Image-Based Rendering for Scenes with Reflections

Sudipta N. Sinha, Microsoft Research
Johannes Kopf, Microsoft Research
Michael Goesele, TU Darmstadt
Daniel Scharstein, Middlebury College
Richard Szeliski, Microsoft Research

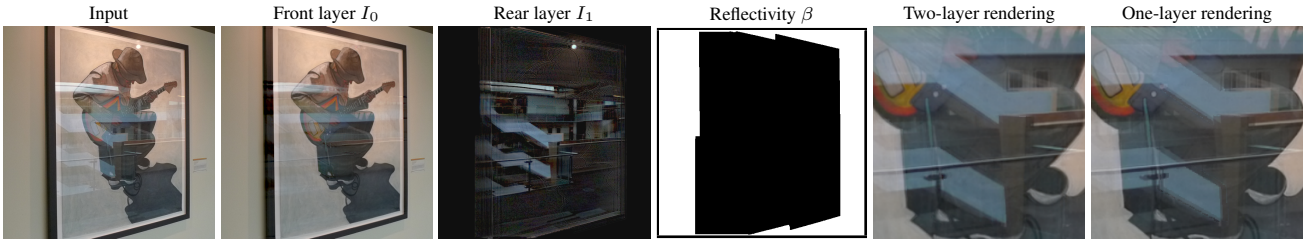


Figure 1: *Two-layer decomposition for the* GUITARIST *image sequence: (a) portion of a sample input image, (b–c) its decomposition into the two layers I_0 and I_1, (d) its reflection map β, (e) detail from an interpolated view using the two-layer model, and (f) using a one-layer model. Notice how the one-layer rendering shows significant ghosting (doubled edges). Please zoom in on the images to see more details.*

Abstract

We present a system for image-based modeling and rendering of real-world scenes containing reflective and glossy surfaces. Previous approaches to image-based rendering assume that the scene can be approximated by 3D proxies that enable view interpolation using traditional back-to-front or z-buffer compositing. In this work, we show how these can be generalized to multiple layers that are combined in an additive fashion to model the reflection and transmission of light that occurs at specular surfaces such as glass and glossy materials. To simplify the analysis and rendering stages, we model the world using piecewise-planar layers combined using both additive and opaque mixing of light. We also introduce novel techniques for estimating multiple depths in the scene and separating the reflection and transmission components into different layers. We then use our system to model and render a variety of real-world scenes with reflections.

Keywords: Image-based rendering, image-based modeling, stereo, reflections, layers.

Links: DL PDF

1 Introduction

Image-based modeling and rendering are powerful techniques for creating and visualizing photorealistic models of the real world. Image-based modeling and related photogrammetric and computer vision techniques are now widely used to create accurate texture-mapped models of urban landscapes. However, such models appear "flat" or "cardboard-like", since they do not model the variation in appearance as the viewer moves throughout the environment.

Image-based rendering techniques such as view-dependent texture maps, Lumigraphs, and Light Fields [Debevec et al. 1996; Gortler et al. 1996; Levoy and Hanrahan 1996], provide increased realism by blending between different viewpoints. However, they still fail to realistically depict scenes with reflections and gloss, since they only use a single 3D proxy to locally model the scene.

What happens at surfaces with reflections that violate the assumptions of current image-based rendering systems? The answer is that we observe the superimposed (transparent) motion of two different scenes or textures at the same spatial location (Figure 2). In order to correctly render such scenes, we need to separately render each of these layers, using appropriate per-layer geometry, and then additively combine these images to produce the final rendering (Figure 1).

While techniques have been developed in computer vision to tease apart these layers (see Section 2), they have not been specifically targeted at image-based rendering. In this paper, we develop novel techniques and a complete image-based modeling and rendering system for handling and depicting such scenes.

Our aim in this work is *not* to recover a true decomposition of the 3D scene into surfaces, normals, materials, and reflections or incident illumination. Instead, we take the usual image-based rendering approach of interpolating nearby views to create new images. However, instead of moving each piece of a reference image according to a single depth, we decompose the reflective parts of the scene into two components, i.e., the *transmitted* and *reflected* layers, each with its own 3D proxies, and blend these at rendering time.

2 Previous work

The main idea behind image-based rendering is that we can pre-render or capture realistic views of a 3D scene and then interpolate between them to render novel views [Chen and Williams 1993]. The foundational theory behind these approaches was worked out in the seminal Light Field and Lumigraph papers [Levoy and Hanrahan 1996; Gortler et al. 1996] and consists of interpolating rays sampled from a four-dimensional light field to generate the required rays for a novel view. The quality of light field interpolation and resampling can be dramatically improved by storing or recovering 3D proxies for the scene, since these provide more accurate descriptions of the motions of corresponding pixels and hence higher quality interpolation [Gortler et al. 1996]. Such proxies (billboards

ACM Reference Format
Sinha, S., Kopf, J., Goesele, M., Scharstein, D., Szeliski, R. 2012. Image-Based Rendering for Scenes with Reflections. *ACM Trans. Graph. 31* 4, Article 100 (July 2012), 10 pages.
DOI = 10.1145/2185520.2185596 http://doi.acm.org/10.1145/2185520.2185596.

Copyright Notice
Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or direct commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701, fax +1 (212) 869-0481, or permissions@acm.org.
© 2012 ACM 0730-0301/2012/08-ART100 $15.00 DOI 10.1145/2185520.2185596
http://doi.acm.org/10.1145/2185520.2185596

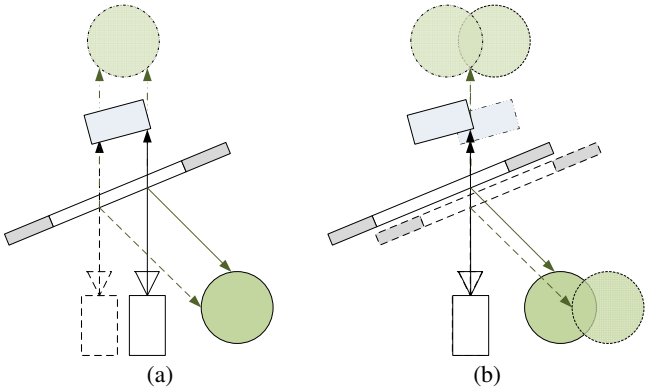


Figure 2: *Basic image formation process. (a) A ray from the camera is reflected from the glass and hits the green sphere and is refracted through the glass and hits the blue box; the dashed cameras and rays indicate the second image, while the dashed circle indicates the (additive) virtual image. (b) The same process, but drawn as if the camera were still and the world (including the reflector) were moving.*

or depth maps) can also be used to store simple descriptions for computing reflections in rendered imagery [Popescu et al. 2006].

Unfortunately, when reflections are present, what we observe is an additive mixing of colors between two (or more) separate layers, each of which moves at a rate depending on the *virtual depth* of the perceived scene component (Figure 2). In this paper, we present a system that estimates separate 3D proxies and color values for different scene components in areas of reflection and transmission. We then use these to interpolate between input views with fewer jumping and ghosting artifacts. While we could use any of a number of 3D representations for proxies, such as global 3D models [Gortler et al. 1996; Buehler et al. 2001], we use piecewise planar models (*aka sprites*) [Shade et al. 1998; Sinha et al. 2009], since these are easier to estimate, compress, and render.

Before we can build such a system, however, we need to develop algorithms to separate the input images into reflected and transmitted components and to compute 3D proxies for each layer. In computer vision, the problem of recovering multiple superimposed motions is known as *motion transparency* and dates back to the early 1990s [Shizawa and Mase 1991; Bergen et al. 1992; Irani et al. 1994; Darrell and Pentland 1995; Ju et al. 1996]. Szeliski *et al.* [2000] provide a good review of this literature. (See [Bhat et al. 2007; Diamant and Schechner 2008; Beery and Yeredor 2008] for extensions of this work to deal with more general scenes, multiple reflections inside glass, and frequency-domain separation techniques.) Tsin *et al.* [2006] give an even more extensive review of the multiple motion literature, and also discuss related techniques such as dealing with specular highlights [Bhat and Nayar 1998; Carceroni and Kutulakos 2002], recovering reflected components using polarizing filters [Schechner et al. 1999], and the local analysis of edges and intensity ratios in single images to separate them into translucent layers [Levin et al. 2002].

In their work, Szeliski *et al.* [2000] first estimate the motion of the *dominant* (higher contrast) layer using a robust parametric (affine or projective) optical flow algorithm. After this initial registration, they compute a *min-composite* to recover a lower bound on this first layer's colors, and then use a *max-composite* of the residual difference images to over-estimate the second layer and to recover its associated motion. They then solve a constrained least-squares problem to refine the images associated with both layers. They also discuss the ambiguities inherent in this problem. In Section 6, we use similar techniques to perform our per-layer image recovery, but use a more accurate piecewise-planar motion recovery algorithm and a more robust formulation to deal with small misalignments.

Tsin *et al.* [2006] describe a stereo matching algorithm that simultaneously estimates two disparities (inverse depth values) at each pixel using an additive color mixing model. To compute these disparities, the algorithm compares quadruplets of color pixel values to see if they are consistent with a paired disparity hypothesis (d_0, d_1). A "graph cut" [Boykov et al. 2001] combinatorial optimization algorithm is then run to estimate the front and rear disparity surfaces $d_0(x, y)$ and $d_1(x, y)$ that simultaneously minimize the matching costs and maximize a smoothness term. Since the number of disparity pair hypotheses at each pixel is quite large (D^2, where D is the number of disparities), their algorithm is quite slow. In this paper, we introduce an alternative algorithm that first aggregates the cost volume using semi-global matching [Hirschmüller 2008] (Section 4), then globally estimates a small number of likely scene planes, and finally uses graph cut optimization to assign each pixel to one or two planes (Section 5), resulting in a much faster algorithm that is also less sensitive to radiometric miscalibrations.

The preceding papers all assume that the reflectors in the scene are planar, resulting in stable virtual images for the reflections (Figure 2). What happens if the reflectors are curved or irregular? For simple reflectors such as lightly curved parabolic mirrors, a stable virtual image is still observed, albeit at an incorrect depth. This inaccuracy, however, will not affect the performance of an image-based rendering system. For more undulating reflectors, the epipolar motion (flow) of pixels becomes less regular and rectilinear. Criminisi *et al.* [2005] analyze this case in more detail, and also develop an algorithm for detecting and removing saturated highlights using epipolar plane image (EPI) analysis.

3 Problem formulation

In order to recover and re-render the two-layer geometry and colors at a reflective surface, we first need to define the image formation process that occurs at such surfaces. Figure 2 shows a diagram for the two-layer model, where a camera sees light transmitted through a material such as glass as well as reflected light, which produces a stable virtual image. For materials such as glossy textured surfaces (paintings matted with glass, textured countertops), the geometry corresponding to the "transmitted" light coincides with the reflective surface. However, our system is designed for general geometries where neither virtual image depth coincides with the reflective material, e.g., the glass case shown in Figure 6a.

At reflective material boundaries, the incident reflected light is attenuated by a fraction $\beta \in [0, 1]$ that depends on the reflective material's BRDF or Fresnel reflection coefficient.[1] The transmitted component can potentially undergo a similar attenuation, but for most materials, such as plate glass or clear coat or varnish, it passes through the interface unattenuated. Given these conditions, the observed composite color C can be described by

$$C = I_0 + \beta I_1, \tag{1}$$

where I_0 denotes the transmitted light and I_1 denotes the reflected light. The reflective fraction β is non-zero at reflective surfaces and zero at matte opaque surfaces, indicated in gray in Figure 2.

When we try to recover the quantities I_0, I_1, and β (along with their appropriate geometries, which we introduce shortly), there arise a few subtle ambiguities. The first of these is that we cannot disambiguate between larger reflection coefficients β and larger

[1]The BRDF can also low-pass filter or blur the incident virtual image. However, we fold this blurring process into the reflection image itself.

amounts of incident light I_1. To resolve this ambiguity, we assume that reflective surfaces have a constant non-zero coefficient β. We can therefore fold this constant factor into I_1 and assume that the β values are either 1 (in two-layer regions of the image) or 0 (in single-layer regions).

The second ambiguity is that unless we use additional cues such as polarization, we cannot distinguish between the transmitted and reflected components, since both appear as stable virtual images. In this paper, therefore, we do not attempt to distinguish between them, and simply call the one whose virtual image appears closer to the camera (smaller depth) as the *front layer* I_0 and the other as the *rear layer* I_1. Equation (1) therefore describes the linear superposition of the front layer I_0 with an optional rear layer I_1.

Our image-based rendering system starts with a collection of input reference images C_i. Each image is decomposed into its front and rear layers I_0 and I_1 and reflectivity map β, along with the inverse depths (disparities) d_0 and d_1 associated with each layer [Szeliski et al. 2000; Tsin et al. 2006]. This decomposition is computed by predicting the appearance of nearby input images C_j from the decomposition of $C_i = \{I_0, I_1, \beta\}$ and then minimizing the discrepancy between the predicted (synthesized) and observed (reference) images.[2]

The image formation equations for nearby views are given by

$$C_j(x) = I_0(u(x, d_0)) + \beta(u(x, d_\beta))\, I_1(u(x, d_1)), \qquad (2)$$

i.e., by warping, multiplying, and adding the appropriate source images. The function $u(x, d_l)$ maps a pixel x in image j into a source pixel address in reference image i, based on the disparity map $d_l(x)$ associated with that particular layer. The form of such maps is a rational linear equation in x and d, which can be derived from the relative pose and intrinsic calibration parameters of the two cameras i and j or equivalently their camera matrices [Szeliski 2010, (2.70) and (11.3)]. These maps are piecewise continuous, but can have tears (gaps) at depth discontinuities where no pixels get mapped to the second view (resulting in a zero intensity contribution).

In practice, there arises a third ambiguity associated with Equation (2). Since the reflective material associated with β is not directly observable, we cannot reliably estimate the depth map d_β associated with this surface, only a range of plausible values.[3] For this reason, we set $d_\beta = d_0$, i.e., we assume that the deformation of the reflector, which affects which pixels in image j see both layer components, is the same as that of the front layer. For single-layer surfaces (the gray regions in the plane in Figure 2), this produces the correct model. For transmissive/reflective pixels, it is still a sensible choice since, in the absence of discontinuities in d_0, it correctly predicts which pixels in the image are reflective. In future work, we would like to develop algorithms for estimating a plausible value for d_β as part of the optimization process.

Minimizing the difference between the observed images and the images synthesized using (2) introduces a set of coupled problems. Given known depths d_0, d_1, and reflectivity values β, solving for I_0 and I_1 is a constrained least-squares problem [Szeliski et al. 2000]. Conversely, if the per-layer colors I_0 and I_1 and the β-map are known for all frames, the problem reduces to two uncoupled stereo matching problems. Unfortunately, we begin with no knowledge of any of these variables. In the next section, we develop a multi-view transparent stereo matching technique that produces multiple plane hypotheses, which are then used in Section 5 to assign each pixel to one or two planes, thereby producing the d_0, d_1, and β maps for every image. Section 6 then describes how we recover the color values using our own variant of regularized weighted constrained least squares. Our rendering approach using the resulting per-image models is described in Section 7.

[2]The actual renderer (Section 7) uses two-view interpolation, but for the purposes of multi-view stereo and layer color recovery, we use the single view interpolation formula.

[3]Consider again the glass case in Figure 6a. The 3D motion of this case determines which pixels in the image will show the reflected virtual image and which will not.

4 Two-layer stereo matching

Before we begin our reconstruction process, we first estimate the locations and intrinsic calibration parameters of the input cameras using feature matching followed by a photogrammetric structure from motion (SfM) pipeline similar to the one described by Snavely *et al.* [2006]. Next, in order to compute two-layer depth map proxies for each reference image, we perform asymmetric stereo matching on calibrated image subsets by extending the semi-global matching approach [Hirschmüller 2008] to handle two layers, as described below.

Computing the disparity space image (DSI): For each input (reference) image, we construct a 3D matching cost volume (defined below) using a plane-sweep framework. The depth range of the stack of fronto-parallel planes is computed based on the depths of the SfM points visible in the reference view, while their spacing is inversely proportional to depth. The minimum of the spacing that corresponds to roughly one pixel shifts in each of the two neighboring views is selected. This corresponds to linear *disparities* (pixel motions) in the case of regularly spaced sideways-looking motion.

The images used for computing the matching costs are selected by finding the N neighboring views which have the maximum number of SfM 3D points in common with the reference view. These images are then warped into the reference view using the set of 2D homographies induced by each individual plane in the plane sweep volume. We use pairwise normalized cross correlation (NCC) as the photo-consistency measure and use it to derive the matching cost volume, i.e., the disparity space image (DSI) [Szeliski 2010]. For small window sizes, NCC performs similarly to matching gradient images, since it subtracts the local mean from each window and hence accentuates high frequencies. We expect such measures to be less sensitive to the addition of secondary light in regions where two images (reflected and transmitted) are being mixed.

To compute the matching cost for a particular plane, we densely compute NCC between the reference image and the j-th warped image. At each pixel p, the final matching cost c_p is computed as

$$c_p \;=\; \sum_{j=1}^{N} (1 - \mathrm{NCC}(\mathbf{x}_p, \mathbf{y}_p^j)), \qquad (3)$$

where vectors $\mathbf{x}_p$ and $\mathbf{y}_p^j$ are $\mu \times \mu$ pixel patches centered at pixel p in the reference image and the j-th warped image respectively. In all our experiments, we set $N = 2$ and $\mu = 3$. The construction of the matching cost volume described so far is identical to a traditional single-layer stereo matching approach.

Semi-global matching: We perform stereo matching using an extension of the semi-global matching (SGM) approach of [Hirschmüller 2008], which is known to be comparable to global methods in terms of accuracy, without being as computationally expensive. In SGM, a particular pixel's depth is computed by aggregating information from pixels along several 1D paths in the image that meet at that pixel. SGM uses dynamic programming to compute the lowest aggregated cost for all pixels on each of these 1D paths. A summed cost is then computed by adding up the costs of the individual paths to obtain a set of aggregated costs $C_p(d)$ at each pixel p. Finally the depth or disparity of each pixel is computed independently using a winner-takes-all strategy.

In our method, we perform cost aggregation using paths from the eight cardinal and ordinal directions. While computing the aggregated cost, we add a small constant penalty for small depth differences. This is similar to the way single pixel disparities are handled in [Hirschmüller 2008]. For larger depth changes, we add a larger penalty, which is adapted to the gradient magnitude to favor depth discontinuities at strong intensity edges in the image. We modify the final step of the original approach as follows. Whenever possible, we compute two depth estimates per pixel instead of a single depth estimate as in the original approach. For each pixel p, we analyze its 1D distribution of aggregated costs $C_p(d)$ within the full depth range of the plane sweep volume. We compute all local minima of this sampled 1D function. Specifically, we compute a set of depth hypotheses $\{d_i^*\}$ such that $C_p(d_i^*) < C_p(d)$ for a local 1D neighborhood where $d_i^* - w_1 \leq d \leq d_i^* + w_1$. Here, the parameter w_1 controls the size of the depth interval for each local minimum and is specified in terms of pixel disparities. In our experiments, w_1 is set to 2. Sub-pixel refinement is performed for each depth estimate by computing a quadratic fit to the adjacent cost values in the depth interval.

At each pixel, the depth corresponding to the global minimum is selected as the *primary* depth estimate d_p. To select the *secondary* depth estimate d_s, all hypotheses in the set $\{d_i^*\}$ within a depth interval $d_p \pm w_2$ are first discarded. Here w_2 is set to 5. If at most two hypotheses remain, the one with the smaller cost is selected as the secondary depth estimate. However, if more than two depth hypotheses are available, the secondary depth estimate d_s is not computed at such pixels, since all the available hypotheses are likely to be unreliable.

The two-layer depth map computed in this manner contains either one or two depth estimates at pixels with valid depths. Two slices of the matching cost volume are shown for visualization in Figure 3 for one of the images in the GUITARIST sequence, both before as well as after the cost aggregation step. The example in Figure 3(a–d) demonstrates that for strong reflections, two distinct layers exist in the matching cost volume after the aggregation step. This aggregation step is particularly important for recovering the depth of the reflected layer since the gradients in this layer are typically attenuated and the evidence in the original matching cost volume is weaker compared to the transmitted layer. However, in spite of the cost aggregation, the per-pixel depth estimates for the reflected layer are often sparse and noisy. This is addressed in the next stage by generating piecewise planar depth maps for both layers.

5 Computing piecewise-planar 3D proxies

In our system, we use two separate 3D proxies for each image to model the geometry associated with the light coming from the reflected and transmitted scene components. Each of these two layers is represented by a piecewise-planar depth map defined in terms of a set of 3D scene planes and a labeling of each pixel to one of these planes.

Our two-layer proxies are computed as follows. First, a set of 3D planes corresponding to the dominant scene surfaces as well as the virtual reflective surfaces are recovered. Next, a pixel labeling is computed such that each image pixel is assigned to one of the 3D scene planes. This labeling determines the extent of each 3D polygon in the piecewise-planar depth map for each of the two layers. A binary pixel labeling is also computed, which indicates which pixels are reflective and require two layers to model observed reflections and which pixels are opaque, and hence can be modeled with a single layer.

Our approach is related to prior work on piecewise-planar stereo [Furukawa et al. 2009; Sinha et al. 2009; Zebedin et al. 2008; Gallup et al. 2010]. However, unlike these methods, which assume opaque surfaces, our approach explicitly models reflections and computes two-layer depth maps. Reflections were modeled using two-layer disparity maps in [Tsin et al. 2006] using a nested plane-sweep approach and enumerating all pairs of disparities in the disparity range. This is computationally expensive when handling a large disparity range. In contrast, we use a piecewise-planar representation for the proxies and propose a more efficient graph cut based energy minimization formulation by first identifying scene planes and then restricting the labels to a small set of planes and plausible overlapping pairs of planes.

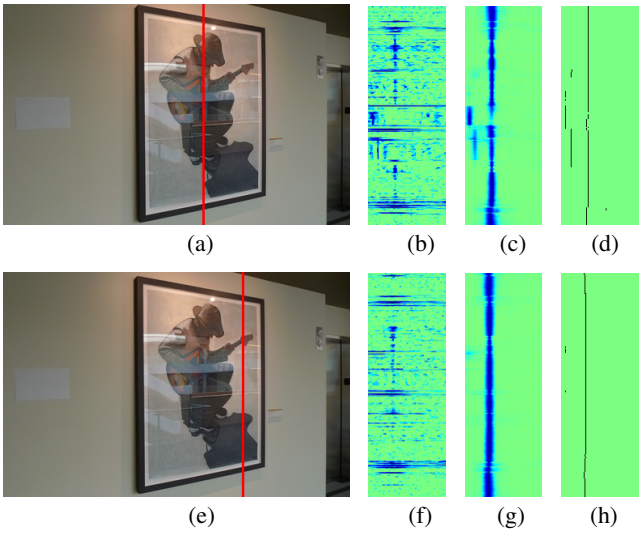
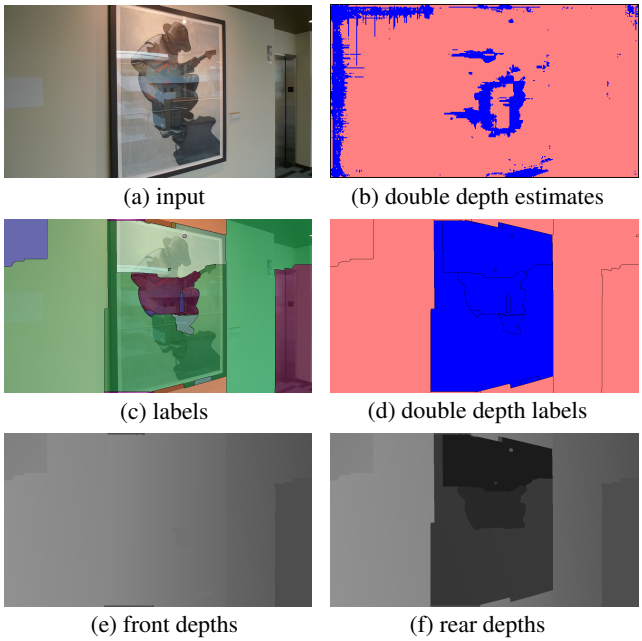


Figure 3: MATCHING COST VOLUME: *(a) For the vertical scanline (in red) in the reference image a vertical slice of the original matching cost volume is visualized in (b). Blue represents low matching costs, disparity increases from left to right. (c) The corresponding slice in the aggregated cost volume after performing SGM. (d) The detected minima. Note how two minima are extracted at pixels where strong reflections are present, giving rise to two depth values at those pixels. (e–h) A different vertical slice of the cost volume without strong reflections. In this case the 1D slice of the depth map is mostly single-layered.*

Plane extraction: We now describe our approach for recovering a set of scene planes for both layers by robustly fitting multiple planes to a subset of the 3D points $\{X_i\}$ obtained from the stereo matching step described in the previous section. All 3D points corresponding to pixels with only primary depth estimates d_p are first selected. For pixels with both primary and secondary depth estimates d_p and d_s respectively, the two corresponding 3D points are selected when $d_p < d_s$; otherwise only the point corresponding to d_p is selected.

A set of planar 3D patches or *surfels* $\{s_j\}$ are now computed by performing local plane-fitting on the selected 3D points. Each surfel s_j is represented using a 3D point X_j^s and a normal vector N_j^s. To compute the surfels, the reference image is divided into bins defined on the 2D pixel grid. The 3D points $\{X_i\}$ are assigned to these bins based on their 2D image projections. At a pixel j, up to two surfels are computed from all 3D points which project within a 7×7 pixel neighborhood of j using sequential RANSAC after which the surfel parameters are re-estimated from the inliers using total least-squares [Weingarten et al. 2004]. Unstable surfels that have fewer than 10 inliers or lie at a grazing angle to the camera ray are pruned. Even though regions such as blank walls may not generate any surfels, hypotheses for such surfaces may still be generated from markings such as labels or posters lying on these walls.



(a) input (b) double depth estimates (c) labels (d) double depth labels (e) front depths (f) rear depths

Figure 4: *(a) An image from the* GUITARIST *sequence. (b) Corresponding binary mask indicates pixels with two depth estimates (in blue). (c) The pixel-labeling computed using graph-cut based energy minimization. (d) The corresponding binary mask indicates image segments having two layers (blue) and a single layer (red). (e – f) The front and rear layer depth maps induced by the pixel-to-plane labeling. The pixel intensity in the visualized depth maps is proportional to inverse depth. Note that the rear depth map is visualized here with the depth of the front layer wherever a single layer was estimated.*

The set of multiple planes $\Pi = \{\pi_i\}$ is now computed using a seed-and-grow approach for robustly clustering the surfels on the basis of coplanarity. Our clustering approach is motivated by prior work in variational shape approximation and mesh simplification [Cohen-Steiner et al. 2004]. The seed-and-grow clustering aims to minimize the total approximation error given by the objective $\sum_{ji} f(s_j, \pi_i)$, where the function $f(s, \pi) = |d - d_\pi|/d$ measures the error incurred by assigning surfel s to plane π. d and d_π denote the depth of the surfel point X^s and the depth of the 3D point, where the ray through the camera's center of projection and X^s intersects the plane π.

We construct a graph where the nodes represent the surfels and edges are present between those pairs of surfel nodes that correspond to pixels within a distance of w pixels from each other in the image (we use $w = 5$). Iterative seed-and-grow clustering is now performed on this graph using the approach proposed by Cohen-Steiner *et al.* [2004]. On convergence, the planes corresponding to the M largest clusters with at least m surfels in each cluster are selected as the representative scene planes. In our implementation, parameters M and m are set to 100 and 32 respectively.

Identifying overlapping planes: The surfel clusters induce a one-to-many partial labeling $\mathcal{L}$ of pixels to planes. We inspect $\mathcal{L}$ for the presence of pixels assigned to pairs of overlapping planes and enumerate all such pairs. This list is further pruned to select pairs of planes for which the pixel count in $\mathcal{L}$ is greater than a threshold (set to 100 in our implementation). The M_1 planes in the set Π and the M_2 overlapping plane pairs denoted as $Q = \{(\pi_u, \pi_v)\}$ where $u, v \in (1 \ldots M_1)$ constitute the set of labels $\{1, \ldots (M_1 + M_2)\}$ used in the subsequent graph cut optimization stage.

Graph cut optimization: Given the set of labels that represents the extracted planes and the restricted subset of plane pairs, we formulate the estimation of the two-layer piecewise-planar depth map as a multi-label MRF optimization problem [Boykov et al. 2001; Sinha et al. 2009]. The pairwise MRF is defined on a graph with the set of pixels $\mathcal{P}$ as nodes and all pairs of adjacent pixels on a 4-connected grid denoted by $\mathcal{N}$ as edges. We compute the labeling L that minimizes the energy

$$E(L) = \sum_{p \in \mathcal{P}} E_p(l_p) + \sum_{(p,q) \in \mathcal{N}} E_{pq}(l_p, l_q). \quad (4)$$

The unary term $E_p(l_p)$ measures the penalty of assigning pixel p to label $l_p \in \{1, \ldots (M_1 + M_2)\}$ based on how well the corresponding plane or planes approximate the observed depths at that pixel. The pairwise term $E_{pq}(l_p, l_q)$ measures the penalty of assigning pixels p and q to labels l_p and l_q respectively. The label l_p represents an individual plane $\pi_{l_p} \in \Pi$ when $1 \leq l_p \leq M_1$ and a pair of overlapping planes $(\pi^1_{l_p}, \pi^2_{l_p}) \in Q$ when $M_1 < l_p \leq (M_1 + M_2)$. We obtain an approximation to the Maximum a Posteriori (MAP) labeling using the α-expansion algorithm [Boykov et al. 2001].

Unary term (E_p): The penalty of assigning a pixel p to a particular label l_p is obtained by computing the sum

$$E_p(l_p) = E^Z_p(l_p) + E^O_p(l_p),$$

where the term E^Z_p measures the geometric error of the piecewise-planar approximation induced by label l_p and the term E^O_p is used to encourage opacity (one layer) and reflectance (two layers) selectively at different pixels in the image. The energy terms for these different cases are defined in Equations (10–13) in Appendix A. The key idea is to encourage a tight fit to the observed data while penalizing reconstructed depth values that do not contribute to a fitted plane.

Pairwise term (E_{pq}): The pairwise term is modeled using a contrast-sensitive Potts model [Boykov et al. 2001]. For neighboring pixels p and q, $E_{pq}(l_p, l_q) = \gamma_0 + \gamma_1 \exp\left(-\gamma_2 |\mathbf{I}_p - \mathbf{I}_q|^2\right)$, when $l_p \neq l_q$, where $\mathbf{I}_p$ and $\mathbf{I}_q$ are the intensities at pixels p and q. The parameter values were set empirically to $\gamma_0 = 1.0$ and $\gamma_2 = 0.001$; γ_1 is set to 10.0 when at least one of the two labels l_p and l_q corresponds to a pair of planes but at the same time the two labels do not share any plane in common, otherwise γ_1 is set to 5.0. A higher penalty is assigned in the former case since a depth discontinuity is unlikely to occur at the same pair of pixels simultaneously in both the front and rear layers of the two-layer depth map.

Reflectivity Field: The labeling L obtained via graph cuts implicitly produces a binary labeling of all pixels, which serves as the estimate of the reflectivity field. For pixels assigned to overlapping planes in L, the corresponding β value is set to 1 to indicate that these pixels are reflective; otherwise, it is set to 0 to represent opaque pixels.

6 Recovering component colors

Once we have computed the depth and reflectivity fields d_0, d_1, and β for each reference image, we can estimate the corresponding layer colors I_0 and I_1 using a weighted constrained least-squares approach similar to the one described by Szeliski *et al.* [2000]. Since d_0, d_1, and β are known and fixed, we can describe the effect of warping the images originally given in (2) using known sparse warping matrices,

$$\boldsymbol{c}_j = \boldsymbol{W}_{j0}\, \boldsymbol{i}_0 + \boldsymbol{W}_{j0}\, \boldsymbol{\beta} \cdot \boldsymbol{W}_{j1}\, \boldsymbol{i}_1, \quad (5)$$

where $\boldsymbol{i}_0$, $\boldsymbol{i}_1$, and $\boldsymbol{\beta}$ are vectorized (raster-order) representations of the layer colors I_0, I_1, and β, the $\boldsymbol{W}_{jl}$ are the sparse warping

matrices corresponding to layer l in input image j, and $\cdot$ represents component-wise multiplication. In practice, we never actually form the sparse $\boldsymbol{W}$ matrices, but use image warping with bilinear resampling for the β images (to avoid ringing) and bicubic resampling for the color layers to transform images between the reference and neighboring views (and vice-versa).

Constrained least squares: The synthesized images $\boldsymbol{c}_j$ are then compared against the input images $\hat{\boldsymbol{c}}_j$, and the values of the unknown layer colors are recovered by minimizing a weighted constrained least-squares problem,

$$\min \sum_j \|\boldsymbol{w}_j \cdot (\boldsymbol{c}_j - \hat{\boldsymbol{c}}_j)\|^2 \quad \text{s.t.} \quad 0 \leq \boldsymbol{i}_l \leq 1. \tag{6}$$

The per-pixel weighting terms $\boldsymbol{w}_j$ are normally 1 but are set to 0 whenever an input pixel (band) is saturated and the corresponding synthesized pixel value is at least as large.[4] In our current implementation, each reference image is matched against its ± 2 nearest neighbors.

Gamma curves: Note that the layer color values $\boldsymbol{i}_l$ are restricted to lie between 0 (black) and 1 (white).[5] The source images we use, which are downsampled versions of regular JPEG images, represent colors in a gamma-mapped sRGB space, so we undo this gamma correction to obtain a linear (calibrated) color space when we read in the images, and we apply the inverse correction when we write out the per-layer component colors. Our image-based renderer performs a similar transformation in order to faithfully model the additive mixing of pixel irradiances. To compensate for auto white balance (AWB), auto gain control (AGC), and exposure differences, we also compute per-color-channel histograms of the residual errors $\boldsymbol{c}_j - \hat{\boldsymbol{c}}_j$ vs. the original color values $\hat{\boldsymbol{c}}_j$, fit a parabolic curve with 0 values at the $[0, 1]$extrema to these errors, and add this offset to the residual as part of the optimization criterion.

Initialization: To initialize the constrained least-squares optimization, we first compute a min-composite of the input images aligned to the reference frame, compute the difference between single-layer synthesized images and the input, and then compute a max-composite of the resulting difference images—please see [Szeliski et al. 2000] for a more detailed description of this process and why it produces sensible initial solutions. Note that this initialization does not affect the final solution value, since we have a convex optimization problem, but simply speeds up the computations.

Conjugate gradient descent: In order to exploit the special characteristics of our optimization problem, we use a constrained nonlinear conjugate gradient descent algorithm.[6] At each step, we synthesize the images $\boldsymbol{c}_j$ by warping and compositing the current layer estimates using (5) and then compute the difference (residual error) images. We then weight these errors by the saturated pixel weights $\boldsymbol{w}_j$ (which can vary from iteration to iteration), multiply the layer 1 residual by the warped β field, warp the residuals back to the reference image, and add them up to produce the current gradient estimate. Gradient components at pixels that are already on the $[0, 1]$ boundary are set to 0 if they point outward.

Next, the gradient vector is conjugated with the previous search direction using the Polak-Ribiere formula [Nocedal and Wright 2006] to produce a new descent direction. We then perform a line search with local quadratic fits to the energy to determine the step size. During this process, whenever the update produces color values outside the $0 \leq \boldsymbol{i}_l \leq 1$ range, they are clipped. More detailed descriptions of (nonlinear) conjugate gradients can be found in [Nocedal and Wright 2006]. The resulting algorithm produces reasonable results, but is sensitive to noise and small misregistrations between the images.

Regularization: To compensate for misalignments and noise sensitivity, we add a regularization term. The quadratic objective function (6) is augmented with two regularizing terms

$$\lambda_1(\|\boldsymbol{D}\,\boldsymbol{i}_0\|^p + \|\boldsymbol{D}\,\boldsymbol{i}_1\|^p) + \lambda_0\|\boldsymbol{i}_1\|^p, \tag{7}$$

where $\boldsymbol{D}$ indicates the horizontal and vertical derivatives between adjacent color pixels, $1 < p \leq 2$ is the regularization exponent, and λ_1 and λ_0 are the regularization parameters that control the desired smoothness and a bias towards 0 (black) pixels in layer 1, respectively. We experimented with both quadratic $p = 2$ and non-quadratic $p \in [1.2, 1.5]$ regularization, and found that while the latter produces slightly sharper results, the former converges much more quickly, so that is what we use. The regularization parameters were tuned to $\lambda_0 = 0.001$ and $\lambda_1 = 0.5$.

Matching the reference image: In our image-based renderer, we would like the synthesized image $\boldsymbol{c}_i$ to match the input reference image $\hat{\boldsymbol{c}}_i$, since then we can display the original artifact-free image when an image-to-image transition terminates [Zitnick et al. 2004; Snavely et al. 2006]. In order to do this, we need to make (6) be exact whenever $j = i$. We can do this by splitting any accumulated error between the two layers (unless one of them is already at the feasible $[0, 1]$ boundary). To make the gradient descent work better, we also modify the computed gradient so that the sum of the two layer gradients at each pixel (in non-saturated regions) is zero, i.e., we subtract any average gradient component between the two layers.

Downweighting strong edges: Small misalignments between images due to inaccuracies in the 3D modeling, e.g., deviations from planarity, can lead to "echoes" in the reconstructed image at sharp edges. In order to compenstate for this, we set the weighting function $\boldsymbol{w}_j$ in (6) to $g_{\min}^2/(g_{\min}^2 + g^2)$, where g is the largest absolute finite difference (gradient) between a pixel and its four neighbors and $g_{\min}$ is set to 5 gray levels. To further mitigate the effects of misalignments, we also low-pass filter the residual errors $\boldsymbol{c}_j - \hat{\boldsymbol{c}}_j$ in (6) with a Gaussian blur of $\sigma = 1$.

Figures 1 and 6 show the results of running our layer separation algorithm on six different data sets. As you can see, our approach does a good job of separating the reflected from the transmitted light, although, as also observed by Szeliski *et al.* [2000], it sometimes has trouble correctly assigning low-frequency components to the appropriate layer. It also occasionally produces other artifacts, as discussed in Section 8 below.

7 Image-based rendering with reflections

Our representation supports a simple image-based rendering technique that we implemented on standard graphics hardware. Any IBR algorithm is most successful when the novel views are close to the input camera poses. Since most of our scenes are object-centric linear sequences, we first describe an algorithm for fitting a smooth approximating path to the estimated camera locations, which we use to navigate around the scene in both our videos and in the interactive viewer that we built.

Assuming that there is a central object in our scene and that the input cameras are facing this object, we can estimate its location as the point in space where the camera optical axes intersect. Since

[4] The resulting problem is still convex.

[5] It is not strictly necessary to constrain $\boldsymbol{i}_l \leq 1$, but this makes the problem well-posed at saturated pixels.

[6] We use nonlinear conjugate gradient descent because the (sub-) gradients differ from iteration to iteration, depending on which pixels are on the $[0, 1]$ boundary or saturated, and which ones are in the interior (quadratic) region.

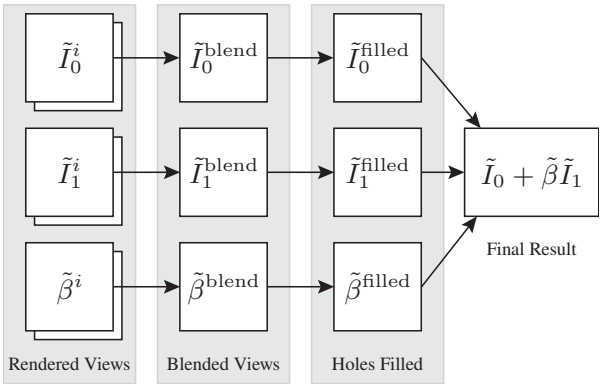


Figure 5: *Block diagram of our rendering algorithm. Please refer to the text for the notation.*

our datasets are captured in a casual way, the camera axes typically do not intersect in a single point. Hence, we estimate the object location as the point that is closest to all camera axes by solving a small linear least-squares system.

Next, we fit a least-squares plane to all camera locations and use this plane to embed the camera path. We project all the camera locations and the object center onto that plane, and then compute a smooth arc around the object that passes near the camera locations. The novel views in our supplementary videos are generated by moving the camera along this arc while always facing the estimated object center. In our interactive viewer the user moves the camera along this path by dragging the mouse. When the user releases the mouse, we allow the camera to drift towards the nearest reference view, where the rendering looks perfect by design. This camera behavior feels intuitive and enhances realism in an interactive session, as can be seen in the accompanying video.

Given the parameters of a novel view, we render the scene by blending the I_0, I_1, and β estimates from multiple input cameras. First, we select the set of cameras we want to use and their relative contributions. Since we are working with simple linear sequences, we simply project the input camera locations onto the path and use the closest camera on either side of the novel camera, i.e., we blend the closest left and right cameras along the path. The blending weights are given by their relative distance to the novel camera. Note that while we blend only two cameras to produce the results for our paper, our algorithm is capable of rendering from more than two views and is formulated below in a more general setting. Buehler *et al.* [2001] describe selection and blending heuristics for more general, unstructured scenes.

We index the set of cameras we are rendering from with an index i. Let $\tilde{I}_0^i, \tilde{I}_1^i, \tilde{\beta}^i$ denote the warped components of camera i, rendered as seen from the novel view. Note that not every pixel in these renderings is defined, since there might be visible gaps. Let $\alpha^i \in \{0, 1\}$ indicate whether a pixel has a contribution from camera i or not and w^i be the relative weights of the cameras. We now blend the cameras using

$$\tilde{I}_0^{\text{blend}} = \frac{\sum_i w^i \alpha^i \tilde{I}_0^i(x)}{\sum_i w^i \alpha^i} \tag{8}$$

and equivalent equations for $\tilde{I}_1^{\text{blend}}$ and $\tilde{\beta}^{\text{blend}}$. In other words, if a pixel is seen by multiple cameras, we blend their relative contributions. If a pixel is seen by only one camera, it takes that value. Pixels that are not seen by any camera remain undefined. We combine the α^i maps using

$$\alpha^{\text{blend}} = \max_i \alpha^i. \tag{9}$$

Name	#Images	Resolution	T_s	T_d	T_p	T_l	Timings
GUITARIST	36	1024 × 576	0:02	0:04	1:00	0:32	1:38 (0:03)
GLASS CASE	41	1024 × 576	0:05	0:16	1:10	0:31	2:02 (0:03)
STATUE	39	1024 × 576	0:05	0:10	1:08	0:30	1:54 (0:03)
GALLERY	76	1024 × 768	0:08	0:22	2:20	0:57	3:47 (0:03)
TABLE	20	576 × 1024	0:02	0:10	0:23	0:16	0:51 (0:03)
CAR	23	640 × 480	0:01	0:03	0:46	0:14	1:04 (0:03)

Table 1: DATASETS/TIMINGS: *For each sequence, the image resolution, the image count and details of the running times of our approach on a single CPU core (hour:minutes) are listed. Columns T_s, T_d, T_p and T_l list the timings for the various stages; namely, the SfM, stereo matching, 3D proxy computation and layer separation stages respectively. The last column lists the total running times and the average per image timing is specified in brackets.*

To generate nicer looking visuals, we fill any remaining holes due to disocclusions, i.e., pixels where α^{blend} takes a zero value, using a smooth membrane, implemented using a strategy similar to the *pull-push* algorithm described by Gortler *et al.* [1996].

In the pull phase, we compute a full RGBα Laplacian pyramid for the $(\tilde{I}_0^{\text{blend}}, \alpha^{\text{blend}})$ and corresponding $\tilde{I}_1^{\text{blend}}$ and $\tilde{\beta}^{\text{blend}}$ RGBα pairs. Here α can take on fractional values, except at the finest level of the pyramid. In the push phase, we work our way down the pyramid, starting at the coarsest, single-pixel level, for which there are no $\alpha = 0$ pixels. We first divide the current RGBα values by α to get full opacity un-multiplied colors. At finer levels, we upsample the coarser parent level and add in the signed Laplacian details, and then divide again by α to get full opacity values.

Once we have obtained the filled-in maps $\tilde{I}_0^{\text{filled}}$, $\tilde{I}_1^{\text{filled}}$, and $\tilde{\beta}^{\text{filled}}$, we use Equation (1) to compute the final result.

8 Experimental results

In order to test the validity of our image-based modeling and rendering approach, we collected a variety of image sequences using regular hand-held photography. Since our system does better when the images have constant exposure and color balance, we either locked the exposure on the camera, or chose subsets of images where the exposure was fairly constant. The sequences were mostly taken while moving horizontally and keeping the camera roughly fixated on a reflective object of interest.

Figures 1 and 6 show the results of running our layer separation algorithm on six different data sets. Each row shows one reference frame, along with the two recovered layer colors and the reflectivity map. Due to lack of space, we mostly do not show the depth maps and other intermediate results corresponding to these images. Figure 4 shows the intermediate results (raw depth values and final plane, reflectivity, and depth labellings) for the GUITARIST sequence. The intermediate results for the other sequences are given in the supplemental materials, which can be found on the conference DVD and our project Web page.

In the supplemental materials we also compare our layer color separation results against the simpler algorithms used in [Szeliski et al. 2000; Tsin et al. 2006]. Overall, the new algorithm, which has additional terms penalizing extra light in the secondary reflection layer and terms to deal with small misalignments, attributes less stray light to the reflection layer.

Table 1 shows the timings for each of our stages on all six sequences. The average running time per image was under 3 minutes when processing them at a resolution of 1024 × 576 pixels on a single core 3GHz CPU on a PC with 24GB RAM. Both the proxy computation and layer separation stages, which dominate the

running time, are easy to parallelize on multi-core platforms as the per-image proxy representation in our method allows multiple images to be processed simultaneously.

To compare with [Tsin et al. 2006], we modified the two-layer piecewise planar proxy computation stage in our method. Instead of using the proposed method, we implemented a method that considers the same number of paired depth hypotheses as [Tsin et al. 2006] and has similar computational complexity to their method. Specifically, we used a set of fronto-parallel planes that induced uniform disparity steps in the image, and used all the nested pair of planes as labels in the graph cut optimization. When the disparity range is manually set to 30 pixels for images at 1024×536 resolution, this generated 435 nested plane pairs or labels. Note that this is a conservative estimate of the disparity range; in practice it is often as high as 70-80 pixels, since it is determined automatically from feature correspondences. For the GUITARIST sequence, this graph cut optimization with 435 paired hypotheses took 7 min 52 seconds per image on average. Two such graph cut problems must be solved in [Tsin et al. 2006], once for the front layer and once for the rear layer. In comparison, our graph cut step took 70 seconds with 45 labels (32 plane hypotheses and 13 plane pairs) on average. With the additional overhead for estimating candidate planes, our method is overall about six times faster.

Looking at the results in Figures 1 and 6, we see that our approach mostly does a good job of separating the reflected from the transmitted light. However, as noted in [Szeliski et al. 2000], our system sometimes has trouble correctly assigning low-frequency components to the appropriate layer. We can also occasionally notice faint high-frequency "echoes" of strong edges. These are due to small misalignments between the images mostly due to the approximations induced by our piecewise-planar 3D proxies. While the edge downweighting and residual low-pass filtering terms introduced in Section 6 help mitigate these effects, some echoes can still be seen, particularly in sequences such as STATUE, where the piecewise-planar nature of our proxies fails to capture the detailed shape of the face of the statue.

Additional artifacts that can be seen by viewing the image-based rendering video sequences include incorrect geometry, due to failures and frame-to-frame inconsistencies in the stereo matching pipeline, and incorrect reflectivity estimates. Some of these artifacts are most visible in the CAR sequence, where overestimating two-layer regions around the occlusion boundary of the car and failures of the stereo in the strongly slanted regions of the fenders leads to ghosting. Improving these estimates using global 3D modeling and by re-estimating the β component during our least-squares appearance recovery stage (6) is a promising area for future research.

9 Conclusions

In this paper, we have developed a system for estimating two-layer image-based models for rendering scenes with reflections and gloss. The use of two 3D proxies per image, one for modeling transmitted light and the other for modeling reflected light, greatly increases the realism of image-based rendering in such scenes. Our system consists of a transparent (multiple-depth) stereo algorithm, a two-layer piecewise-planar depth labeling component, a regularized color separation algorithm that models effects such as saturation and slight misregistrations, and an interactive hardware-accelerated image-based rendering viewer. Because we use two regular images for each input reference view along with compact piecewise-planar proxies, we can efficiently represent and transmit the information required for our image-based renderer. Our experimental results demonstrate a notable improvement in rendering quality over traditional single-proxy systems.

In future work, we plan to improve the quality of the reflectivity estimate β by jointly estimating it as part of our least-squares appearance recovery stage (6). We also plan to improve our stereo matching components by using the results of per-frame depth estimation to reinforce and correct each other, and to develop a more detailed representation of surface shape, which should help reduce artifacts due to misalignments.

In even longer-term research, we would like to model small local variations in reflector geometry, e.g., the sag and bowing in glass that occurs at window edges. We would also like to model spatially varying reflectivity, which occurs due to the Fresnel effect and is most pronounced when looking at reflections in water and through windows at varying incidence angles. These extensions will widen the range of reflective and glossy scenes that can be successfully modeled and further increase realism in image-based rendering.

References

BEERY, E., AND YEREDOR, A. 2008. Blind separation of superimposed shifted images using parameterized joint diagonalization. *IEEE Transactions on Image Processing 17*, 3 (March), 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 (September), 886–896.

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

BHAT, P., ZITNICK, C. L., SNAVELY, N., AGARWALA, A., AGRAWALA, M., COHEN, M., CURLESS, B., AND KANG, S. B. 2007. Using photographs to enhance videos of a static scene. In *Eurographics Symposium on Rendering*, 327–338.

BOYKOV, Y., VEKSLER, O., AND ZABIH, R. 2001. Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence 23*, 11 (November), 1222–1239.

BUEHLER, C., BOSSE, M., MCMILLAN, L., GORTLER, S. J., AND COHEN, M. F. 2001. Unstructured Lumigraph rendering. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, ACM, SIGGRAPH '01, 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.

CHEN, S., AND WILLIAMS, L. 1993. View interpolation for image synthesis. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '93, 279–288.

COHEN-STEINER, D., ALLIEZ, P., AND DESBRUN, M. 2004. Variational shape approximation. *ACM Trans. Graph. 23* (August), 905–914.

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 (January), 51–85.

DARRELL, T., AND PENTLAND, A. 1995. Cooperative robust estimation using layers of support. *IEEE Transactions on Pattern Analysis and Machine Intelligence 17*, 5 (May), 474–487.

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

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

FURUKAWA, Y., CURLESS, B., SEITZ, S. M., AND SZELISKI, R. 2009. Manhattan-world stereo. In *Computer Vision and Pattern Recognition (CVPR 2009)*, 1422–1429.

GALLUP, D., FRAHM, J.-M., AND POLLEFEYS, M. 2010. Piecewise planar and non-planar stereo for urban scene reconstruction. In *Computer Vision and Pattern Recognition (CVPR'10)*, 1418–1425.

GORTLER, S. J., GRZESZCZUK, R., SZELISKI, R., AND COHEN, M. F. 1996. The Lumigraph. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, 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 (February), 328–341.

IRANI, M., ROUSSO, B., AND PELEG, S. 1994. Computing occluding and transparent motions. *International Journal of Computer Vision 12*, 1 (January), 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. In *Computer Vision and Pattern Recognition (CVPR'96)*, 307–314.

LEVIN, A., ZOMET, A., AND WEISS, Y. 2002. Learning to perceive transparency from the statistics of natural scenes. In *(NIPS)*, MIT Press, 1247–1254.

LEVOY, M., AND HANRAHAN, P. 1996. Light field rendering. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, ACM, SIGGRAPH '96, 31–42.

NOCEDAL, J., AND WRIGHT, S. J. 2006. *Numerical Optimization*, second ed. Springer, New York.

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

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

SHADE, J., GORTLER, S., HE, L., AND SZELISKI, R. 1998. Layered depth images. In *ACM SIGGRAPH 1998 Conference Proceedings*, 231–242.

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

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

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 (August), 835–846.

SZELISKI, R., AVIDAN, S., AND ANANDAN, P. 2000. Layer extraction from multiple images containing reflections and transparency. In *Computer Vision and Pattern Recognition (CVPR'2000)*, vol. 1, 246–253.

SZELISKI, R. 2010. *Computer Vision: Algorithms and Applications*. Springer, New York.

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

WEINGARTEN, J. W., GRUENER, G., AND DORF, A. 2004. Probabilistic plane fitting in 3D and an application to robotic mapping. In *International Conference on Robotics and Automation (ICRA)*, 927–932.

ZEBEDIN, L., BAUER, J., KARNER, K. F., AND BISCHOF, H. 2008. Fusion of feature- and area-based information for urban buildings modeling from aerial imagery. In *European Conference on Computer Vision (ECCV'08)*, 873–886.

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 (August), 600–608.

A Formulas for unary matching terms

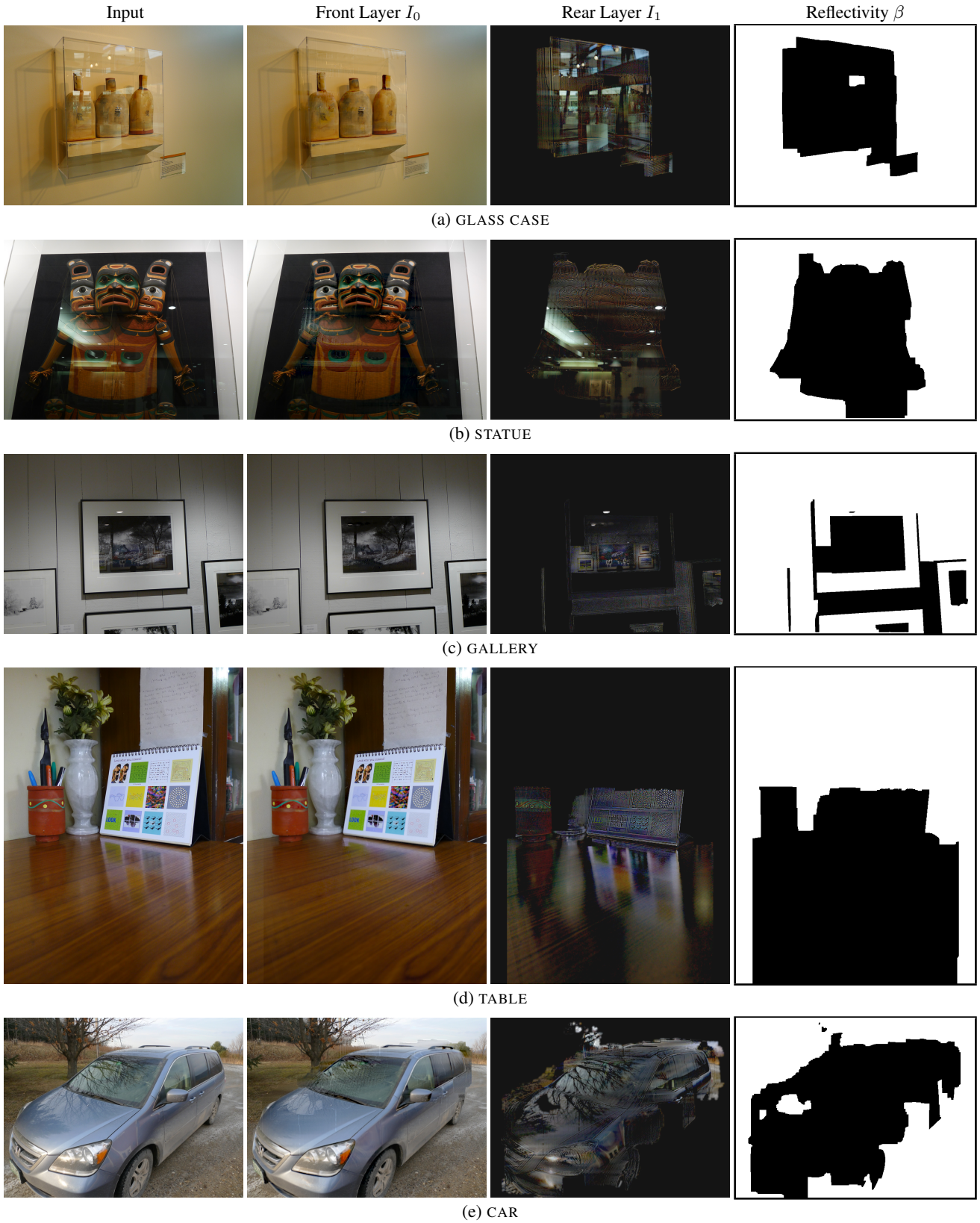
In this appendix, we give the detailed formulas for the energy terms introduced in Section 5. Here, z_1 and z_2 denote the two depths at pixel p. Similarly, y_1 and y_2 denote the depths induced by planes $\pi^1_{l_p}$ and $\pi^2_{l_p}$ at p. Depending on whether one or two depths are available and whether l_p corresponds to a single plane or two planes, there are four cases to be considered when defining the geometric term E^Z_p and opacity term E^O_p that encourages a single layer:

- 1 sample, 1 plane:
$$E^Z_p = g(z_1, y_1), \quad E^O_p = 0 \tag{10}$$
- 2 samples, 1 plane:
$$E^Z_p = \min(g(z_1, y_1), g(z_2, y_1)), \quad E^O_p = K_1 \tag{11}$$
- 1 sample, 2 planes:
$$E^Z_p = \min(g(z_1, y_1), g(z_1, y_2)), \quad E^O_p = 1 - g(y_1, y_2) \tag{12}$$
- 2 samples, 2 planes:
$$E^Z_p = \min(g(z_1, y_1)+g(z_2, y_2), g(z_1, y_2)+g(z_2, y_1)),$$
$$E^O_p = \begin{cases} 1 - g(y_1, y_2) & \text{if } \mathcal{C}(p) > T_2 \\ 1 - g(y_1, y_2) + K_2 & \text{if } \mathcal{C}(p) \le T_2 \end{cases} \tag{13}$$

Parameter K_1 in (11) is a positive constant that penalizes choosing a single layer at pixels where stereo matching produced two depth estimates, while (12) penalizes assigning two planes with very similar induced depths y_1 and y_2 to a pixel. In (13), K_2 is another positive constant that penalizes assigning two depths to pixels which have low contrast in the image, i.e., where the intensity gradient magnitude measured by the function $\mathcal{C}(p)$ is smaller than a threshold T_2 (set to 0.03 for color values in the interval $(0, 1)$). Both constants K_1 and K_2 are set to 1.0 in our implementation. The function $g(z, y) = h(|y - z|/z, T_1)$ measures the approximation error between the depth z and the plane-induced depth y, where

$$h(x, T_1) = \begin{cases} 0 & \text{if } x \le 0, \\ 1 & \text{if } x \ge T_1 \\ (1 - (1 - (x/T_1))^2)^3 & \text{if } 0 \le x \le T_1 \end{cases} \tag{14}$$

and the robustness parameter T_1 is set to 0.1 in all our experiments.



(a) GLASS CASE

(b) STATUE

(c) GALLERY

(d) TABLE

(e) CAR

Figure 6: *Experimental results for the (a)* GLASS CASE*, (b)* STATUE*, (c)* GALLERY*, (d)* TABLE*, and (e)* CAR *image sequences. From left to right we see a sample input image, its front and rear layer decompositions I_0 and I_1, and the reflectivity map β. Please see the text for a discussion of these results, zoom in to get a closer look at the image details, and see the video to see the resulting image-based renderings.*