Fast synthesis of atmospheric image effects

Miles Hansard
miles.hansard@qmul.ac.uk
School of EECS, Queen Mary University of London.

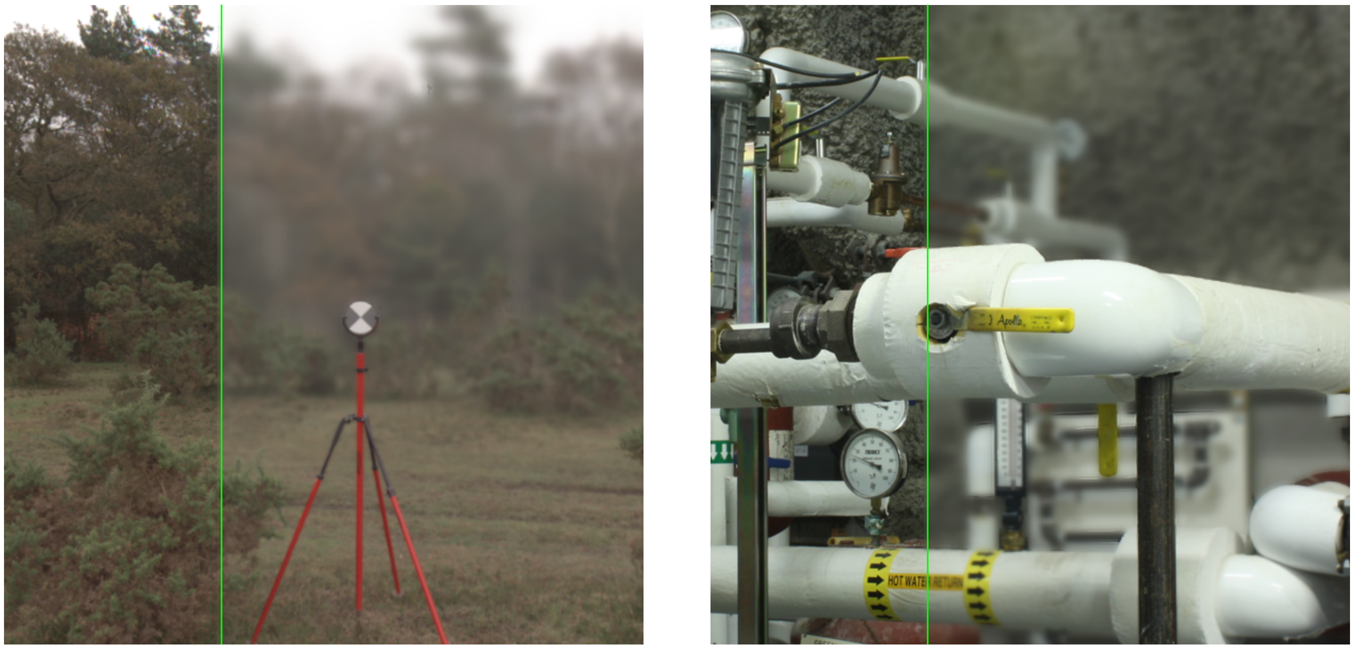


Figure 1: Synthetic haze effects have been automatically added to the right two-thirds of each image, based on the associated depth maps. There is a realistic loss of contrast and sharpness, in the distant parts of each scene, owing to synthetic absorption and scattering effects. Note that the closest objects remain clearly visible. These effects can be computed in real time, by 2D inhomogeneous diffusion, in a GPU shader.

ABSTRACT

This paper introduces a new method for adding synthetic atmospheric effects, such as haze and fog, to real RGBD images. The given depth maps are used to compute per-pixel transmission and and spatial frequency values, which determine the local contrast and blur, based on physical models of atmospheric absorption and scattering. A fast 2D inhomogeneous diffusion algorithm is developed, which is capable of computing and rendering the effects in real time. The necessary pre-processing methods, including sky identification and matting, are also explained. A GPU implementation is described, and evaluated on a range of RGBD data, including that from outdoor lidar and indoor structured light systems.

1 INTRODUCTION

The synthesis of atmospheric image effects, such as haze and fog, is of interest for three reasons. Firstly, it is well known that these effects can significantly increase the realism of computer generated imagery [O'Neil 2005; Wroński 2016]. Secondly, it may be necessary to add haze or fog to real video footage or photographs, for cinematic or stylistic effects [Christiansen 2006; Narasimhan and Nayar 2003]. Thirdly, there is considerable interest in removing real fog and haze effects from photographs [Cai et al. 2016]; these dehazing methods can be trained on synthetic degraded images, for which the ground truth originals are available. The approach described here is relevant to all three of these applications. The required input is RGBD imagery, which can be acquired in several different ways. Both outdoor lidar [Adams et al. 2016] and indoor structured light [Scharstein et al. 2014] data sets are used in the present paper. It would also be possible to use stereo matching or single view reconstruction methods, in the absence of scanner data. In the case of computer generated imagery, the depth buffer can be used; indeed this is easier than dealing with real imagery, because the projection of the sky (and/or other light sources) is known.

There is a large computer graphics literature on rendering in participating media, such as haze or fog [Cerezo et al. 2005]. The possible approaches range from simple linear attenuation (`glFog` in OpenGL), to physically based volumetric simulations [Premože et al. 2004; Stam 1995; Sun et al. 2005], and machine learning approaches [Kallweit et al. 2017]. Volumetric *rendering* methods are not directly relevant to the present work, because they typically require complete 3D scene models, although significant precomputation may be possible [Bruneton and Neyret 2008]. The method developed in sections 3–4 only requires access to a depth map, and is primarily aimed at real imagery. Hence, although some light transport theory is needed (in section 2), the following literature review will focus on '2.5D' or *screen-space* rendering methods, which are more directly comparable to the present approach.

1.1 Previous work

This work is partially inspired by that of Elek, Ritschel and Seidel, who introduced a screen-space rendering method, for scattering media [Elek et al. 2013]. That work makes use of a depth-based blur parameterization, as will be done here, in order to post-process computer generated imagery. Their algorithmic approach is to build a mipmap of the rendered image, in which the scale of the Gaussian blur increases from one level to the next. The known depth of each pixel can then be used as an index into this structure, via the blur parameterization, so that an appropriate value can be retrieved.

There are two difficulties with this approach, as discussed by Elek et al. Firstly, there is inevitable brightness 'leakage', across depth boundaries. This is because the filter scale is constant within each level, so (for example) a dark background pixel may acquire unrelated luminance from a bright foreground object, at coarse scales. This problem is addressed by masking the unwanted contributions, during the rendering process, with reference to the depth maps. The second difficulty is that a discrete structure (the mipmap) is used to represent a continuous phenomenon (blurring). This gives rise to discontinuities in the image, if the scale indexing is done naively. Elek et al. address this problem by generating the scale indices from a blurred depth map, and then linearly interpolating between the two nearest levels in the mipmap. These remedies work well in practice, but they are somewhat removed from the underlying scattering theory. Furthermore, it is unclear how sensitive these approximations would be to depth errors (e.g. in lidar data), because Elek et al. work with computer generated imagery, and have access to the full resolution depth buffer.

The algorithms developed in the present paper are more similar to those in certain synthetic *depth of field* methods, although the latter are typically designed for computer generated imagery. Lee et al. use a filtered mipmap, comparable to that described above, for depth of field effects [Lee et al. 2009]. A different approach is taken by Bertalmío et al., who use inhomogeneous diffusion to obtain space-variant blur, as do Kosloff and Barsky [Bertalmío and Fort 2004; Kosloff and Barsky 2007]. These latter works use *explicit* methods to solve the diffusion PDE; this approach is very slow, as will be shown here (in section 5.3). Kass et al. use an *implicit* method, and propose a GPU algorithm to solve the resulting tridiagonal system of equations [Kass et al. 2006]. The present work is also based on inhomogeneous diffusion, but uses a different method to solve the PDE.

Finally, a *splatting* method for depth of field (and motion blur) effects has been introduced [Leimkühler et al. 2018]. The splatting operations are performed in the Laplacian domain, where they are very efficient, using pre-computed PSFs. This method, which is applied to computer generated scenes, is not directly related the present approach.

1.2 Contributions

The organization and contributions of this work are as follows. Section 2 describes the optical effects of fog and haze, including a simple scattering approximation (2.2), based on the work of Premože et al. [Premože et al. 2004]. Section 3 develops a screen-space inhomogeneous diffusion model of atmospheric effects, involving a new blur parameterization, and an efficient scale space representation (3.2). This approach was not considered in the literature reviewed above. Section 4 describes the practical aspects of the model, including a new sky segmentation procedure (4.1), and an efficient GPU implementation (4.2). The accuracy and speed of the model are evaluated in section 5.

2 ATMOSPHERIC DEGRADATION

This section contains an overview of the model, which emphasizes the role of spatial scale in the rendering process (2.1). In particular, the mapping from viewing distance to spatial scale is defined, in relation to a physical scattering model (2.2).

2.1 Attenuation and blur

The proposed model depends on the optical transmission $\tau(\mathbf{x})$, which is a function of the range map $R(\mathbf{x})$. The latter encodes the estimated distance to the visible scene patch at image location $\mathbf{x} = (x, y)$, as estimated by a lidar scanner, for example. If κ_a and κ_s are the *absorption* and *scattering* coefficients, respectively, then the Beer-Lambert attenuation model is

$$\tau(\mathbf{x}) = \exp\Big(-(\kappa_a + \kappa_s)\,R(\mathbf{x})\Big). \tag{1}$$

Hence the transmission is exponentially decaying [Middleton 1960], in a homogeneous medium, as determined by the *attenuation* coefficient $\kappa_a + \kappa_s$. The absorption process represents light that was lost, whereas the scattering process represents light that was diverted.

Let $\mathbf{E}(\mathbf{x}, \sigma_0)$ be the original RGB image, in which the *inner scale* σ_0 encodes the spatial resolution, as determined by the sensor and sampling process [Florack et al. 1994; Koenderink 1984]. This image is converted to RGBA, and *pre-multiplied* [Blinn 1994] by the inital transmission map $\tau(\mathbf{x}, \sigma_0) = \tau(\mathbf{x})$, to give the depth-attenuated RGBA image

$$\mathbf{F}(\mathbf{x}, \sigma_0) = \tau(\mathbf{x}, \sigma_0)\,\Big(\mathbf{E}(\mathbf{x}, \sigma_0),\, 1\Big). \tag{2}$$

The local spatial frequency of this image should now be band-limited to $\sigma(\mathbf{x})$, where the desired scale is determined from the known distance $R(\mathbf{x})$, as will be explained in section 2.2. This space-variant blurring procedure, which will be developed in section 3, has the approximate form of a convolution integral

$$F\Big(\mathbf{x}, \sigma(\mathbf{x})\Big) \approx \int G\Big(\mathbf{u}, \sigma(\mathbf{x})\Big)\, F\Big(\mathbf{x} - \mathbf{u}, \sigma_0\Big)\, d\mathbf{u} \tag{3}$$

where $G\Big(\mathbf{x}, \sigma(\mathbf{x})\Big)$ is a variable 2D Gaussian kernel, and F refers to each of the *four* channels F_r, F_g, F_b and F_a. If the visible surface is continuous and fronto-parallel, then the convolution form (3) is exact; but in general, there will be no fixed kernel shape G associated with the diffusion process of section 3.

The final colour is rendered by composing the blurred surface colour $\mathbf{F}\Big(\mathbf{x}, \sigma(\mathbf{x})\Big)$ with the background airlight colour $\mathbf{B}$, recalling that the former has pre-multiplied opacity:

$$\mathbf{F}\Big(\mathbf{x}, \sigma(\mathbf{x})\Big) \leftarrow \mathbf{F}\Big(\mathbf{x}, \sigma(\mathbf{x})\Big) + \Big(1 - \tau\Big(\mathbf{x}, \sigma(\mathbf{x})\Big)\Big)\mathbf{B}(\mathbf{x}). \tag{4}$$

Note that the *blurred* transmission $\tau\big(\mathbf{x}, \sigma(\mathbf{x})\big)$ has been retrieved from the alpha channel of $\mathbf{F}$. The airlight colour $\mathbf{B}$, which can be estimated from sky regions of the image, is typically constant (and approximately achromatic). This results in an appropriate depth-dependent loss of contrast, in the final image.

The above model (4) gives visually plausible results, in practice. An important feature is that the transmission map $\tau\big(\mathbf{x}, \sigma(\mathbf{x})\big)$ has been blurred in the same way as the (pre-multiplied) colour channels. This is very different from the naive *non*-premultiplied composition, $\tau(\mathbf{x}, \sigma_0)\,\mathbf{E}\big(\mathbf{x}, \sigma(\mathbf{x})\big) + \big(1 - \tau(\mathbf{x}, \sigma_0)\big)\mathbf{B}$, which would allow high spatial frequency information in the transmission map $\tau(\mathbf{x}, \sigma_0)$ to mix inconsistently with colour information in $\mathbf{E}$ at local scale $\sigma(\mathbf{x})$. Unlike (4), this would give rise to ghost-like edges in distant regions of the image.

A physical interpretation of (4) is that blurred image regions are associated with uncertainty about *which* surface patches reflected the observed radiance. Hence there should be corresponding uncertainty about the attenuations, because these depend on the distances to the uncertain patches. From this perspective, the procedure (2–4) can be seen as a kind of normalized filtering [Knutsson and Westin 1993], with a space-variant kernel.

2.2 Scattering

The Beer-Lambert equation (1) gives the optical transmission $\tau(\mathbf{x})$ as a function of distance $R(\mathbf{x})$ and the absorption and scattering coefficients κ_a, κ_s. An analogous equation for optical blur $\sigma(\mathbf{x})$ has been derived by Premože et al., based on path integral approximations [Ashikhmin et al. 2004; Premože et al. 2004]:

$$\sigma(\mathbf{x}) = \sqrt{\frac{1}{2}\left(\frac{2\kappa_a}{3R(\mathbf{x})} + \frac{4}{(1-\nu)\kappa_s R^3(\mathbf{x})}\right)^{-1}}. \tag{5}$$

The directional concentration parameter $\nu \in [-1, 1]$ is the average projection of the scattering direction onto the forward direction, which in turn depends on the chosen phase function [Premože et al. 2004]. For example, if $\nu = 1$, then no scattering occurs, because the two directions must always be collinear.

The blur definition (5) was used directly by Elek et al. in their rendering scheme [Elek et al. 2013]. It would be useful to have a more intuitive approximation of this function, but the corresponding Taylor series is divergent. Nonetheless, the definition (5) can be put into the form $(a\,R^{3/2})(b + c\,R^2)^{-1/2}$, which can then be developed in a generalized binomial expansion. The result can be expressed as a multiple of the transformed variable $R^{3/2}(\mathbf{x})$, as follows:

$$\sigma(\mathbf{x}) \approx \frac{\sqrt{(1-\nu)}}{2\sqrt{2}}\,\kappa_s^{1/2}\,R^{3/2}(\mathbf{x}). \tag{6}$$

Note that the absorption coefficient κ_a does not appear in this new approximation. Also note that the physical dimensions are consistent, because R has units of length, while κ_s has units of inverse length. In practice, there are overall scale factors, involving the camera parameters (e.g. focal length), in the relationship between distance and blur. These unknowns will be combined, along with the directional concentration coefficient ν, into an overall dimensionless parameter λ. It will also be assumed, for convenience, that the range R is divided by $R_{\max}$, while κ_a and κ_s are multiplied by the same number, to compensate. With these normalizations in mind, the proposed scale parameterization is, based on (6):

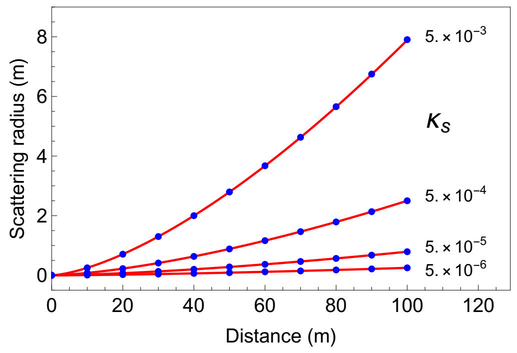


Figure 2: Red lines: the scattering function (5) from [Premože et al. 2004], plotted in physical units, for four values of the scattering coefficient κ_s. Blue dots: the proposed approximation (6), which is satisfactory for rendering purposes. The directional concentration is set to $\nu = 0.9$, which corresponds to a typical forward scattering effect [Elek et al. 2013]; the absorption coefficient is set to a high value $\kappa_a = 0.005$, in all cases.

$$\sigma(\mathbf{x}) = \sigma_0 + R^{3/2}(\mathbf{x})\left(\sigma_{\max} - \sigma_0\right) \quad \text{for} \quad R \in [0, 1] \tag{7}$$

$$\text{where} \quad \sigma_{\max} = \sigma_0 + \lambda\,\kappa_s^{1/2}.$$

Now the closest points $R(\mathbf{x}) = 0$ will retain the original scale σ_0, while the furthest points $R(\mathbf{x}) = 1$ will appear at scale $\sigma_{\max}$. Hence the effect of λ is to set the maximum possible blur, in the final image, given the scattering parameter κ_s and initial resolution σ_0. The intermediate scales increase as $R^{3/2}$, in accordance with (6). Note that the parameterization (7) corresponds to the plots in figure 2, after vertical scaling and offset. The slope of $\sigma(\mathbf{x})$ becomes zero as $R(\mathbf{x}) \to 0$, which means that any objects in the closest part of the scene will remain quite sharp.

In summary, the main parameters of the model are $(\kappa_a, \kappa_s, \lambda)$, where the absorption coefficient κ_a has no effect on the blur, according to the approximation (6). The scattering coefficient κ_s affects both attenuation and blur, as light is *diverted* from its original path, onto another.

3 INHOMOGENEOUS DIFFUSION

Diffusion tends to equalize the spatial distribution of intensity $F(\mathbf{x})$. In particular, an uphill concentration gradient ∇F induces a downhill flux $\mathbf{w} = -\varrho\nabla F$, where ϱ is the diffusion rate. If intensity is conserved, then the change over time t is the negative divergence of the flux, $\partial F/\partial t = -\mathrm{div}(\mathbf{w})$. These observations can be combined, to form the classical equation for spatial diffusion [Crank 1975]:

$$\begin{aligned}\frac{\partial}{\partial t}F(\mathbf{x}, \sigma) &= \mathrm{div}\big(\varrho(\mathbf{x})\nabla F(\mathbf{x}, \sigma)\big) \\ &= \varrho(\mathbf{x})\nabla^2 F(\mathbf{x}, \sigma) + \nabla\varrho(\mathbf{x}) \cdot F(\mathbf{x}, \sigma).\end{aligned} \tag{8}$$

If the diffusion coefficient $\varrho(\mathbf{x})$ is constant over the image, then the convective term $\nabla\varrho(\mathbf{x}) \cdot F(\mathbf{x}, \sigma)$ is zero. It is well known that Gaussian blurring arises as a solution to the remaining equation,

$\partial F/\partial t = \varrho\nabla^2 F$. Specifically, the Green's function that solves this equation, by convolution with the initial image $F(\mathbf{x},\sigma_0)$, is the normalized Gaussian [Koenderink 1984; Lindeberg 1994]

$$G(\mathbf{x},t,\varrho) = \frac{1}{4\pi\varrho t}\exp\left(\frac{-|\mathbf{x}|^2}{4\varrho t}\right) \tag{9}$$

where $2\varrho t$ is interpreted as the variance σ^2 of the kernel. This means that, as the image diffuses, the spatial scale at time t is

$$\sigma_t = \sqrt{\sigma_0^2 + 2\varrho t}. \tag{10}$$

Here the inner scale σ_0 is again interpreted as the resolution of the original image, at the initial level of the resulting scale-space [Florack et al. 1994; Koenderink 1984]. The plan in the following subsections is to let $\varrho(\mathbf{x})$ *vary* across the image, so that at time T, the desired scale $\sigma(\mathbf{x})$ is reached at every location $\mathbf{x}$. A standard numerical approach will be developed first (3.1), followed by a much more efficient approach (3.2).

3.1 Discretization

The diffusion equation (8) can be solved numerically, by taking a forward difference in time $F(\mathbf{x},\sigma_{t+1}) - F(\mathbf{x},\sigma_t)$, and equating this to a combination of spatial differences that approximate the divergence [Weickert 1997]. A 'Forward Euler' update for $F(\mathbf{x},\sigma_{t+1})$ can then be obtained by moving the $F(\mathbf{x},\sigma_t)$ term to the spatial side of the equation. A standard discretization scheme [Perona and Malik 1990] is used here, as described below.

Conceptually, the weighted gradient $\varrho(\mathbf{x})\nabla F(\mathbf{x},\sigma)$ is evaluated at the four locations $\mathbf{x} + (\pm 1/2, 0)$ and $\mathbf{x} + (0, \pm 1/2)$, by averaging adjacent pairs of ϱ samples, and differencing adjacent pairs of F samples, at the corresponding locations. The divergence can then be estimated by differencing the previous estimates, horizontally and vertically, and summing the results. The whole procedure reduces to a weighted average of the four neighbourhood differences around $\mathbf{x}$. Combining the above arrangements leads to the following algorithm [Perona and Malik 1990], which applies in each of the four image channels:

$$\begin{gathered} F(\mathbf{x},\sigma_{k+1}) \leftarrow F(\mathbf{x},\sigma_k) + \sum_{\mathbf{u}_k} \varrho(\mathbf{x},\mathbf{u}_k)\Big(F(\mathbf{x}+\mathbf{u}_k,\sigma_k) - F(\mathbf{x},\sigma_k)\Big) \\ \text{for}\quad k = 0,\ldots,N-1 \quad\text{and}\quad F = F_{\mathrm{r}},\, F_{\mathrm{g}},\, F_{\mathrm{b}},\, F_{\mathrm{a}}. \end{gathered} \tag{11}$$

Here $\mathbf{u}_k$ ranges over the four neighbourhood offsets, $(\pm 1, 0)$ and $(0, \pm 1)$. These do not yet depend on k; the subscript has been included for later developments. The coefficients $\varrho(\mathbf{x},\mathbf{u}_k)$ are defined as follows, based on the given diffusion rates $\rho(\mathbf{x})$:

$$\varrho(\mathbf{x},\mathbf{u}) = \tfrac{1}{2}\Big(\rho(\mathbf{x}+\mathbf{u}) + \rho(\mathbf{x})\Big). \tag{12}$$

The per-pixel rates $\rho(\mathbf{x})$ are set in relation to the total time T required to reach the maximum required blur, $\sigma_{\max}$, where the latter is easily determined from the maximum range $R_{\max}$ and setting of λ in the parameterization (7). Specifically, the rate $\rho(\mathbf{x}) \in [0, \rho_{\max}]$ is interpolated in proportion to $\sigma^2(\mathbf{x}) \in [\sigma_0^2, \sigma_{\max}^2]$, as follows:

$$\begin{gathered} T = \frac{\sigma_{\max}^2 - \sigma_0^2}{2\,\rho_{\max}} \qquad N = \mathrm{ceiling}(T) \\ \rho(\mathbf{x}) = \frac{\sigma^2(\mathbf{x}) - \sigma_0^2}{2N} \qquad \mathbf{u} \in \Big\{(\pm 1, 0), (0, \pm 1)\Big\} \\ \text{so that}\quad \sigma_k(\mathbf{x}) = \sqrt{\sigma_0^2 + 2k\rho(\mathbf{x})} \quad\text{and}\quad \sigma_N(\mathbf{x}) = \sigma(\mathbf{x}). \end{gathered} \tag{13}$$

The maximum rate $\rho_{\max}$ can be set to an upper limit of 1/4, as determined by standard numerical stability analysis of the iterative scheme (11), in 2D [Crank 1975]. Note that there are a whole number N of iterations, and that rounding *up* the total time T makes the rates slightly conservative, with respect to $\rho_{\max}$. Also note that if $\sigma_{\max} = \sigma_0$ then $\sigma(\mathbf{x}) = \sigma_0$ as expected, because $\rho(\mathbf{x})$ must be zero in this case.

3.2 Invariant formulation

The original diffusion process (8) runs in a scale space $(\mathbf{x},\sigma)$ with the usual Euclidean metric $\mathrm{d}s^2 = \mathrm{d}\mathbf{x}^2 + \mathrm{d}\sigma^2$. Eberly proposes to replace this with the Riemannian metric $\mathrm{d}s^2 = \mathrm{d}\mathbf{x}^2/\sigma^2 + \mathrm{d}\sigma^2/\sigma^2$, so that both spatial distances and scale differences are measured *relative* to the current scale σ [Eberly 1994]. This metric achieves translation invariance, both within and *between* scales. The corresponding scale space derivative operator $\sigma\nabla$ is obtained from the invariant gradient $\mathrm{d}F$, where the latter is constructed via the differential forms $\big(\frac{\mathrm{d}\mathbf{x}}{\sigma}, \frac{\mathrm{d}\sigma}{\sigma}\big)$. The derivative operator is then $\big(\sigma\frac{\partial F}{\partial \mathbf{x}}\mathrm{d}\mathbf{x},\, \sigma\frac{\partial F}{\partial \sigma}\mathrm{d}\sigma\big) = \mathbf{G}^{-1}\big(\frac{\partial F}{\partial \mathbf{x}}\frac{\mathrm{d}\mathbf{x}}{\sigma},\, \frac{\partial F}{\partial \sigma}\frac{\mathrm{d}\sigma}{\sigma}\big)$, where $\mathbf{G} = \mathbf{I}_3/\sigma^2$ is the 3×3 diagonal metric tensor, and $\mathbf{x} = (x, y)$ as usual.

Using these definitions, Eberly investigates the modified diffusion equation

$$\sigma\frac{\partial}{\partial\sigma}F(\mathbf{x},\sigma) = \sigma^2\nabla^2 F(\mathbf{x},\sigma) \tag{14}$$

which runs 'multiplicatively' in scale. This is of great practical significance here, because the original equation is *very* slow to reach large scales, owing to the relation $t \propto \sigma^2$ implied by (10). The modified equation (14) is effectively a re-parameterization of (8) with respect to σ. This can be seen by dividing both sides of (14) by σ^2, and then using the chain rule $\frac{\partial F}{\partial \sigma} = \frac{\partial F}{\partial t}\frac{\partial t}{\partial \sigma} = \sigma\frac{\partial F}{\partial t}$. It may be noted that a closely related *difference of Gaussians* representation is used to compute the SIFT image descriptor [Lowe 2004].

Eberly proposes a finite difference scheme in σ [Eberly 1994], by identifying the left-hand side of (14) with

$$\sigma\frac{\partial}{\partial\sigma}F(\mathbf{x},\sigma) = \lim_{\varrho\to 0}\frac{F(\mathbf{x},\sigma\mathrm{e}^{\varrho}) - F(\mathbf{x},\sigma)}{\varrho} \tag{15}$$

which can be seen from l'Hôpital's rule. Meanwhile, the right hand side of (14) is identified with

$$\begin{gathered} \sigma^2\nabla^2 F(\mathbf{x},\sigma) = \lim_{\varepsilon\to 0} F(\mathbf{x},\sigma) - \frac{1}{4\varepsilon^2}\sum_{\mathbf{v}(\varepsilon,\sigma)} F\Big(\mathbf{x} + \mathbf{v}(\varepsilon,\sigma),\ \sigma\Big) \\ \text{where}\quad \mathbf{v}(\varepsilon,\sigma) \in \Big\{(\pm\varepsilon\sigma, 0), (0, \pm\varepsilon\sigma)\Big\}. \end{gathered} \tag{16}$$

This results from replacing the constant spatial increment δ with the scaled increment $\varepsilon = \delta/\sigma$. Finally, setting $\varepsilon = 1$, it is straightforward to combine (15) and (16) into the form of the original update equation (11). It is important to note that the scale parameter is now sampled *geometrically* as

$$\sigma_k = \sigma_0 e^{k\varrho} \tag{17}$$

and therefore advances quickly, compared to the Euclidean scale space relation (10).

The above scheme will now be made space-variant, by analogy with (13). The interpolation takes place in the logarithmic space $\log(\sigma(\mathbf{x})) \in [\log(\sigma_0), \log(\sigma_{\max})]$, instead of in the quadratic space of variances. Hence the new configuration for equation (11), including the variable spatial neighbourhood from (16), is as follows:

$$\begin{aligned} S &= \frac{\log(\sigma_{\max}/\sigma_0)}{\rho_{\max}} \qquad N = \mathrm{ceiling}(S) \\ \rho(\mathbf{x}) &= \frac{\log\big(\sigma(\mathbf{x})/\sigma_0\big)}{N} \qquad \mathbf{u}_k \in \big\{(\pm\sigma_k, 0), (0, \pm\sigma_k)\big\} \\ &\text{so that} \quad \sigma_k(\mathbf{x}) = \sigma_0 \exp\big(k\rho(\mathbf{x})\big) \quad \text{and} \quad \sigma_N(\mathbf{x}) = \sigma(\mathbf{x}). \end{aligned} \tag{18}$$

The maximum rate $\rho_{\max}$ can be set to an upper limit of 1/4, as before. The final coefficient $\varrho(\mathbf{x})$ for (11) is also obtained as before (12). The values $F(\mathbf{x} + \mathbf{u}_k, \sigma)$ are obtained by bilinear interpolation, because the coordinates $\mathbf{x} + \mathbf{u}_k$ will not be integer values, in general.

This scheme looks computationally unappealing, because *twenty* bilinear interpolations must be performed per pixel, per iteration if the definitions (18) are used in (11). These interpolations are counted as $(F_r, F_g, F_b, F_a, R) \times$ (4 neighbours). However, this is a very easy proposition on a GPU, which executes bilinear interpolation in hardware, in parallel. In fact, section 5.3 will show that the computational cost is negligible, compared to the reduced number of iterations implied by (17), as opposed to (10).

4 IMPLEMENTATION

This section describes the practical aspects of the model, including some supporting procedures. In particular, section 4.1 describes how the data are preprocessed, including the treatment of holes and sky regions in the scanner data. Section 4.2 describes the details of the GPU shader implementation.

4.1 Data pre-processing

A simple RANSAC procedure [Fischler and Bolles 1981] is used to fill any holes in the range map, by sampling from the available values on each hole boundary. The value that minimizes a robust function of the gradient $\nabla R(\mathbf{x})$, across the boundary, is used to fill the hole.

A more difficult problem is posed by the sky, which has indefinite range, in outdoor images. Furthermore, the horizon is often a very complicated interface, especially when foliage is involved. This interface is usually the least reliable part of the data, for the following reasons. Firstly, the range data may be poor, owing to beam divergence at far distances, which results in mixed sky/object returns. Less obviously, the image data can be poor, owing to saturation and leakage of bright sky across fine object structures. These issues can give rise to bad visual artefacts, because the image and range data can be inconsistent. This problem could be solved by fully segmenting the sky, which may consist of many disjoint regions, and matting it into the horizon. However, it can be argued that this difficult task is unnecessary, in the present context. Recall that both sky and non-sky structures are ultimately composed with the airlight (4). If the latter has a single representative colour, then 'incorrect' transmission has little or no effect on sky regions; it simply changes the blending between two almost identical colours. Indeed, it can be argued that there is no 'correct' transmission for the sky.

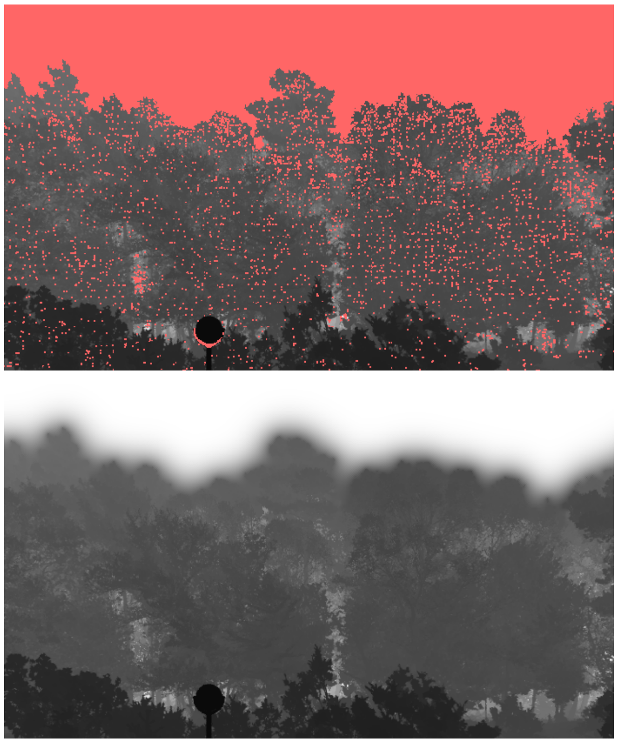


Figure 3: Range pre-processing. Top: part of an outdoor lidar scan [Adams et al. 2016], with lighter points being further away. The missing data (red) includes small surface patches, as well as the entire sky. Bottom: the same data after preprocessing. The holes have been filled, and the uncertain horizon has been matted into the sky, which has been identified and associated with the maximum distance (white). None of the original data have been altered by these processes.

From this perspective, the full segmentation problem can be avoided, provided that the horizon interface is made *consistent* with the data. This means that the known range values should not be changed, and that there should be no hard transition at the interface. This ensures, in turn, that no hard edges can be created by incorrect sky composition. These requirements are satisfied by the following procedure, which tapers outwards from the observed range values, into the sky. A typical result is visualized in figure 3.

Firstly, the majority of the sky is identified, by finding the largest connected components of undefined range values. These regions are all assigned a constant range value $S_{\max} > R_{\max}$, where $R_{\max}$ is the maximum observed range (often the maximum range of the scanner). Next, a constant-width margin of uncertainty is identified at the sky interface, by thresholding the distance transform [Maurer et al. 2003] of the binary scene mask (support of R) at some value μ.

Let $S_0(\mathbf{x}) = S_{\max}$ be the initial range values in the margin. These values are then adjusted, in order to solve the 2D Poisson equation $\nabla^2 S = 0$, subject to Dirichlet boundary conditions $S(\mathbf{x}) = R(\mathbf{x})$ on the scene side of the margin, and $S(\mathbf{x}) = S_{\max}$ on the sky side. The solution is easily obtained, by iteratively replacing each margin estimate $S(\mathbf{x})$ by the average of its four neighbours $S(\mathbf{x} + \mathbf{u})$. This naive Jacobi iteration is very slow in general; however, in the present application, the maximum propagation distance μ is small by definition, as is the total area of the margin. In practice, a good solution is obtained after about 2μ iterations, given that it takes about μ iterations for information to propagate across the margin. Note that this procedure slightly *extrapolates* the available data, but does *not* blur it. In fact, the Poisson solution can be meaningfully related to the diffusion equation (8). In particular, the same result would be obtained by diffusing $-\log\big(\tau/(\kappa_\mathrm{a} + \kappa_\mathrm{s})\big)$ to a steady state, $\partial S/\partial t = 0$, subject to the given boundary conditions.

4.2 Real-time rendering

The model has been implemented in OpenGL 3.2 ES [Leech 2015], with shaders written in GLSL `320es`. This version of the API is widely available on mobile devices, although the present results were obtained on a laptop. The general strategy is to use two full resolution textures, one for the source $F(\mathbf{x}, \sigma_k)$, and one for the target $F(\mathbf{x}, \sigma_{k+1})$ in the diffusion iteration (11). The target texture is rendered in an offscreen framebuffer, although a progressive display could also be shown. There is no need to copy the current target texture back to the source, for the next iteration; the two texture bindings can simply be swapped. The `GL_RGBA32F` format, in which each component is a 32bit IEEE float, is used for the two textures, as well as for the depth map. Additional tests are reported for half-precision `GL_RGBA16F` textures, in section 5.

A total of $N + 2$ shader passes are performed, in the current implementation: one to premultiply the transmission (2), followed by N diffusion iterations (11), and a final pass to composite the image with the airlight (4). This could be reduced to N passes, by modifying the first and last diffusion iterations, but it would then be slightly less convenient to integrate the method with other rendering processes or shaders.

The four components of $\mathbf{F}$ are processed in parallel, using native operations on the GLSL `vec4` type. Bilinear interpolation of $F(\mathbf{x} + \mathbf{u}_k, \sigma_k)$ is done in hardware, so only a single texture fetch is performed in the shader, for each interpolated value [Leech 2015].

5 EXPERIMENTS

Some initial results from the new model are reported in the following sections, starting with example images in section 5.1. It was claimed in section 3.2 that the invariant reformulation is faster than the naive Euler scheme of section 3.1. The accuracy and speed of the invariant scheme will be established, in sections 5.2 and 5.3, respectively.

5.1 Appearance

The model has been tested on images from two data sets. The SYNS data [Adams et al. 2016] comprises outdoor lidar scans, with co-registered spherical panoramas. The Middlebury 2014 data [Scharstein et al. 2014] comprises indoor structured light scans, with co-registered DSLR images. These datasets therefore represent a good range of challenges. The outdoor SYNS data requires sky segmentation, and has some inevitable discrepancies between the range and colour data (e.g. due to moving foliage). The Middlebury data has excellent depth maps, but the high contrast and hard edged scenes are very revealing of any rendering artefacts. All images are of size 1600×1040 in these experiments.

Figures 4 and 5 show some example results, which were obtained using the fast invariant scheme described in section 3.2. Recall from section 2.2 that normalized units $R' = R/R_{\max}$ and $\kappa' = \kappa R_{\max}$ are used here. For example, the reported coefficients, for the SYNS scene, should be divided by $R_{\max} = 46$m, to obtain physical values.

It is notable that depth discontinuities, such as the vertical black pole on the right, are well rendered in figure 4. Recall that no additional logic or masking was used at these boundaries; they were properly maintained by the inhomogeneous diffusion scheme. Some minor artefacts can be seen around edges on the right of figure 4, but these are not detectable in natural imagery. It is notable that the complex sky interface is well rendered, in figure 5, despite the simplicity of the approach described in section 4.1.

Finally, it should be mentioned that the fast diffusion method allows interactive control of the $(\kappa_\mathrm{a}, \kappa_\mathrm{s}, \lambda)$ parameters. In fact the whole solution can be recomputed, without noticeable delay, after any adjustment.

5.2 Accuracy

The results of the standard and accelerated schemes are visually identical, when presented side by side, on a range of images. It is difficult to quantify the accuracy of the haze rendering, owing to lack of ground truth data. However, it is straightforward to quantify the accuracy of the underlying diffusion schemes, by setting all target scales to a reference value $\sigma(\mathbf{x}) = \sigma_{\mathrm{ref}}$, and then comparing the result to that from a standard Gaussian convolution with kernel width σ_{ref}. There are three possible sources of error, in the RMS differences. Firstly, accumulation of rounding error is potentially the most serious problem for both Forward Euler schemes (13,18). The accumulation is known to be catastrophic if the limit $\rho_{\max} \leq 1/4$ is not observed [Crank 1975], and may be problematic as it is approached. Secondly, both schemes use the same four-neighbour discretization of the Laplacian operator, which may cause anisotropy artefacts. Thirdly, the invariant scheme (18) takes large spatial steps, as the iterations progress, and also requires bilinear interpolation of every image sample. Note that *none* of these problems affect the reference convolution.

The result of this test is clear and consistent — although perhaps not as expected. Data for the Middlebury '`Pipes`' [Scharstein et al. 2014] image are reported, owing to the high contrast, and wide range of spatial frequencies in this example. It is clear from figure 6 (top) that the original scheme (13) is unstable in 16bit precision. It is stable in 32bit precision, but still much less accurate than the invariant scheme (18), in either precision. This shows that although the the invariant scheme (18) has two additional sources of error (increasing step size and bilinear interpolation), these are *completely subsumed* by the rounding error that accumulates during the many additional iterations of the original scheme (13).

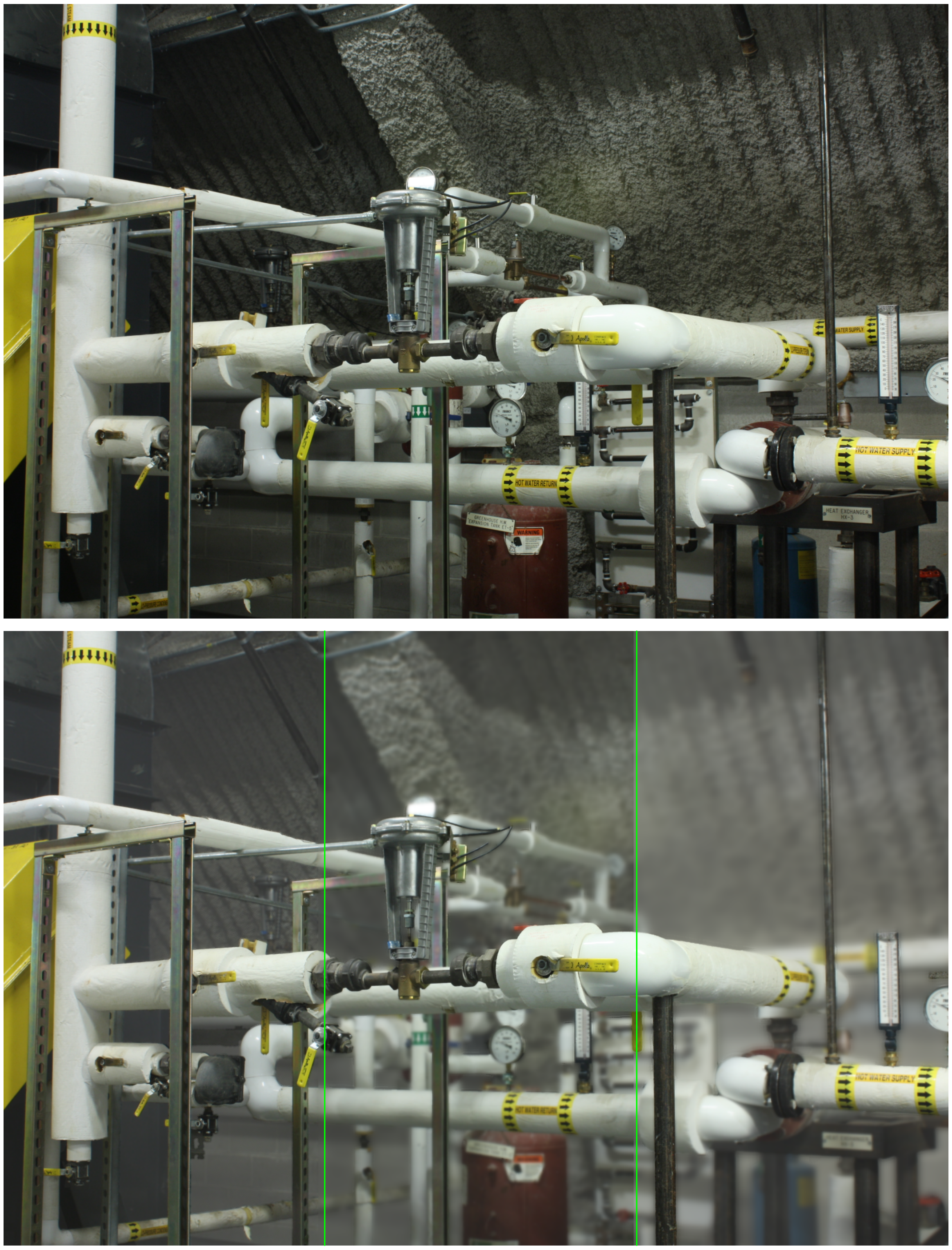


Figure 4: Top: original reference image from the Middlebury data set [Scharstein et al. 2014]. Note that the entire scene is in good focus. Bottom: example synthetic steam effects. Bottom left: Absorption only ($\kappa_a = 0.25$) does not give a realistic effect; distant objects have reduced contrast, but are too sharp. Bottom middle: scattering, to an equivalent attenuation level ($\kappa_s = 0.25$, $\lambda = 15$), shows realistic blurring in the distance. Note that the yellow handle in the foreground is still as sharp as the original (above). Bottom right: a more extreme example, with scattering, additional absorption, and an increase in the maximum blur scale ($\kappa_a = 0.25$, $\kappa_s = 0.25$, $\lambda = 30$). Note the dramatic difference in resolution between the vertical black pole, and the background.

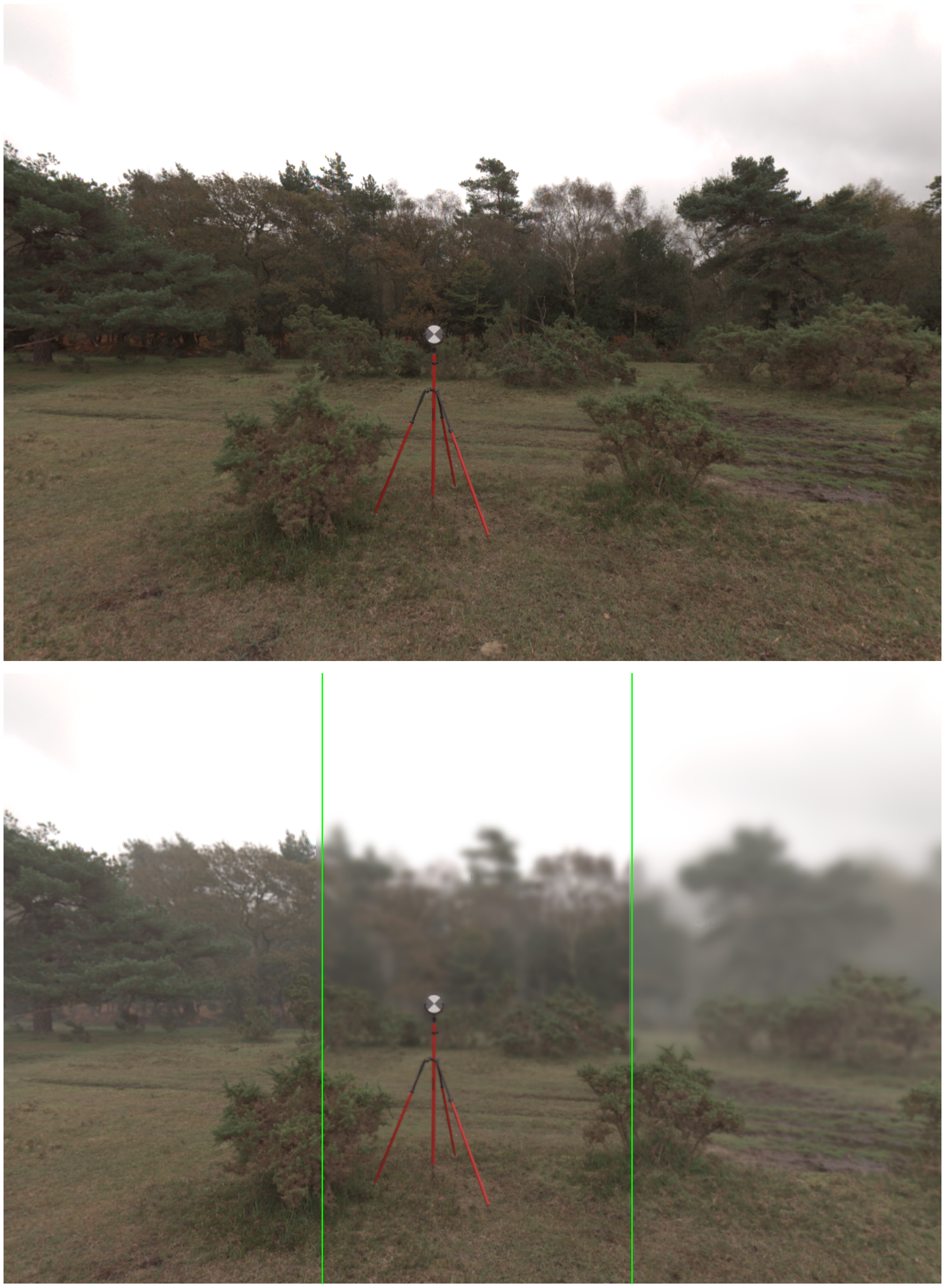
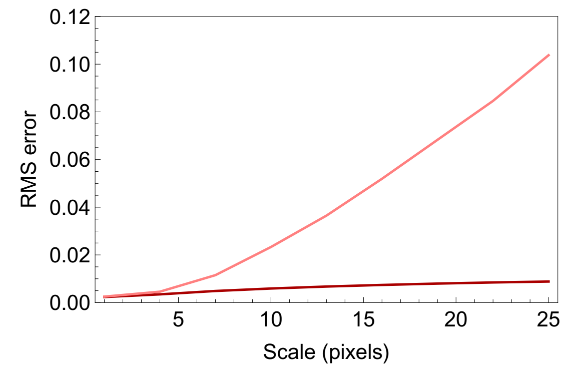


Figure 5: Top: original reference image from the SYNS data set [Adams et al. 2016]. Bottom: example synthetic fog effects. Bottom left: Absorption only ($\kappa_a = 0.25$) does not give a realistic effect; distant objects have reduced contrast, but are too sharp. Bottom middle: scattering, to an equivalent attenuation level ($\kappa_s = 0.25$, $\lambda = 15$), shows realistic blurring in the distance. Note that the target in the foreground is still as sharp as the original (above). Bottom right: a more extreme example, with scattering, additional absorption, and an increase in the maximum blur scale ($\kappa_a = 0.25$, $\kappa_s = 0.25$, $\lambda = 30$). The foreground grass remains clear.



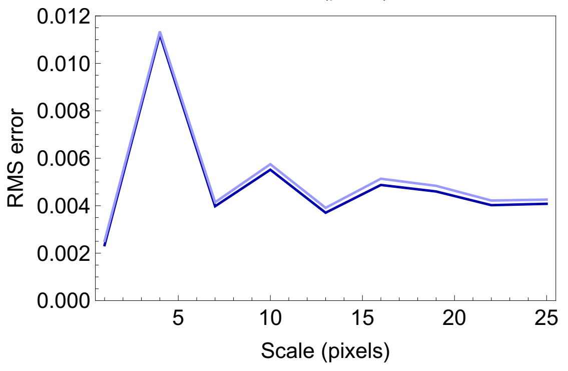
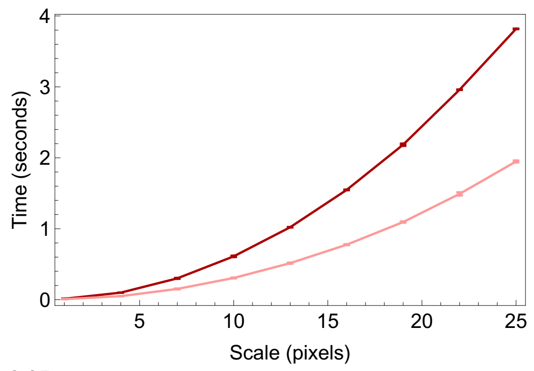


Figure 6: Accuracy of isotropic diffusion blurring on a floating point image (in the range $[0,1]$). Top: the original diffusion scheme (13) performs well in 32bit precision (dark red), but suffers from severe accumulation of error in 16bit precision (light red). Bottom: the invariant numerical scheme (18) is approximately ten times more accurate, noting the vertical axis limit, in both 32bit (dark blue) and 16bit (light blue).



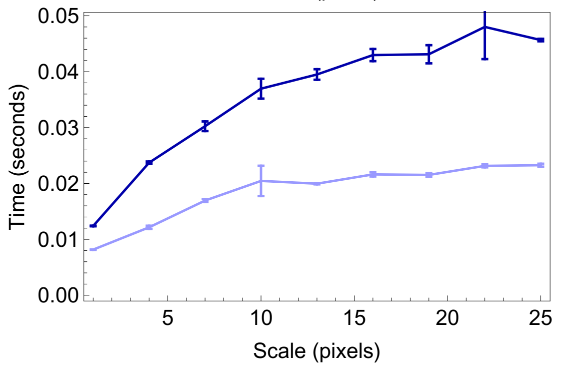


Figure 7: Speed of isotropic diffusion blurring on a $1600{\times}1120$ floating point image. Top: the original diffusion scheme (13) is very slow to reach large scales, in both 32bit (dark red) and 16bit (light red) precision. The quadratic relationship between scale and time (13) is clearly visible. Bottom: The invariant scheme (18) is almost a hundred times faster, noting the vertical axis limit, in both 32bit (dark blue) and 16bit (light blue). Error bars represent ± 1 standard deviation, and show essentially no variation, for these GPU-based implementations.

The oscillation of the invariant error in figure 6 (bottom) *is* attributable to bilinear interpolation. This depends on how the sample points $\mathbf{x}+\mathbf{u}_k$ happen to align, on average, with the pixel grid (11). However, this effect soon becomes irrelevant as the high spatial frequencies are blurred out of both the reference and test images; hence the observed damping of this oscillation.

5.3 Speed

Performance of the OpenGL implementation was measured using the `GL_TIME_ELAPSED` timer query [Leech 2015], which returns the time taken for all GPU tasks to complete (including any CPU time between shader passes). This accounts for all pixel premultiplication, diffusion, and compositing operations, in the present implementation. The tests were performed on an ordinary laptop, with Intel integrated graphics (i915/UHD620), under Linux. Averages were taken over five runs of any measurement, although in practice there is essentially no variation in GPU timings, owing to the simplicity of the shaders. The same 1600×1120 Middlebury [Scharstein et al. 2014] '`Pipes`' image is used for this test, although in fact the timings are independent of the image content, for any fixed scale σ_{ref}.

The results of this test are clear, and not surprising, as shown in figure 7. The timings for the original scheme (top) reflect the quadratic relationship between scale and time (13); more than 900 iterations were required to reach the maximum scale of 25px, as opposed to 13 iterations for the invariant scheme (18). The 16bit version is approximately twice as fast as the 32bit version, at the cost of accuracy, as already seen in fig. 6. The invariant scheme (bottom) is almost 100 times faster, with a similar factor of two between the 16bit and 32bit versions. Note that the 25px maximum scale in these tests is quite extreme; the corresponding $D \times D$ convolution kernel is 101px $\times$ 101px, using the convention $D = 1 + 2\,\text{ceiling}\big(2\sigma_{\text{ref}}\big)$.

6 DISCUSSION

A new approach to image-based atmospheric effects has been developed, which is both fast and physically motivated. It has been shown that although there are problems with the obvious GPU implementation, these can be solved by a simple re-parameterization of the diffusion equation [Eberly 1994]. The model has been developed for real-world RGBD data, but is equally applicable to computer generated imagery. Future work will generalize the model in two ways. Firstly, the 3D distribution of haze will be made inhomogeneous, and time dependent. Secondly, the scalar diffusion rate will be replaced by a diffusion tensor, in order to model directional effects, such as rain.

REFERENCES

W. J. Adams, J. H. Elder, E. W. Graf, J. Leyland, A. J. Lugtigheid, and A. Muryy. 2016. The Southampton-York Natural Scenes (SYNS) dataset: Statistics of surface attitude. *Scientific Reports* 6 (2016), 35805:1–16.

M. Ashikhmin, S. Premože, R. Ramamoorthi, and S. Nayar. 2004. *Blurring of light due to multiple scattering by participating medium: A path integral approach.* Technical Report CUCS-017-04. Department of Computer Science, Columbia University.

M. Bertalmío and P. Fort. 2004. Real-time, accurate depth of field using anisotropic diffusion and programmable graphics cards. In *Proc. 3DPVT.* 1124–1140.

J. F. Blinn. 1994. Compositing, part I: Theory. *IEEE Computer Graphics and applications* 14(5) (1994), 83–87.

E. Bruneton and F. Neyret. 2008. Precomputed atmospheric scattering. *Computer graphics forum* 27, 4 (2008), 1079–1086.

B. Cai, X. Xu, K. Jia, C. Qing, and D. Tao. 2016. Dehazenet: An end-to-end system for single image haze removal. *Proc. IEEE Trans. Image Processing* 25, 11 (2016), 5187–5198.

E. Cerezo, F. Pérez, X. Pueyo, F.J. Seron, and F. X. Sillion. 2005. A survey on participating media rendering techniques. *The Visual Computer* 21(5)) (2005), 303–28.

M. Christiansen. 2006. Climate: Air, water, smoke, clouds in after effects. In *Adobe After Effects 7.0 Studio Techniques.* Pearson, 371–392.

J. Crank. 1975. *The mathematics of diffusion.* Oxford University Press.

D. Eberly. 1994. A differential geometric approach to anisotropic diffusion. In *Geometry-Driven Diffusion in Computer Vision*, B. M. ter Ham Romeny (Ed.). Kluwer, 371–392.

O. Elek, T. Ritschel, and H.-P. Seidel. 2013. Real-time screen-space scattering in homogeneous environments. *IEEE Computer Graphics and applications* 50 (2013), 53–65.

M.A. Fischler and R.C. Bolles. 1981. Random Sample Consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Comm. ACM* 24, 6 (1981), 381–395.

L. M. J. Florack, B. M. ter Haar Romeny, J. J. Koenderink, and M. A. Viergever. 1994. Linear scale-space. *Journal of Mathemetical Imaging and Vision* 4 (1994), 325–351.

S. Kallweit, T. Müller, B. McWilliams, M. Gross, and J. Novak. 2017. Deep scattering. *ACM Transactions on Graphics* 36(6)) (2017), 231:1–11.

M. Kass, A. Lefohn, and J. Owens. 2006. *Interactive depth of field using simulated diffusion on a GPU.* Technical Report. Pixar Technical Memo. 1124–1140 pages.

H. Knutsson and C.-F. Westin. 1993. Normalized and differential convolution. In *Proc. CVPR.* 515–523.

J. Koenderink. 1984. The structure of images. *Biological Cybernetics* 50 (1984), 363–370.

T.J. Kosloff and B.A. Barsky. 2007. An algorithm for rendering generalized depth of field effects based on simulated heat diffusion. In *Proc. ICCSA.* 1124–1140.

S. Lee, G.J. Kim, and S. Choi. 2009. Real-time depth-of-field rendering using anisotropically filtered mipmap interpolation. *IEEE Trans. Visualization and Computer Graphics* 15, 3 (2009), 453–464.

J. Leech (Ed.). 2015. *OpenGL ES version 3.2.* Khronos Group.

T. Leimkühler, H-P. Seidel, and T. Ritschel. 2018. Laplacian kernel splatting for efficient depth-of-field and motion blur synthesis or reconstruction. *ACM Trans. Graphics* 37, 4 (2018), 55:1–11.

T. Lindeberg. 1994. Scale-space theory: A basic tool for analysing structures at different scales. *Journal of Applied Statistics* 21(2) (1994), 224–270.

D. G. Lowe. 2004. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision* 60, 2 (2004), 91–110.

C. R. Maurer, R. Qi, and V. Raghavan. 2003. A linear time algorithm for computing exact Euclidean distance transforms of binary images in arbitrary dimensions. *IEEE Trans. PAMI* 25, 2 (2003), 265–270.

W. E. Knowles Middleton. 1960. Bouguer, Lambert, and the theory of horizontal visibility. *Isis* 51, 2 (1960), 145–149.

S.G. Narasimhan and S.K. Nayar. 2003. Shedding light on the weather. In *Proc. CVPR.* 665–672.

S. O'Neil. 2005. Accurate atmospheric scattering. In *GPU Gems 2*, M. Pharr (Ed.). Addison-Wesley.

P. Perona and J. Malik. 1990. Scale-space and edge detection using anisotropic diffusion. *IEEE Trans. PAMI* 12(7) (1990), 629–639.

S. Premože, M. Ashikhmin, R. Ravimoorthi, and S. Nayar. 2004. Practical rendering of multiple scattering effects in participating media. In *Eurographics symposium on rendering.* 363–374.

D. Scharstein, H. Hirschmüller, Y. Kitajima, G. Krathwohl, N. Nesic, X. Wang, and P. Westling. 2014. High-resolution stereo datasets with subpixel-accurate ground truth. In *German Conference on Pattern Recognition (GCPR 2014).* 31–42.

J. Stam. 1995. Multiple scattering as a diffusion process. In *Proc. Rendering Techniques.* 41–50.

B. Sun, R. Ramamoorthi, S.G. Narasimhan, and S.K. Nayar. 2005. A practical analytic single scattering model for real time rendering. *ACM Trans. Graphics* 24, 3 (2005), 1040–1049.

J. Weickert. 1997. A review of nonlinear diffusion filtering. In *Proc. International Conference on Scale-Space Theories in Computer Vision.* 1–28.

B. Wroński. 2016. Volumetric fog and lighting. In *GPU Pro*, W. Engel (Ed.). CRC Press.