Pop-Up Light Field: An Interactive Image-Based Modeling and Rendering System

HEUNG-YEUNG SHUM and JIAN SUN
Microsoft Research Asia
SHUNTARO YAMAZAKI
University of Tokyo
and
YIN LI and CHI-KEUNG TANG
University of Science and Technology, Hong Kong

In this article, we present an image-based modeling and rendering system, which we call *pop-up light field*, that models a sparse light field using a set of *coherent layers*. In our system, the user specifies how many coherent layers should be modeled or popped up according to the scene complexity. A coherent layer is defined as a collection of corresponding planar regions in the light field images. A coherent layer can be rendered free of aliasing all by itself, or against other background layers. To construct coherent layers, we introduce a Bayesian approach, *coherence matting*, to estimate alpha matting around segmented layer boundaries by incorporating a coherence prior in order to maintain coherence across images.

We have developed an intuitive and easy-to-use user interface (UI) to facilitate pop-up light field construction. The key to our UI is the concept of human-in-the-loop where the user specifies where aliasing occurs in the rendered image. The user input is reflected in the input light field images where pop-up layers can be modified. The user feedback is instant through a hardware-accelerated real-time pop-up light field renderer. Experimental results demonstrate that our system is capable of rendering anti-aliased novel views from a sparse light field.

Categories and Subject Descriptors: I.3.6 [**Computer Graphics**]: Methodology and Techniques—*interaction techniques*; I.3.7 [**Computer Graphics**]: Three-Dimensional Graphics and Realism; I.4.6 [**Image Processing and Computer Vision**]: Segmentation

General Terms: Algorithms, Experimentation

Additional Key Words and Phrases: Image-based rendering, interactive techniques, light field, lumigraph, layers, matting

1. INTRODUCTION

Central to many image-based rendering (IBR) systems is the goal of interpolating accurately between the sample images in order to generate novel views. In IBR, rendering a desired pixel is often equivalent

The research of Yin Li and Chi-Keung Tang is supported in part by the Research Grant Council of Hong Kong Special Administration Region, China: HKUST6246/00E and AOE/E-01/99, while Yin Li was an intern at Microsoft Research Asia.
Authors' addresses: H.-Y. Shum and J. Sun, Microsoft Research Asia, 3/F Sigma Building, Zhichun Rd/Beijing, China; email: {hshum;t-jiansu}@microsoft.com; S. Yamazaki, 3-27-4-301 Higashiohi, Shinagawa, Tokyo 140-0011, Japan; email: shun@cvl.iis.u-tokyo.ac.jp; Y. Li and C.-K. Tang, Computer Science Department, Hong Kong University of Science and Technology, Clear Water Bay, Hong Kong, email: liyin@ust.hk; cktang@cs.ust.hk.

to interpolating intensity values of some input pixels. Such an interpolation, however, depends on the correspondence between the rendered pixel and those pixels from the input sample images. Accurate correspondence between these pixels can be obtained if we have a large number of input images or an accurate geometric model of the scene.

The issue of realistic rendering using dense image samples in IBR has been very well explored in the forms of Light Field [Levoy and Hanrahan 1996] and Lumigraph [Gortler et al. 1996]. In light field rendering, a simple focal plane is sufficient to establish accurate correspondence between interpolating pixels. When sampling is sparse, however, conventional light field rendering would cause aliasing because of erroneous correspondence. Some researchers designed reconstruction filter more sophisticated than a simple linear interpolation filter to reduce the ghosting effects [Stewart et al. 2003]. On the other hand, the Facade system [Debevec et al. 1996] shows that, from only a few photographs, realistic rendering of a building's architecture can be achieved because the reconstructed geometry model provides accurate correspondence for view-dependent texture mapping. A formal analysis of the tradeoff between the number of images and the amount of geometry (in terms of the number of depth layers associated with each pixel) is presented in Chai et al. [2000].

Many IBR systems have proposed different geometric proxies [Buehler et al. 2001] to alleviate the problem introduced by undersampled light fields. In practice, however, acquiring an approximate yet *continuous* proxy model is a challenging task. A continuous proxy model is needed for these IBR systems because every desired ray must intersect a point on the geometric proxy in order to establish the correspondence between interpolating pixels [Schirmacher et al. 2000]. This explains why most IBR systems today have used a very simple geometric proxy (e.g., a focal plane in dynamically reparameterized light field [Isaksen et al. 2000]) for a complex scene, but can construct a relatively complex proxy for a single object (e.g., the lion model in Lumigraph [Gortler et al. 1996] or the car model in unstructured Lumigraph [Buehler et al. 2001]). The image-based visual hull [Matusik et al. 2000] is another geometry proxy that can be constructed and updated in real-time from silhouettes of a single object. If a single global geometric proxy is not sufficient, local geometric proxies can also be used to improve rendering quality such as the view-dependent geometry [Rademacher 1999] used in impostors [Schaufler 1995], Microfacet Billboarding [Yamazaki et al. 2002], and Billboard Clouds [Décoret et al. 2003]. These local geometric proxies are mostly for single objects as well.

The Problem. We address the following problem in this article. Can we use a relatively sparse set of images of a complex scene and produce photorealistic virtual views free of aliasing? A straightforward approach would be to perform stereo reconstruction or to establish correspondence between all pixels of the input images. The geometric proxy is a depth map for each input image. Unfortunately, state-of-the-art automatic stereo algorithms are inadequate for producing sufficiently accurate depth information for realistic rendering. Typically, the areas around occlusion boundaries [Kolmogorov and Zabih 2002; Kang et al. 2001] in the scene have the least desirable results, because it is very hard for stereo algorithms to handle occlusions without prior knowledge of the scene.

Our Approach. We approach this problem by suggesting that it is not necessary to reconstruct accurate 3D information for each pixel in the input light field. Our solution is to construct a pop-up light field by segmenting the input–sparse light field into multiple coherent layers. Pop-up light field differs from other layered modeling and rendering approaches (e.g., Lengyel and Snyder [1997], Shade et al. [1998], and Baker et al. [1998]) in a number of ways. First, the number of layers needed in a pop-up light field is not pre-determined. Rather, it is decided interactively by the user. Second, the user specifies the layer boundaries in key frames. The layer boundary is then propagated to the remaining frames automatically. Third, our representation is simple. Each layer is represented by a planar surface without the need for per-pixel depth. Fourth and most importantly, our layers are coherent so that anti-aliased rendering using these coherent layers is achieved. Each coherent layer must have sufficiently small

depth variation so that anti-aliased rendering of the coherent layer itself becomes possible. Moreover, to render each coherent layer with its background layers, not only is accurate layer segmentation is required on every image, but segmentation across all images must be consistent as well.

To segment the layers, the user interface is key. A good user interface can enable the user to intuitively manipulate the structure of a pop-up light field so that virtual views with the desired level of fidelity can be produced. By having a human-in-the-loop for pop-up field construction, the user can specify where aliasing occurs in the rendered image. Then, corresponding layers are refined accordingly. More layers are popped up, refined, and propagated across all images in the light field until the user is satisfied with the rendering quality (i.e., no aliasing is perceived).

Outline of this Article. The rest of this article is organized as follows. After reviewing related work in Section 2, we introduce in Section 3 the representation of pop-up light field, which consists of a set of coherent layers, and our novel *coherence matting* that maintains the layer consistency across frames in the light field. Section 4 details the operations in the user interface. Section 5 describes our real-time hardware-accelerated pop-up light field rendering algorithm. Experimental results are presented in Section 6. Discussion and concluding remarks are made in Section 7.

2. RELATED WORK

Interactive Modeling Systems. Many image-based interactive modeling systems use only one image, which impose or assume certain geometric constraints on the scenes (e.g., Tour in Pictures [Horry et al. 1997] models the scene by a simple spidery mesh. In single view metrology [Criminisi et al. 1999], 2D projections of 3D parallel lines must be present in the input image so that the user can click on them for computing vanishing points). An elaborate modeling system was proposed in Mok et al. [2001] where depth values are assigned to pixels in a single picture. Interactive single-view systems are difficult to generalize to a sparse light field, that still consists of many images. Furthermore, while it is straightforward to perform interactive image segmentation on a single image, consistent propagation of image regions to different images is a challenging task.

The UI for designing a multiview/multiframe interactive modeling system has been a challenge. Most available movie editing tools are in fact manual systems, requiring frame-by-frame editing and consistency maintenance. In Debevec et al. [1996], an interactive modeling system is proposed that makes use of a depth map derived from a sparse set of views (image-based) and a modeling program (geometry-based). Plenoptic editing [Seitz and Kutulakos 1998] first recovers a 3D voxel model from a sparse light field, and then applies traditional 3D warping to the recovered model. Thus, this automatic system shares the same shortcomings with stereo reconstruction. Recently, feature-based light field morphing [Zhang et al. 2002] has been proposed to morph two light fields. The key is an easy-to-use UI for feature specification. The feature polygons are then used to compute the visibility map and perform ray space warping. Consistency is maintained by ray correspondence.

Layer Modeling and Rendering. Layers have been proved successful [Lengyel and Snyder 1997; Baker et al. 1998; Shade et al. 1998] in image-based rendering. Coherent layers [Lengyel and Snyder 1997] are constructed from 3D models for efficient rendering. In layered depth image (LDI), a two-step rendering algorithm is proposed to render sprites (layers) with depth. While great progress has been made in automatic reconstruction of layers [Wang and Adelson 1994; Baker et al. 1998; Ke and Kanade 2002] in the computer vision community, the layers are not accurate enough for artifact-free rendering. In pop-up light field, the user interactively determines how many layers are to be popped up, initializes the layers in key frames, and modifies the layers by inspecting the rendering quality.

Matting. Layered representations always demand accurate alpha matting along the layer boundary (e.g., the sprites in Shade et al. [1998]). To estimate alpha matting from images, blue screen matting was proposed in Smith and Blinn [1996]. Recently, a Bayesian approach for digital matting was

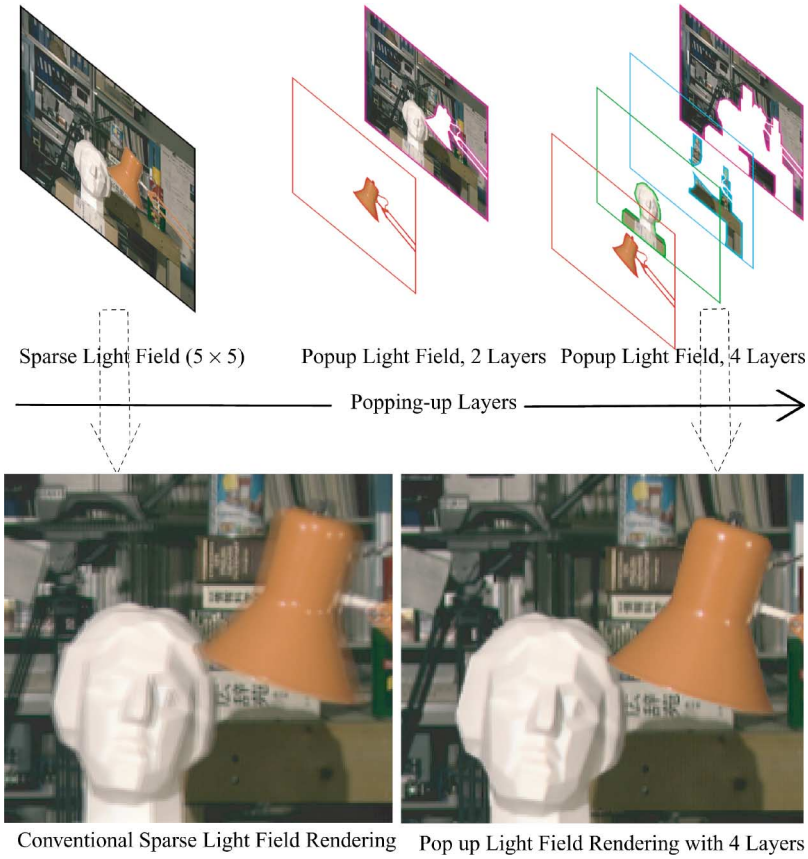


Fig. 1. An example of rendering with pop-up light fields. Rendering using the 5 × 5 Tsukuba light field data set is shown in the top left. Aliasing is clearly visible near the front objects in the bottom left image because the input light field is sparse. The top row shows that the pop-up light field splits the scene gradually into 4 coherent layers, and achieves an anti-aliased rendering as shown in the bottom right image.

proposed in Chuang et al. [2001], that provides an excellent survey on other patented matting algorithms [Mishima 1993; Berman et al. 2000; Ruzon and Tomasi 2000]. Video matting based on the same Bayesian framework has been reported in Chuang et al. [2002]. However, video matting is not designed to have consistent alpha mattes across frames. More accurate alpha matte can be estimated when the multi-backgrounds are available Wexler et al. [2002]. In image-based opacity hulls [Matusik et al. 2002], multi-background matting is used to acquire alpha mattes to construct a visual hull with view dependent opacity.

3. POP-UP LIGHT FIELD REPRESENTATION

3.1 An Example

When a light field is undersampled, conventional light field rendering [Levoy and Hanrahan 1996] results in aliasing. Figure 1 shows the rendering of a sparse light field with 5 × 5 Tsukuba images. The bottom left image is rendered with the 5 × 5 sparse light field by setting a single focal plane in the scene [Isaksen et al. 2000]. Double images can be easily observed on the front objects.

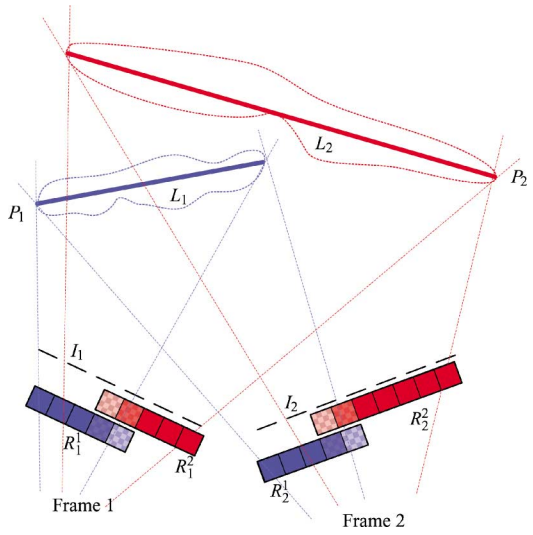


Fig. 2. A light field (with images I_1 and I_2) can be represented by a set of coherent layers (L_1 and L_2). A coherent layer is a collection of layered images in the light field. For instance, L_1 is represented by layered image R_1^1 (from I_1) and R_2^1 (from I_2). Each layered image has an alpha matte associated with its boundary. Part of the scene corresponding to each layer (e.g., L_1) is simply modeled as a plane (e.g., P_1).

The bottom right of Figure 1 shows that anti-aliased rendering can be achieved using four layers, each of which employs a simple planar surface as its geometric proxy. By splitting the scene into multiple layers, the depth variation in each layer becomes much smaller than that in the original sparse light field.

The pop-up light field is represented by a collection of *coherent layers*. A key observation in our pop-up light field representation is that the number coherent layers that should be modeled or "popped up" depends on the complexity of the scene and how undersampled the input light field is. For a sparser light field, more layers need to be popped up for anti-aliased rendering.

3.2 Coherent Layers

We represent a coherent layer L_j by a collection of corresponding layered image regions R_j^i in the light field images I^i. These regions are modeled by a simple geometric proxy without the need for accurate per-pixel depth. For example, a global planar surface (P_j) is used as the geometric proxy for each layer L_j in the example shown in Figure 2. To deal with complicated scenes and camera motions, we can also use a local planar surface P_j^i to model the layer in every image i of the light field.

A layer in the pop-up light field is considered coherent if the layer can be rendered free of aliasing by using a simple planar geometric proxy (global or local). Anti-aliased rendering occurs at two levels when

(1) the layer itself is rendered; and
(2) the layer is rendered with its background layers.

Therefore, to satisfy the first requirement, the depth variation in each layer must be sufficiently small, as suggested in Chai et al. [2000]. Moreover, the planar surface can be adjusted interactively to achieve the best rendering effect. This effect of moving the focal plane has been shown in Isaksen et al. [2000] and Chai et al. [2000].

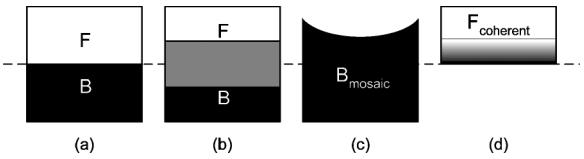


Fig. 3. Illustration of the major steps in coherence matting. (a) The user specifies an approximate segmentation. (b) An uncertain region is added in between the foreground and background. (c) A background mosaic is constructed from multiple under-segmented background images. (d) A coherent foreground layer is then constructed using coherence matting.

However, to meet the second requirement, accurate layer boundaries must be maintained across all the frames to construct the coherent layers. A natural approach to ensuring segmentation coherence across all frames is to propagate the segmented regions on one or more key frames to all the remaining frames [Shade et al. 1998; Zhang et al. 2002]. Sub-pixel precision segmentation may be obtained on the key frames by meticulously zooming on the images and tracing the boundaries. Propagation from key frames to other frames, however, causes inevitable under-segmentation or over-segmentation of a foreground layer. Typically over-segmentation of a foreground layer leads to the inclusion of background pixels, thus introducing ghosting along the occlusion boundaries in the rendered image. A possible example of foreground over-segmentation is exhibited in Figure 4(g) of Shade et al. [1998] where black pixels on the front object's boundary can be observed. To alleviate the rendering artifacts caused by over-segmentation or under-segmentation of layers, we need to refine the layer boundary with alpha matting [Porter and Duff 1984].

Figure 2 illustrates coherent layers of a pop-up light field. All the pixels at each coherent layer have consistent depth values (to be exact, within a depth bound), but may have different fractional alpha values along the boundary.

To produce fractional alpha mattes for all the regions in a coherent layer, a straightforward solution is to apply video matting [Chuang et al. 2002]. The video matting problem is formulated as a Maximum A Posterior (MAP) estimation as in Bayesian matting [Chuang et al. 2001],

$$\arg\max_{F,B,\alpha} P(F, B, \alpha|C) = \arg\max_{F,B,\alpha} L(C|F, B, \alpha) + L(F) + L(B) + L(\alpha) \quad (1)$$

where $L(\cdot) = log P(\cdot)$ is log likelihood, C is the observed color for a pixel, and F, B and α are foreground color, background color and alpha value to be estimated, respectively. For color image, C, F and B are vectors in RGB space. In Bayesian matting and video matting, the log likelihood for the alpha $L(\alpha)$ is assumed constant so that $L(\alpha)$ is dropped from Eq. (1).

In video matting, the optical flow is applied to the trimap (the map of foreground, background and uncertain region), but not to the output matte. The output foreground matte is produced by Bayesian matting on the current frame, based on the propagated trimap. Video matting works well if we simply replay the foreground mattes against a different background. However, these foreground mattes may not have in-between frame coherence that is needed for generating novel views.

3.3 Coherence Matting

We propose a novel approach, called *coherence matting* to construct the alpha mattes in a coherent layer that have in-between frame coherence. The workflow of our approach is similar to video matting and is illustrated in Figure 3. First, the user-specified boundaries are propagated across frames. Second, the uncertain region along the boundary is determined. Third, the under-segmented background regions from multiple images are combined to construct a sufficient background image. Fourth, the alpha matte for the foreground image (in the uncertain region) is estimated. The key to our approach is at the fourth step in Figure 3(d) where we introduce a coherent feathering function across the corresponding layer

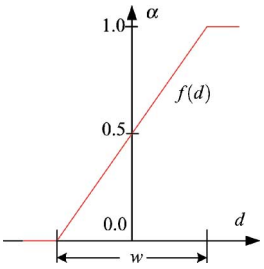


Fig. 4. The feathering function used in coherence matting.

boundaries. Note that, for a given layer, a separate foreground matte is estimated independently for each frame in the light field, and the coherence across frames is maintained by foreground boundary consistency.

$L(B)$ in Eq. (1) can be dropped since we can explicitly estimate the background (see Sect. 4.4). By incorporating a coherence prior on the alpha channel $L(\alpha)$ across frames, coherence matting can be formulated as

$$L(F, B, \alpha|C) = L(C|F, B, \alpha) + L(F) + L(\alpha) \tag{2}$$

where the log likelihood for the alpha $L(\alpha)$ is modeled as:

$$L(\alpha) = -(\alpha - \alpha_0)^2/\sigma_a^2 \tag{3}$$

where $\alpha_0 = f(d)$ is a feathering function of d and σ_a^2 is the standard deviation. d is the distance from the pixel to the layer boundary. The feathering function $f(d)$ defines the α value for surrounding pixels of a boundary. In this article, the feathering function is set as $f(d) = (d/w) * 0.5 + 0.5$, where w is feathering width, as illustrated in Figure 4.

Assume the observed color distribution $P(C)$ and sampled foreground color distribution $P(F)$ (from a set of neighboring foreground pixels) are all of Gaussian distribution:

$$L(C|F, B, \alpha) = -\|C - \alpha F - (1 - \alpha)B\|^2/\sigma_C^2 \tag{4}$$

$$L(F) = -(F - \overline{F})^T \Sigma_F^{-1} (F - \overline{F}) \tag{5}$$

where σ_C is the standard deviation of the observed color C, $\overline{F}$ is the weighted average of foreground pixels and Σ_F is the weighted covariance matrix. Taking the partial derivatives of (2) with respect to F and α and forcing them to equal zero, results in the following equations:

$$F = \frac{\Sigma_F^{-1}\overline{F} + C\alpha/\sigma_C^2 - B\alpha(1 - \alpha)/\sigma_C^2}{\Sigma_F^{-1} + I\alpha^2/\sigma_C^2} \tag{6}$$

$$\alpha = \frac{(C - B) \cdot (F - B) + \alpha_0 \cdot \sigma_C^2/\sigma_a^2}{\|F - B\|^2 + \sigma_C^2/\sigma_a^2} \tag{7}$$

α and F are solved alternatively by using (6) and (7). Initially, α is set to α_0.

3.4 Rendering With Coherence Matting

Bayesian matting [Chuang et al. 2001] and video matting [Chuang et al. 2002] solve the matting from the equation

$$\alpha = \frac{(C - B) \cdot (F - B)}{\|F - B\|^2},$$

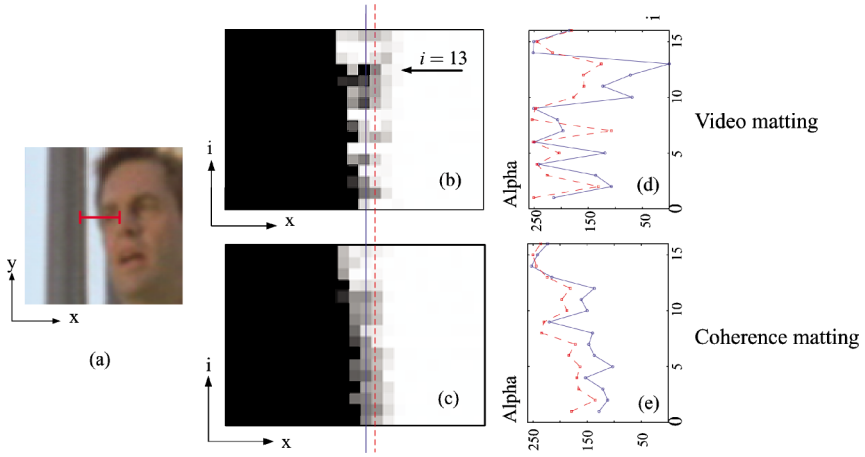


Fig. 5. Comparison between video matting and coherence matting: (a) is a small window on one frame in the Plaza data (Figure 12); (b) and (c) are two alpha epipolar plane images (α-EPI) corresponding to the red line in (a), using the algorithm of video matting and coherence matting respectively; (d) and (e) are the alpha curves of two adjacent columns, marked as blue and red lines in (b) and (c); (d) corresponds to video matting and shows a large jump at $i = 13$, which causes an accidental transparency within the face; (e) corresponds to coherence matting, which provides a more reasonable result.

which works well in general but becomes unstable when $F \approx B$. In comparison, Eq. (7) of the coherence matting can be solved more stably because applying the coherence prior on α results in a nonzero denominator. The coherence prior behaves similar to the smoothness constraint commonly used in visual reconstruction (e.g., shape from shading [Horn and Brooks 1989]).

The spatial inconsistency of the alpha matte from video matting can be observed in Figure 5. We plot the alpha epipolar plane image (α-EPI) of a video matting result. Similar to the conventional EPI [Baker and Bolles 1989], for a short segment of scanline from the Plaza sequence, we stack the alpha values along this segment for all of the 16 frames ((b) and (c)). The alpha values along 2 lines (solid and dotted) in the α-EPI are plotted in (d) and (e). Each line represents the alpha values of the corresponding pixels across 16 frames. A close inspection of (b) around frame $i = 13$(video matting method), shows that the alpha value changes from about 126 to 0, then to 180, (the range of alpha is from 0 to 255) indicating a small part of the face accidently becomes transparent.

The temporal incoherence of the alpha matte from video matting can be more problematic during rendering. The fluctuation of alpha values along both dotted and solid lines will generate incoherent alpha values and thus cause rendering artifacts as we change viewpoints (along axis i). This phenomenon can be more clearly observed in the accompanying videotape. Figure 5(e) shows the same solid and dotted lines with coherence matting results. Both lines have much less fluctuation between neighboring pixels and appear temporally smoother than their counterparts in Figure 5(d).

4. POP-UP LIGHT FIELD CONSTRUCTION

To construct a pop-up light field, we design an easy-to-use user interface (UI). The user can easily specify, refine, and propagate layer boundaries and indicate rendering artifacts. More layers can be popped up and refined until the user is satisfied with the rendering quality.

4.1 UI Operators

Figure 6 summarizes the operations in the pop-up light field construction UI. The key is that a human is in the loop. The user supplies the information needed for layer segmentation, background construction,

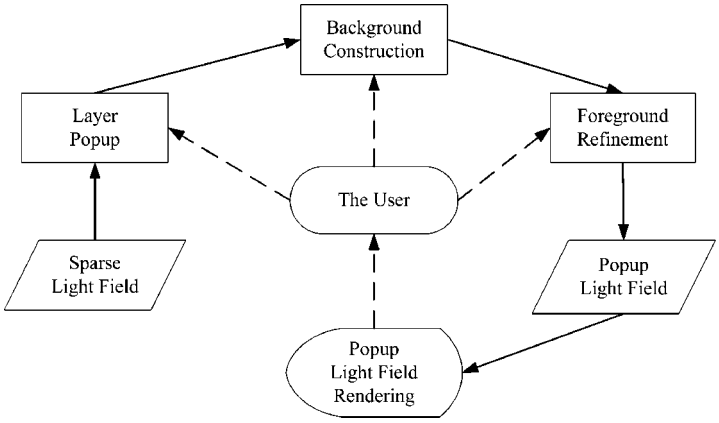


Fig. 6. Flowchart of pop-up light field construction UI.

and foreground refinement. By visually inspecting the rendering image from the pop-up light field, the user also indicates where aliasing occurs and thus which layer needs to be further refined. The user input or feedback is automatically propagated across all the frames in the pop-up light field. The four steps of operations in the UI are summarized as follows.

(1) *Layer pop-up*. This step segments layers and specifies their geometries. To start, the user selects a key frame in the input light field, specifies regions that need to be popped up, and assigns the layer's geometry by either a constant depth or a plane equation. This step results in a coarse segmentation represented by a polygon. The polygon region and geometry configuration can be automatically propagated across frames. Layers should be popped up in order of front-to-back. More details of layer pop-up are shown in Section 4.3.

(2) *Background construction*. This step obtains background mosaics that are needed to estimate the alpha mattes of foreground layers. Note that the background mosaic is useful only for the pixels around the foreground boundaries, for example, in the uncertain region as shown in Figure 3. More details of background construction are discussed in Section 4.4.

(3) *Foreground refinement*. Based on the constructed background layers, this step refines the alpha matte of the foreground layer by applying the coherence matting algorithm described in Section 3.3. Unlike layer pop-up in Step 1, foreground refinement in this step should be performed in back-to-front order.

(4) *Rendering feedback*. Any modification to the above steps will update the underlying pop-up light field data. The rendering window will be refreshed with the changes as well. By continuously changing the viewpoint, the user can inspect for rendering artifacts. The user can mark any rendering artifacts such as ghosting areas by brushing directly on the rendering window. The corresponding frame and layer will then be selected for further refinement.

4.2 UI Design

Figure 7 shows the appearance of our UI, including five workspaces where the user interacts with a frame and a layer in the pop-up light field. These workspaces and their functionalities are explained as follows.

Lower middle. The user chooses an active frame by clicking on the *frame navigator*, shown at the lower middle in Figure 7. The active frame appears in the *editing frame view*, shown at the upper left in Figure 7.

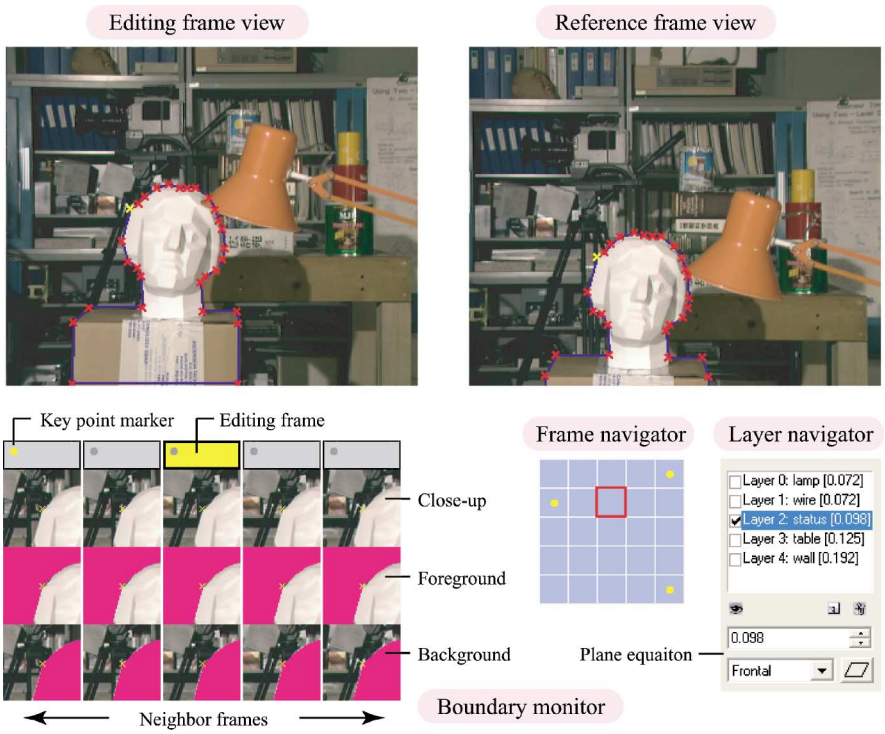


Fig. 7. The UI for Pop-up light field construction.

Upper left. In the *editing frame view*, the user can create or select an active layer and edit its polygon region. This active layer is displayed in blue polygons with crosses for each editable vertex. The information of the active layer is available in the *layer navigator*, shown at the lower right of Figure 7.

Lower right. From the *layer navigator*, the user can obtain the active layer's information. The user can select, add, or delete layers in the list. By selecting the layer in the check box, the user can turn on/off a layer's display in the *editing frame view*, *reference frame view* (shown at the upper right of Figure 7) and the rendering window. The plane equation of the active layer is displayed and can be modified through keyboard input. Layer equations can also be set through adjusting the rendering quality in the rendering window.

Upper right. The *reference frame view* is used to display another frame in the light field. This workspace is useful for a number of operations where correspondences between the reference frame view and the editing frame view need to be considered, such as specifying plane equations.

Lower left. To fine tune the polygon location for the active layer, the *boundary monitor* (lower left of Figure 7) shows close-up views of multiple frames in the light field. The first row shows the close-up around the moving vertex. The second and third rows show the foreground and background of the active layer composed with a fixed background selected by the user. For instance, using mono fuchsia color in Figure 7 as the background, it is easy for the user to observe over-segmentation or under-segmentation of the foreground across multiple frames simultaneously.

Not shown in the figure is the rendering window on which the user can render any novel view in real-time and can inspect the rendering quality. The user can also specify the frontal plane's equation for an

active layer by sliding the plane depth back and forth until the best rendering quality (i.e., minimum ghosting) is achieved. If the ghosting cannot be completely eliminated at the occlusion boundaries, the layer's polygon must be fine tuned. The user can brush on the ghosting regions, and the system can automatically select the affected frame and layer for modification. The affected layer is front-most and closest to the specified ghosting region.

To specify the slant plane equation for a layer, the user needs to select at least four pairs of corresponding points on the *editing frame view* and the *reference frame view*. The plane equation can be automatically computed and then used for rendering.

Also not shown in the above figure is a dialog box where the user can specify the feathering function. Specifying a feathering curve is useful for the coherence matting algorithm described in Section 3.3.

4.3 Layer Pop-Up

To pop up a layer, the user needs to segment and specify the geometry of the layer for all frames in the light field. This section discusses the operations by which the user interacts with the system and the underlying algorithms.

4.3.1 *Layer Initialization.* We use polygons to represent layer boundaries, since the correspondence between polygons can be maintained well in all frames by the corresponding vertices. The user can specify the layer's boundary with a polygon (e.g., using the polygon lasso tool in Adobe Photoshop) and edit the polygon by dragging the vertices. The editing will be immediately reflected in the *boundary monitor* window and in the rendering window (Sect. 4.2).

First of all, the user needs to inspect the rendering window by changing the viewpoint and decide which region is going to be popped up (usually the front-most, non-ghosting object). The user then selects a proper key frame to work with and draws a polygon on the frame.

Then, the user needs to specify the layer's geometry. For a frontal plane, the layer depth is the one that achieves the best rendering quality which can be observed on the rendering window by the user. For a slant plane, the user specifies at least four pairs of corresponding points on at least two frames to estimate the plane equation.

Once the layer geometry is decided, the polygon on the first key frame can be propagated to all other frames by back-projecting its vertices, resulting in a coarse segmentation of the layer on all frames in the light field. All vertices on the key frame are marked as key points. At this stage, the layer has a global geometry which is shared across all the frames. Moreover, an accurate polygon boundary for layer initialization is not necessary. Because of occlusions and viewpoint changes, propagated polygon boundaries inevitably need to be refined.

4.3.2 *Layer Refinement.* The following aspects need to be considered in layer refinement.

Boundary Refinement in a Key Frame. All vertices on any frame can be added, deleted, and moved. Once a vertex is modified, it is marked as a key point. The position of the modified vertex will be propagated across frames at once and the layer region will be updated in several UI workspaces. To adjust a vertex position, the user can observe how well foreground and background colors are separated in the *boundary monitor* window, or how much the ghosting effect is removed in the rendering window.

Boundary Propagation across Multiple Frames. For a specific vertex on the layer boundary, if it is a nonkey point on frame I_P, we want to interpolate its image coordinate from the corresponding key points in other frames. If there is only one key point in other frames, we compute the coordinate by back-projecting the intersection point of the layer plane and the viewing ray from the key point. Otherwise, we select two or three neighboring frames that contain the key points. Then, we compute the coordinate by back-projecting the 3D point which has a minimal sum of distances to the viewing rays from key points in these frames.

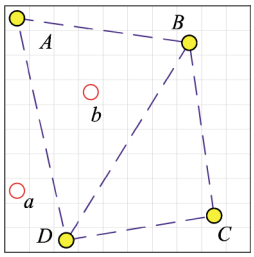


Fig. 8. Neighboring frames selection for 2D camera array. The solid (yellow) dots are frames including key point and hollow (red) dots are frames to be interpolated.

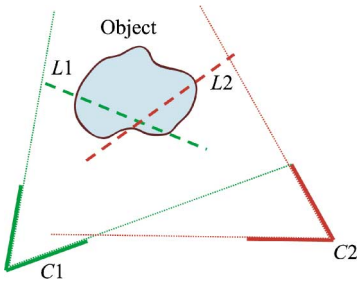


Fig. 9. Local geometry.

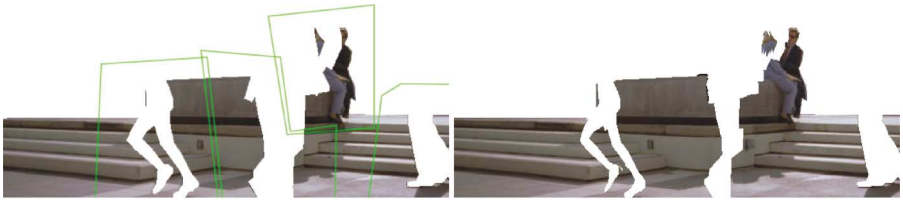
For an 1D camera array, we select the two frames closest to the left and right of the frame I_P that contain the key points. For a 2D camera array, we first compute the Delaunay triangulation in the camera plane, using all frames containing key points. If frame I_P is in the interior of a triangle, the three frames on the triangle vertices are neighboring frames. For example in Figure 8, A, B, and D are the neighboring frames of frame b. If frame I_P is in the exterior of all triangles, we select two frames I_0 and I_1 that maximize the angle $\angle I_0 I_P I_1$ in the camera plane. For example, A and D are the neighboring frames of frame a, as shown in Figure 8.

Note that the key points are those that have been modified by the user. They don't necessarily exist on key frames.

Coherence Matting. It is difficult to accurately describe a layer boundary simply by using polygons. It is hard for the user to manually adjust a boundary with subtle micro geometry to sub-pixel accuracy. A pixel is often blended with colors from both foreground and background due to the camera's point spread function. Therefore, we choose not to require the user to specify very accurate sub-pixel boundary positions. Instead, a coherence matting algorithm is applied to further refine the layer boundary. Polygon editing (in a frame and across frames) and coherence matting can be alternatively performed with assistance from the user.

Local Geometry. When the viewpoint changes significantly, a single planar geometry may not be sufficient to achieve anti-aliased rendering, such as cameras $C1$ and $C2$ in Figure 9. Therefore, we introduce a local geometry representation in our system which allows each frame to have its own planar equation $L1$ and $L2$ as illustrated in Figure 9.

Using the same UI as in Section 4.3.1, the plane equation can be estimated for each frame. Our system allows the user to specify only a few key frames' geometry, and interpolates the plane equations for frames in between. Similar to the neighboring frames selection algorithm in the boundary propagation step, we can select two (for 1D camera array) or three (for 2D camera array) key frames for interpolation.



(a) (b)

Fig. 10. (a) The background mosaic operator uses the polygon lasso operator to segment the layer into regions. (b) The resulting background mosaic fills in many missing pixels in (a). Although (b) still has many missing pixels, it is enough for coherence matting of the foreground.

For the frontal plane model, we interpolate the depth of the plane. For the 3D plane model, we interpolate the plane orientation while keeping the intersecting line if using two key frames, or the intersecting point if using three key frames. Once the plane equation is estimated for each frame, the same rendering algorithm can be applied as in global geometry.

4.4 Constructing the Background

The algorithm of coherence matting in Section 3.3 assumes that the background for the uncertain regions (where matting is estimated) is known. A key observation is that because the uncertain regions are located around foreground boundaries, they can only appear on neighboring frames in the light field where these regions are disoccluded. Our background reconstruction algorithm fills the disoccluded region using (warped) pixels from neighboring frames.

Once the foreground is popped up, we obtain the background image by removing the foreground image, as shown in Figure 10(a). Moreover, the background boundary is eroded by a few pixels (typically two pixels) before constructing the background mosaic because a possible under-segmentation of the foreground may leave some mixed foreground pixels on the background around boundaries.

An automatic algorithm is designed to construct the background, which warps the neighboring images to fill the holes using the background layer's geometry. This method works well if the the background is well approximated by plane, for example, in Figure 1.

However, when the background contains objects with relatively large depth variation, we need to further subdivide the background layer into sub layers, each of which can be represented as one plane. As shown in Figure 10(a), a background layer is segmented manually into four sub layers using polygons. This time, the location of the polygon is not critical. Instead, the criterion here is to group the background boundaries into a better planar approximation.

The sub layers are propagated from the key frame, where the user specifies the division, to all other frames using the existing background layer geometry. This propagation requires less accuracy as long as it covers the same group of boundaries. The relative motion of the sub layer across frames is estimated hierarchically, starting from translation to affine, and from affine to perspective transform [Szeliski and Shum 1997]. Only the pixels visible in both frames are used to estimate parameters. Figure 10(b) shows the resulting mosaic. Note that we need not create a hole-free mosaic, as a few pixels surrounding the occlusion boundaries are adequate for coherence matting.

5. REAL-TIME RENDERING OF POP-UP LIGHT FIELD

An integral part of our UI is the real-time pop-up light field renderer which provides the user instant feedback on the rendering quality. It is based on previous light field and Lumigraph rendering systems [Gortler et al. 1996; Isaksen et al. 2000; Buehler et al. 2001]. Our rendering algorithm includes

three steps: (1) splitting a light field into layers, (2) rendering layers in back-to-front order, and (3) combining the layers.

5.1 Data Structure

The data structure used in our rendering algorithm is shown below.

```
struct PopupLightField {
    Array<CameraParameter>  cameras;
    Array<Layer>  layers;
};
struct Layer {
    Array<Plane>  equations;
    Array<Image>  images;
};
struct Image {
    BoundingBox  box;
    Array2D<RGBA>  pixels;
};
```

The pop-up light field keeps the camera parameters associated with all the input frames. Each layer in the pop-up light field has corresponding layered images, one for each frame. Each layered image has a corresponding plane equation, so as to represent the local geometry. If global geometry is applied to a layer, all equations are the same for images in this layer.

Since these corresponding layered images vary their shapes in different views, they are stored as an array of images on each layer. Layers can be overlapping in the pop-up light field and each layered image is modified independently by mosaicing and coherence matting. Therefore it is necessary to keep both the color and opacity of the images for each layer separately. Each layered image is stored as an RGBA texture image of the foreground colors with its opacity map, and a bounding box as well. The opacity (alpha value) of the pixel is zero when this pixel is out of the foreground.

5.2 Layered Rendering Algorithm

The scene is rendered layer-by-layer using texture-mapped triangles in back-to-front order. Then the layers are sequentially combined by alpha blending. Our rendering scheme is based on Heigl et al. [1999] and Buehler et al. [2001], but extended to multiple layers. The pseudo-code of our rendering algorithm is shown below.

```
ClearFrameBuffer()
T ←CreateRenderingPrimitives()
for all layers Layer from back to front do
        for all triangles △ ∈ T do
                SetupProjectiveTextureMapping(△)
                Render(△)
                BlendToFrameBuffer(△)
        end for
end for
```

After initializing a frame buffer, we generate a set of triangular polygons on which the original images are blended and drawn. We first project the camera positions onto the image plane and triangulate these projection points together with the image plane's four corner points into a set of triangles.

Then, we assign a triple of texture images $\{I_i\}_{i=1}^{3}$ to each triangle which are blended across the triangle when rendering. The blending ratio $\{w_i^k\}_{k=1}^{3} (0 \le w_i^k \le 1, \sum_{k=1}^{3} w_i^k = 1)$ for three images are also assigned

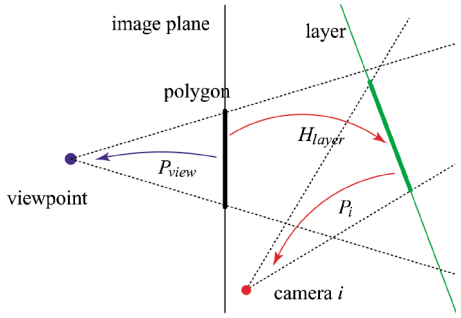


Fig. 11. Set-up of projective texture mapping.

to each of the three vertices and linearly interpolated across the triangle. The exact blending ratio based on ray angles is not necessarily distributed linearly on the screen. If the size of a triangle is not small enough with respect to the screen size, we subdivide the triangle into four triangles iteratively. On the vertex which is the projection of a I_i's camera, the blending ratio w_i^k is calculated using the following equation.

$$\begin{aligned} w_i^k &= 1 \quad \text{if camera } i \text{ is projected onto the } k\text{th vertex} \\ &= 0 \quad \text{otherwise.} \end{aligned}$$

For the vertex which is not the projection of a camera, the weights are calculated using the angle between the ray through the camera and the ray through the vertex [Buehler et al. 2001].

Then, each layer is rendered by blending texture images $\{I_i\}$, using blending ratios $\{w_i^k\}$. At the point other than the vertices on the triangle, the blending ratios $\{\tilde{v}_i\}$ are calculated by interpolating $\{w_i^k\}_{k=1}^3$. Using $\{I_i\}$ and $\{\tilde{v}_i\}$, the pixels on the triangle are drawn in the color $\sum_{i=1}^3 \tilde{v}_i I_i$.

The texture images are mapped onto each triangle projectively as illustrated in Figure 11. Let P_{view} be the projection matrix for the rendering camera (to produce the novel view), P_i be the projection matrix for the camera corresponding to I_i, and H_{layer} be a planar homography from the triangle to the layer plane. Then, the texture image I_i is mapped onto the triangle using a projection matrix $P_i H_{layer}$.

5.3 Hardware Implementation

Light field rendering can be accelerated using graphics hardware. Although several hardware-accelerated light field rendering approaches have been proposed [Gortler et al. 1996; Isaksen et al. 2000; Heigl et al. 1999; Buehler et al. 2001], we cannot use them directly for pop-up light field rendering. In these previous approaches, texture images are blended by multi-pass rendering of triangles and alpha-blending them in the frame buffer. In a layered rendering algorithm, we must alpha-blend layers using alpha values assigned in texture images, which means each layer must be rendered onto the frame buffer in a single pass.

One straightforward way is to copy the frame buffer into memory and composite them after rendering each layer. We implemented this method and found that it is too slow for interactive usage. Instead, we have developed a single-pass rendering method using multitexture mapping and programmable texture blending, which is available on modern graphics hardware.

In order to blend all textures on a single triangle, we first bind three different textures assigned to each triangle, then assign three blending ratios $\{w_1, w_2, w_3\}$ as the primary color {R, G, B} on each vertex. The primary color is smoothly interpolated on the triangle. Hence, the interpolated blending ratios $\{\tilde{v}_i\}$ are obtained simply by referring to the primary color at an arbitrary point on the triangle.

Table I. Performance of Rendering

Data	Resolution (w × h × #cameras)	Size of Original LF (MB)	Size of Pop-up LF (MB [#layers])	Frames per second
Tsukuba	384 × 288 × 25	8.3	19.4 [5]	62.5
Plaza	640 × 486 × 16	14.9	38.3 [16]	58.8
Pokemon	512 × 384 × 81	47.8	117.8 [5]	31.3
Fur	1136 × 852 × 23	66.8	140.4 [5]	46.9
Statuette	1136 × 852 × 43	124.8	161.2 [4]	37.1

Table II. User Interaction

Data	Tsukuba	Plaza	Pokemon	Fur	Statuette
# frame	25	16	81	23	43
# layer	4	16	5	5	4
points/frame	129	379	90	167	47
key points/frame	6.3	62.4	8.6	16.1	12.3
Time (hours)	≈0.5	≈4	≈1	≈1	≈1.5

Then the texture images on the triangle can be blended using the blending equation programmed in the pixel shader in graphics hardware.

The layers can be composed simply by alpha-blending each triangle on the frame buffer when it is rendered because the triangles are arranged without overlap in a layer and each triangle is drawn in a single pass.

Our rendering system has been implemented using OpenGL and its extensions for multi-texturing and per-pixel shading, and tested on a PC (CPU 660 MHz, memory 768 MB) equipped with an NVidia GeForce4 or ATI Radeon9700 graphics card with 128 MB of graphics memory. The performance of rendering is shown in Table I.

6. EXPERIMENTAL RESULTS

We have constructed several pop-up light fields from several real scenes. The “Tsukuba” data set and the “Plaza” sequence are courtesy of Prof. Ohta of University of Tsukuba, and Dayton Taylor, respectively. “Pokemon” data is captured by a computer-controlled vertical X-Y table in our lab. Data sets of “Statuette” (with unstructured camera motion) and “Furry toys” (with the camera moving along a line) are captured by a Canon G2 Digital Camera.

As shown in Table I, rendering of all the pop-up light fields can be done in real-time (with a frame rate greater than 30). Table II summarizes the amount of work required to construct these pop-up light fields. For most scenes in our experiments, it takes a couple of hours for a graphics graduate student in our lab to interactively model the pop-up light field. It, however, took the student 4 hours to construct the pop-up light field from the Plaza sequence where 16 layers are segmented.

Tsukuba. Some rendering results of the Tsukuba 5 × 5 light field are shown in Figure 1. It is demonstrated that with 4 layers, an anti-aliased rendering can be achieved. In our experiment, we have found that the same rendering quality can be achieved with 7 layers if the light field is down-sampled to 3 × 3.

Plaza. Figure 12 shows an aliasing-free novel view we rendered using the pop-up light field constructed from the Plaza sequence, which is a collection of only 16 images. The sequence was captured by a series of “time-frozen cameras” arranged along a line or curve. Because the scene is very complex, stereo reconstruction is very difficult. Note that nearly perfect matting is achieved for the floating papers in the air. The boundaries for the foreground characters are visually acceptable, made possible mainly by the coherent layers produced by coherence matting.

Fig. 12. Result of pop-up light field rendering of the Plaza sequence rendered from a novel viewpoint (in the position midway between the 11th and 12th frames). The input consists of only 16 images. We have used 16 layers to model the pop-up light field.

Fig. 13. Results on Pokemon 9 × 9: comparison of conventional light field rendering and pop-up light field rendering.

Pokemon. Figure 13 again demonstrates the progressive improvement of visual quality when more layers are popped up. With 5 layers, anti-aliased rendering of pop-up light fields (Pokemon 9 × 9) is achieved. The four layers that model the three toys and the background use frontal-parallel planes while the table plane is slanted.

Statuette. For complicated scenes, instead of using a global planar surface defined in the world coordinate system, local geometry should be used. Figure 14 shows the rendering result from a sequence of 42 images taken with unstructured camera motion. If a global planar surface is set as a frontal-parallel plane in the frame (Figure 14(a)), rendering at a very different viewpoint will have noticeable artifacts, as shown in Figure 14(b). Figure 14(c) shows a good rendering result using view-dependent geometry. Specifically, we have changed the plane orientations for different views.

Furry Rabbit. To show the efficacy of our coherence matting methodology, we use a sparse light field that captures 23 images of a Furry Rabbit with the camera path along a line. Figure 15 compares the

(a) (b) (c)

Fig. 14. Results on sparse images taken with unstructured camera positions. (a) The global planar surface is set as a frontal-parallel plane in this view, (b) rendering result from another view with the global plane, (c) rendering result from the same view of (b) with local geometry.

Fig. 15. Comparison of results on furry rabbit, with video matting (middle image) and with coherence matting (right image). The alpha matte from coherence matting is smoother that the one from video matting in the rendering image.

results with video matting and with coherence matting. The zoomed-up views of the left ear demonstrate that coherence matting obtains a more consistent matte than video matting.

7. DISCUSSION

Motivation. Our work was inspired by the real-time 3D model acquisition system of Rusinkiewicz et al. [2002], in which the user specifies areas that need to be modeled depending on the current merge model from multiple scans. Though the goal and methodology are very different, the key concept in our system is similar: our user is in the modeling loop and specifies, through a real-time pop-up light field renderer, where aliasing occurs and how the scene should be further modeled.

Another motivation stems from our frustration with not being able to get accurate per-pixel depth using stereo or other vision techniques. Pop-up light field is an image-based modeling technique that does not rely on accurate 3D depth/surface reconstruction. Rather, it is based on accurate layer extraction/segmentation in the light field images. In a way, we trade a difficult correspondence problem in 3D reconstruction for another equally difficult segmentation problem. However, for a user, it is much easier to specify accurate contours in images than accurate depth for each pixel.

Coherence Matting vs. Prefiltering. In a way, coherence matting is similar to prefiltering the alpha channel in the foreground layer. It was suggested in Levoy and Hanrahan [1996] and Chai et al. [2000] that prefiltering can be used to reduce aliasing at the expense of lower rendering resolution. With coherence matting, we do not need to prefilter the entire light field. It is adequate to prefilter only occlusion boundaries.

Moreover, we handle background and foreground layers differently in pop-up light field. Because we have multiple images, we can construct a complete background mosaic from under-segmented background layers. For the foreground image, we prefilter (by coherence matting) the pixels around the boundary before rendering. The rendering artifact is that the boundary of the foreground image is slightly blurred.

Limitations of Our Approach. Our pop-up light field takes on sparse light field as input. It cannot handle specular highlights and other significant appearance changes. It also assumes each layer itself can be rendered correctly by the sparse light field. Our method cannot guarantee an alias-free result if the layer is too complicated, such as a tree that cannot be segmented accurately, or a single object that violates plenoptic sampling criterion. Our layer-based approach does not handle topological changes properly. Moreover, coherence matting does not work well for semi-transparent surfaces and long hairs because the prior $L(\alpha)$ used in our formulation is approximately modeled as a point spread function.

REFERENCES

BAKER, H. AND BOLLES, R. 1989. Generalizing epipolar-plane image analysis on the spatiotemporal surface. *Int. J. Comput. Vis. 3*, 1, 33–39.

BAKER, S., SZELISKI, R., AND ANANDAN, P. 1998. A layered approach to stereo reconstruction. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'98)*. IEEE Computer Society Press, Los Alamitos, Calif., 434–441.

BERMAN, A., DADOURIAN, A., AND VLAHOS, P. 2000. Method for removing from an image the background surrounding a selected object. *U. S. Patent 6*, 134, 346.

BUEHLER, C., BOSSE, M., MCMILLAN, L., GORTLER, S., AND COHEN, M. 2001. Unstructured lumigraph rendering. In *Proceedings of ACM SIGGRAPH'01*. ACM, New York, 425–432.

CHAI, J.-X., TONG, X., CHAN, S.-C., AND SHUM, H.-Y. 2000. Plenoptic sampling. In *Proceedings of ACM SIGGRAPH'00*. ACM, New York, 307–318.

CHUANG, Y.-Y., AGARWALA, A., CURLESS, B., SALESIN, D. H., AND SZELISKI, R. 2002. Video matting of complex scenes. In *Proceedings of ACM SIGGRAPH'02*. ACM, New York, 243–248.

CHUANG, Y.-Y., CURLESS, B., SALESIN, D. H., AND SZELISKI, R. 2001. A Bayesian approach to digital matting. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'01)*. II:264–271.

CRIMINISI, A., REID, I., AND ZISSERMAN, A. 1999. Single view metrology. In *Proceedings of 7th International Conference on Computer Vision (ICCV'99)*. ACM, New York, 434–441.

DEBEVEC, P. E., TAYLOR, C. J., AND MALIK, J. 1996. Modeling and rendering architecture from photographs: a hybrid geometry- and image-based approach. *Proceedings of ACM SIGGRAPH'96*. ACM, New York, 11–20.

DÉCORET, X., DURAND, F., SILLION, F., AND DORSEY, J. 2003. Billboard clouds for extreme model simplification. In *Proceedings of the ACM Siggraph 2003*. ACM, New York.

GORTLER, S. J., GRZESZCZUK, R., SZELISKI, R., AND COHEN, M. F. 1996. The lumigraph. In *Proceedings of ACM SIGGRAPH'96*. ACM, New York, 43–54.

HEIGL, B., KOCH, R., POLLEFEYS, M., DENZLER, J., AND GOOL, L. V. 1999. Plenoptic modeling and rendering from image sequences taken by a hand-held camera. In *Symposium fur Mustererkennung*. 94–101.

HORN, B. K. P. AND BROOKS, M. J. 1989. *Shape From Shading*. MIT Press, Cambridge, Mass.

HORRY, Y., ANJYO, K.-I., AND ARAI, K. 1997. Tour into the picture: Using a spidery mesh interface to make animation from a single image. In *Proceedings of ACM SIGGRAPH'97*. ACM, New York, 225–232.

ISAKSEN, A., MCMILLAN, L., AND GORTLER, S. J. 2000. Dynamically reparameterized light fields. In *Proceedings of ACM SIGGRAPH'00*. ACM, New York, 297–306.

KANG, S. B., SZELISKI, R., AND CHAI, J. 2001. Handling occlusions in dense multi-view stereo. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'01)*. IEEE Computer Society Press, Los Alamitos, Calif., 103–110.

KE, Q. AND KANADE, T. 2002. A robust subspace approach to layer extraction. In *Proceedings of the IEEE Workshop on Motion and Video Computing*. IEEE Computer Society Press, Los Alamitos, Calif., 233–242.

KOLMOGOROV, V. AND ZABIH, R. 2002. Multi-camera scene reconstruction via graph cuts. In *Proceedings of the European Conference on Computer Vision*. 82–97.

LENGYEL, J. AND SNYDER, J. 1997. Rendering with coherent layers. In *Proceedings of ACM SIGGRAPH'97*. ACM, New York, 233–242.

LEVOY, M. AND HANRAHAN, P. 1996. Light field rendering. In *Proceedings of ACM SIGGRAPH'96*. ACM, New York, 31–42.

MATUSIK, W., BUEHLER, C., RASKAR, R., GORTLER, S. J., AND MCMILLAN, L. 2000. Image-based visual hulls. In *Proceedings of ACM SIGGRAPH'00*. ACM, New York, 369–374.

MATUSIK, W., PFISTER, H., NGAN, A., BEARDSLEY, P., ZIEGLER, R., AND MCMILLAN., L. 2002. Image-based 3d photography using opacity hulls. In *Proceedings of ACM SIGGRAPH'02*. ACM, New York, 427–437.

MISHIMA, Y. 1993. Soft edge chroma-key generation based upon hexoctahedral color space. *U.S. Patent 5,355,174*.

MOK, B., CHEN, M., DORSEY, J., AND DURAND, F. 2001. Image-based modeling and photo editing. In *Proceedings of ACM SIGGRAPH'01*. ACM, New York, 433–442.

PORTER, T. AND DUFF, T. 1984. Compositing digital images. *Comput. Graph. 18*, 3 (July), 253–259.

RADEMACHER, P. 1999. View-dependent geometry. In *Proceedings of ACM SIGGRAPH'99*. ACM, New York, 439–446.

RUSINKIEWICZ, S., HALL-HOLT, O., AND LEVOY, M. 2002. Real-time 3d model acquisition. In *Proceedings of ACM SIGGRAPH'02*. ACM, New York, 438–446.

RUZON, M. A. AND TOMASI, C. 2000. Alpha estimation in natural images. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'00)*. IEEE Computer Society Press, Los Alamitos, Calif., 18–25.

SCHAUFLER, G. 1995. Dynamically generated impostors. In *Proceedings of the GI Workshop on Modeling, Virtual Worlds, Distributed Graphics*. 129–135.

SCHIRMACHER, H., HEIDRICH, W., AND SEIDEL, H.-P. 2000. High-quality interactive lumigraph rendering through warping. In *Graphics Interface*. 87–94.

SEITZ, S. AND KUTULAKOS, K. 1998. Plenoptic image editing. In *Proceedings of 7th International Conference on Computer Vision (ICCV'98)*. 17–24.

SHADE, J., GORTLER, S., WEI HE, L., AND SZELISKI, R. 1998. Layered depth images. In *Proceedings of ACM SIGGRAPH'98*. ACM, New York, 231–242.

SMITH, A. R. AND BLINN, J. F. 1996. Blue screen matting. In *Proceedings of ACM SIGGRAPH'96*. ACM, New York, 259–268.

STEWART, J., YU, J., GORTLER, S., AND MCMILLAN, L. 2003. A new reconstruction filter for undersampled light fields. In *Eurographics Symposium on Rendering, 2003*.

SZELISKI, R. AND SHUM, H.-Y. 1997. Creating full view panoramic image mosaics and environment maps. In *Proceedings of ACM SIGGRAPH 1997*. ACM, New York, 251–258.

WANG, J. AND ADELSON, E. 1994. Representing moving images with layers. *IEEE Trans. Image Proc. 3*, 233–242.

WEXLER, Y., FITZGIBBON, A., AND ZISSERMAN., A. 2002. Image-based environment matting. In *Proceedings of the 13th Eurographics Workshop on Rendering*, Italy.

YAMAZAKI, S., SAGAWA, R., KAWASAKI, H., IKEUCHI, K., AND SAKAUCHI, M. 2002. Microfacet billboarding. In *Proceedings of the 13th Eurographics Workshop on Rendering*. 175–186.

ZHANG, Z., WANG, L., GUO, B., AND SHUM, H.-Y. 2002. Feature-based light field morphing. In *Proceedings of ACM SIGGRAPH'02*. ACM, New York, 457–464.

Received May 2003; revised October 2003, February 2004; accepted February 2004