Image and Video-Based Modeling and Rendering

Richard Szeliski

Vision-Based Modeling Group
Microsoft Research
e-mail: szeliski@microsoft.com
http://www.research.microsoft.com/~szeliski/

Abstract

Obtaining photo-realistic geometric and photometric models is an important component of image-based rendering systems that use real-world imagery as their input. Applications of such systems include novel view generation and the mixing of live imagery with synthetic computer graphics. This paper reviews a number of image-based representations (and their associated reconstruction algorithms) we have developed in the last few years. It begins by reviewing some recent approaches to the classic problem of recovering a depth map from two or more images. It then describes some of our newer representations and reconstruction algorithms, including volumetric representations, layered plane-plus-parallax representations (including the recovery of transparent and reflected layers), and multiple depth maps. The paper also presents our work in video-based rendering, in which we synthesize novel video from short sample clips by discovering their (quasi-repetive) temporal structure.

1 Introduction

Image-based rendering is one of the most exciting new research areas in computer graphics. Instead of rendering objects or scenes using a traditional polygon-based approach, IBR uses large amounts of imagery (photographic or synthetic) to achieve a high degree of realism and efficient rendering speeds.

Examples of image-based rendering systems include panoramic image mosaics [6, 26, 22], Concentric Mosaics [21], view interpolation and view morphing [5, 18], lightfields and Lumigraphs [14, 10], and Layered Depth Images and Sprites with Depth [20]. Central to each of these systems is the idea of taking two or more images and generating novel views using suitable combinations of image warping and blending.

To make this process more effective, i.e., to get by with fewer sample images for a given level of quality, it is often imperative to either establish accurate correspondences between the various input images, or to build accurate full 3-dimensional models. This aspect of image-based rendering is often call *image-based modeling*. Fortunately, researchers in

computer vision have been developing such *3-D reconstruction* techniques for decades. The advent of image-based rendering gives new impetus to this field. However, when stereo matching is used in such applications, there are several demanding requirements that are not present in more traditional robotics applications.

First of all, the stereo algorithm must be able to assign correct (or at least reasonable) depths at *all* pixels, especially those near depth discontinuities. In Section 2, I discuss how some recent stereo algorithms are able to avoid the systematic "fattening" of layers associated with traditional area-based methods. A second requirement is the ability to pull mattes, i.e., to separate foreground and background elements while correctly describing the true colors of individual pixels. A third requirement is to generate novel views with as few gaps (missing pixels) as possible, and to also account for partially occluded regions during the matching process. The single depth map representation used in Section 2 is inadequate on all of these counts.

Section 3 describes how a *volumetric* representation of space, combined with real-valued opacities, can be used to overcome most of these problems. Section 4 describes a different, more compact, representation based on arrangements of colored quasi-planar cutouts, which can also overcome these problems. Section 5 describes how to use multiple depth maps (and associated images) to solve the problem of partially occluded areas, and how this representation can also serve as a preliminary step towards a more complete reconstruction of the scene.

While the approaches described above all require a static scene (so that 3-D correspondences can be established), a generalization of image-based rendering to temporally varying scenes is also possible. In Section 6, I present our first steps in *video-based rendering*, which enable us to take short-duration video clips and turn them into infinitely repeating *video textures*. I then conclude the paper with a comparison of the approaches presented and some prospects for further research.

2 Depth maps

The classical problem of computing a dense depth map from two or more images has been extensively studied [8]. In this section, we first present a formulation for this problem, and then discuss several recently developed algorithm that attempt to accurately solve for depth near discontinuities.

2.1 Generalized disparity space

Assume we are given as input a collection of K images, $I_1(x, y), I_2(x, y), \ldots, I_K(x, y)$, captured by K cameras with known projection (camera) matrices, $\mathbf{P}_1, \mathbf{P}_2, \ldots, \mathbf{P}_K$. To formulate the multiframe stereo problem, we use a *generalized disparity space*, which can be any projective sampling (collineation) of 3-D space [7, 25]. This space is a generalization of the notion of *disparity space* [11, 16], i.e., the enumeration of all possible disparities at every pixel. The goal of stereo matching is then to find the elements in disparity space which lie on the surfaces of the objects in the scene. The benefits of such an approach include the equal and efficient treatment of a large number of images [7] and the possibility of modeling occlusions [11].

To formulate the generalized disparity space, we first choose a *virtual camera* position and orientation. This virtual camera may be coincident with one of the input images, or it can be chosen based on the application demands and the desired accuracy of the results. For instance, if we wish to regularly sample a volume of 3-D space, we can make the camera orthographic, with the camera's (x, y, d) axes being orthogonal and evenly sampled (as in [19]).

Having chosen a virtual camera position, we then choose the orientation and spacing of the *disparity planes*, i.e., the constant d planes. The relationship between d and 3-D space can be projective. For example, we can choose d to be inversely proportional to depth, which is the usual meaning of disparity. The information about the virtual camera's position and disparity plane orientation and spacing can be captured in a single 4×4 matrix $\hat{\mathbf{M}}$, which represents a collineation of 3-D space, $w(x\ y\ d\ 1)^T = \hat{\mathbf{M}}(X\ Y\ Z\ 1)^T$.

2.2 Area-based approaches

Having presented the representation used for describing the output of the matching algorithm, we can now state its goal: For each (x, y) location in disparity space, find the disparity d that aligns corresponding locations in the input images (ignoring, for now, the possibility that pixels may be occluded). In traditional area-based correlation, the quality of a match is measured by comparing windows centered at corresponding locations, for example, using the sum of squared intensity differences (SSD) [12].

A more general way of characterizing area-based algorithms is the following [16]:

1. For each disparity under consideration, compute a per-pixel matching cost, e.g., squared intensity difference or variance of colors across the k input images.

2. Aggregate support spatially (e.g., by summing over a window, or by diffusion).

3. At each pixel (x, y), find the best matching disparity d based on the aggregated support.

4. Compute a sub-pixel disparity estimate.

Let us look at the components in this framework in more detail.

At the base of any matching algorithm is a matching cost that measures the similarity of corresponding locations. Matching costs can be defined locally (at pixel level), or over a certain area of support. Examples of local costs are absolute intensity differences, squared intensity differences, binary pixel matches, edges, filtered images, and measures based on gradient direction or gradient vectors. Matching costs that are defined over a certain area of support include correlation and non-parametric measures. These can be viewed as a combination of the matching cost and aggregation stages. More than two images are used in multiframe stereo to increase stability of the algorithm [15].

Aggregating support is necessary for stable matching. A support region can either be two-dimensional at a fixed disparity (favoring fronto-parallel surfaces), or three-dimensional in *x-y-d* space (supporting slanted surfaces). Two-dimensional evidence aggregation has been done using square windows (traditional), Gaussian convolution, multiple windows anchored at different points [11], and windows with adaptive sizes [12].

Sub-pixel disparity estimates can be computed by fitting a curve to the matching costs at the discrete disparity levels [12]. This provides an easy way to increase the resolution of a stereo algorithm with little additional computation. However, to work well, the intensities being matched must vary smoothly.

2.3 Global optimization approaches

Optimization (regularization) approaches start with the same computation of matching costs as area-based techniques, but then add a controlled smoothness penalty (prior) on the disparity field $d(x, y)$. A variety of optimization algorithms can then be used to find a good solution to this problem [16, 3].

An elegant mathematical approach to formulating these energy function and finding their minimum is to use a Bayesian (probabilistic) estimation framework. The Bayesian model of stereo image formation consists of two parts. The first part, a *prior model* for the disparity surface, uses a traditional Markov Random Field (MRF) to encode preferences for smooth surfaces [9]. The second part of a Bayesian model is the *data* or *measurement model* which accounts for differences in intensities between corresponding image locations. The posterior distribution, $p(\vec{d}|I_1, \ldots, I_K)$ can be derived from the prior and measurement

models using Bayes' rule,

$$p(\vec{d}|I_1, \ldots, I_K) \propto p_P(\vec{d}) p_M(I_1, \ldots, I_K|\vec{d}). \quad (1)$$

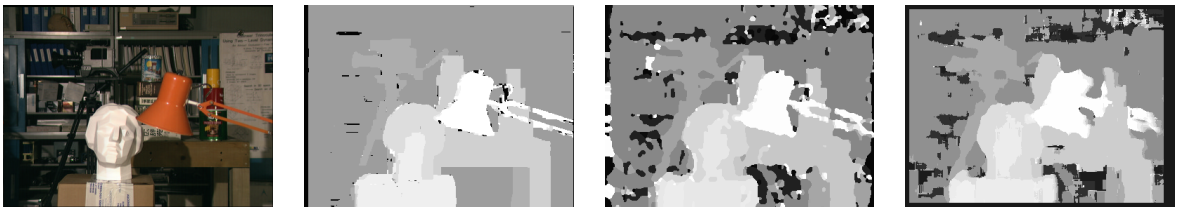
A variety of techniques have been developed for maximizing the posterior distribution. Two of the most popular are the Gibbs Sampler [9] and mean field theory. The Gibbs Sampler randomly chooses values for each $d_{i,j}$ site according to the local distribution determined by the current guesses for a site's neighbors [9]. This process will in theory converge to a statistically optimal sample, given enough time. Mean field theory updates an estimate of the *mean* value of $d_{i,j}$ at each site using a deterministic update rule derived from the original probability distribution [9].

The Gibbs Sampler and its variants can produce good solutions, but at the cost of long computation times. Mean field techniques, on the other hand, are not very good at modeling ambiguous estimates, such as multiple potential matches at each pixel. Instead of using either of these two traditional approaches, we developed a novel estimation algorithm based on modeling the probability distribution of $d(x, y)$ at each site [16] . To do this, we associate a scalar value between 0 and 1 with each possible discrete value of d at each pixel (x, y), and require that the probabilities sum up to 1. This representation is therefore the same as that used by aggregation-based algorithms, i.e., it explicitly models all possible disparities at each pixel, rather than modeling a single estimated disparity as in traditional Gibbs Sampler or mean-field approaches.

The algorithm is initialized by calculating the probability distribution for each pixel (x, y) based on the intensity errors between matching pixels, i.e., using the measurement model. To derive the update formula, we approximate the true Markov Random Field distribution with a *factored* approximation, i.e., we assume that the neighboring disparity columns have independent distributions. Minimizing the Kullback-Leibler divergence between the true posterior Gibbs distribution and its factored (mean-field) approximation leads to a set of update formulas on the probability distributions that use non-linear *diffusion* (see [16] for details).

The results of running this algorithm on difficult stereo pairs are quite promising. The algorithm is particularly good at correctly matching pixels near depth discontinuities, since the robust smoothness constraint can be violated at the appropriate places, and also at stereograms that have a lot of potential matches, such as random-dot stereograms.

Another recent development in optimization-based stereo matching is the use of graph algorithms [3]. Here, techniques from discrete optimization are used to find good minima (in some cases, even global minima) of the global energy function. These algorithms have both good discontinuity localization, since they are based on robust smoothness models,



Input image | Graph cuts | Bayesian diffusion | SAD (13×13)

Figure 1: Results for several stereo algorithms on the University of Tsukuba imagery.

and also excel at filling in good disparities inside uniform color/intensity regions (since they take large steps in state space).

Figure 1 shows the results of applying three different matching algorithms on a multi-camera stereo data set provided by the University of Tsukuba. You can readily see that the two energy minimization-based approaches (Graph cuts and Bayesian) have much crisper depth discontinuities, compared with the sum of absolute differences (SAD) technique. The graph-cut approach also does an excellent job of filling in uniform intensity areas.

3 Volumetric representations

The depth map representation presented in the previous section is unable to represent and hence render partially occluded background regions. This is due to our insistence (enforced during the winner-take-all stage) that only a single depth value be assigned to each pixel in the reference image.

What if we were to relax this assumption? What if in addition to being able to have several depth along each ray in the reference image, we also represented the colors of these pixels and their (potentially partial) opacities? In principle, we should be able to represent and reason about partially occluded pixels, and to correctly estimate the color values of mixed pixels. These are the intuitions that led to the development of the volumetric stereo reconstruction algorithm presented in [25]. (Simultaneously with our work, Seitz and Dyer [19] developed a volumetric stereo algorithm that uses binary (filled/empty) opacities and a front-to-back plane sweep (*voxel coloring*) algorithm.)

The algorithm starts by performing the same matching cost computation, aggregation, and winning depth value selection as described in the previous section. However, instead of insisting that every pixel in the reference image pick a winning depth, we only select depth values that have a good match (good aggregated evidence), using a threshold to mark other pixels as currently "unassigned". These pixels will typically not have correspondences

either because they are partially occluded, or because they are mixed pixels with different backgrounds in different images. A new (x, y, d) volume can now be created, where each cell now contains a color value, initially set to the mean color computed in the first stage, and the opacity is set to 1 for cells which are winners, and 0 otherwise.

Once we have an initial (x, y, d) volume containing estimated RGBA (color and 0/1 opacity) values, we can re-project this volume into each of the input cameras. In our approach, we interpret the (x, y, d) volume as a set of (potentially) transparent acetates stacked at different d levels. Each acetate is first warped into a given input camera's frame using the known homography $\mathbf{H}_k$. Once the layers have been resampled, they are then composited using the standard *over* operator.

After the re-projection step, we refine the disparity estimates by preventing visible surface pixels from voting for potential disparities in the regions they occlude. More precisely, we build an (x, y, d, k) *visibility map*, which indicates whether a given camera k can see a voxel at location (x, y, d) [25].

Once we have computed the visibility volumes for each input camera, we can update the list of color samples we originally used to get our initial disparity estimates to obtain a distribution of colors in (x, y, d, k) where each color has an associated visibility value. Voxels that are occluded by surfaces lying in front in a given view k will now have fewer (or potentially no) votes in their local color distributions. We can therefore recompute the local mean and variance estimates using weighted statistics, where the visibilities $V(x, y, d, k)$ provide the weights.

With these new statistics, we are now in position to refine the disparity map. In particular, voxels in disparity space that previously had an inconsistent set of color votes (large variance) may now have a consistent set of votes, because voxels in (partially occluded) regions will now only receive votes from input pixels that are not already assigned to nearer surfaces.

Figure 2 shows the results of this algorithm when run on a cropped portion of the *SRI Trees* multibaseline stereo dataset. A small region (64×64 pixels) was selected in order to better visualize pixel-level errors. While the overall reconstruction is somewhat noisy, the final reconstruction with a synthetic blue layer inserted shows that the algorithm has done a reasonable job of assigning pixel depths and computing partial transparencies near the tree boundaries.

From this example, we see that the volumetric approach is a much more powerful representation for dealing with partially occluded regions and mixed pixels. Unfortunately, this power comes at the expense of two problems: the depth are quantized, which can lead to aliasing effects, and the representation has a very large number of degrees of freedom, which makes it difficult to find the optimal solution. The first problem could be fixed,

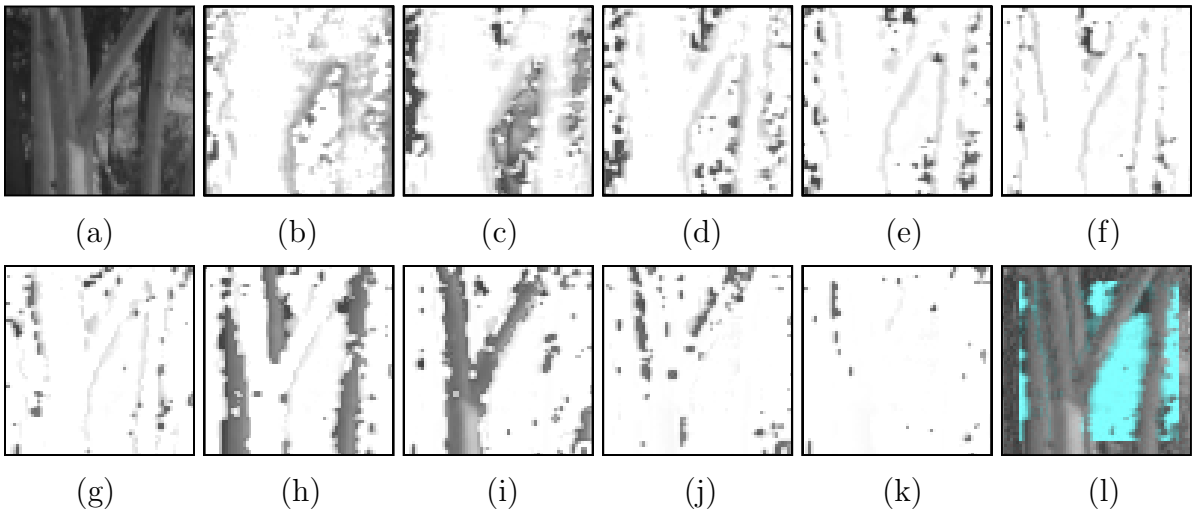


Figure 2: Volumetric reconstruction example: (a) cropped subimage from *SRI Trees* data set, (b–k) disparity layers $d = 0 \ldots 9$, (l) re-synthesized input image with inserted $d = 4$ blue layer.

in principle, by using fractional disparities (although these would have to be relative to one preferred camera). The second problem we address in the next section, where a much more parsimonious description is used.

4 Layered representations

To overcome the problem with the volumetric representation, we draw some inspiration from recent work in layered motion estimation [29]. Here, the goal is to decompose the images into sub-images, commonly referred to as *layers,* such that the pixels within each layer move in a manner consistent with a parametric transformation. The motion of each layer is determined by the values of the parameters. An important transformation is the 8–parameter homography (collineation), because it describes the motion of a rigid planar patch as either it or the camera moves.

While existing techniques have been successful in detecting multiple independent motions, layer extraction for scene modeling has not been fully developed. One fact that has not been exploited is that, when simultaneously imaged by several cameras, each of the layers implicitly lies on a fixed plane in the 3D world. Another omission is the proper treatment of transparency. With a few exceptions, the decomposition of an image into layers that are partially transparent has not been attempted. In contrast, scene modeling using multiple partially transparent layers is common in the graphics community.

In our own work [1], we have developed a framework for reconstructing a scene as a

collection of approximately planar layers. Each of the layers has an explicit 3D plane equation and is recovered as a *sprite*, i.e. a colored image with per-pixel opacity (transparency). To model a wider range of scenes, a per-pixel depth offset relative to the plane is also added.

Our layered approach to stereo shares many of the advantages of the volumetric approach. In addition, it offers a number of other advantages:

- The combination of the global model (the plane) with the local correction to it (the per-pixel depth offset) results in very robust performance. In this respect, the framework is similar to the *plane + parallax* work of [13].

- The output (a collection of approximately planar regions) is more suitable than a discrete collection of voxels for many applications, including, rendering [20] and video parsing.

Our representation consists of a collection of L approximately planar layers, each of which is an alpha-matted color image (layer sprite) $L_l(x, y)$ with *pre-multiplied opacities*. We also associate a homogeneous vector $\mathbf{n}_l$ with each layer (which defines the plane equation of the layer via $\mathbf{n}_l^T \mathbf{x} = 0$) and a per-pixel residual depth offset $Z_l(x, y)$.

To estimate the sprite layers, we have developed an algorithm that consists of two phases [1]. In the first part, we assume boolean opacities to get a first approximation to the structure of the scene. Since the opacities are boolean, each point in each image I_k is only the image of a point on one of the layers L_l. We therefore introduce boolean masks B_{kl} which denote the pixels in image I_k that are images of points on layer L_l, from which we can compute masked input images $M_{kl} = B_{kl} \cdot I_k$. Estimating the homographies that relate these image fragments to each other allows us to compute both the camera locations and the plane equations [27].

In the second part of our algorithm, we use the initial estimates of the layers made by the first part as input into a re-synthesis algorithm that refines the layer sprites L_l, *including* the opacities α_l. A more detailed description of the complete algorithm can be found in [1].

Figure 3 shows some results of applying our algorithm to five images from a 40-image stereo data set taken at a graphics symposium. Figure 3(a) shows the middle input image, Figure 3(b) shows the initial pixel assignment to layers, Figure 3(c) shows the recovered planar depth map, and Figure 3(f) shows the residual depth map for one of the layers. Figures 3(d) and (e) show the recovered sprites. Figure 3(g) shows the middle image re-synthesized from these sprites. Finally, Figures 3(h–i) show the same sprite collection seen from a novel viewpoint (well outside the range of the original views), first with and

Figure 3: Layered stereo results: (a) third of five images; (b) initial segmentation into six layers; (c) recovered depth map (darker denotes closer); (d) and (e) the five layer sprites; (f) residual depth image for fifth layer. (g) re-synthesized third image (note extended field of view). (h) novel view without residual depth; (i) novel view with residual depth (note the "rounding" of the people).

then without residual depth correction. The gaps in Figure 3 correspond to parts of the scene that were not visible in any of the five input images.

To summarize, the layered approach to 3D reconstruction represents the scene as a collection of approximately planar layers. Each layer consists of a plane equation, a layer sprite image, and a residual depth map. The framework exploits the fact that each layer implicitly lies on a fixed plane in the 3D world. This is both the algorithm's strength (using a compact description) and its weakness (it is limited to scenes where objects are "cutouts with relief"). The layered approach also requires solving a combinatorial optimization problem, since the number of layers needs to be determined, as well as figuring out the assignment of pixels to layers [28].

4.1 Transparent layers

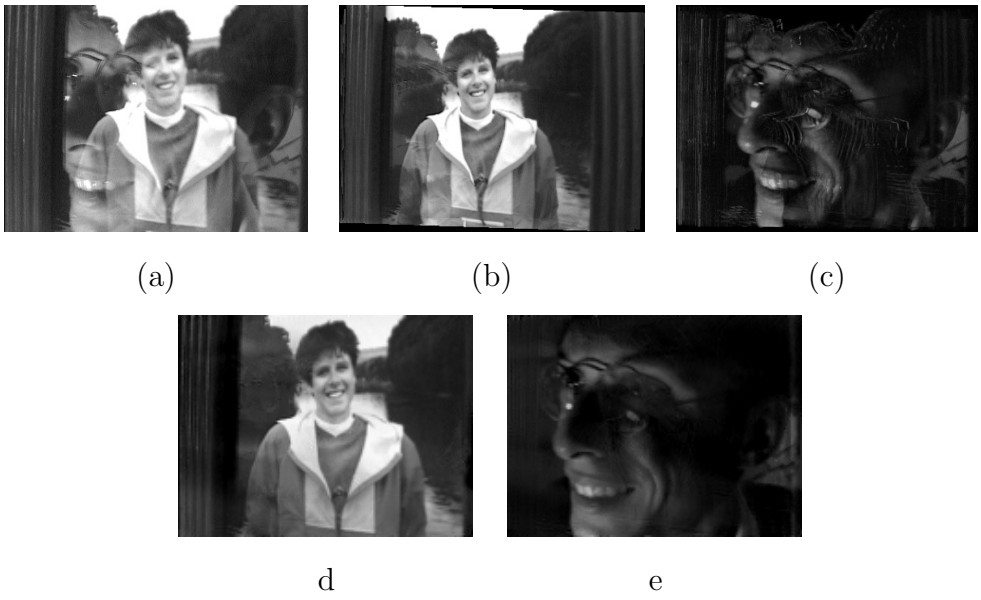
While the system described in [1] can be used to recover opaque layers that occlude one another, a different approach is required when the layers are *transparent*. In the case of a reflection in a window, the light reflecting off the glass is being added to the light transmitted through the window. Hence, the compositing equation is now a purely additive mixing of warped layers

$$m_k(x) = \sum_{l=0}^{L-1} W_{kl} \circ f_l(x), \quad f_l(x) \geq 0, \tag{2}$$

where W_{kl} are the warping operators corresponding to the layers motions, and $f_l(x)$ are the layer intensites, which are known to be non-negative [24]. To pull apart the transparent layers, we must estimate these two quanities.

To accomplish the first step, we use *dominant motion estimation* to lock onto the larger or higher contrast signal. However, instead of taking an average or median of the registered input images, we take a *min-composite* (minimum of registered pixel values at each pixel), since the contaminating layers are known to be non-negative [24]. Once we have subtracted this *over-estimate* of the dominant layer, we can then register the *residual* signals, and take their *max-composite* to get an initial second layer estimate.

Once the motions have been computed, the layer values themselves can be estimated using constrained least-squares, which is a simple quadratic programming problem. Figure 4a shows one image from an input sequence that consists of the reflection of a face in a photograph taken from a moving camera [2]. Figures 4b shows the *min*-composite of the dominant layer (the picture) while Figure 4c shows the *max*- composite of the reflection layer. Figures 4d and 4e show the results obtained using the constrained least-squares algorithm.



(a) (b) (c)

d e

Figure 4: Experimental results for the *Michael and Lee* sequence [2]: (a) first input image, (b) dominant (picture) layer *min-composite*, (c) reflection layer *max-composite*, (d)-(e) final estimated picture and reflection layers.

5 Multiple depth maps

In our most recent stereo algorithm, we have developed an alternative to volumetric and layered representations that can also represent and reason about semi-occluded regions. Rather than estimating a single depth map, we associate a depth map with *each* input image (or some subset of them) [23]. Furthermore, we try to ensure consistency between these different estimates using a *depth compatibility* constraint, and reason about occlusion relationships by computing pixel *visibilities*. Our representation can be used as is for image-based rendering (view interpolation) applications, or it can be used as a low-level representation from which segmentation and layer extraction (or 3D model construction) can take place.

To formulate the multi-view stereo problem, we take the matching costs for all reference images and sum them together.

This *brightness compatibility* term, which measures the degree of agreement in brightness or color between corresponding pixels, can be written as

$$\mathcal{C}(\{\mathbf{u}_s\}) = \sum_{s \in S} \sum_{t \in \mathcal{N}(s)} w_{st} \sum_{\mathbf{x}_s} v_{st}(\mathbf{x}_s) \rho \left(I_s(\mathbf{x}_s) - I_t(\mathbf{x}_t) \right) . \tag{3}$$

The images I_s form the set S of *keyframes* (or *key-views*) for which we will estimate a depth

(a) (b) (c)

Figure 5: Results of multi-view stereo algorithm: (a) depth estimate for first frame; (b) warped (resampled) images without visibilities; (c) with visibility computation.

estimate $d_s(\mathbf{x}_s)$. The decision as to which images are keyframes is problem-dependent, much like the selection of I and P frames in video compression. For 3D view interpolation, one possible choice of keyframes would be a collection of *characteristic views*.

Images $I_t, t \in \mathcal{N}(s)$ are *neighboring frames* (or views), for which we require that corresponding pixel brightnesses (or colors) agree. The pixel coordinate $\mathbf{x}_t$ corresponding to a given keyframe pixel $\mathbf{x}_s$ with depth d_s can be computed according to the rigid motion model. The constants w_{st} are the *inter-frame weights* which dictate how much neighboring frame t will contribute to the estimate of d_s. Corresponding pixel brightness or color differences are passed through a robust penalty function ρ.

The visibility factor $v_{st}(\mathbf{x}_s)$, which encodes whether pixel $\mathbf{x}_s$ is *visible* in image I_t, can be computed by comparing corresponding depth values, i.e., checking whether $d_t(\mathbf{x}_t) \leq d_s(\mathbf{x}_s)$.

The cost function used in [23] consists of two additional terms. The controlled *depth compatibility* constraint, enforces *mutual consistency* between depth estimates at different neighboring keyframes. The controlled *depth smoothness* constraint, encourages the depth maps to be piecewise smooth. The shape of this robust penalty function is affected by the brightness/color difference between neighboring pixels (see [23] for details).

Our algorithm operates in two phases. During an initialization phase, we estimate the depths independently for each keyframe. Since we do not yet have any good motion estimates for other frames, the depth compatibility term $\mathcal{C}_T$ is ignored, and no visibilities are computed (i.e., $v_{st} = 1$). In the second phase, we enforce depth compatibility and compute visibilities based on the current collection of depth estimates $\{d_s\}$. Details on the optimization algorithm can be found in [23].

Figure 5 shows some representative results from running our algorithm. The depth map estimated by the algorithm is shown in Figure 5a. Figure 5b shows the results of warping the last image to the first image, based on the depth computed in the first image. Displaying these warped images as the algorithm progresses is a very useful way to debug

Candle flame Balloons Watering Can Fish Tank

Figure 6: Selected stills from generated video textures.

the algorithm and to assess the quality of the motion estimates. Figure 5c shows the same warped image with invisible pixels flagged as black. Notice how the algorithm correctly labels most of the occluded pixels to the right of the two people's heads.

The experimental results we have obtained so far are encouraging, but still leave room for improvement. In particular, the smoothness of the final estimates and the *sharpness* of the motion discontinuities is not as high as that obtainable with layered models [1]. This is particularly true in occluded regions: layered models will apply the layer's motion to the occluded regions, while we use a weak smoothness constraint.

6 Video-Based Rendering

In our most recent work, we have started looking at the question: if image-based rendering is about generating novel views from a collection of images, how can we generate novel *videos* from some sample video footage? We call this new area of inquiry *video-based rendering* [17].

Previous examples of video-based rendering include Video Rewrite [4], in which a set of talking-head videos was analyzed so that the appropriate lip motion could be synthesized to go along with new speech. In our own work, which we call *Video Textures*, we have developed an algorithm that takes as input a short video clip, and then generates a similar looking continuous, randomly varying video of infinite duration [17].

The central element of a Video Texture is a description of the *looping structure* present in the original video, i.e., the set of transitions where the video can be jumped backwards (or forwards) without introducing visual or temporal discontinuities. The "Candle Flame" image in Figure 6 shows these transitions overlayed as red arcs over a still from the video texture. Using this technique, we have been able to successfully analyze and synthesize such diverse video element as fire (candles and bonfires), wind (the algorithm

automatically detects independently moving regions, such as those color coded in the "Balloons" still), waterfalls and water fountains, children on swings, people running, and someone smiling for a "video portrait" [17]. By separately analyzing an object's appearance and motion, we can also generate synthetic trajectories, potentially under a user's or program's control. The Fish Tank video texture shows a complex video consisting of several elements (bubbles, swaying plants), including some fish that can be scripted to follow paths or to react to mouse events (e.g., to follow the cursor). In future work, we are planning to incorporate Video Texture elements into more complex scenes that have been modeled using more traditional image-based modeling techniques.

7 Discussion and Conclusions

In this paper, I have presented a number of representation and algorithms for reconstructing 3D scenes and objects from multiple images and videos. My emphasis has been on techniques that are well suited to image-based rendering, i.e., approaches that can re-synthesize observed and novel views with a high degree of realism.

The simplest representation is a single depth map, which is very compact and can yield good results when the amount of occlusion is not large, i.e., when the surface is smoothly varying (e.g., a human face) and the range of viewpoints is limited. The volumetric representation (with partial opacities) can be used to represent and reason about partially occluded regions and mixed pixels. Unfortunately, it also has many degrees of freedom, which makes it tricky to find the best reconstruction. Layered representations have the same advantages as volumetric ones, and are potentially more compact, and hence easier to recover. Furthermore, they can represent transparent motion as well as regular opaque objects. However, determining the best number of planes and the correct pixel assignment is a tricky problem, which we are currently trying to solve. Finally, multiple depth maps can be used to obtain some of the same advantages with respect to partially occluded regions, and also to model the variation in appearance between viewpoints. Unfortunately, they are not guaranteed to have consistent representations of 3D shape, and also do not correctly predict the appearance of mixed pixels. In addition to these static representations, temporal analysis of (single or multiple) videos can be used to create dynamic versions of image-based models.

Unfortunately, all of the representations suggested so far have their limitations. Still, a tremendous amount of progress has been made in recent years in obtaining better and better stereo reconstructions, especially for image-based rendering applications where recovering the true shape of a scene is not paramount. I expect that by re-visiting issues

in representation, e.g., by more closely studying the role of discontinuities in shape and depth representations, we will be able to make even further progress, and thereby expand the utility and applicability of image and video-based modeling techniques.

References

[1] S. Baker, R. Szeliski, and P. Anandan. A layered approach to stereo reconstruction. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'98)*, pages 434–441, Santa Barbara, June 1998.

[2] M. J. Black and P. Anandan. The robust estimation of multiple motions: Parametric and piecewise-smooth flow fields. *Computer Vision and Image Understanding*, 63(1):75–104, 1996.

[3] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. In *Seventh International Conference on Computer Vision (ICCV'99)*, pages 377–384, Kerkyra, Greece, September 1999.

[4] C. Bregler, M. Covell, and M. Slaney. Video rewrite: Driving visual speech with audio. *Computer Graphics (SIGGRAPH'97)*, pages 353–360, August 1997.

[5] S. Chen and L. Williams. View interpolation for image synthesis. *Computer Graphics (SIGGRAPH'93)*, pages 279–288, August 1993.

[6] S. E. Chen. QuickTime VR – an image-based approach to virtual environment navigation. *Computer Graphics (SIGGRAPH'95)*, pages 29–38, August 1995.

[7] R. T. Collins. A space-sweep approach to true multi-image matching. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'96)*, pages 358–363, San Francisco, California, June 1996.

[8] U. R. Dhond and J. K. Aggarwal. Structure from stereo—a review. *IEEE Transactions on Systems, Man, and Cybernetics*, 19(6):1489–1510, November/December 1989.

[9] S. Geman and D. Geman. Stochastic relaxation, Gibbs distribution, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6(6):721–741, November 1984.

[10] S. J. Gortler, R. Grzeszczuk, R. Szeliski, and M. F. Cohen. The lumigraph. In *Computer Graphics Proceedings, Annual Conference Series*, pages 43–54, Proc. SIGGRAPH'96 (New Orleans), August 1996. ACM SIGGRAPH.

[11] S. S. Intille and A. F. Bobick. Disparity-space images and large occlusion stereo. In *Proc. Third European Conference on Computer Vision (ECCV'94)*, volume 1, Stockholm, Sweden, May 1994. Springer-Verlag.

[12] T. Kanade and M. Okutomi. A stereo matching algorithm with an adaptive window: Theory and experiment. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(9):920–932, September 1994.

[13] R. Kumar, P. Anandan, and K. Hanna. Direct recovery of shape from multiple views: A parallax based approach. In *Twelfth International Conference on Pattern Recognition (ICPR'94)*, volume A, pages 685–688, Jerusalem, Israel, October 1994. IEEE Computer Society Press.

[14] M. Levoy and P. Hanrahan. Light field rendering. In *Computer Graphics Proceedings, Annual Conference Series*, pages 31–42, Proc. SIGGRAPH'96 (New Orleans), August 1996. ACM SIGGRAPH.

[15] M. Okutomi and T. Kanade. A multiple baseline stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(4):353–363, April 1993.

[16] D. Scharstein and R. Szeliski. Stereo matching with nonlinear diffusion. *International Journal of Computer Vision*, 28(2):155–174, July 1998.

[17] A. Schödl, R. Szeliski, D. H. Salesin, and I. Essa. Video textures. In *Computer Graphics (SIGGRAPH'2000) Proceedings*, pages 489–498, New Orleans, July 2000. ACM SIGGRAPH.

[18] S. M. Seitz and C. M. Dyer. View morphing. In *Computer Graphics Proceedings, Annual Conference Series*, pages 21–30, Proc. SIGGRAPH'96 (New Orleans), August 1996. ACM SIGGRAPH.

[19] S. M. Seitz and C. M. Dyer. Photorealistic scene reconstrcution by voxel coloring. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'97)*, pages 1067–1073, San Juan, Puerto Rico, June 1997.

[20] J. Shade, S. Gortler, L.-W. He, and R. Szeliski. Layered depth images. In *Computer Graphics (SIGGRAPH'98) Proceedings*, pages 231–242, Orlando, July 1998. ACM SIGGRAPH.

[21] H.-Y. Shum and L.-W. He. Rendering with concentric mosaics. In *SIGGRAPH'99*, pages 299–306, Los Angeles, August 1999. ACM SIGGRAPH.

[22] H.-Y. Shum and R. Szeliski. Construction of panoramic mosaics with global and local alignment. *International Journal of Computer Vision*, 36(2):101–130, February 2000.

[23] R. Szeliski. A multi-view approach to motion and stereo. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'99)*, volume 1, pages 157–163, Fort Collins, June 1999.

[24] R. Szeliski, S. Avidan, and P. Anandan. Layer extraction from multiple images containing reflections and transparency. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2000)*, volume 1, pages 246–253, Hilton Head Island, June 2000.

[25] R. Szeliski and P. Golland. Stereo matching with transparency and matting. *International Journal of Computer Vision*, 32(1):45–61, August 1999. Special Issue for Marr Prize papers.

[26] R. Szeliski and H.-Y. Shum. Creating full view panoramic image mosaics and texture-mapped models. *Computer Graphics (SIGGRAPH'97)*, pages 251–258, August 1997.

[27] R. Szeliski and P. Torr. Geometrically constrained structure from motion: Points on planes. In R. Koch and L. Van Gool, editors, *European Workshop on 3D Structure from Multiple Images of Large-Scale Environments (SMILE)*, pages 171–186, Freiburg, Germany, June 1998.

[28] P. H. S. Torr, R. Szeliski, and P. Anandan. An integrated Bayesian approach to layer extraction from image sequences. In *Seventh International Conference on Computer Vision (ICCV'98)*, pages 983–990, Kerkyra, Greece, September 1999.

[29] J.Y.A. Wang and E.H. Adelson. Layered representation for motion analysis. In *Proceedings of the 1993 Conference on Computer Vision and Pattern Recognition*, pages 361–366, 1993.