplicate of the source; (**C**) Output image, computed by summing together (A) and (B).

4.1 Disability Glare

Disability glare is a sensation of excessive or uncontrolled brightness, prevalent across a wide range of eye conditions. It was simulated by additively combining the source image with a Gaussian-blurred, gain-adjusted copy of itself. The blur was computed using spatial convolution. For efficiency, the blur was implemented using two-passes of a 1D Gaussian kernel (vertical + horizontal), thus:

$$P_{h[i,j]} = \sum_{k=-m}^{m} \omega_k * \lambda P_{[i,j+k]}$$

$$P_{hv[i,j]} = \sum_{k=-m}^{m} \omega_k * P_{h[i+k,j]}$$

where $P_{[i,x]}$ is the original (i x j) matrix of pixels, λ is a scalar which controls the intensity of the glare, ω are weights from a 1D Gaussian kernel of length $2m$, normalized to sum to 1, and P_h and P_{hv} represent the pixel values after applying horizontal and horizontal+vertical blur, respectively.

This algorithm executes at a rate of $O(2N)$, versus $O(N^2)$ for a two-dimensional kernel. To further improve speed, the source image was also downsampled using bilinear interpolation, prior to blurring. The result of this operation, P_{hv} was stored in a temporary rendertexture (Fig. 2B), which was then blitted onto the main frame-buffer using additive blending. The overall effect is of saturation and loss of spatial detail in regions of high intensity (Fig Fig. 2C). The spatial extent of the glare can be manipulated by varying the standard deviation of the Gaussian kernel, while the magnitude of the glare can be manipulated using the intensity-scaling constant λ.

4.2 Blur

Forms of blur are extremely common across all types of visual impairments. Blur can be caused by a range of factors, including a sparsening of photoreceptors, or an increase in spatial pooling across neurons [9]. Unlike with the glare effect (above), spatial convolution is inappropriate as a generalized blurring algorithm for two reasons. First, because the kernel sizes required to simulated low-vision can become prohibitively large (a problem which will only be compounded in future as pixel density increases). Second, because the size of the kernel is liable to vary pixel-by-pixel, meaning that the procedural Gaussian kernel cannot be cached in advanced, and must be (re-)computed independently, for every pixel. Instead, blurring was simulated using the multiresolution pyramid approach proposed previously by Geisler and Perry [2]. In brief, the source image is converted into a multiresolution pyramid (or: *mipmap*), by repeatedly smoothing the image and downsampling by a factor of two (see Fig. 3A). The magnitude of blur can be controlled by sampling from different levels of the pyramid: interpolating between adjacent levels as required. Unlike Geisler and Perry, the pyramid was not computed explicitly, but was generated implicitly as part of the default OpenGL pipeline. The overall result is an extremely
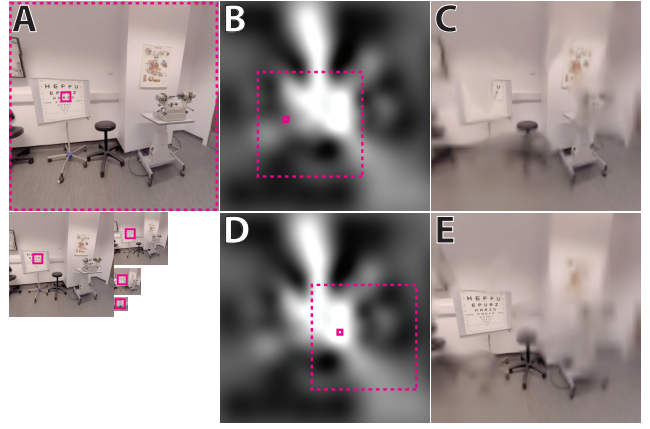

Figure 3: Blur. (**A**) Source image, including the image pyramid. Purple rectangles indicate equivalent locations in each image; (**B**) The reference texture, which specifies the level of the image pyramid to sample from. The reference was centered on the user's point of gaze – purple dashed area shows the relative location of the source image given the current point of fixation. (**C**) Output image, computed by combining (A) and (B). (**D–E**) Same as (B–C), for a different point of fixation.
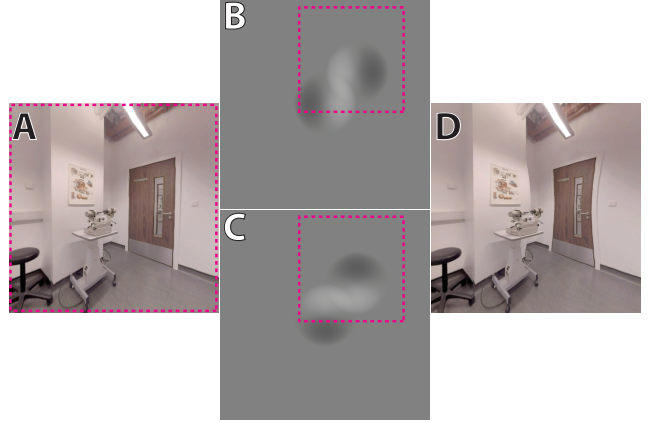

Figure 4: Spatial Distortions. (**A**) Source image. (**B**) Horizontal reference texture. (**C**) Vertical reference texture. (**D**) Output image.

efficient, blurring operation, similar to, though not identical with, a Gaussian smoothing process. The only substantive constraint of this approach is that it requires the source texture to be a power of two in size. This was achieved by using bilinear resampling as required.

The spatial extent of the blur was controlled using a reference texture (or: *resolution map*): a 24-bit texture, the pixel-wise intensity of which specified the blur magnitude for each corresponding pixel in the source image (see Fig. 3B). Crucially, the reference texture was twice the size of the source image, and was transposed prior to every screen refresh such that it was always centered on the user's current point of gaze. In this way, the blur was localized not in screen coordinates, but on the user's retina/visual-field. When simulating a particular individual's vision, the values for the reference texture were derived by interpolating between empirical visual field measurements, made using static threshold perimetry.

4.3 Spatial Distortions

Spatial distortions (or: *Metamorphopsia*) are prevalent in disorders of central vision, such as AMD and diabetic macular edema [DME]. They are caused by a range of factors, including physical swelling

of the retina, leading to photoreceptors becoming misaligned [5].

It was simulated by replacing individual pixels with additively displaced texture samples. Furthermore, non-stationary could be introduced by modulating the offset values by a periodic temporal signal. As with the Blur filter, offsets were encoded by a gaze-contingent reference texture (or: *deformation map*). However, in this case two reference textures were required – i.e., in order to express pairs of cardinal offsets in cartesian coordinates (see Fig. 4). These reference textures were generated by summing N simulated force fields. Each force was specified by four values: its center coordinates, $\{\overline{X}, \overline{Y}\}$, in pixels, its radius, r, and its magnitude, λ, where λ could be either negative (compressive) or positive (expansive). This was used to computed two matrices of additive displacement values, $\{\Delta_x, \Delta_y\}$, as follows.

For a given pixel, $P_{[i,j]}$, and each force, $k = 1 \ldots N$, the horizontal and vertical offset of the pixel from the center of the force field was computed:

$$x_k = j - \bar{X}_k$$
$$y_k = i - \bar{Y}_k$$

These values were used to compute a distortion weight, ω, the magnitude of which was inversely proportional to the distance of the pixel from the centroid of the force, in units of radii:

$$\omega_k = \frac{\lambda_k r_k - \sqrt{x_k^2 + y_k^2}}{r_k \sqrt{2}}$$

$$\omega_k = \begin{cases} \min\{\omega_k, 0\}, & \text{if } \lambda_k < 0 \\ \max\{\omega_k, 0\}, & \text{otherwise} \end{cases}$$

Together, these values were used to compute the signed displacement of the pixel; with the overall total displacement, Δ_x, Δ_y, computed by summing across all N the forces:

$$\Delta_{x[i,j]} = \sum_{k=1}^{N} \omega_k x_k$$
$$\Delta_{y[i,j]} = \sum_{k=1}^{N} \omega_k y_k$$

These values were clamped to fall within a range -0.1 to $+0.1$. However, for storage within an (unsigned) 24-bit texture, they were remapped to a positive, 0 to 1, range. This process resulted in two matrices of values of the type shown in Fig. 4B-C, wherein dark values encode negative shifts, positive values encode positive shifts, and grey values are neutral (no shift).

4.4 Perceptual Filling-in

When visual input is absent from a region of the visual field, patients typically do not perceive a 'hole', but rather report that any occluded features are simply absent from the visual scene [1, 8]. In effect, the visual system 'fills in' the missing information. The most common example of this phenomenon is the blindspot of the healthy eye. This phenomenon can be simulated by removing part of the image, and then reconstructing the missing regions through interpolation. In computer science, this is a problem known as inpainting.

Recently several promising algorithms have been developed for real-time inpainting of complex scenes [3]. However, for this first approximation, missing pixels were replaced the average of their nearby healthy neighbors. Specifically, for a given missing pixel, $P_{[i,j]}$, we began by locating the nearest non-missing pixel along each
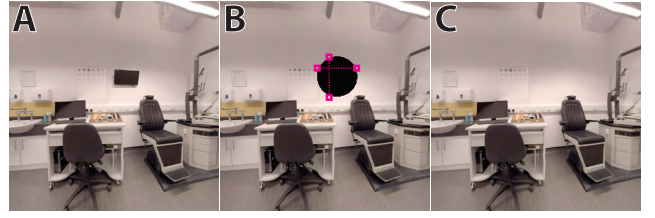

Figure 5: Perceptual Filling-in. (**A**) The source image; (**B**) The area to be removed and inpainted. This was controlled by a gaze-contingent reference texture: typically a binarized version of Fig. 3B. The purple lines show how the four nearest cardinal pixels were located to replace a given pixel in the occluded region; (**C**) The output image, after inpainting. Note that the letter chart is no longer visible.
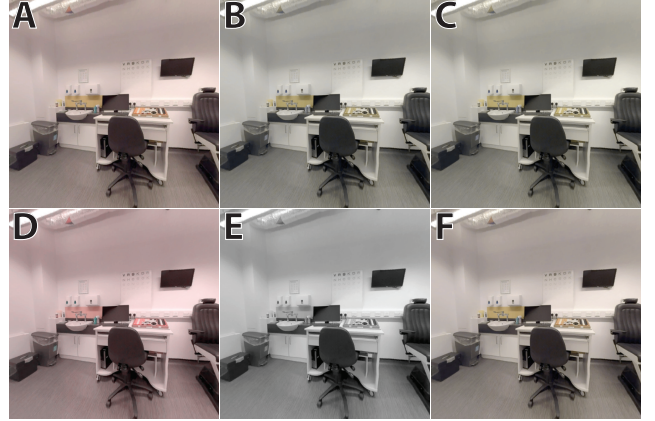

Figure 6: Color Vision Deficits. (**A**) The source image; (**B**) Protanopia; (**C**) Deuteranopia; (**D**) Tritanopia; (**E**) Monochromacy; (**F**) Protanomaly (partial Protanopia).

of the four cardinal directions. We then replaced the missing pixel with the linear-weighted sum of these four pixels:

$$P_{R[i,j]} = \omega_1 P_{R[i,j+\Delta_1]} + \omega_2 P_{R[i+\Delta_2,j]} + \omega_3 P_{R[i,j-\Delta_3]} + \omega_4 P_{R[i-\Delta_4,j]}$$
$$P_{G[i,j]} = \omega_1 P_{G[i,j+\Delta_1]} + \omega_2 P_{G[i+\Delta_2,j]} + \omega_3 P_{G[i,j-\Delta_3]} + \omega_4 P_{G[i-\Delta_4,j]}$$
$$P_{B[i,j]} = \omega_1 P_{B[i,j+\Delta_1]} + \omega_2 P_{B[i+\Delta_2,j]} + \omega_3 P_{B[i,j-\Delta_3]} + \omega_4 P_{B[i-\Delta_4,j]}$$

where the weights, ω, were inversely proportional to the pixels' relative distance:

$$\omega_k = \frac{(1/\Delta_k)^2}{\sum_{k=1}^{n=4} (1/\Delta_k)^2}$$

The overall effect is a simple, fast inpainter that is able to crudely preserve local features (see Fig. 5). One of the principle downsides of this approach is that it is liable to introduce noticeable blur. This was not considered a substantive concern for the present application, however, since patient-reported 'filling in' effects are often co-morbid with a loss of acuity (i.e., blurring). Accordingly, this effect was used intended to be used primarily in conjunction with the blur effect (above), and the spatial extent of the inpainting effect was controlled using the same reference texture as the blur effect, binarized via using an arbitrary scalar threshold.

4.5 Color Vision Deficits

Abnormalities in color perception can be simulated by selectively attenuating one or more color channel (Fig. 6). This required RGB

pixel vectors to first be converted into LMS space: a color space designed to represent the responses of the three cones in the human eye. For efficiency, in the present work, the three key steps (transformation, attenuation, inverse transformation) were combined into a single matrix multiplication, following the method and values published previously by Machado, Oliveira and Fernandes [4]. Thus, for a given pixel, the RGB values for a given color vision deficit (CVD) were obtained using:

$$\begin{bmatrix} R_s \\ G_s \\ B_s \end{bmatrix} = \Gamma_{norm}^{-1} \Gamma_{CVD} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

where Γ_{norm} is a 3x3 matrix that transforms RGB values to the opponent-color space of a normal trichromat, while Γ_{CVD} is the analogous transform that maps the RGB color space to a particular instance of opponent-color space (i.e., to a color abnormality of a given type and severity). The values for Γ_{norm} are constant, and were derived from the cone spectral sensitivity functions of a representative healthy observer [10]. The values of Γ_{CVD} were derived from values published by Machado and colleagues, and varied depending on the type of impairment (protanomaly, deuteranomaly, tritanomaly) and the severity of the impairment. Note that this approach allows us to simulate gradations of impairment (e.g., protanomaly, as well as complete protanopia). Note also that this approach requires to make certain assumptions about the spectral output properties of the display screen, as well as the psychophysical properties of the user's spectral sensitivity functions. The results are therefore only an approximation. However, anecdotal reports from individuals with CVD have indicated a reasonable degree of accuracy.

5 SYSTEM IMPLEMENTATION

Hardware: The system was implemented on two platforms: an iPhone 7 (together with a Homido v2 VR holster), and a Fove0 VR headset. The Fove0 incorporates near-infrared corneal-reflectance eye-tracking technology, allowing continuous (120 Hz) measurement the orientation of each eye independently. The iPhone 7 provided a cable-less solution, for greater portability.

Software: The system was programmed in Unity 5.5/C#, using the Google Cardboard (iPhone 7) and Fove (Fove0) frameworks. Image manipulations were performed primarily using Cg/HLSL fragment shaders, with key parameters precalculated using compute shaders. The use of Cg/HLSL shaders meant that image manipulations were performed primarily on the GPU, thereby allowing relatively complex operations to be performed in near real-time.

Operation: In VR mode, the source image was either a 3D virtual environment, or a pre-made panoramic image projected onto the side of a sphere. Pass-through Augmented Reality [AR] was also supported by taking the source image from a live camera feed. Head movements were recorded by the devices integrated gyroscope, and allowed 360-degree spherical viewing of VR content. And where eye-tracking data were available (Fove0 version only), simulations were positioned relative to the user's gaze. Visual impairments could be applied independently to each eye, and multiple impairments could be stacked up, with no lag or dropping of frames. An example output is shown in Fig. 7, and a video of the system in use can be viewed here: `www.ucl.ac.uk/~smgxprj/videos/vr_info.mp4`. A browser-based prototype has also been implemented in WebGL, and can be accessed here: `www.ucl.ac.uk/~smgxprj/vr`.

6 CONCLUSIONS

We have demonstrated that it is possible to simulate a wide range of visual impairments using existing commercial VR/AR technology. We are currently undertaking empirical validations of their accuracy
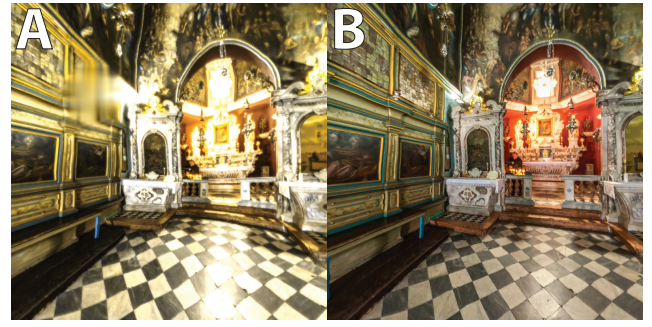

Figure 7: Summary of all the visual impairments presented here. (**A**) Abnormal left eye, exhibiting disability glare, visual field loss (blurring and filling-in), spatial distortions (metamorphopsia), and a color vision deficit (protanomaly); (**B**) Healthy right eye (no impairments).

and utility. A number of important questions also remain, including how to relate simulation parameters to patients' medical records, how to simulate more complex, non-linear aspects of vision [11], and how to combine simulations with structured environmental data (e.g., such as that returned from ARKit/ARCore). However, based on the present work, we believe VR/AR could already be a powerful tool for improving our understanding and awareness of eye-disease.

REFERENCES

[1] D. P. Crabb, N. D. Smith, F. C. Glen, R. Burton, and D. F. Garway-Heath. How does glaucoma look?: patient perception of visual field loss. *Ophthalmology*, 120(6):1120–1126, 2013.

[2] W. S. Geisler and J. S. Perry. Real-time simulation of arbitrary visual fields. In *Proceedings of the 2002 symposium on Eye tracking research & applications*, pp. 83–87. ACM, 2002.

[3] N. Kawai, T. Sato, and N. Yokoya. Diminished reality based on image inpainting considering background geometry. *IEEE transactions on visualization and computer graphics*, 22(3):1236–1247, 2016.

[4] G. M. Machado, M. M. Oliveira, and L. A. Fernandes. A physiologically-based model for simulation of color vision deficiency. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1291–1298, 2009.

[5] E. Midena and S. Vujosevic. Metamorphopsia: An overlooked visual symptom. *Ophthalmic research*, 55(1):26–36, 2016.

[6] D. Pascolini and S. P. Mariotti. Global estimates of visual impairment: 2010. *British Journal of Ophthalmology*, 96(5):614–618, 2012.

[7] H. A. Quigley. Glaucoma. *The Lancet*, 377(9774):1367–1377, 2011.

[8] V. S. Ramachandran and R. L. Gregory. Perceptual filling in of artificially induced scotomas in human vision. *Nature*, 350(6320):699–702, 1991.

[9] T. Redmond, D. F. Garway-Heath, M. B. Zlatkova, and R. S. Anderson. Sensitivity loss in early glaucoma can be mapped to an enlargement of the area of complete spatial summation. *Investigative ophthalmology & visual science*, 51(12):6540–6548, 2010.

[10] V. C. Smith and J. Pokorny. Spectral sensitivity of the foveal cone photopigments between 400 and 500 nm. *Vision research*, 15(2):161–171, 1975.

[11] W. B. Thompson, G. E. Legge, D. J. Kersten, R. A. Shakespeare, and Q. Lei. Simulating visibility under reduced acuity and contrast sensitivity. *JOSA A*, 34(4):583–593, 2017.