Scam Light Field Rendering

Jingyi Yu Leonard McMillan
Laboratory of Computer Science
Massachusetts Institute of Technology
{jingyi,mcmillan}@graphics.lcs.mit.edu

Steven Gortler
Division of Engineering and Applied Science
Harvard University
sjg@graphics.lcs.mit.edu

Abstract

In this paper we present a new variant of the light field representation that supports improved image reconstruction by accommodating sparse correspondence information. This places our representation somewhere between a pure, two-plane parameterized, light field and a lumigraph representation, with its continuous geometric proxy. Our approach factors the rays of a light field into one of two separate classes. All rays consistent with a given correspondence are implicitly represented using a new auxiliary data structure, which we call a surface camera, or ***scam****. The remaining rays of the light field are represented using a standard two-plane parameterized light field. We present an efficient rendering algorithm that combines ray samples from scams with those from the light field. The resulting image reconstructions are noticeably improved over that of a pure light field.*

1. Introduction

Light fields are simple and versatile scene representations that are widely used for image-based rendering [11]. In essence, light fields are simply data structures that support the efficient interpolation of the radiance estimates along specified rays. A common organization for light fields is a two-plane parameterization in which the intersection coordinates of a desired ray on two given planes determines the set of radiance samples used to interpolate an estimate. A closely related representation to a light field is the lumigraph [8]. A lumigraph incorporates an approximate geometric model, or proxy, in the interpolation process, which significantly improves the quality of the reconstruction. Unfortunately, every desired ray must intersect some point on the geometric proxy in order to estimate its radiance in a lumigraph. Thus, a continuous, albeit approximate, scene model is required for lumigraph rendering. Acquiring an adequate scene model for lumigraph rendering can be difficult in practice. In fact, most lumigraphs have been limited to scenes composed of a single object or a small cluster of scene elements. The geometric scene proxy used by a lumigraph can be created using computer vision methods or with a 3-D digitizer. Geometric information about the scene is important for eliminating various reconstruction artifacts that are due to undersampling in light fields. An analysis of the relationship between image sampling density and geometric fidelity was presented by [4]. Chai, et al, presented a formal bound on the accuracy with which a geometric proxy must match the actual geometry of the observed scene in order to eliminate aliasing artifacts in the image reconstruction. As with the lumigraph model they assume that a geometric proxy can be identified for any requested ray.

Acquiring dense geometric models of a scene has proven to be a difficult computer vision problem. This is particularly the case for complicated scenes with multiple objects, objects with complicated occlusion boundaries, objects made of highly reflective or transparent materials, and scenes with large regions free of detectable textures or shading variations. The wide range of depth extraction methods that have been developed over the past 40 years, with the objective of extracting geometric models, have met with only limited success. Even with the recent development of outward-looking range scanners it is still difficult to create a dense scene model. However, both passive stereo and active range scanners are usually able to establish the depth or correspondence of a sparse set of scene points with a reasonably high confidence. The primary objective of this research is to incorporate such sparse geometric knowledge into a light field reconstruction algorithm in an effort to improve the reconstruction of interpolated images.

Our light field representation factors out those radiance samples from a light field where correspondence or depth information can be ascertained. We introduce a new data structure that collects all of those rays from a light field that are directly and indirectly associated with a 3D point correspondence. This data structure stores all of the light field rays through a given 3D point, and, there

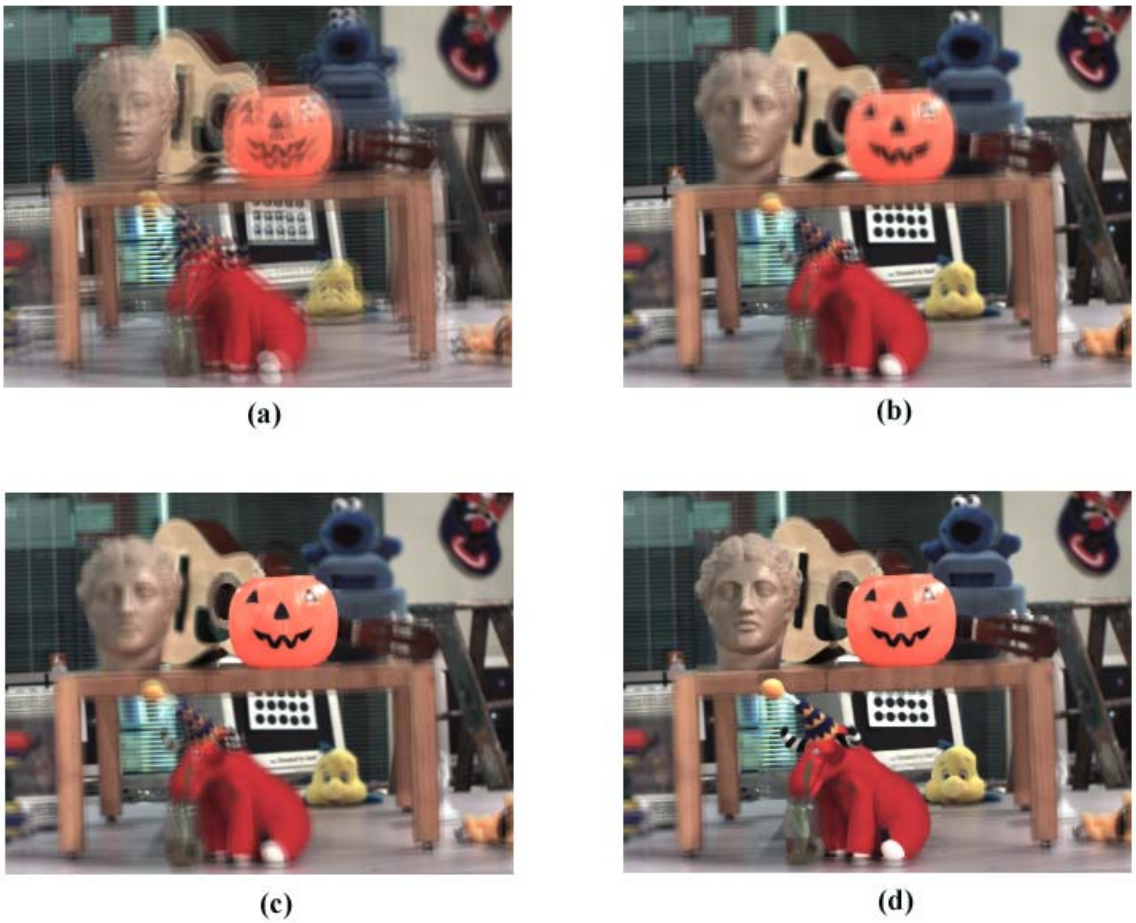


Figure 1: Light field rendering and scam rendering: (a) light field rendering with zero disparity; (b) light field rendering with optimal disparity; (c) scam rendering with correspondences of the pumpkin head; (d) scam rendering with correspondences of multiple objects in the scene;

fore, it is similar to a pinhole camera anchored at the given correspondence. Since this virtual pinhole camera is most often located at a surface point in the scene, we call it a surface camera, or *scam* for short. Once the rays associated with a scam are determined, they can be removed from the light field. We call this partitioning of rays into scams a *factoring* of the light field. Ideally, every ray in a light field would be associated with some scam, and, thus, we would call it fully factored. The resulting scam light field would generate reconstructions comparable to those of a lumigraph, although the two representations would be quite different. The utility of a scam renderer lies in its ability to improve light field reconstructions with a set of scams that are a small subset of a fully factored light field. In this paper we describe a new light field representation composed of a collection of implicit scam data structures, which are established by sparse correspondence information, and an associated light field, which is used to interpolate rays for those parts of the scene where no scam information has been established. We describe how to factor all of the rays associated with a specified scam from a light field when given as few as two rays from the scam (i.e. a correspondence) or the depth of a single known point. We then describe the necessary bookkeeping required to maintain the scams and light field representations. Next we describe an efficient two-pass rendering algorithm that incorporates scam information, and thus, sparse correspondence information, to improve light field reconstructions. Finally, we show results of light field renderings using our new representation with varying degrees of geometric sparseness.

2. Previous work and background

Early image-based rendering models such as 3D warping [13], view interpolation [5] and layered-depth images [15], etc, consist of a set of images of a scene and their associated depth maps. The rendering quality relies heavily on the correctness of these depth maps, which are usually derived from stereo algorithms. Light field ren-

dering [11] synthesizes new views by interpolating a set of densely sampled images without associated depth information. When undersampled, the light field rendering exhibits aliasing artifacts, as is shown in Figure 1(a). Plenoptic sampling [4] analyzed the relationship between the sample rate and the geometrical information. Plenoptic sampling also suggested that one can minimize the aliasing artifacts by placing the object or focal plane at a distance from the camera plane that is consistent with the scene's the average disparity, as shown in Figure 1(b). Lumigraph rendering [8] addresses the issue of sparse sampling by introducing geometric information and showed that the rendering quality is significantly improved with approximate geometry proxies even in a sparsely sampled light field. Our work assumes that it is difficult to provide dense sampled geometric proxies, or dense correspondences. We design an algorithm to render views using a sparse set of correspondences and a sparsely sampled light field.

In conventional light fields, a two parallel plane parameterization is commonly used to represent rays, where each ray is parameterized in the coordinates of the camera plane (*s, t*) and a image plane (*u, v*). Surface light fields [17] suggested an alternative ray parameterization where rays are parameterized over the surface of a prescanned geometry model. We combine both parameterizations in our algorithm.

For simplicity, we assume uniform sampling of the light field and use the same camera settings as Gu et al [9], where the image plane lies at $z = -1$ and the camera plane lies at $z = 0$. This leads to the parameterization of all rays passing through point (p_x, p_y, p_z) as

$$\mathbf{r} = (0,0,-p_x/p_z,-p_y/p_z) + s\cdot(1,0,1+1/p_z,0) + t\cdot(0,1,0,1+1/p_z) \quad (1)$$

We use a slightly different parameterization of [9]; we parameterize each ray as the 4-turple (*s, t, u, v*), where (*u, v*) is the pixel coordinate in camera (*s, t*). This simplifies the ray equation (1) to

$$\mathbf{r}(s,t,u,v) = (0,0,-p_x/p_z,-p_y/p_z) + s\cdot(1,0,1/p_z,0) + t\cdot(0,1,0,1/p_z) \quad (1')$$

The ray-point equations (1) and (1') indicate that all rays passing through the same 3D point lie on an *s-t* plane in the 4D ray space, which we call the point's *constraint plane*.

In a calibrated setting, each correspondence identifies a unique 3D point; therefore, it also identifies a constraint plane. Our goal is to first factor all of the rays associated with the constraint plane of each correspondence using a special data structure called "surface camera" or *scam*. We then suggest an algorithm that synthesizes new views from these scams using a reconstruction algorithm similar

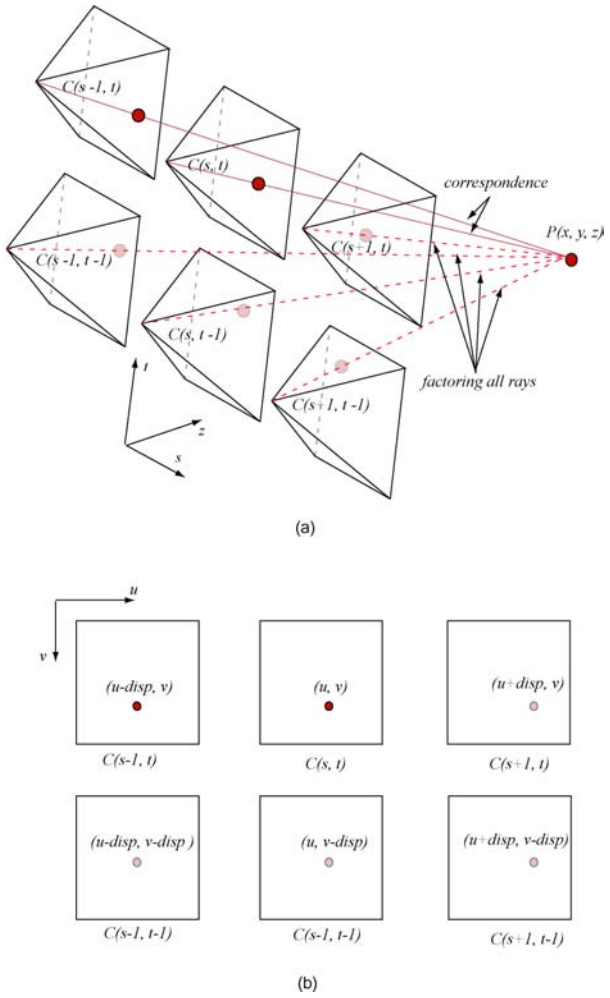


Figure 2: Factoring scam: (a) a correspondence is specified as two points and can be factored to all cameras by back projection the 3D point; (b) by normalizing the a correspondence with unit disparity, we can factoring all rays associated with a scam.

to the rebinning approach described for unstructured cameras in the lumigraph. For desired rays that are not interpolated by any scam, we use the light field approach to render them. In addition, we present an interactive rendering system that allows users to provide or remove correspondences and re-renders the view in real-time. Figure 1(c) and 1(d) illustrates the various rendering results using the scam rendering algorithm with correspondence information from multiple objects.

3. Scam factoring

Correspondences can always be specified as scalar disparities along epipolar lines. We will assume that all source mages have to be rectified such that their epipolar

planes lie along pixel rows and columns. In this setting disparities can be describe the horizontal and/or the vertical shifts between the corresponding pixels of image pairs, as is shown in Figure 2. Each correspondence is represented as two rays $r_1(s_1,t_1,u_1,v_1)$ and $r_2(s_2,t_2,u_2,v_2)$, which pass through the same 3D point. Assuming uniform sampling, and applying the constraint plane equation (1'), we have

$$\frac{u_2-u_1}{s_2-s_1}=\frac{v_2-v_1}{t_2-t_1}=\frac{1}{p_z}=disp \qquad (2)$$

where *disp* is the disparity of a correspondence. We can rewrite the point's constraint plane equation (1) in terms of its disparity as

$$\mathbf{r}=(0,\,0,\,u_0,\,v_0) + s\cdot(1,\,0,\,disp,\,0) + t\cdot(0,\,1,\,0,\,disp) \qquad (3)$$

This construction transforms each correspondence into a constraint plane defining a set of 4-dimensional rays. The values of u_0 and v_0 can be determined directly from the correspondence rays, $\boldsymbol{r_1}$ and $\boldsymbol{r_2}$. All other rays can be factored from the given light field by setting the values of *s* and *t* and solving for the appropriate *u* and *v* values consistent with the constraint plane of the correspondence. The constraint plane solutions at integer values of *s* and *t* are equivalent to placing a virtual camera at the 3D point and computing rays from that point through the camera centers lying on the camera plane, as is shown in Figure 2(a). We implicitly store constraint plane as an image parameterized over the same domain as the camera plane. We call this image a "surface camera" or *scam*. To index the rays of a scam, we solve the disparity equation (3) for all data camera locations shown in Figure 2(b). Because these rays do not necessarily pass through the pixels (samples) of the data cameras, we bilinearly interpolate their color in the data image. The complete factoring algorithm is shown as follows:

Generate scam for each correspondence
for each correspondence *S* **do**
  normalize *S* in form (u_0 , v_0 , *disp*)
  for each data camera *C(s, t)* **do**
    calculate the projection (*u, v*) of *S* in camera *C(s, t)* from the disparity equation (3)
    bilinearly interpolate *P*(*u, v*) in *C(s, t)*
    store *P* as pixel *(s, t)* in $scam_s$
  end for
end for

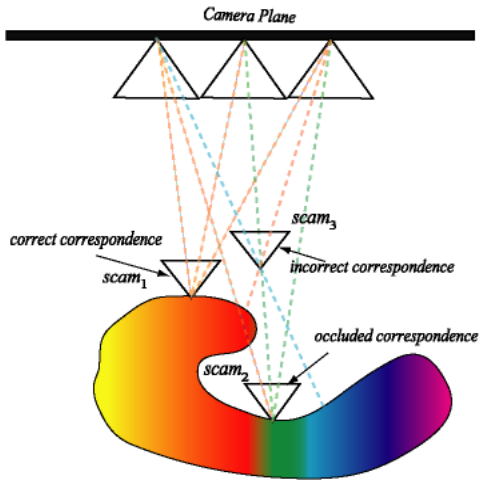


Figure 3: $scam_1$ is close to the real surface and is not occluded by another other parts of the scene; $scam_2$ is occluded by other parts of the scene; $scam_3$ is from the incorrect correspondence and is far away from the real surface.

4. Scam representation

Scam images can be used to analyze the accuracy of correspondences, the presence of occlusions, and the consistency of the correspondence's surface reflectance. If a correspondence is not close to the real surface, then we expect to observe greater variations in its scam images, as shown in $scam_3$ of Figure 3 and Figure 4(b). If a correspondence is near the actual surface and it is not occluded by other parts of the scene, then its scam image indicates the radiance received at all data cameras and thereby represents the surface's local reflectance radiance, shown as $scam_1$ of Figure 3. Moreover, if the surface is Lambertian, then these scams are expected to have constant color everywhere, as is shown in Figure 4(c). If the surface's reflectance exhibits view dependencies such as specular highlights, we expect to observe smooth radiance variations over the scam images. Figure 4(d) shows the scam of the specular highlight on the pot in Figure 4. When a correspondence lies close to the occlusion boundary of an object, then we expect to see specific abrupt color transitions in its scam image. Rays that are not occluded should have consistent colors, while occluded rays might exhibit significant color variations and discontinuities, as shown in $scam_2$ of Figure 3 and Figure 4(a). Since we bilinearly interpolate each scam image, we model the scene with "smooth occlusion" by implicitly interpolating between points on either side of the occlusion boundaries.

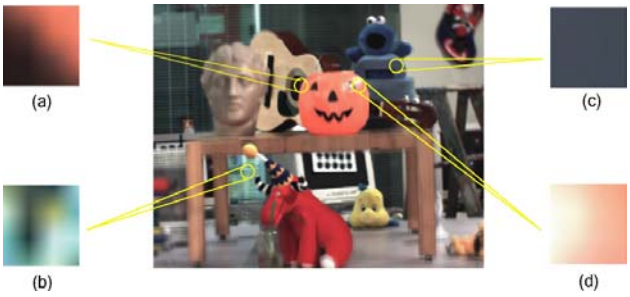


Figure 4: In the illustration above, scam (a) is close to the occlusion boundaries; scam (b) is away from the real surface; scam (c) is close to the real Lambertian surface and is not occluded; scam (d) is close to the specular highlight on the real surface;

Because correspondences are usually not very accurate, we can estimate a measure of the quality of correspondences by calculating the distribution and the variance of the colors within scams. The color distribution in incorrect correspondences should be discontinuous, non-compact, and its variance is expected to be high. For the correct correspondences and unoccluded surfaces, we expect to see more uniform and continuous color variations, and, therefore, low color variance. For correspondences on simple occlusion boundaries or with simple view dependencies, we can characterize them by modeling bimodal color distributions from their scam images. We will discuss how to use this measurement to improve the reconstruction in Section 5.

5. Scam rendering

In order to render a correspondence from an arbitrary view, we need to first project its scam onto the desired view. We do this by constructing a ray from the desired view that is consistent with the correspondence. We then use the scam data structure to interpolate the reflected radiance. The explicit coordinates of the 3D point are not needed to construct the ray. We can instead compute the intersection of the correspondence's constraint plane with the camera's image plane. This is particularly efficient with our representation.

5.1. Projecting correspondences

We describe all virtual cameras in form (*s', t', z'*), where *z'* is the distance to the data camera plane. If the camera is on the camera plane, i.e., at (*s', t', 0*), we can calculate the projection of the correspondence in the new view using the disparity equation (3) as

$$(u,\ v) = (u_0 + s' \cdot disp,\ v_0 + t' \cdot disp) \qquad (4)$$

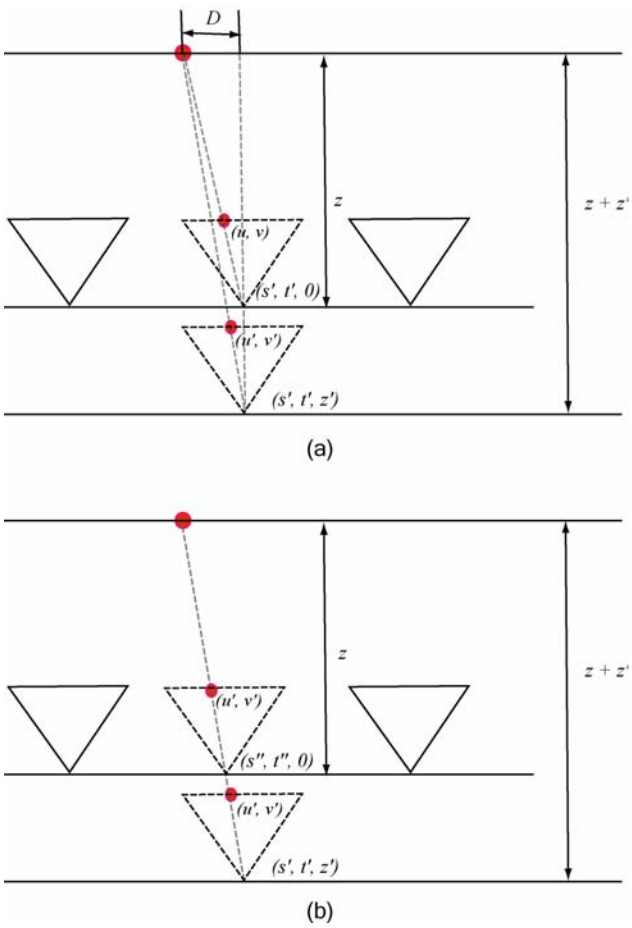


Figure 5: Projecting the correspondence in the new view: (a) we project the correspondence in camera (*s', t', z'*) by first projecting it in camera (*s', t', 0*) then calculating its position in camera (*s', t', z'*) using the geometric relationship; (b) we construct the ray that passes the correspondence from camera (*s', t', z'*) and compute its intersection with the original camera plane.

And we can query the color of the projected correspondence by interpolating point (*s, t*) in its scam image. To determine the projection of a correspondence in a camera (*s', t', z'*) off the camera plane, we first calculate its projection *(u, v)* in camera *C'(s', t', 0)*. Because *C'* is on the camera plane, we can simply calculate *(u, v)* as (4).

We then apply the geometry relationship as is shown in Figure 5(a) and use the depth-disparity equation (2), we have

$$\frac{u'}{u} = \frac{v'}{v} = \frac{D/(z+z')}{D/z} = \frac{z}{z+z'} = \frac{1}{1+z' \cdot disp} \qquad (5)$$

Therefore the projection of the correspondence in the new camera (*s', t', z'*) can be computed as

$$(u',\ v') = \left((u_0 + s' \cdot disp)/(1 + z' \cdot disp),\ (v_0 + t' \cdot disp)/(1 + z' \cdot disp) \right) \quad (6)$$

We then calculate the intersection point (*s", t", 0*) of the ray with the original camera plane. Notice the correspondence should project to the same pixel coordinates in both camera (*s', t', z'*) and (*s", t", 0*), as is shown in Figure 5(b). Therefore by reusing disparity equation (3), we can calculate (*s", t"*) as

$$\begin{aligned}(s'', t'') &= \left((u' - u_0)/disp,\ (v' - v_0)/disp \right) \\ &= \left((s' - z' \cdot u_0)/(1 + z' \cdot disp),\ (t' - z' \cdot v_0)/(1 + z' \cdot disp) \right)\end{aligned} \quad (7)$$

The color of the projected correspondence is then bilinear interpolated at (*s", t"*) in the scam image of the correspondence.

5.2. Color blending

Once we project all correspondences onto the new camera, we need to synthesize the image from these scattered projection points. We use a color-blending algorithm similar to unstructured lumigraph [2]. We assume correspondences are comparatively accurate and therefore its projection only influences a limited range of pixels around it in the new view. In practice, we only blend correspondences projected in a pixel's 1-ring neighborhood. If there isn't any, we then render the pixel directly from the light field. We use the following weight function to blend these correspondences:

$$\begin{aligned}weight(corresp\ i) = &\ metric_{smoothness}\ (scam\ i) \\ &+ metric_{distance}(corresp\ i) \\ &+ metric_{disparity}\ (corresp\ i)\end{aligned} \quad (8)$$

The first term of the weight evaluates the quality of the scam, where we use the color variance as a simple measurement, as is discussed in Section 4. The second term measures how close the projection is to the pixel, where closer projections get higher weight. The last term distinguishes closer correspondences from far away ones by their disparities. For a boundary pixel, there could be multiple correspondences with difference disparities around it. Since closer objects are expected to be more important, we assign larger weight to those of large disparities. Furthermore, we assign continuous metric functions for all terms to maintain the smooth transition from scam rendered parts to light field rendered parts. The complete two-pass rendering algorithm is shown as follows:

Synthesize view *C(s', t', z')*
for each correspondence *S* **do**
  calculate ray *r(s", t")* that passes *S* and *C(s', t', z')* using equation (7)
  interpolate *r* in the scam image of *S*
  calculate the projection *P(u', v')* of *S* in *C* using equation (6)
  compute the weight of *S* using equation (8) and add *S* to *P*'s 1-ring pixels' scam list
end for
for each pixel *P(u, v)* in the synthesized image **do**
  if *P*'s scam list is not empty **do**
    color blend all correspondences in *P*'s scam list with calculated weights
  end if
  else do
    use light field to render *P*
  end else
end for

6. Result

We have applied our algorithm to correspondences generated from a range of different stereo algorithms [7][3]. We have also developed a user-guided stereo algorithm in which the user to specifies image regions to be correlated because local correspondences are faster to generate and are more reliable and the user can focus their efforts on important features. We have tested our algorithm on varying degrees of sparseness and quality of the correspondences.

The pumpkin dataset shown in Figure 6 is constructed from a 4x4 sparse light field. We allow users to specify the regions to correlate and generate correspondences on the fly using a dynamic-programming based stereo algorithm [7]. Previous results have shown correspondences generated by [7] are not always reliable, however, because our color-blending algorithm is robust, we are able to synthesize high quality views from these set of low-fidelity correspondences. Figure 6(a) renders the new image using standard light filed rendering methods with the focal plane optimally placed at the depth associated with the average disparity as suggested by plenoptic sampling [4]. Aliasing artifacts are still visible because the light field is undersampled. We then add in correspondences for the head sculpture, the pumpkin, the fish, and part of the red stuffed animal by correlating rectangular regions using algorithm [7], as is shown in Figure 6(a). As a result, the reconstruction in the yellow regions of the synthesized image 6(b), are significantly improved. Boundaries between the light field rendered parts and scam rendered parts are also very smooth.

The office scene light field in Figure 7 illustrates the gradual improvements using scam rendering with more and more correspondences. Figure 7(a) is a synthesized view using light field rendering with the optimal disparity. Figure 7(b) improves the reconstruction of the red stuffed animal with scam rendering associating its correspondences. Figure 7(c) renders the scene with additional correspondences from different parts of the scene. Figure 7(d) reconstructs the reflected highlights on the background by providing correspondences of the background while these view dependencies are usually difficult to achieve in traditional light field rendering systems.

7. Conclusions and future work

In this paper we have presented a light field decomposition technique and an image-based rendering algorithm for light fields with a sparse collection of correspondences. We use a special data structure called a scam to store light field regions associated with correspondences and to accelerate interpolation of arbitrary rays in it with two-plane parameterization. We implemented the algorithm in an interactive and real-time system that allows users to aid in the assignment of new correspondences and quickly re-renders the view. We have tested our algorithm on correspondences with varying degrees of sparseness and show it is robust with low-fidelity correspondences. Our reconstructions are comparable to those of a lumigraph while it doesn't require complete geometric models.

For those parts of the image without accurate correspondence information, our method uses the traditional light field method for interpolating the radiance at the desired ray. As a result, in these regions, we expect to see aliasing artifacts due to under-sampling in the light field.

However, there are special cases where such artifacts are less apparent, in particular, in areas of low texture. Our method generates effective reconstructions in these regions where, one should note, it is also difficult to establish correspondences. Using traditional stereo vision methods, it is also difficult to establish accurate correspondence near occluding boundaries and on specular surfaces. However, if any high-confidence correspondence can be established from any image pair from the set of all light field images, our technique will generally provide reasonable reconstructions.

In the future, we would like to extend our scam representation as an alternative modeling method to the image-based and geometry-based approaches. We also want to study the surface radiance properties from scams by better characterizing their color variance and distributions. In addition, our scam rendering algorithm is closely related to the point-based rendering algorithms and we hope to investigate and apply relative techniques to improve scam rendering.

8. References

[1] E.H. Adelson and J. Bergen. The plenoptic function and the elements of early vision. In Computational Models of Visual Processing, pages 3-20. MIT Press, Cambreidge, MA, 1991

[2] C. Buehler, M. Bosse, L. McMillan, S.Gortler, and M.Cohen. Unstructured lumigraph rendering. Computer Graphics(SIGGRAPH '01), pages 425-432, August 2001.

[3] C. Buehler, S. Gortler, M. Cohen, L. McMillan. "Min Surfaces for Stereo." To appear in Proceedings of ECCV 2002.

[4] Jin-Xiang Chai, Xin Tong, Shing-chow Chan, and Heung-Yeung Shum. Plenoptic sampling. SIGGRAPH 00, pages 307-318.

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

[6] P. Debevec, C. Taylor, and J. Malik. Modeling and rendering architecture from photographs. SIGGRAPH96, pages 11-20.

[7] Olivier D. Faugeras. Three dimensional computer vision: A geometric Viewpoint, pages 198-200. The MIT Press.

[8] Steven J. Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F.Cohen. The lumigraph. SIGGRAPH96, pages 43-54.

[9] Xianfeng Gu, Steven J. Gortler, and Michael F. Cohen. Polyhedral geometry and the two-plane parameterization. Eurographics Rendering Workshop 1997, pages 1-12, Jun 1997.

[10] A. Isaksen, L. McMillan, and S. Gortler. Dynamically reparameterized lightfields. SIGGRAPH'00, pages 297-306.

[11] M. Levoy and P. Hanrahan. Lightfield rendering. SIGGRAPH96, pages 31-42.

[12] L. McMillan and G. Bishop. Plenoptic modeling: An image-based rendering system. Computer Graphics (SIGGRPH'95), pages 39-46, August 1995.

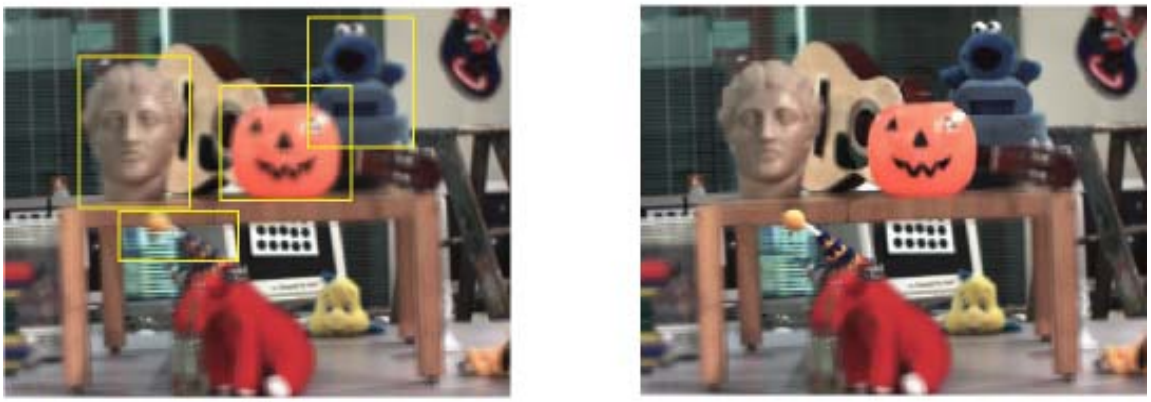
[13] W. Mark, L. McMillan, and G. Bishop. Post-rendering 3d warping. In Proc. symposium on I3D graphics, pages 7-16, 1997

[14] Hartmut Schirmacher, Wolfgang Heidrich, and Hans-Peter Seidel. Adaptive acquisition of lumigraphs from synthetic scenes. Computer Graphics Forum. 18(3):151-160, September 1999. ISSN 1067-7055

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

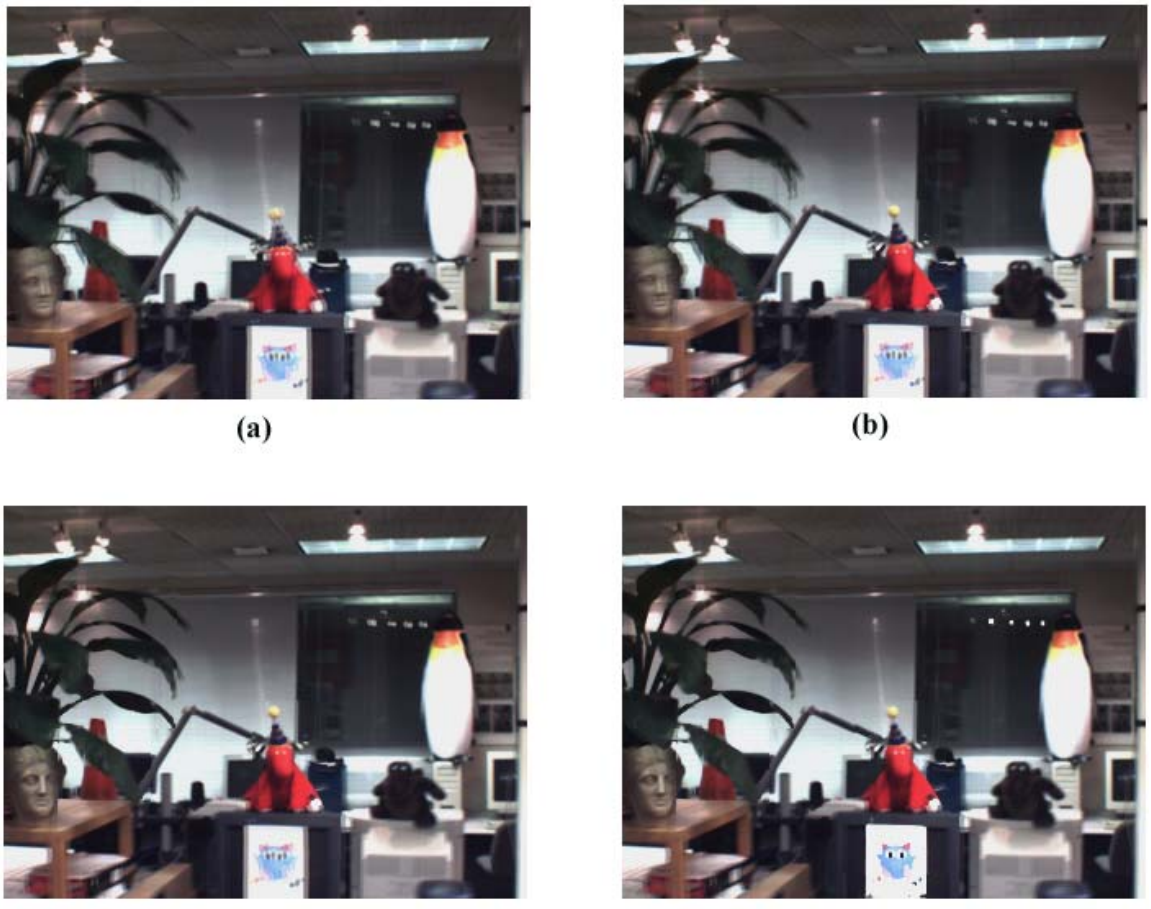
[16] Heung-Yeung Shum and Li-Wei He. Rendering with concentric mosaics. SIGGRAPH99. pages 299-306

[17] Daniel N. Wood, Danieal I. Azuma, Ken Aldinger, Brian Curless, Tom Duchamp, David H. Salesin, and Werner Stuetzle. Surface light fields for 3d photography. SIGGRAPH00, pages 287-296



(a) (b)

Figure 6: Light field and scam rendering: (a) light field rendering with optimal disparity; (b) scam rendering view with correspondences for all yellow rectangle regions in (a) while rest of the scene rendered by light field with optimal disparity.



(a) (b)

(c) (d)

Figure 7: Scam rendering with view dependencies: (a) light field rendering with optimal disparity; (b) scam rendering with correspondences of the unicorn; (c) scam rendering with correspondences from multiple objects; (d) scam rendering further adds in correspondences of reflected highlights.