Image-Based Rendering in the Gradient Domain

Johannes Kopf
Microsoft Research

Fabian Langguth
TU Darmstadt

Daniel Scharstein
Middlebury College

Richard Szeliski
Microsoft Research

Michael Goesele
TU Darmstadt

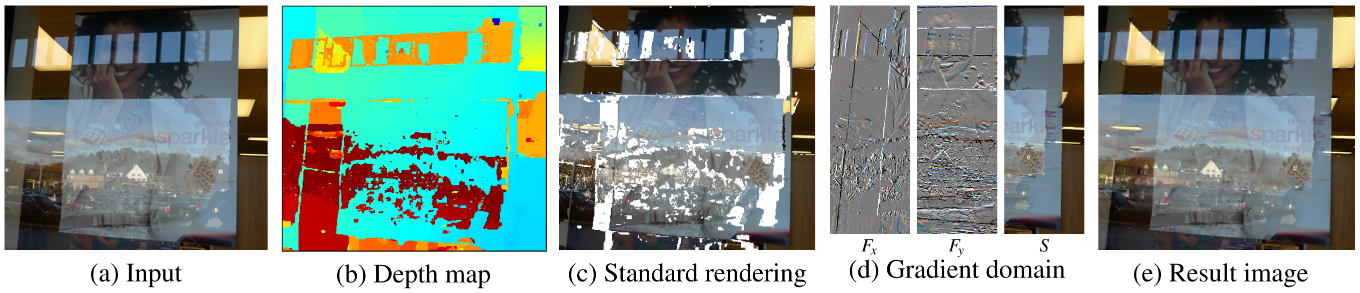


Figure 1: *Standard image-based rendering synthesizes novel views of a scene by reprojecting the input image (a) using a coarse estimated depth map (b). The technique cannot handle scenes with reflections and has problems in untextured regions, e.g., holes show up in the reprojection (c). We perform image-based rendering in the gradient domain. Our technique synthesizes horizontal and vertical gradient fields F_x and F_y, as well as an approximate solution S (d). The final image is obtained through Poisson integration (e). Our technique naturally handles reflections and untextured regions.*

Abstract

We propose a novel image-based rendering algorithm for handling complex scenes that may include reflective surfaces. Our key contribution lies in treating the problem in the gradient domain. We use a standard technique to estimate scene depth, but assign depths to image gradients rather than pixels. A novel view is obtained by rendering the horizontal and vertical gradients, from which the final result is reconstructed through Poisson integration using an approximate solution as a data term. Our algorithm is able to handle general scenes including reflections and similar effects *without* explicitly separating the scene into reflective and transmissive parts, as required by previous work. Our prototype renderer is fully implemented on the GPU and runs in real time on commodity hardware.

CR Categories: I.3.3 [Computer Graphics]: Picture/Image Generation—Viewing algorithms

Keywords: Image-based rendering, gradient domain processing

Links: DL PDF WEB VIDEO CODE

1 Introduction

Traditional computer graphics can produce photo realistic renderings of scenes, but it requires meticulous modeling and fine tuning of every scene detail including geometry, texture, and lighting, as well as sophisticated and often slow global light transport simulations. In contrast, image-based rendering takes a lighter approach by starting from a set of captured photos of a real scene, recovering the camera parameters and coarse 3D proxies, and then reprojecting the input photos using the proxies as seen from a novel viewpoint. This technique can maintain the photographic quality of the input images without requiring explicit modeling of fine scene details. Image-based rendering methods are widely in use today, particularly for navigating urban and street level scenes.

A great challenge for all image-based rendering methods is to reliably recover scene depth in poorly textured areas. The resulting uncertainty typically requires strong regularization and can lead to rendering artifacts. Another problem is that traditional image-based rendering methods recover only a single depth value per input pixel. Thus, reflective or glossy scenes, which require modeling multiple separate layers at different depths, cannot be handled, and lead to ghosting. Sinha et al. [2012] addressed this issue for scenes with reflections by decomposing each input photo into up to two additive layers, each with their own estimated depth. The technique, however, has difficulties recovering accurate depths and discerning reflective and non-reflective scene parts, which leads to various artifacts.

In this paper, we introduce a new approach to image-based rendering that operates in the gradient domain. We use a standard regularized multi-view stereo algorithm to recover the scene depth. However, we interpret the result as the depth of the image *gradients* rather than *pixels*. We then project the gradients to their new locations as seen from the novel viewpoint and reconstruct the image through Poisson integration. To provide a data term for the Poisson solver, we directly render the effect of a gradient moving within the image. The details of our method are described in Sections 3–6.

This approach has many advantages over traditional image-based rendering. Most importantly, any depth estimation technique works best in the vicinity of strong gradients that can be reliably found in neighboring images. In less textured regions, where depth estimation is hardest, the gradients are nearly zero and thus do not contribute much to our results. In contrast, previous methods tend to produce artifacts in such regions. In addition, image-based rendering in the gradient domain is particularly well suited for handling reflections and other non-Lambertian effects as well as reasoning about occlusions. This is due to the fact that gradients in natural images are sparse, and, hence, the gradients of two mixed layers can be easily separated. (In fact, Levin et al. [2004] have used this observation to unmix two reflective layers in a single image.)

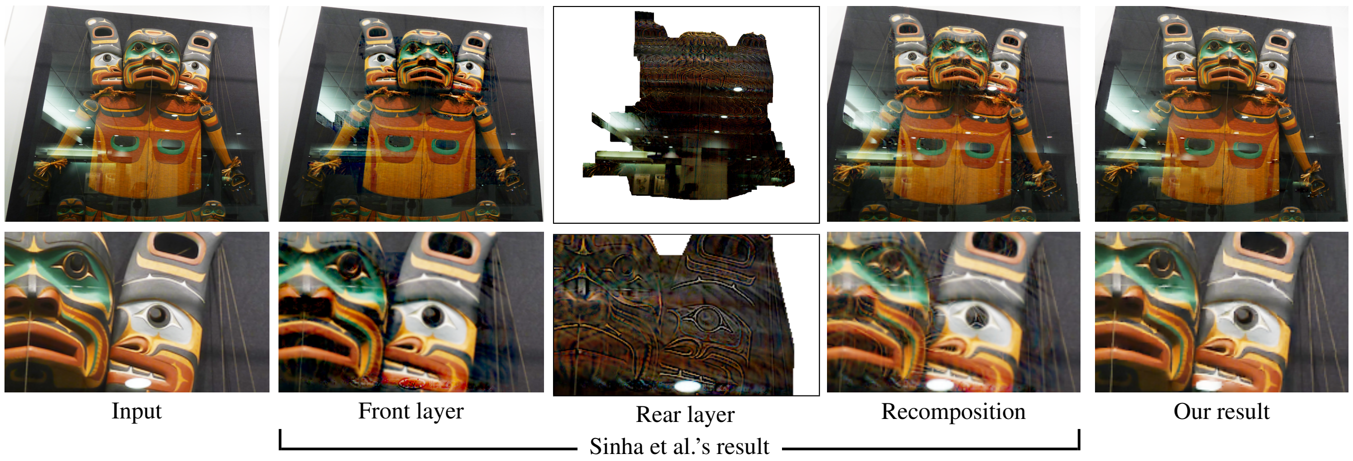


Figure 2: *Sinha et al.'s method [2012] explicitly decomposes the input images into transmitted and reflected components. This separation is not always clean. Note the ringing artifacts in the rear layers, that are also visible in their recomposed result. Our result does not suffer from ringing artifacts.*

2 Previous work

Image-based rendering takes pre-rendered or captured images of 3D scenes and interpolates these images to create novel in-between views [Shum et al. 2007]. While image-based rendering can be performed purely in ray space without the need for any 3D proxy geometry [Levoy and Hanrahan 1996], more accurate results (for the same amount of data) can be obtained by mapping input views onto some estimated proxy geometry and then blending between adjacent views [Chen and Williams 1993; Debevec et al. 1996; Gortler et al. 1996; Buehler et al. 2001]. Over the years, a wide variety of algorithms and representations have been developed to recover and model such geometry, including global polyhedral models [Debevec et al. 1996; Gortler et al. 1996; Buehler et al. 2001] and piecewise planar "impostors" [Shade et al. 1998; Popescu et al. 2006; Sinha et al. 2009].

Several recent approaches are concerned with handling scenes where accurate depth estimation is challenging. Eisemann et al. [2008] present a technique to correct misaligned projections on coarse 3D proxies. Goesele et al. [2010] turn uncertain pixels into randomized "ambient point clouds", effectively replacing reconstruction errors by less objectionable noise. Chaurasia et al. [2013] hallucinate plausible depth in poorly reconstructed regions based on appearance similarity to well reconstructed regions.

Even with 3D geometry, the movement of visual features in scenes that contain both reflected and transmitted light cannot be correctly modeled, since two different motions can be present at such locations. (Reflections also do not obey rigid *epipolar geometry* when the reflective surfaces are strongly curved or undulating [Criminisi et al. 2005].) A lot of research has been done in recovering such *transparent motions* in computer vision using both layered motion models [Shizawa and Mase 1991; Bergen et al. 1992; Irani et al. 1994; Ju et al. 1996; Szeliski et al. 2000] as well as more complex models that can handle multiple reflections or use frequency-domain analysis [Diamant and Schechner 2008; Beery and Yeredor 2008]. The separation and modeling of specular highlights has also received a lot of attention [Bhat and Nayar 1998; Carceroni and Kutulakos 2002], as has the separation of reflections using polarizing filters [Schechner et al. 1999], focus [Schechner et al. 2000] and the analysis of transparency in single images [Levin et al. 2004].

Relatively less work, however, has focused on recovering transparent and reflected motions for the purpose of image-based rendering. Szeliski et al. [2000] demonstrate how to model a scene with local planar depth approximations and to then separate the colors of the transmitted and reflected light using constrained least squares. Tsin et al. [2006] recover general depth maps corresponding to the transmitted and reflected light using a graph-cut optimization framework that estimates up to two depths per pixel. Most recently, Sinha et al. [2012] introduce a two-stage approach that uses semi-global stereo matching [Hirschmüller 2008] followed by a piecewise-planar approximation to model complex scenes with reflections and gloss. After estimating the contribution of transmitted and reflected light in each image, they develop a real-time image-based rendering system that blends between the original images using this additive two-layer decomposition. While the results they demonstrate often work well, their approach sometimes produces visual artifacts near curved surfaces and in areas where either the scene reflectivity and/or the 3D geometry is inaccurately estimated. Most often, these artifacts are visible in the separated layers, which can have "ghosts" corresponding to the other layer or ringing due to errors in the least-squares fitting stage (Figure 2).

In this paper, we sidestep the need to estimate two complete proxy geometries and to separate each input image into transmitted and reflected light. Instead, we concentrate on getting good motion (depth) and occlusion estimates at strong gradients in the image, and then use Poisson reconstruction to synthesize each new frame from the motion of the displaced gradients.

Several recent image interpolation techniques synthesize results in the gradient domain. Mahajan et al. [2009] present a technique that determines optimal linear paths in image space between pairs of pixels. They use gradients as part of the matching cost and to reduce artifacts during rendering. The interpolated frames are recovered using a 3D space-time Poisson reconstruction, where the images to be interpolated are used as boundary conditions. Linz et al. [2010] present a related approach where interpolated views are generated by warping gradient images based on a dense flow estimate followed by a similar Poisson integration. Our algorithm differs from these approaches in that it more deeply exploits the fundamental properties of gradients such as sparsity and separation of reflections: the ability to move individual gradients enables handling reflections and other non-Lambertian effects. Our system also uses a full 3D scene model and can render the scene from arbitrary viewpoints. We do not use the input images as boundary conditions since we are not restricted to interpolating images, but instead propose a novel data term to regularize the 2D Poisson reconstruction.

3 Overview

Given a set of input photos, our goal is to synthesize images from novel viewpoints by performing image-based rendering in the gradient domain. In traditional image-based rendering, the 3D scene is modeled as (potentially coarse) geometric proxies and then used to reproject the input images into a novel view. In this work, instead, we consider how the image of a scene changes if we move the viewpoint and interpret this as movement of the image gradients (i.e., the edges between pixels) in the rendered image. In order to achieve physically correct movement, we assign 3D positions to the gradients, which enables us to render them from any viewpoint.

It is straightforward to assign 3D positions to surface texture gradients. If the gradient is caused by a scene discontinuity (occlusion boundary), however, it is less obvious what 3D position it should be assigned. In this case we assign the gradient the depth of the occluder and achieve (at least approximately) the correct behavior. Finally, there are gradients on semi-transparent reflective surfaces, e.g., a sheet of glass with the underlying scene shining through. This case is very challenging for traditional reconstruction methods. However, the gradients from the two layers are typically still well separated, due to the property that gradients in natural images are sparse. For this reason it is unlikely that edges from the two layers coincide; we will most likely observe no (strong) gradient at a pixel, or only a single (strong) gradient. See Figure 3 for an illustration. This property has been used previously to separate reflections in a single image [Levin et al. 2004].

3.1 Scene reconstruction

The first step in most image-based rendering systems is estimating the camera parameters as well as (dense) depth maps for the input images. We use a standard approach for this stage in our system. In particular, we use a structure from motion pipeline similar to the one described by Snavely et al. [2006] to estimate the location, orientation, and intrinsic parameters for the input cameras. Next, we run plane sweep-based multi-view stereo matching with normalized cross-correlation (NCC) as the photometric consistency measure and combine it with graph cuts [Kolmogorov and Zabih 2002] to extract a dense depth map for each input view. In all of our experiments, we use 256 discrete depth labels.

Given two horizontally or vertically neighboring pixels p_1 and p_2, we compute pairwise regularization weights

$$w_{12} = \begin{cases} 0 & d_1 = d_2, \\ 0.005 & \|d_1 - d_2\| = 1, \\ 0.200 & \|d_1 - d_2\| > 1, \ \ \|\mathbf{g}_1\| < \varepsilon, \ \ \|\mathbf{g}_2\| < \varepsilon, \\ c_\angle & \|d_1 - d_2\| > 1, \ \ \|\mathbf{g}_1\| > \varepsilon, \ \ \|\mathbf{g}_2\| > \varepsilon, \\ 0.005 & \text{else (high and low gradient magnitude)}, \end{cases} \quad (1)$$

where d_1, d_2 are the depth labels, and $\mathbf{g}_1, \mathbf{g}_2$ are the color gradients at the pixel positions. $c_\angle = \mathbf{g}_1^\top \mathbf{g}_2 / \|\mathbf{g}_1\| \|\mathbf{g}_2\|$ is the dot product of the normalized gradient vectors. We used $\varepsilon = 0.075$ in all our experiments. This formula favors consistent depth between gradients with low magnitude (as they most likely belong to the same surface) and between gradients with a higher magnitude and similar direction (since they likely belong to the same edge). Depth discontinuities are preferred between two gradients with a high and a low magnitude.

In contrast to traditional approaches, we only need reliable depth estimates at pixels with non-negligible gradient magnitudes. Fortunately, these are just the locations where any vision-based 3D reconstruction method works best. Figure 3 shows the reconstruction results with and without regularization. In the right column, we

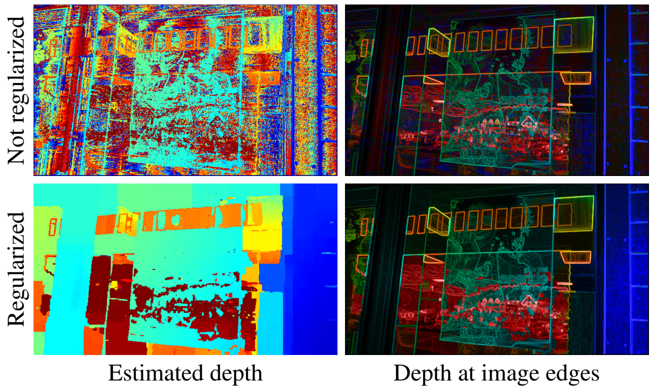


Figure 3: *Stereo reconstruction, unregularized (top) and regularized (bottom). In the right column we modulated the depth maps by gradient magnitude to visualize that the depth is correct at image edges. Regularization produces more consistent results, while still being correct at strong gradients.*

modulate the depth map by the gradient magnitude to show that the depth is correct at image edges (despite regularization that favors piecewise planar surfaces). In other words, the depth is correct in places where it matters for our method, whereas wrongly estimated depths away from image edges will affect traditional image-domain rendering methods, where they lead to artifacts.

3.2 Rendering

Our method reconstructs the novel view first in the gradient domain, by computing two gradient fields F_x, F_y, for horizontal and vertical gradients, respectively, and then obtains the final color image I through integration. Note that whenever we talk about gradients in this paper, we are referring to forward differences, i.e., the gradients can be thought to be localized on the right and bottom edges of pixels for horizontal and vertical gradients, respectively (see Figure 5).

Computing the interpolated gradient field is easy: we start with an empty image and simply splat the gradients where they project to in the novel view using additive blending. The challenge lies in the integration step, which recovers the color image. Due to noise or missing data, the gradient fields are generally not fully integrable. Thus, we obtain the color image with the best matching gradients by solving a Poisson problem [Pérez et al. 2003]. However, this solution is only defined up to an arbitrary additive constant. Unlike previous work [Pérez et al. 2003; Mahajan et al. 2009; Linz et al. 2010] we do not have a fixed boundary and thus cannot use boundary conditions to obtain a non-singular linear system. Instead we regularize the solution with a weakly weighted data term.

We tried several simple data terms, such as a constant color or the depth reprojected input image (Figures 6a–b). However, our particular application gives us a better option (Figure 6d). Since we start with regular images (the inputs) and render the scene from a different but similar perspective, we can create the result images using a pure image-domain rendering approach. More specifically, we observe that shifting a gradient in an image by a number of pixels will simply change the value of the pixels over which the gradient passed by the gradient magnitude, which is either added or subtracted depending on the direction of the movement. This can be implemented in a simple rendering step (see Sections 4 and 5 for details). In a perfect world where the 3D reconstruction would yield accurate and noiseless results, this would theoretically give perfect results. However, since vision is not perfect, this solution contains

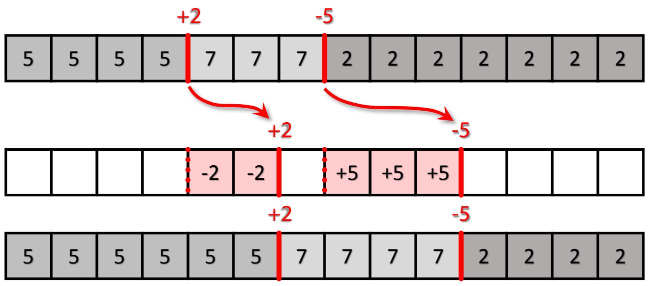


Figure 4: *Computing the approximate solution in the special case of pure horizontal camera motion. The top row shows a scanline from the source image with the two non-zero gradients highlighted. The gradients shift to their new locations (on the same scanline because of the restricted camera motion model). As gradients shift right across the image, their value is* subtracted *from the underlying pixels (bottom rows).*

artifacts. It still suffices, however, as an approximate solution (or data term), S, which we can use to weakly bias the additive offset of the Poisson problem.

Overall, we solve the following optimization problem:

$$\min_I \left(\tfrac{\partial}{\partial x}I - F_x\right)^2 + \left(\tfrac{\partial}{\partial y}I - F_y\right)^2 + \lambda\left(I - S\right)^2, \tag{2}$$

where $\lambda = 0.1$ is used to weakly bias the solution towards our approximation. We implemented a simple solver using the Conjugate Gradient Method [Shewchuk 1994] in CUDA that runs in real time on the GPU. Implementation details are provided in the supplementary material.

As with other view interpolation systems, we render a novel view in our system always from *two* reference images. For this, we find the respective closest input camera to the left and to the right of the novel view. We then compute the term images F_x, F_y, S separately for both inputs and combine each corresponding pair of terms into a single term image using linear blending weights proportional to the inverse distance of the cameras. Finally, we solve a single combined Poisson system using Equation 2.

4 Horizontal camera motion

In this section, we describe a simple special case that illustrates the key concepts of our method while avoiding some of the complications that arise in the general case. In this setting, the camera moves horizontally (basically corresponding to a rectified setting). In this case, the epipolar lines are parallel to the x-axis and all points move only horizontally between the original and novel viewpoints, without any scaling. This makes splatting the gradient fields F_x and F_y particularly easy: we distribute each gradient's value to the two nearest pixels using linear splat kernels.

To compute the approximate solution S, we initialize it by copying the input photo and then shift the horizontal gradients one by one from their original locations to their new locations, updating S as the gradients move across each pixel (Figure 4). For this operation, we consider the gradients to be located on the boundary between pixels (red lines in Figure 4). A gradient that shifts right is *subtracted* from the image, and a gradient that shifts left is *added*. This operation is only applied to the horizontal gradients in this section. The vertical gradients do not affect the approximate solution S. In practice, we implement this operation by rendering a one pixel wide horizontal line for each gradient, connecting the original and new locations, using the appropriate color value and additive blending.

Given F_x, F_y, and S we can now solve the Poisson problem given in Equation 2 using the conjugate gradient solver.

5 General camera motion

Handling general camera motion involves the same steps as the horizontal case described before: splatting the gradient fields F_x, F_y, and computing an approximate solution S. However, it also poses some new challenges that we did not encounter before. First of all, gradients can now move both horizontally and vertically. Moreover, the mapping from original to novel view can involve rotation and scaling (of pixels) that we have to handle.

Let us define some notation first. We consider the pixels in the original image as fronto-parallel squares in 3D world space, whose corner vertices $\mathbf{v}_0, \ldots, \mathbf{v}_3$ are located at the depth estimated as described in Section 3. The horizontal gradient is assumed to be located at the right boundary, $\overline{\mathbf{v}_1\mathbf{v}_3}$, and the vertical gradient at the bottom boundary, $\overline{\mathbf{v}_2\mathbf{v}_3}$, colored red and green in Figure 5, respectively.

Let $\mathbf{T}_\mathrm{o} = \mathbf{P}\mathbf{M}_\mathrm{o}$ and $\mathbf{T}_\mathrm{n} = \mathbf{P}\mathbf{M}_\mathrm{n}$ be the transformation matrices that map homogeneous world vectors to screen space for the original and novel view, respectively. $\mathbf{P}$ is a 4×3 projection matrix, and $\mathbf{M}_\mathrm{o}, \mathbf{M}_\mathrm{n}$ are 4×4 model-view matrices, all defined for a right handed coordinate system (e.g., as in the OpenGL API).

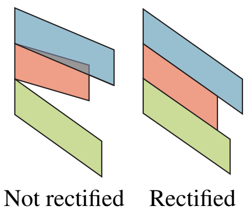
5.1 Splatting gradients

When generating the gradient term images, we need to splat line segments rather than points, to account for zoomed and rotated gradient boundaries in the novel view. For each horizontal and vertical gradient, we rasterize a 1 pixel-thick line segment connecting $\mathbf{T}_\mathrm{n}\mathbf{v}_i$ and $\mathbf{T}_\mathrm{n}\mathbf{v}_j$, where i and j index the two corners involved. The gradient values are splatted in an additive manner as before. Figure 5c shows the spines of these anti-aliased lines as dark red and green lines.

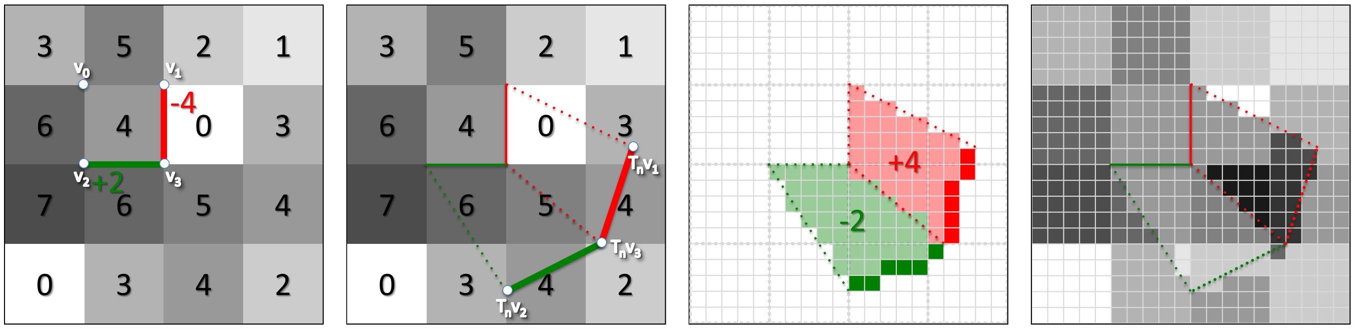
5.2 Computing the approximate solution

The goal of computing the approximate solution remains the same as before, but we now have to consider both horizontal and vertical gradients. We first compute the exact continuous geometry of the region affected by a gradient by forming a quad Q connecting the two pixel corners in the original and novel views, i.e., $Q = (\mathbf{T}_\mathrm{o}\mathbf{v}_i, \mathbf{T}_\mathrm{o}\mathbf{v}_j, \mathbf{T}_\mathrm{n}\mathbf{v}_j, \mathbf{T}_\mathrm{n}\mathbf{v}_i)$. In Figure 5b we show the resulting geometry for a pixel's horizontal and vertical gradients. The quads are rasterized to determine the pixels involved (Figure 5c). For a horizontal/vertical gradient that moves right/down, its value is *subtracted* from the frame buffer, otherwise it is *added* (Figure 5d). A simple way to implement this condition is to test whether the quad is front- or back-facing.

5.3 Rectified streaks

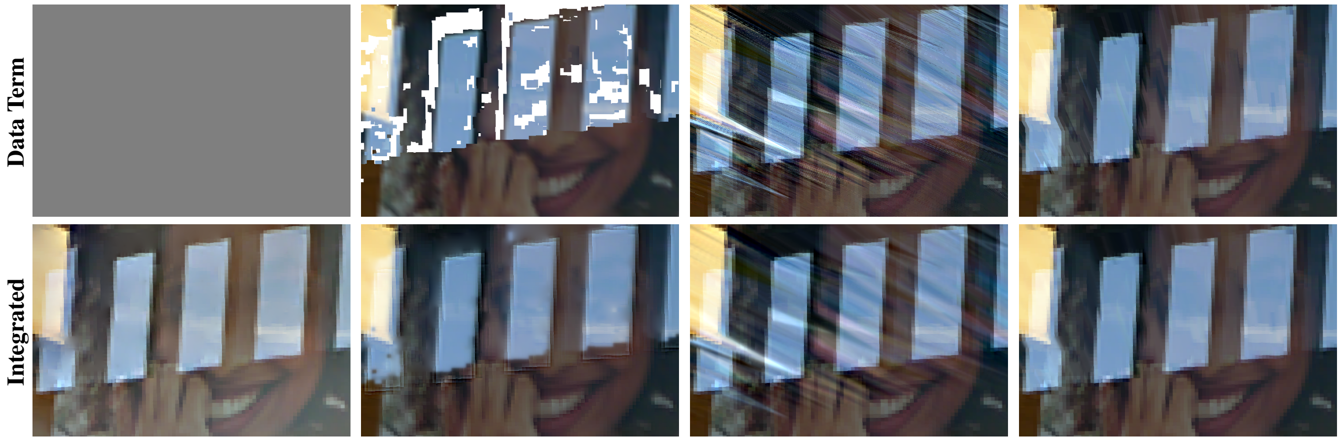


A complication when computing the approximate solution in the general case is that gradients are not all moving in the same direction anymore. This can lead to artifacts, e.g., gaps opening between neighboring gradients and clutter due to gradients overlapping chaotically (see inset figure). We correct this behavior by rectifying the original and novel cameras, so that all gradient shift directions become aligned. (Alternatively, we could use a more sophisticated



(a) Gradients in original view (b) Gradients in novel view (c) Rasterization (d) Updated data term

Figure 5: *Computing the approximate solution in the general case. Each gradient (a) generates a quad connecting its locations in the original and novel view (b). The quad is rasterized (c) and splatted with additive blending updating the underlying image (d). In Figures (c) and (d), the small pixels can also be thought of as the super-samples used in anti-aliasing. The dark red and green lines in (c) are the spine of the 1-pixel wide anti-aliased lines used to splat the gradients into the gradient buffer.*



(a) Middle gray data term (b) Depth reprojection data term (c) Our *unrectified* data term (d) Our *rectified* data term

Figure 6: *Comparing different data terms (top row) for regularizing the Poisson reconstruction (bottom row): (a) a constant middle gray data term does not always reproduce the correct colors, (b) setting the data term to a depth reprojected image produces artifacts in poorly reconstructed (typically untextured) regions, (c) our* unrectified *data term suffers from clutter, (d) our* rectified *data term yields the best results.*

model for the pixel geometry, deviating from the fronto-parallel assumption.)

Rectifying two cameras involves rotating them such that the epipolar lines are aligned with the x-axis [Loop and Zhang 1999]. We achieve this by replacing the original model-view matrices with new ones,

$$\mathbf{M}_\mathrm{o} := \begin{bmatrix} \mathbf{R} & -\mathbf{R}\mathbf{t}_\mathrm{o} \\ \mathbf{0}^\top & 1 \end{bmatrix}, \quad \mathbf{M}_\mathrm{n} := \begin{bmatrix} \mathbf{R} & -\mathbf{R}\mathbf{t}_\mathrm{n} \\ \mathbf{0}^\top & 1 \end{bmatrix}, \tag{3}$$

where $\mathbf{t}_\mathrm{o}$ and $\mathbf{t}_\mathrm{n}$ are the positions of the original and novel cameras, and $\mathbf{R} = \left[\mathbf{r}_\mathrm{right}\ \mathbf{r}_\mathrm{up}\ -\mathbf{r}_\mathrm{front}\right]^\top$ is the usual rotation matrix.

The camera right vector is set parallel to $\overline{\mathbf{t}_\mathrm{o}\mathbf{t}_\mathrm{n}}$,

$$\mathbf{r}_\mathrm{right} = \frac{\mathbf{t}_\mathrm{n} - \mathbf{t}_\mathrm{o}}{\|\mathbf{t}_\mathrm{n} - \mathbf{t}_\mathrm{o}\|}, \tag{4}$$

and the two other base vectors are obtained through cross products,

$$\mathbf{r}_\mathrm{front} = \frac{\mathbf{w}_\mathrm{up} \times \mathbf{r}_\mathrm{right}}{\|\mathbf{w}_\mathrm{up} \times \mathbf{r}_\mathrm{right}\|}, \text{ and} \tag{5}$$

$$\mathbf{r}_\mathrm{up} = \mathbf{r}_\mathrm{right} \times \mathbf{r}_\mathrm{front}. \tag{6}$$

Here, $\mathbf{w}_\mathrm{up}$ is the global world space up vector.

There are some issues with this approach. First, the rotation changes the novel view, so we would have to use image warping in a post-process to get the desired result. Warping is undesirable because it can introduce sampling artifacts. More critically, the rectified views can become extremely distorted under certain conditions, in particular, if one camera center approaches the frustum of the other.

We remedy both problems by computing an extra screen space homography H that maps the rectified novel view back to the *non-rectified* novel view, and applying this homography to both novel and original geometry *before* rasterization, i.e., $\mathrm{H} = \mathbf{T}_\mathrm{o}\mathbf{T}_\mathrm{n}^+ = \mathbf{T}_\mathrm{o}\mathbf{T}_\mathrm{n}^\top \left(\mathbf{T}_\mathrm{n}\mathbf{T}_\mathrm{n}^\top\right)^{-1}$. The final transformation matrices are therefore $\mathbf{T}'_\mathrm{o} = \mathrm{H}\mathbf{T}_\mathrm{o}$ and $\mathbf{T}'_\mathrm{n} = \mathrm{H}\mathbf{T}_\mathrm{n}$. This procedure avoids post-process image warping and any problems due to distortion and produces improved results, as shown in Figures 6c–d.

6 Occlusions and disocclusions

A major challenge for image-based rendering approaches are occlusions and disocclusions. In traditional image-based rendering

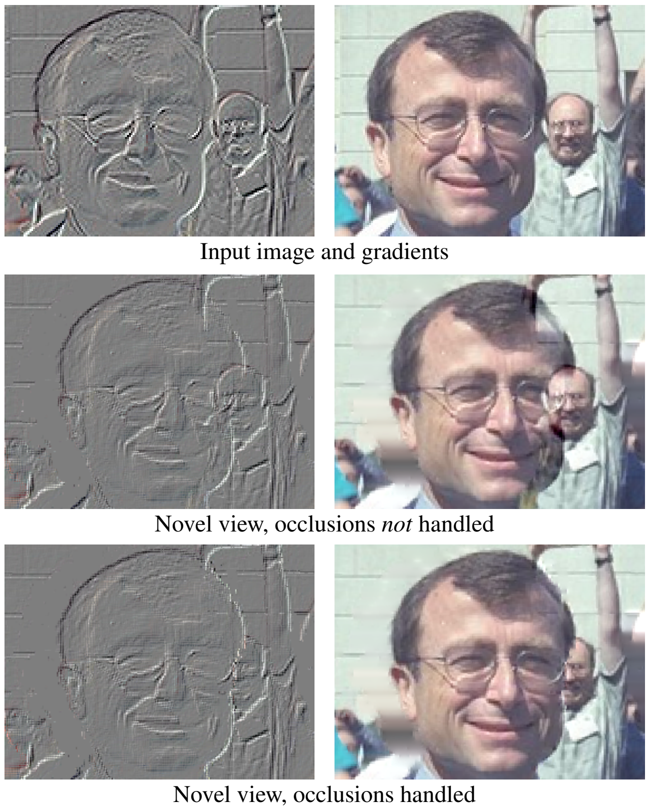

Input image and gradients
Novel view, occlusions *not* handled
Novel view, occlusions handled

Figure 7: *Occlusions and disocclusions. Note how the disocclusions to the left of the neck are smoothly filled in by the Poisson integration.*

approaches, handling occlusions requires careful reconstruction of the proxy geometry at the boundary in order to avoid artifacts due to sharp but incorrect edges. Disocclusions are similarly complex since the geometry that becomes disoccluded needs to be modeled as well, e.g., using a multi-layered representation [Shade et al. 1998; Zitnick et al. 2004].

In the gradient domain, occlusions must be handled differently, because gradients with different depths do not necessarily occlude each other; for reflective or semi-transparent surfaces, gradients on different layers just add together.

A naïve approach is to not handle occlusions at all, i.e., splat all gradients to their projected position and integrate the resulting gradient field. If the input image density is high enough, one can rely on blending between images from different viewpoints to avoid artifacts at the cost of larger use of resources. In many cases, this works surprisingly well since in particular disocclusions are filled in smoothly and consistently by this approach (see Figure 7, to the left of the head). However, in scenes with high amounts of occlusions, this leads to ghosting artifacts when gradients of opaque surfaces are added together. In the following, we therefore develop a more principled approach for detecting and handling occlusions and disocclusions.

6.1 Detection and modeling

As we cannot use depth as a sole indicator for occlusions due to reflections, we need to do an explicit search for gradients that vanish at occlusion boundaries. Given a reference view, we first detect whether its gradients are still visible in one of the neighboring views

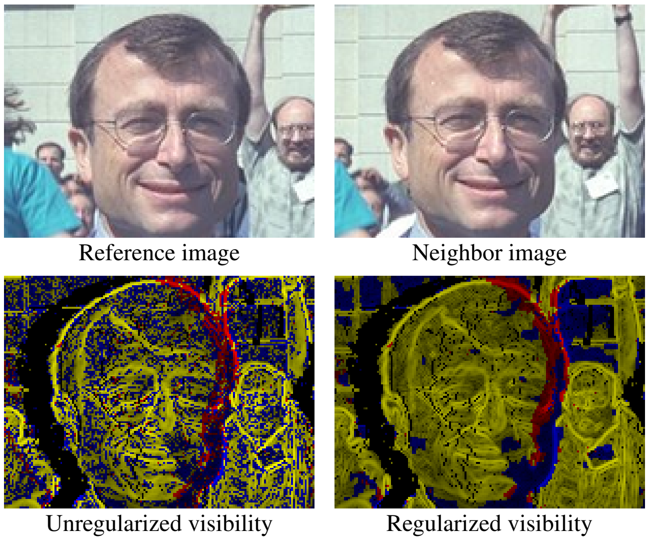


Figure 8: *Input images and gradients classified according to their visibility in a neighboring view. Yellow: gradient visible. Blue: gradient occluded. Red: both occluded and visible gradients are projected to this location. All colors are weighted with the gradient magnitude to improve visualization.*

at (or near) the location predicted by the epipolar geometry. This problem is closely related to the reconstruction as described in Section 3.1 but requires a more robust formulation due to the fact that we need to argue locally (for a single neighboring image) and not globally for the complete set of input images.

In order to determine whether a gradient is visible in a neighboring view, we compute a similarity measure c between single-pixel gradients which is defined as the product of the angle cost $c_\angle = \mathbf{g}_1^\top \mathbf{g}_2 / \|\mathbf{g}_1\| \|\mathbf{g}_2\|$ (as introduced near Equation 1) and a cost c_s that measures the intensity difference between two gradients:

$$c = c_\angle^k \cdot c_s, \tag{7}$$

$$c_s = \min\left(\frac{\|\mathbf{g}_n\|}{\|\mathbf{g}_r\|}, \frac{\|\mathbf{g}_r\|}{\|\mathbf{g}_n\|}\right). \tag{8}$$

$\mathbf{g}_r$ is the gradient in the reference view at location $\mathbf{T}_\mathrm{o}\mathbf{v}$, $\mathbf{g}_n$ is the gradient in the neighboring view at the corresponding location $\mathbf{T}_\mathrm{n}\mathbf{v}$, and k determines the angular selectivity of the measure (we always use $k = 10$). The intuition behind this is that gradients should only match if they have a similar direction as well as a similar magnitude. In order to be robust to various errors (e.g., miscalibration, slightly non-planar reflectors), we do not strictly enforce the epipolar geometry. Instead of using c directly we compute $c_{max} = \max_{\mathcal{N}} c$ in a 5×5 pixel neighborhood $\mathcal{N}$ around $\mathbf{T}_\mathrm{n}\mathbf{v}$. In other words, we compute the maximal similarity score within the neighborhood $\mathcal{N}$ around the location $\mathbf{T}_\mathrm{n}\mathbf{v}$ predicted by the epipolar geometry.

To finally determine whether a given gradient is visible in each of the other views, we solve per-view a binary graph-cut with the data terms c_{max} (occluded) and $1 - c_{max}$ (visible) for the respective labels. The regularization weights between neighboring pixels are computed with a simple Potts model, assigning 0 if both labels are identical and 0.3 if they differ (see Figure 8).

For the rendering approach described in the next section, we not only need to determine which gradients eventually become occluded, but also the 3D position of their occluders. Since this is generally ambiguous in the reflective case, we use a simple heuristic: If a gradient $\mathbf{v}$ becomes occluded between neighboring views i

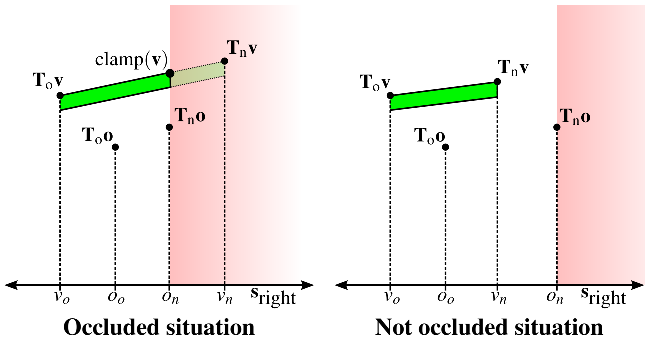


Figure 9: *Rendering occlusions. The candidate and occluding gradient positions in the original (o) and novel (n) views are projected along the $\mathbf{s}_{right}$ vector and their values are compared. The occluded half-space is indicated by red shading.* Left: *the gradient is occluded since $v_o < o_o$ but $v_n > o_n$; the quad is clamped.* Right: *the gradient is not occluded since $v_o < o_o$ and $v_n < o_n$.*

and j, its occluder $\mathbf{o}$ must cross the epipolar line of $\mathbf{v}$ between i and j and it must be closer to the camera than $\mathbf{v}$. We therefore search along the epipolar line and select the first gradient with magnitude larger than 0.001 and with a smaller depth than $\mathbf{v}$ as the occluder $\mathbf{o}$.

6.2 Rendering

Let $\mathbf{v}$ be a gradient and $\mathbf{o}$ its occluder. If the novel camera position is not exactly on the connecting line between the input camera positions in which $\mathbf{v}$ and $\mathbf{o}$ are defined, they will generally not intersect in screen space. Therefore, to be more robust we consider projections onto a common screen space axis $\mathbf{s}_{\text{right}}$, where $\mathbf{s}_{\text{right}}$ is the right camera axis $\mathbf{r}_{\text{right}}$ projected into screen space. The resulting original and new projections onto this axis are therefore

$$v_{\text{o}} = (\mathbf{T}_{\text{o}}\mathbf{v})^{\top}\mathbf{s}_{\text{right}}, \quad v_{\text{n}} = (\mathbf{T}_{\text{n}}\mathbf{v})^{\top}\mathbf{s}_{\text{right}}, \tag{9}$$

with analogous definitions for o_{o} and o_{n}.

If the order of v and o flips when comparing original and novel views, i.e.,

$$(v_{\text{o}} < o_{\text{o}}) \text{ XOR } (v_{\text{n}} < o_{\text{n}}), \tag{10}$$

the gradient $\mathbf{v}$ is occluded. In other words, o_{n} defines a half-space in which $\mathbf{v}$ is occluded. Figure 9 illustrates this situation.

When rendering the gradient terms F_x and F_y, we evaluate Equation 10 in the vertex shader, and simply discard occluded gradients. When rendering the approximate solution S we cannot simply discard vertices, because this fails to account for the additive sweeping effect of the partially occluded gradient (until it becomes occluded). Instead, we clamp the quad Q (see Section 5.2) against the half space defined by o_{n}, i.e., we replace $Q \leftarrow \big(\mathbf{T}_{\text{o}}\mathbf{v}_i, \mathbf{T}_{\text{o}}\mathbf{v}_j, \text{clamp}(\mathbf{v}_j), \text{clamp}(\mathbf{v}_i)\big)$, where

$$\text{clamp}(\mathbf{v}) = \mathbf{T}_{\text{o}}\mathbf{v} + (\mathbf{T}_{\text{n}}\mathbf{v} - \mathbf{T}_{\text{o}}\mathbf{v})\,\frac{o_{\text{n}} - v_{\text{o}}}{v_{\text{n}} - v_{\text{o}}}. \tag{11}$$

This clamping operation is visualized in Figure 9.

7 Results

We have developed a prototype renderer implementation in C++ using OpenGL. The Poisson integration is implemented with CUDA (please see the supplementary material for implementation details). We used this implementation to generate all the figures in the paper and in the supplementary material.

Render step	Average	Max	Min	St. dev.
Horz. gradients	1.056	1.242	0.738	0.174
Vert. gradients	1.132	1.313	0.814	0.170
Approx. solution	4.721	8.763	1.742	1.650
Poisson	13.343	14.033	13.105	0.358
Total	**22.616**	**27.856**	**18.761**	**2.089**

Table 1: *Rendering times (in milliseconds) averaged across 5 different datasets showing the distribution of times according to rendering components / stages.*

Performance: All rendering stages except the approximate solution run in constant time for a given input image size. The approximate solution stage is fill rate bound, and its performance depends on the visible length of the streaks.

In Table 1, we summarize timings for 5 different datasets. Each dataset uses input images of size 640×480 and is rendered at an output size of 720×480 on an NVIDIA GTX 680. For each dataset, we measured performance for ten random novel views along an arc connecting the input cameras, yielding the aggregate statistics shown in Table 1.

Results: We tested our approach on a variety of input sequences captured using hand-held photography under mostly horizontal camera motion. (We use primarily one-dimensional motion to make the interactive view navigation simpler, as was done by Sinha et al. [2012].) Please see our supplementary video and Web page for more results, since these show the visual artifacts much more clearly than the still images in this paper.

Figure 10 shows view extrapolation results produced from just a single reference color image and depth map. As you can see, standard image-based rendering has great difficulty dealing with the two depths present in the wood grain table top and the reflected window. In contrast, our technique handles this reflective scene without difficulty. Figure 2 and Figure 11 compare our results to [Sinha et al. 2012] and show the advantage of not having to separate various layers. Results for a non-reflective scene are shown in Figure 12. Compared to regular image-based rendering or piecewise planar solutions, our technique introduces fewer artifacts when the proxy geometry contains small errors. In particular, it tends to produce higher quality for datasets with wide base lines, as long as the proxy geometry is accurate. In the supplementary material we test this by rendering a scene using only a fraction of all input images, while the proxies were computed from the full dataset.

Limitations: Compared to previous image-based rendering approaches, our new technique almost always produces improved results. However, there are still cases where it produces visible artifacts. The most common case is when the stereo algorithm associates incorrect depth values with a gradient or edge. This occurs most often in two situations: for horizontal edges, since their depth is hardest to estimate for mostly horizontal camera motion, and for cluttered random textures. A challenging example for the second case is the TREE scene, which we included in the supplementary material. The occlusion detection algorithm also occasionally makes mistakes, which results in ghosted gradients and edges extending beyond the regions of a reflective surface. Finally, our rendering approach requires the use of specialized shaders and higher-end graphics hardware, which means that it may not run on all computing platforms (e.g., under-powered mobile phones).

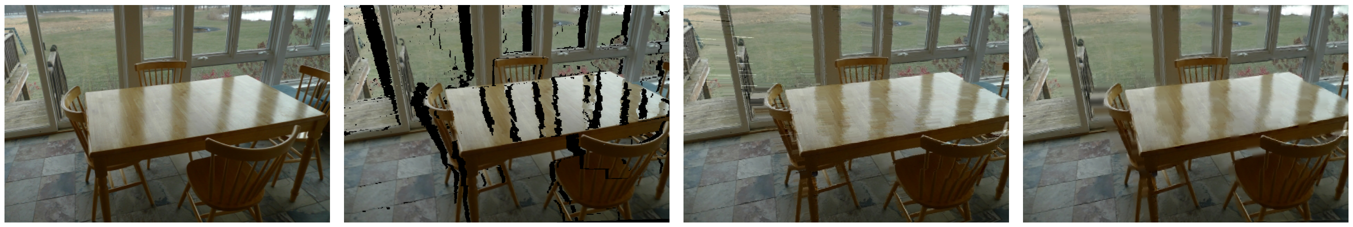


Figure 10: *View extrapolation using a single image and depth map for the* SUNROOM *dataset. The same proxy geometry was used for standard image-based rendering and our method.*

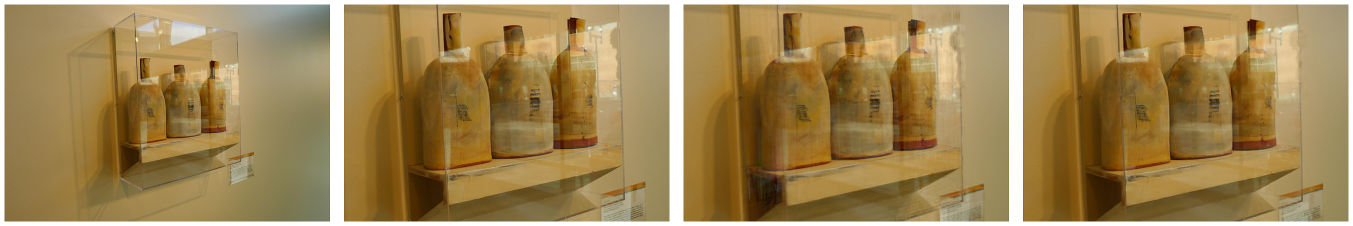


Figure 11: *For the* MUSEUM *dataset, the planar proxies of the reflection layers used by Sinha et al. cause incorrect reflections in the bottom left. Please see the video to see these artifacts more clearly.*

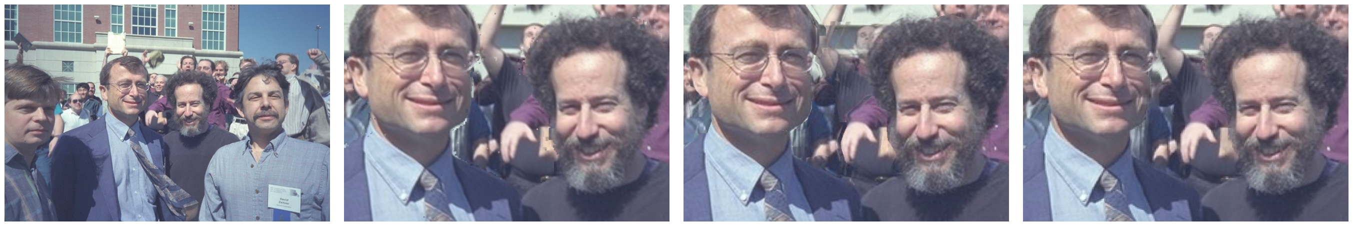


Figure 12: *Input images and sample novel views (zoomed in) for the* CONFERENCE *dataset. The same proxy geometry was used for standard image-based rendering and our method. Please see the video for details.*

8 Discussion and conclusions

In this paper, we have developed novel gradient-based 3D reconstruction and image-based rendering algorithms. Compared to traditional image-based modeling and rendering systems, which associate one or more depths and colors with each pixel in a reference view, our method associates depths with gradients and reconstructs both a novel gradient image and a lower-fidelity guide image from the gradient motions. This has the advantage of both handling multiple depths such as seen in reflective surfaces and mitigating depth errors. For example, traditional image-based rendering may produce "floater pixels" or "holes" at pixels with erroneous depths in low-texture regions. With our approach, errors at such pixels have little effect, since they contribute very little (if at all) to the final rendering.

While our results look promising, there are several areas we would like to explore in future work. One such area is to re-examine the resampling we perform when forward rendering gradients as antialiased lines. Using a two-pass rendering algorithm, which first computes a displacement field in the new view and then resamples the gradient image, may improve the quality of the rendering [Shade et al. 1998]. We would also like to explore improved methods for estimating depths, including the use of alternative (e.g., gradient-based) matching cost functions as well as representations and regularizers that support multiple depth estimates per pixel [Tsin et al. 2006]. Finally, we would like to apply our new techniques to an even wider range of image-based rendering scenarios, e.g., large-scale outdoor environments and scenes with curved reflectors.

Acknowledgments

The CONFERENCE dataset is courtesy of Dayton Taylor. Daniel Scharstein acknowledges support by NSF grant IIS-0917109. Michael Goesele acknowledges support by DFG Emmy Noether fellowship GO 1752/3-1.

References

BEERY, E., AND YEREDOR, A. 2008. Blind separation of superimposed shifted images using parameterized joint diagonalization. *IEEE Transactions on Image Processing 17*, 3, 340–353.

BERGEN, J. R., BURT, P. J., HINGORANI, R., AND PELEG, S. 1992. A three-frame algorithm for estimating two-component image motion. *IEEE Transactions on Pattern Analysis and Machine Intelligence 14*, 9, 886–896.

BHAT, D. N., AND NAYAR, S. K. 1998. Stereo and specular reflection. *International Journal of Computer Vision 26*, 2, 91–106.

BUEHLER, C., BOSSE, M., MCMILLAN, L., GORTLER, S. J., AND COHEN, M. F. 2001. Unstructured Lumigraph rendering. *Proc. SIGGRAPH 2001*, 425–432.

CARCERONI, R. L., AND KUTULAKOS, K. N. 2002. Multi-view scene capture by surfel sampling: From video streams to non-rigid 3D motion, shape and reflectance. *International Journal of Computer Vision 49*, 2/3, 175–214.

CHAURASIA, G., DUCHENE, S., SORKINE-HORNUNG, O., AND DRETTAKIS, G. 2013. Depth synthesis and local warps for plausible image-based navigation. *ACM Transactions on Graphics 32*, 3, Article no. 30.

CHEN, S., AND WILLIAMS, L. 1993. View interpolation for image synthesis. *Proc. SIGGRAPH '93*, 279–288.

CRIMINISI, A., KANG, S. B., SWAMINATHAN, R., SZELISKI, R., AND ANANDAN, P. 2005. Extracting layers and analyzing their specular properties using epipolar-plane-image analysis. *Computer Vision and Image Understanding 97*, 1, 51–85.

DEBEVEC, P. E., TAYLOR, C. J., AND MALIK, J. 1996. Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach. *Proc. SIGGRAPH '96*, 11–20.

DIAMANT, Y., AND SCHECHNER, Y. Y. 2008. Overcoming visual reverberations. *Proc. Computer Vision and Pattern Recognition (CVPR'08)*, 1–8.

EISEMANN, M., DE DECKER, B., MAGNOR, M., BEKAERT, P., DE AGUIAR, E., AHMED, N., THEOBALT, C., AND SELLENT, A. 2008. Floating textures. *Computer Graphics Forum (Proc. Eurographics 2008) 27*, 2, 409–418.

GOESELE, M., ACKERMANN, J., FUHRMANN, S., HAUBOLD, C., KLOWSKY, R., STEEDLY, D., AND SZELISKI, R. 2010. Ambient point clouds for view interpolation. *ACM Transactions on Graphics (Proc. SIGGRAPH 2010) 29*, 4, Article no. 95.

GORTLER, S. J., GRZESZCZUK, R., SZELISKI, R., AND COHEN, M. F. 1996. The Lumigraph. *Proc. SIGGRAPH '96*, 43–54.

HIRSCHMÜLLER, H. 2008. Stereo processing by semiglobal matching and mutual information. *IEEE Transactions on Pattern Analysis and Machine Intelligence 30*, 2, 328–341.

IRANI, M., ROUSSO, B., AND PELEG, S. 1994. Computing occluding and transparent motions. *International Journal of Computer Vision 12*, 1, 5–16.

JU, S. X., BLACK, M. J., AND JEPSON, A. D. 1996. Skin and bones: Multi-layer, locally affine, optical flow and regularization with transparency. *Proc. Computer Vision and Pattern Recognition (CVPR '96)*, 307–314.

KOLMOGOROV, V., AND ZABIH, R. 2002. Multi-camera scene reconstruction via graph cuts. *Proc. European Conference on Computer Vision 2002 (ECCV 2002)*, 82–96.

LEVIN, A., ZOMET, A., AND WEISS, Y. 2004. Separating reflections from a single image using local features. *Proc. Computer Vision and Pattern Recognition 2004 (CVPR 2004) 1*, 306–313.

LEVOY, M., AND HANRAHAN, P. 1996. Light field rendering. *Proc. SIGGRAPH '96*, 31–42.

LINZ, C., LIPSKI, C., AND MAGNOR, M. 2010. Multi-image interpolation based on graph-cuts and symmetric optical flow. *Proc. Vision, Modeling and Visualization 2010 (VMV 2010)*, 115–122.

LOOP, C., AND ZHANG, Z. 1999. Computing rectifying homographies for stereo vision. *Proc. Computer Vision and Pattern Recognition (CVPR '99)*, 125–131.

MAHAJAN, D., HUANG, F.-C., MATUSIK, W., RAMAMOORTHI, R., AND BELHUMEUR, P. 2009. Moving gradients: a path-based method for plausible image interpolation. *ACM Transactions on Graphics (Proc. SIGGRAPH 2009) 28*, 3, Article no. 42.

PÉREZ, P., GANGNET, M., AND BLAKE, A. 2003. Poisson image editing. *ACM Transactions on Graphics (Proc. SIGGRAPH 2003) 22*, 3, 313–318.

POPESCU, V., MEI, C., DAUBLE, J., AND SACKS, E. 2006. Reflected-scene impostors for realistic reflections at interactive rates. *Computer Graphics Forum (Proc. Eurographics 2006) 25*, 3, 313–322.

SCHECHNER, Y. Y., SHAMIR, J., AND KIRYATI, N. 1999. Polarization-based decorrelation of transparent layers: The inclination angle of an invisible surface. *Proc. International Conference on Computer Vision (ICCV '99)*, 814–819.

SCHECHNER, Y. Y., KIRYATI, N., AND SHAMIR, J. 2000. Blind recovery of transparent and semireflected scenes. *Proc. Computer Vision and Pattern Recognition (CVPR 2000)*, 38–43.

SHADE, J., GORTLER, S., HE, L., AND SZELISKI, R. 1998. Layered depth images. *Proc. SIGGRAPH '98*, 231–242.

SHEWCHUK, J. R. 1994. An introduction to the conjugate gradient method without the agonizing pain. Tech. rep., Carnegie Mellon University.

SHIZAWA, M., AND MASE, K. 1991. A unified computational theory of motion transparency and motion boundaries based on eigenenergy analysis. *Proc. Computer Vision and Pattern Recognition (CVPR '91)*, 289–295.

SHUM, H.-Y., CHAN, S.-C., AND KANG, S. B. 2007. *Image-Based Rendering*. Springer, New York, NY.

SINHA, S. N., STEEDLY, D., AND SZELISKI, R. 2009. Piecewise planar stereo for image-based rendering. *Proc. International Conference on Computer Vision (ICCV 2009)*, 1881–1888.

SINHA, S. N., KOPF, J., GOESELE, M., SCHARSTEIN, D., AND SZELISKI, R. 2012. Image-based rendering for scenes with reflections. *ACM Transactions on Graphics (Proc. SIGGRAPH 2012) 31*, 4, Article no. 100.

SNAVELY, N., SEITZ, S. M., AND SZELISKI, R. 2006. Photo tourism: exploring photo collections in 3D. *ACM Transactions on Graphics (Proc. SIGGRAPH 2006) 25*, 3, 835–846.

SZELISKI, R., AVIDAN, S., AND ANANDAN, P. 2000. Layer extraction from multiple images containing reflections and transparency. *Proc. Computer Vision and Pattern Recognition (CVPR 2000)*, 246–253.

TSIN, Y., KANG, S. B., AND SZELISKI, R. 2006. Stereo matching with linear superposition of layers. *IEEE Transactions on Pattern Analysis and Machine Intelligence 28*, 2, 290–301.

ZITNICK, C. L., KANG, S. B., UYTTENDAELE, M., WINDER, S., AND SZELISKI, R. 2004. High-quality video view interpolation using a layered representation. *ACM Transactions on Graphics (Proc. SIGGRAPH 2004) 23*, 3, 600–608.