Scam Light Field Rendering

by

Jingyi Yu

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of
Master of Science in Electrical Engineering and Computer Science
at the
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

January 2003

© Jingyi Yu. All rights reserved.
The author hereby grants to MIT permission to reproduce and distribute publicly paper and electronic copies of this thesis document in whole or in part.

Author..............................................................
Department of Electrical Engineering and Computer Science
January 30, 2003

Certified by..............................

Leonard McMillan
Associate Professor
Thesis Supervisor

Accepted by............
Arthur C. Smith
Chairman, Department Committee on Graduate Students
Scam Light Field Rendering

by

Jingyi Yu

submitted to the Department of Electrical Engineering and Computer Science
on January 30, 2003, in partial fulfillment of the
requirements for the degree of
Master of Science in electrical Engineering and Computer Science

Abstract

In this thesis 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.

Thesis Supervisor: Leonard McMillan
Title: Associate Professor
Contents

1 Introduction ............................................. 8

2 Background and Previous Work ...................... 12

3 Scam Parameterization ................................. 20
   3.1 Ray Parameterization .............................. 20
   3.2 Correspondence Parameterization ............... 22
   3.3 Scam Factoring .................................... 23

4 Scam Representation ..................................... 26
   3.4 Interpolate Scam Images ......................... 26
   3.5 Scam Classification ............................... 28
   3.6 Measure Scam Quality ............................ 30

5 Scam Rendering .......................................... 32
   5.1 Projecting Correspondences .................... 32
   5.2 Color Blending .................................... 34
   5.3 Implementation ................................... 37

6 Scam Generation ......................................... 38
   6.1 Correlate Scanlines under Epipolar Constraints 39
   6.2 Generating Correspondences between Scanlines . 41
      6.2.1 Uniform Shearing and Brensenham’s Algorithm 41
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.2.2 Dynamic Programming</td>
<td>43</td>
</tr>
<tr>
<td>6.3 Continuous Light Field Reconstruction</td>
<td>46</td>
</tr>
<tr>
<td>7 Results</td>
<td>50</td>
</tr>
<tr>
<td>8 Future Work and Conclusion</td>
<td>55</td>
</tr>
</tbody>
</table>
## List of Figures

<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 2-1</td>
<td>Light field rendering, system setup and rendering algorithm</td>
<td>14</td>
</tr>
<tr>
<td>Figure 2-2</td>
<td>Light field rendering algorithm</td>
<td>15</td>
</tr>
<tr>
<td>Figure 2-3</td>
<td>Light field rendering and scam rendering of a sparse light field</td>
<td>18</td>
</tr>
<tr>
<td>Figure 3-1</td>
<td>Two plane parameterization for light field</td>
<td>21</td>
</tr>
<tr>
<td>Figure 3-2</td>
<td>Correspondence parameterization</td>
<td>22</td>
</tr>
<tr>
<td>Figure 3-3</td>
<td>Factoring scam</td>
<td>24</td>
</tr>
<tr>
<td>Figure 4-1</td>
<td>Generating scam images</td>
<td>27</td>
</tr>
<tr>
<td>Figure 4-2</td>
<td>Comparison between scam images</td>
<td>29</td>
</tr>
<tr>
<td>Figure 4-3</td>
<td>Classifying scam images</td>
<td>29</td>
</tr>
<tr>
<td>Figure 5-1</td>
<td>Projecting the correspondence in the new view</td>
<td>33</td>
</tr>
<tr>
<td>Figure 5-2</td>
<td>Color blending scams</td>
<td>35</td>
</tr>
<tr>
<td>Figure 6-1</td>
<td>Correlating scanlines under Epipolar constraints</td>
<td>40</td>
</tr>
<tr>
<td>Figure 6-2</td>
<td>Brensenham’s Algorithm</td>
<td>41</td>
</tr>
<tr>
<td>Figure 6-3</td>
<td>Uniform disparity</td>
<td>42</td>
</tr>
<tr>
<td>Figure 6-4</td>
<td>Dynamic Programming</td>
<td>43</td>
</tr>
<tr>
<td>Figure 6-5</td>
<td>Comparing uniform disparity and dynamic programming</td>
<td>45</td>
</tr>
<tr>
<td>Figure 6-6</td>
<td>Triangulating two scanlines</td>
<td>47</td>
</tr>
</tbody>
</table>

5
Figure 6-7 Intercrossing of two correspondences in 4D space 48

Figure 7-1 Scam rendering system 52

Figure 7-2 Light field rendering and scam rendering for the toys scene 53

Figure 7-3 Light field rendering and scam rendering with view dependencies 54

Figure 8-1 Relationship between light field, scam and surface light field rendering 56
Acknowledgements

I would like to thank many people who helped with this research. First of all, I would like to thank my advisor Professor Leonard McMillan, whose mentoring, advising and friendship was most important to me while I worked on this project. To me, Professor McMillan is not only my research advisor but also my most sincere friend. I would also like to thank Professor Steven Gortler of Harvard University who was also instrumental in developing key sections of the scam light field rendering work in general. Without their help, I would never have been able to complete this work. Also their help in writing the Pacific Graphics 2002 paper [23] and presentation on this subject formed the textual basis for this thesis.

This work was funded through support NSF Career Grant CCR-9975859.

I would like to thank Junhua Shen, Weiwei Gao, Chang Zhou, Yile Ding, Jingyang Li, Jingou Huang, the MIT LCS Computer Graphics Group, my roommate Rui Fan and my family for their emotional support and friendship while I worked on this research. This support made it possible to keep going when things got especially difficult.

Once again, I would like to thank my most sincere friend and respectable teacher Professor Leonard McMillan, who most of all made this work possible.
Chapter 1

Introduction

Light fields are simple and versatile scene representations that are widely used for image-based rendering [13]. 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 [9]. 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, therefore, 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 and do not require to be accurate.

In this thesis, 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.

1.1 Contributions

The contributions of this thesis include:

(1) A novel data structure called “Surface CAMera” or scam, which is used to represent correspondences and to factor rays from a light field that are associated with the correspondences.

(2) A new image-based rendering algorithm using the rebinning and color blending techniques to render the light field using scams.

(3) An interactive system that allows users to specify or remove scams for real-time rendering.

(4) And finally a theoretical analysis of our approach with a comparison of continuous light field representation and rendering.

1.2 Thesis Overview

I will discuss previous work in image-based and computer vision in Chapter 2, including graphics approaches as light field and lumigraph rendering, as well as vision approaches as shape from shading and correspondence mapping. In chapter 3, I will show how to parameterize correspondences in the 4D light field spacing using two-plane parameterization and introduce of scam representation for correspondences. In chapter 4, I give a method to classify scams in term of accuracy, occlusion conditions and surface properties by measuring radiance distributions in the scam images. In chapter 5, I
describe a forward-mapped scam-rendering algorithm combined with backward-mapped light field rendering that uses rebinning and color-blending. In Chapter 6 I describe several stereo algorithms to generate scams and a theoretical comparison between scam-rendering and simplex-based continuous light field rendering. In Chapter 7 I demonstrates a real-time interactive rendering system that allows users to specify or remove scams and show our rendering results for several sparsely sampled light field where conventional image-based rendering algorithms fail. In Chapter 8 I discuss the possible extensions of our scam representation and scam-rendering algorithm and conclude the thesis.
Chapter 2

Background and Previous Work

This thesis examines new representations and rendering methods for the light field rendering. Previous researchers have examined methods to reduce aliasing artifacts in light field rendering by either densely sampling or providing accurate geometry proxies. In this chapter, I discuss the major issues that arise when rendering from a sparsely sampled light field and what previous researchers have done to avoid these problems.

In recent years, image-based modeling and rendering (IBMR) has become a popular alternative to conventional geometry-based computer graphics. In IBMR, a collection of reference images are used as the primary scene representation [9][14]. Early IBMR systems such as 3D warping [17], view interpolation [5] and layered-depth images [19], etc, rely heavily on the correctness of depth approximations from stereo algorithms. Previous work in IBMR has further shown that the quality of the resulting synthesized images depends on complicated interactions between the parameterization of the given ray space [10][14], the underlying sampling rate [4][14] and the availability of approximate depth information [2][9].

Light field rendering is a special form of image-based rendering. It synthesizes new views by interpolating a set of sampled images without associated depth information. Light field rendering relies on a collection of densely sampled irradiance measurements along rays, and require little or no geometric information about the described scene. Usually, these measurements are acquired using a series of pinhole camera images
acquired along the surface of a parameterized, two-dimensional, manifold, most often on a plane as is shown in Figure 2-1(a).

Standard light field assumes constant scene depth, which is called the focal plane. To render a new ray $r$, it first interests $r$ with the focal plane. Then it collects the closest rays to $r$ that passes through the camera and the intersection point. Finally light field rendering interpolates or blends $r$ from these neighboring rays, shown in Figure 2-1(b). In practice, simple linear interpolation method like bilinear interpolation is used to blend the rays.

If we treat rays as signals, the light field rendering is similar to reconstructing continuous signals from discrete samples. In signal processing, when signals consist of different frequencies, it requires a minimal sampling density, or the Nyquist's limit in order to correctly reconstruct the signal from samples. In the case of insufficient sampling, the reconstruction presents serious aliasing artifacts. In a light field, depth variations of the scene are similar to frequency variations of a signal. By assuming constant scene depth, the light field rendering is only able to correctly render parts of the scene that are close to the assumed depth. Or in other words, it only reconstructs one frequency band of the ray signal. As a result, when sparsely sampled, no matter what depth we assume, light field rendering incurs serious aliasing artifacts, as is shown in Figure 2-2(a) and 2-2(b).

Levoy and Hanrahan [14] suggested that light field aliasing could be eliminated with proper prefiltering. Prefiltering can be accomplished optically by using a camera whose aperture is at least as large as the spacing between cameras. Otherwise, prefiltering can be accomplished computationally by initially oversampling along the camera-spacing dimensions and then applying a discrete low-pass filter, which models a synthetic aperture. In practice, it is usually impractical to do oversampling since it requires cameras be positioned very close to each other or have large aperture. The other major issue is the large storage requirement since dense sampling means a tremendously large number of images. Therefore the only practical way to reducing aliasing in conventional light field is to use band-limited filtering, i.e., low pass filtering. However, low-pass filtering has the side effects of removing high frequencies like sharp edges and view-dependencies, and will incur undesirable blurriness in reconstructed images, as is shown in Figure 2-2(c) and 2-2(d).
Figure 2-1: light field rendering: system setup and rendering algorithm: (a) Conventional light field uses an array of cameras uniformly positioned on a plane [22]; (b) Light field assumes constant scene depth and it renders a new ray by intersecting the ray with the focal plane and back-traces and blends its closest neighboring rays.
Figure 2-2: Light field rendering algorithm: (a) assumes the scene lies at infinity; (b) assumes the scene lies at the optimal focal plane; (c) uses prefiltering on (a); (d) uses prefiltering on (b); (e) uses dynamically reparameterized light field (DRLF) and puts the focal plane on the red stuffed animal with a large aperture; (f) uses DRLF and puts the focal plane on the guitar with large aperture.
A special prefiltering technique to remove aliasing is the dynamic reparameterized light field [12]. The dynamic reparameterized light field techniques synthesize virtual aperture and virtual focus to allow for any particular scene element (depth) to be reconstructed without aliasing artifact. However only those scene elements near the assigned focal depth are clearly reconstructed and all other scene elements are blurred, as is shown in Figure 2-2(e) and 2-2(f). This kind of prefiltering hence has the undesirable side effect of forcing an a priori decision as to what parts of the scene can be clearly rendered thereafter. The introduction of clear and blurry regions of focus in a prefiltered light field is a direct result of the depth-of-field effects seen by a finite (non-pinhole) aperture. In addition, view dependencies like specular high lights will be reduced or even removed because such features are usually not presented in all data cameras or do not have consistent focal depth.

Plenoptic sampling [4] pointed out that aliasing in light field rendering is due to either insufficient sampling or insufficient geometrical information. By further analyzing the relationship between the sample rate and the geometrical information, authors of [4] suggested that one can minimize the aliasing artifacts of light rendering by placing the object or focal plane at a distance from the camera plane that is consistent with the scene's average disparity, as shown in Figure 2-3(b).

Plenoptic sampling also suggested that another efficient way to reduce aliasing is to provide more geometrical information, like the lumigraph rendering [9]. The lumigraph system has the similar setup as the light field but approximates the scene with certain geometry proxies. It showed that the rendering quality can be significantly improved with these geometry proxies with a sparsely sampled light field. To render a new ray \( r \), it first intersects the ray with the model and then blends rays that are close to it in all data cameras. The same idea is applied to arbitrary cameras setups like the unstructured lumigraph rendering [2]. With dense sampling and accurate geometry model, surface light field [21] can further model view-dependency effects. The rendering quality of lumigraph type rendering algorithms entirely depends on the accuracy of the geometry proxies and unfortunately it is usually difficult to provide densely sampled and accurate geometry proxies for a real scene.
Rendering new views from a set of images have also been studied in the computer vision community. Shape-from-image techniques, such as shape-from-shading [11] have long been studied and its major goal is to reconstruct the 3D geometry of the scene from one or more images. These methods usually work for single and simple objects with specific surface models and are not suitable to reconstructing real scenes with complicated occlusions and surface properties. An alternative is to model such scenes as 3D points with depths that are usually stored implicitly in the form of correspondences. Lots of researches have focused on robust correspondence generation algorithms. Most of these algorithms, like graph-cut/maximum flow method [2] and the dynamic programming algorithms [8] are successful on textured Lambertian surfaces but exhibit poor performance on non-textured regions, surface regions with view-dependent reflection, and occlusion boundaries. These conditions lead to inaccurate correspondences for most computer vision algorithms, and hence incorrect geometry reconstruction and problematic rendering. Recent studies [13][15] in computer vision focus on segmenting occlusion boundaries and modeling specular highlight from densely sampled images. These methods usually require dense sampling and high computation cost that are not suitable for real-time light field rendering.

We may also view conventional image-based rendering approaches in the framework of correspondences. For instance, by assuming constant scene depth, light field rendering actually takes advantage of dense correspondences with uniform disparity. Plenoptic sampling suggests the optimal uniform disparity is the average disparity of the scene. Geometry proxies in Lumigraph rendering can also be viewed as dense and accurate correspondences.

In this thesis, we propose an efficient method to utilize a general set of correspondences to render light fields. We give a new representation of correspondences and suggest an algorithm that synthesizes new views from them using a reconstruction algorithm similar to the rebinning approach described for unstructured cameras in the lumigraph. These correspondences do not need to be highly accurate and can be generated by any of the existing stereo algorithms. Since these correspondences provide additional geometry information, sparsely sampled light field can also achieve high quality rendering.
Our approach first factors all of the rays associated with the constraint plane of each correspondence using a special data structure called “surface camera” or scam. We assign a weight to each scam according to its quality and associated disparity. When rendering, we blend them with their weights by a similar algorithm like the unstructured lumigraph. We will show that our algorithm successfully renders occlusions and view-dependencies even though the correspondences of those regions are inaccurate. 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 specify or remove
correspondences and re-renders the view in real-time. Figure 2-3(c) and 2-3(d) illustrates the various rendering results using scam-rendering algorithm with correspondence from multiple objects.
Chapter 3

Scam Parameterization

In this chapter, we study how to parameterize correspondences in a 4D light field so that they can be easily used for rendering new scenes without explicitly reconstructing the 3D geometries.

3.1 Ray Parameterization

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 an image plane \((u, v)\). Surface light fields [21] suggested an alternative ray parameterization where rays are parameterized over the surface of a pre-scanned 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 \(z = 0\), as is shown in Figure 3-1. This leads to the parameterization of all rays passing through point \((p_x, p_y, p_z)\) as

\[
\mathbf{r}(\hat{s}, \hat{t}, \hat{u}, \hat{v}) = (0, 0, -p_z/p_z, -p_y/p_z) + \hat{s} \cdot (1, 0, 1 + 1/p_z, 0) + \hat{t} \cdot (0, 0, 1 + 1/p_z)
\]  

(1)
Figure 3-1: Two plane parameterization for light field: each ray in a light field is uniquely parameterized as a 4-tuple \((s, t, u, v)\), where \((s, t)\) is the camera plane and \((u, v)\) is the image plane.

We use a slightly different parameterization of [9]; we parameterize each ray as the 4-tuple \((s, t, u, v)\), where \((u, v)\) is the pixel coordinate in camera \((s, t)\). This parameterization is more natural and gives a simple ray parameterization as

\[
\mathbf{r}(s, t, u, v) = \hat{\mathbf{r}}(\hat{s}, \hat{t}, \hat{u} - \hat{s}, \hat{v} - \hat{t})
= (0, 0, -p_x/p_z, -p_y/p_z)
+ s \cdot (1/0, 1/p_z, 0) + t \cdot (0, 1, 0, 1/p_z)
\]  

(2)

The ray-point equations (1) and (2) 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.
3.2 Correspondence Parameterization

Figure 3-2: Correspondence parameterization: each correspondence is identified by two rays from the light field and because they both pass the same 3D point in space, each correspondence forms two similar triangles and the ratio represents the disparity of the correspondence. Notice the red ray that passes through the 3D point and \((0, 0)\) on \(s-t\) plane intersects \(u-v\) plane at \((u_0, v_0)\).

In a calibrated setting, each correspondence associates two rays that identify a unique 3D point. By ray-point equation (2), all rays passing through this point should lie on an \(s-t\) constraint plane. And therefore we can represent each correspondence with a constraint plane by computing the 3D point from the correspondence.
Usually correspondences can always be specified as scalar disparities along epipolar lines. We will assume that all source images have to be rectified such that their epipolar planes lie along pixel rows and columns. In this setting disparities can be described as the horizontal and/or the vertical shifts between the corresponding pixels of image pairs. Each correspondence is represented as two rays \( r_1(s',t',u',v') \) and \( r_2(s'',t'',u'',v'') \), which pass through the same 3D point. Assuming uniform sampling, and applying the constraint plane equation (2), we have

\[
\frac{u''-u'}{s''-s'} = \frac{v''-v'}{t''-t'} = \frac{1}{p_z} = disp
\]

where \( disp \) is the disparity of a correspondence. In a geometric point of view, the two rays associated with the correspondence form two similar triangles as is shown in Figure 3-2, one on camera plane and one on image plane where their ratio is the disparity of the correspondence. We then rewrite the point’s constraint plane equation (2) in terms of its disparity as

\[
\mathbf{r} = (0, 0, u_0, v_0) + s \cdot (1, 0, disp, 0) + t \cdot (0, 1, disp)
\]

3.3 Scan Factoring

Our goal is to factor all of the rays associated with a given correspondence and use this information to better interpolate new rays. By using the disparity parameterization, we don’t actually need to determine the 3D position of a correspondence to identify all source rays. Instead, we can directly do ray factoring in image space. The values of \( u_0 \) and \( v_0 \) in constraint plane equation (4) can be determined directly from the correspondence rays, \( r_1 \) and \( r_2 \). 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.
Figure 3-3: 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 correspondence with unit disparity, we can factoring all rays associated with a scam.
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 3-3(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 3-3(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

      bilinearly interpolate $P(u, v)$ in $C(s, t)$

      store $P$ as pixel $(s, t)$ in scam $s$

   end for

end for
Chapter 4

Scam Representation

From the constraint equation (4), each correspondence is on a 2D $s$-$t$ plane in the 4D light field and by collecting all rays associated with a given correspondence, we can form an 2D image of it, which we call the scam image. In this chapter, we study the meaning that scam images imply and how to use scam images to interpolate new rays when rendering new views. An example of scam image is shown in Figure 4-1.

4.1 Interpolating Scam Images

All rays passing through a correspondence should all lie on its constraint plane and scam image and therefore it is important to correctly interpolate the scam image to render new rays. When a correspondence is accurate, i.e., its corresponding 3D point lies close to the real surface and is not occluded in any camera view, its scam image reflects the radiance properties of this point on the surface. For most surfaces, their radiance functions are smooth and have small variance. Similar to reconstructing 2D signals from samples, simple interpolation method like bilinear interpolation is sufficient. If a scam is on the view-dependent spots, it then exhibits variation in intensity. In that case, better reconstruction filters that preserves high frequencies is more appropriate. Furthermore, when a scam is on occlusion boundaries, it exhibits sharp transitions between occluded and non-occluded rays and edge-preserving filters is the correct choice to interpolate the image.
Factor the light field

Figure 4-1: In order to generate a scam image, we first reparameterize the correspondence in form of constraint plane and then factor out all the rays (pixels) in the light field using the scam-factoring algorithm and construct a sampled $s$-$t$ image and finally interpolate the image.
4.2 Scam Classification

The scam image of a correspondence close to the real surface indicates the radiance received at all data cameras and thereby represents the surface’s local reflectance radiance, shown as \textit{scam}_1 of Figure 4-3. Moreover, if the surface is Lambertian, then these scams are expected to have constant color everywhere, as is shown in Figure 4-2(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 4-2(d) shows the scam of the specular highlight on the pumpkin. Finally, if a correspondence is not close to the real surface, then we expect to observe greater variations in its scam images, as shown in \textit{scam}_3 of Figure 4-3 and Figure 4-2(b).

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 \textit{scam}_2 of Figure 4-3 and Figure 4-2(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.

Because correspondences are usually not very accurate, we can further 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, we can characterize them by modeling bimodal color distributions from their scam images using the method described. And for view-dependent spots like specular highlights, we model them as combinations of Gaussian functions.
Figure 4-2: 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.

Figure 4-3: $scam_1$ 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 incorrect correspondence and is far away from the real surface.
4.3 Measure Scam Quality

The quality of the scam depends on its accuracy, occlusion condition and view dependencies. An accurate correspondence on non-occluded Lambertian surface should have uniform color in within its scam image. Correspondences on occlusion boundaries should have partial consistencies as well as sharp changes within their scam images. Accurate correspondences on view-dependent spots like specular highlights have a Gaussian-like smooth distribution. And inaccurate correspondences are expected to have random colors within their scam images.

We can therefore measure the quality of the scam as following; we first calculate the variance of the color in the scam image. If the variance is below certain threshold, then we assume the scam is on the non-occluded Lambertian surface and we assign a large weight to the scam; otherwise we estimate if there is abrupt changes in the image, if so, we fit a bimodal distribution to it and classify it as occlusion boundaries. Otherwise, we fit a 2D Gaussian function to the scam and classify it as a view-dependent scam. If the fitting error is small, we then treat the scam as on occlusion boundaries or on a non-Lambertian surface and assign the weight according the fitting error. Otherwise, we assume the scam is incorrect and assign a small weight or simply discard it. Moreover, we assign different interpolation methods as is discussed in chapter 4.1 in respect to their type. The complete scam classification algorithm is as follows:

Classify scams:

\[
\text{for each scam } S \text{ do} \\
\quad \text{calculate the color variance } var \text{ of } S \\
\quad \text{if } var < threshold_{flat\_color} \\
\quad \quad \text{metric}_{\text{quality}} = \text{metric}_{\text{max}} \\
\quad \quad \text{RENDER\_METHOD} = \text{BI\_LINEAR} \\
\quad \text{end if} \\
\quad \text{else do} \\
\quad \end{align*}


if $S$ has abrupt color changes
  fit a bimodel function to $S$ in term of least square error $err$
  if $err < threshold_{boundary}$
    \[
    metric_{quality} = metric_{max} \cdot \exp(-err)
    \]
    \[
    RENDER\_METHOD = EDGE\_PRESERVE
    \]
    end if
  else do
    \[
    metric_{quality} = metric_{min} \text{ or discard the scam}
    \]
    end else
  end else
end else
end else
else
  fit a 2D Gaussian function to $S$ in term of least square error $err$
  \[
  metric_{quality} = metric_{max} \cdot \exp(-err)
  \]
  \[
  RENDER\_METHOD = BI\_CUBIC
  \]
end else
end else
end for
Chapter 5

Scam Rendering

In this chapter, we study how to render new views using scams. We start with first projecting each scam onto the desired view and then determine the ray that passes through the desired camera and the scam (correspondence). We finally use the scam image to interpolate the reflected radiance with appropriate interpolation methods discussed in Chapter 4. We show that by our scam parameterization, all these computations can be efficiently done in image space and it is not necessary to reconstruct the 3D point of a correspondence in geometry space.

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 (4) as

\[
(u, v) = (u_0 + s' \cdot \text{disp}, v_0 + t' \cdot \text{disp})
\]  

(5)

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).
Figure 5-1: Projecting the correspondence in the new view: (a) we project the correspondence onto camera \((s', t', z')\) by first projecting it onto camera \((s', t', 0)\) and then calculating its projection with geometric relationships; (b) we construct the ray that passes the correspondence from camera \((s', t', z')\) by computing its intersection \((s'', t'', 0)\) with the original camera plane and interpolating it from the scam.
We then apply the geometry relationship as is shown in Figure 5-1(a) and use the depth-disparity equation (3), we have

\[
\frac{u'}{u} = \frac{v'}{v} = \frac{D/(z + z')}{D/z} = \frac{z}{z + z'} = \frac{1}{1 + z' \cdot \text{disp}}
\]  

(6)

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

\[
(u', v') = \left( \frac{(u_0 + s' \cdot \text{disp})}{1 + z' \cdot \text{disp}}, \frac{(v_0 + t' \cdot \text{disp})}{1 + z' \cdot \text{disp}} \right)
\]  

(7)

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 5-1(b). Therefore by reusing disparity equation (3), we can calculate \((s'', t'')\) as

\[
(s'', t'') = \left( \frac{(u' - u_0)}{\text{disp}}, \frac{(v' - v_0)}{\text{disp}} \right)
\]

\[
= \left( \frac{(s' - z' \cdot u_0)}{(1 + z' \cdot \text{disp})}, \frac{(t' - z' \cdot v_0)}{(1 + z' \cdot \text{disp})} \right)
\]  

(8)

The color of the projected correspondence is then bilinear interpolated at \((s'', t'')\) in the scan 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:
Figure 5-2: Color blending scams: to render a desired pixel (yellow star) in the desired view, we first collect all the projected scams (red dots) in the 1-ring neighborhood of the pixel; we then assign weights to these scams and color blend them into the desired pixel.

\[ \text{weight (corresp } i) = \text{metric}_{\text{smoothness}} \text{ (scam } i) + \text{metric}_{\text{dis tan ce}} \text{ (corresp } i) + \text{metric}_{\text{disparity}} \text{ (corresp } i) \] (9)

The first term of the weight evaluates the smoothness of the scam, where we classify and measure their quality as is discussed in Chapter 4.

The second term measures how close that the projection of a scam to the pixel, where closer projections get higher weight. Basically we calculate the projection of the scam in the new view using the $u$-$v$-projection equation (7) and then compute the Euclidian distance between the projection $(u', v')$ and the pixel $(u, v)$ as follows:

\[ \text{metric}_{\text{dis tan ce}} \text{ (corresp } i) = \exp(-\sqrt{(u-u')^2 + (v-v')^2}) \] (10)
The calculation of $u'$ and $v'$ accelerated using the caching method described in Chapter 5.3.

The last term of equation (9) biases on the depth of correspondences in terms of their disparities. For a boundary pixel, there could be multiple correspondences with different disparities associated with it. Obviously closer correspondence should occlude farther-away ones; we therefore assign higher weight to large disparities as follows:

$$\text{metric}_{\text{disparity}}(\text{corresp } i) = k_0 \cdot e^{-1/\text{disp}}$$

where $\text{disp}$ is the disparity of correspondence $i$ and $k_0$ is some constant term for normalization.

Notice the metric functions are chosen to be continuous for all terms to maintain the smooth transition from scam rendered parts to light field rendered parts. We will analyze in the next chapter that our rendering algorithm is equivalent to a simplex-based continuous light field rendering. 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 (8)
  interpolate $r$ in the scam image of $S$
  calculate the projection $P(u', v')$ of $S$ in $C$ using equation (7)
  compute the weight of $S$ using equation (9) 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
5.3 Implementation

Obviously, the computation for rendering a new view is not cheap, especially when rendering views off the camera planes. In the presence of dense correspondences, the major computation comes from the \( u-v, s-t \) projection equation (7) and (8), as well as the color blending weight equation (10). Since we assume discrete and integer disparity values in our model, we can build up tables of the most computational costly terms in these equations, indexing on the disparity value.

For instance, when rendering a new view \( (s', t', z') \), we first compute two tables indexing on the disparity for terms \( s'/(1+z\cdot disp) \) and \( t'/(1+z\cdot disp) \). We also compute the table for term \( z' u_0 / (1+z'\cdot disp) \) indexing on both the disparity and the pixel coordinate. The \( u-v \) and \( s-t \) projection for each scam then can be quickly calculated afterwards by indexing the corresponding disparity and pixel value into the cache table.

For a sparse light field, the maximum disparity is in the magnitude of \( O(100) \) and the image resolution is of \( O(1000) \), hence the cache table is usually less 10MB while saves a huge amount of computations, especially when there are a large number of scams.

We can also apply the same caching method to calculating the color blending weight of each scam. The smoothness term and the disparity term in the color blending weight (9) of the scam are independent of the view position and hence can be pre-computed. The remaining term, i.e., the weight from the projection distance (10) needs to be calculated on the fly while the projection of the scam (7) can be quickly computed using the cache table discussed before.
Chapter 6

Scam Generation

In this chapter, we study efficient methods to generate high quality scams under sparse correspondence constraints between images. We start with correlating regions from a pair of images that users can provide with our interactive tools. We then generate correspondences by sweeping through scanlines between the two regions under epipolar constraints. The epipolar constraints guarantee that two rays associated with each correspondence intersect at a 3D point in object space.

Furthermore we assume two additional constraints:

- Ordering constraint: corresponding points appear in the same order along epipolar lines.
- Piecewise continuity: 3D geometries are piecewise continuous.

Notice the ordering constraint is not always valid, especially for regions close to occlusion boundaries. It, however, prohibits intercrossing between correspondences and allows us to use a large class of dynamic programming based algorithms to generate correspondences. In addition, as is mentioned in previous chapters, our rendering algorithm does not heavily rely on the accuracy of the correspondences for rendering quality since low quality correspondences will be “overwritten” by high quality ones in our quality-biased blending schemes. Piecewise continuity constraint assumes the 3D geometry is continuous, e.g., occlusion boundaries are continuous. This matches well with the continuity assumption in our scam-rendering algorithm where we implicitly
maintain a continuous light field. We will discuss in details this continuous light field property later this chapters.

6.1 Correlating Scanlines under Epipolar Constraints

Given two regions from two images, our goal is to determine pairs of scanlines to be correlated to generate correspondences. Recall the ray-point parameterization in Chapter 3, rays passing through the same 3D point is parameterized as

\[ r = (0, 0, u_0, v_0) + s \cdot (1, 0, \text{disp}, 0) + t \cdot (0, 1, 0, \text{disp}) \]  

(3)

Obviously, if the two images are on the same row or column, i.e., have the same \( t \) or \( s \) coordinates, then the two rays of each correspondence should lie on the same horizontal or vertical scanline as well. In general, two rays \( r(s_1, t_1, u_1, v_1) \) and \( r(s_2, t_2, u_2, v_2) \) of a correspondence between images \( (s_1, t_1) \) and \( (s_2, t_2) \) must satisfy

\[
\begin{align*}
\frac{u_1 - u_2}{v_1 - v_2} &= \frac{u_0 + s_1 \cdot \text{disp} - (u_0 + s_2 \cdot \text{disp})}{v_0 + t_1 \cdot \text{disp} - (v_0 + t_2 \cdot \text{disp})} = \frac{s_1 - s_2}{t_1 - t_2}
\end{align*}
\]  

(12)

In other words, the scanlines to be correlated between the two regions should have the same slope, i.e., the slope of the two images in image space. Figure 6-1 shows an example on how to determine corresponding scanlines between two images.
Figure 6-1: Correlating scanlines under Epipolar Constraint: (a) two images are chosen by user to be correlated; (b) the correspondence scanlines are generated under Epipolar Constraint.
6.2 Generating Correspondences between scanlines

6.2.1 Uniform Shearing and Bresenham’s Algorithm

Once the user assigns the two regions to be correlated, we can determine the direction of scanlines using the method described above and then follow each pair of scanlines to generate correspondences. Notice if the number of pixels on two scanlines is the same, then we will have a simple one to one mapping for all correspondences. More often two scanlines have slightly different number of pixels and in that case we use Bresenham’s algorithm to correlate pixels. Bresenham’s algorithm is widely used for rasterizing lines and in our case we parameterize each scanline in form of its start and end point \((\text{start}, \text{end})\) and to generate correspondences between two scanlines \((\text{start}_1, \text{end}_1)\) and \((\text{start}_2, \text{end}_2)\) is equivalent to rasterize a line starting from \((\text{start}_1, \text{end}_1)\) and ending at \((\text{start}_2, \text{end}_2)\). The complete algorithm is shown as followings:

\[
\text{Correlate-Bresenham}(\text{scanline}_1(\text{start}_1, \text{end}_1), \text{scanline}_2(\text{start}_2, \text{end}_2))
\]

Bensenham Rasterizing \(((\text{start}_1, \text{end}_1), (\text{start}_2, \text{end}_2))\);

for each rasterized pixel \((x, y)\)

Generate Correspondence for \((\text{scanline}_1(x), \text{scanline}_2(y))\);

end for

Figure 6-2: Bresenham’s Algorithm: To generate correspondences two scanlines is equivalent to rasterize a line using Bresenham’s Algorithm.
Figure 6-3: Dynamically reparameterized light field is a special case of correspondence mapping. Boundary points are forced to correlate to some end points.

An easy alternative to Brensenham’s algorithm is to assume uniform disparity between scanlines, as is shown in Figure 6-3 and we correlate internal pixels with a specific disparity and fix the boundary pixels. Furthermore, we can use the local optimal disparity when correlating two scanlines by picking the disparity that minimize the total difference function. Notice here the optimal disparity is different from the one suggested by plenoptic sampling [4]. We obtain our optimality per pair of scanlines and hence our disparity is local while plenoptic sampling assumes global optimal disparity (by assuming different depth layers of geometries have uniform distribution).

Furthermore, by assuming uniform disparity, we can view this scam generation algorithm as to apply the dynamically reparameterized light field on small regions, each of which has a fixed focal plane. As is discussed in Chapter 2, in DRLF, only parts of the scene whose real depth is close to the assumed focal depth can be clearly reconstructed while other parts that are far away from the focal plane or close to occlusion boundaries exhibits serious aliasing artifacts or blurriness if using large aperture. This artifacts, however, is less serious in scam rendering for the following reasons; first of all, unlike DRLF that assumes uniform disparity for the whole light field, different pairs of regions that are correlated can have different disparity in scam rendering; second, because scam rendering biases on the quality of the scams, low quality scams are expected to be “overwritten” by the high quality ones provided later, therefore aliasing artifacts can be reduced or even removed. This is not uncommon, especially with our interactive system that allow users to provide or remove scams.
6.2.1 Dynamic Programming

To obtain better quality scans, we assume the ordering constraint so that we are able to use a large class of dynamic programming algorithms from computer vision research. The basic idea of these dynamic programming-based methods is to transform the problem of finding correspondences into finding the optimal path in a directed graph. For example, to correlate two scanlines with \( m \) and \( n \) pixels, we first form an \( mxn \) correlation graph where each node \((i, j)\) in the graph represents the similarity between pixel \( i \) and \( j \) in its corresponding scanlines. And we can find a best sequence of correspondences between these two scanlines by finding the path from node \((0, 0)\) to node \((m, n)\) with highest overall correlation scores. The ordering constraint guarantees that the correlation graph is directional and hence we may apply dynamic programming algorithms to find these optimal paths as is shown in Figure 6-4.

We start with calculating the correlation \( \text{correl}(i, j) \) between pixel \( i \) in the first scanline and \( j \) in the second. Ideally, we need the linear invariance despite the changes in
illumination. This linear invariance property can be easily achieved by using CIE model with \(xyY\) color space. \(x\) and \(y\) are good indicators of the actual color, regardless of the luminance. We nevertheless have to deal with a new non-linearity in the transform between RGB and CIE \(xyY\). In a light field where illumination remains almost constant for all images, simple RGB color distance also works well in practice. We further assume the minimum depth of the scene as the largest disparity as \(\text{disp}_{\text{max}}\) and assign the weight of each node in the graph as:

\[
correl(i, j) = \begin{cases} 
3 \times 255^2 & \text{if } j > i \text{ or } i > j + \text{disp}_{\text{max}} \text{;} \\
3 \times 255^2 - (r_i - r_j)^2 - (g_i - g_j)^2 - (b_i - b_j)^2 & \text{otherwise;} 
\end{cases}
\]

Denote \(\text{Opt}(i, j)\) as the optimal path that that goes from \((0, 0)\) through \((i, j)\); then we deduce the dynamic programming equation as:

\[
\text{Opt}(i, j) = \max \begin{cases} 
\text{correl}(i, j) + \text{Opt}(i, j - 1) \\
\text{correl}(i, j) + \text{Opt}(i - 1, j) 
\end{cases}
\]

A routine dynamic programming method solves this problem in \(O(N^2)\) time and gives the complete optimal path from \((0, 0)\) to \((m, n)\). Then for each node \((i, j)\) on the path, we correlate pixel \(i\) and \(j\) on the two scanlines as a scam. Figure 6-5 compares the dynamic programming method and the optimal disparity method by showing the Eipolar Images by interpolating correspondences between scanlines. The optimal disparity method aligns most of the features correctly except the white/black stripe since its corresponding geometries is not close to the presumed the optimal focal plane and we observe serious aliasing artifacts in these regions. The dynamic programming method, however, manages to remove these aliasings by correctly aligning them.

The dynamic programming has been used widely in the computer vision field, whose main goal is to completely reconstruct the 3D surface. Unfortunately it has several major artifacts. First of all, it adapts poorly to non-textured regions where it is very sensitive to small noises on uniform-colored surfaces. Second, it cannot handle view-dependencies such as specular highlights where the correlations are low for points on these surfaces. Third, the ordering constraints are not always valid for cameras with large baselines where intercrossing may happen. It becomes more problematic on occlusion boundaries to determine the depth of pixels on these occlusion boundaries.
Figure 6-5: Comparison between optimal disparity and dynamic programming: from top to bottom, a sparse sampled Epipolar image, interpolated Epipolar image using optimal disparity and interpolated Epipolar image using dynamic programming.
Fortunately these defects for most stereo algorithms cast much fewer artifacts on our scam-rendering algorithm. Recall that our scam-factoring algorithm distribute each correspondence into the light field and then measure its quality in its scam image, therefore inaccurate correspondences are given much lower priorities when rendering. As a result, it will be "overwritten" by its good neighbor scams ones and hence has much fewer artifacts. Furthermore, on the occlusion boundaries where most computer vision algorithms fail to reconstruct accurate geometries, our scam-rendering algorithm smoothly blends different layers of correspondences and guarantees high quality rendering. Finally we provide users interactive tools to select regions and methods to correlate them and therefore it helps to solve most of the view-dependency problems. In the result chapter, we will show by different examples of our scam rendering algorithms to illustrate how our algorithm takes advantage of correspondences while removing their defects.

6.3 Continuous Light Field Reconstruction

Rendering a new ray in a light field is equivalent to interpolating it from the sampled rays and it is very desirable to provide a continuous representation of the light field. The simplest continuous representation of space is the simplex-tiled space, e.g., a triangulation of the 2D space or a tetrahedralization of the 3D space. Such representations have great advantages on the rendering: to render a new ray \( r \), we first locate the simplex that it lies in and then calculate the barycentric coordinates of \( r \) in the simplex in respect to all vertexes of the simplex and use them as weights for blending. Randomized algorithms [7] are usually used for tracing consecutive rays and memory caching is used to record types of simplexes to accelerate calculations of the bary-centric coordinates.

The dynamic programming algorithm we mentioned above in fact naturally gives a triangulation between two scanlines. Recall the dynamic programming algorithm determines the optimal path of correspondences and prohibits them over-crossing, we
may define the “edge frontier” as the correspondence generated at each node on the optimal path as pair \((i, j)\), where \(i\) is the pixel index on first scanline and \(j\) on the second. Notice edge frontier \((i, j)\) in the optimal path must go to either \((i + 1, j)\) or \((i, j + 1)\) according to the algorithm; therefore the two consecutive edge frontier must share a vertex as is shown in Figure 6-6. Therefore the two neighboring correspondences form a triangle and it is easy to extend the deduction to the whole scanline and we then form a triangulation between two scanlines.

Figure 6-6: We can triangulate two scanlines directly from the optimal path of the correlation graph.
Figure 6-7: Correspondences generated by dynamic programming algorithm may still intersect in 4D space; here the green and the red correspondences from the horizontal scanlines intersect on the vertical scanline.

It is desirable to tile the 4D light field space with simplexes aligned with the correspondences. Simplexes in 4D are 5-vertex 10-face pentahedras. Notice correspondences in 4D are 2 dimensional constraint planes. To achieve a non-trivial pentahedra-tiled 4D space, we need to align simplexes with constraint planes. Unfortunately it is an extremely difficult task, even with the same set of correspondences that are used for triangulating scanlines from the dynamic programming algorithm in 2D. The major difficulty lies in that, although in 2D two correspondences do not intersect under our ordering constraint, the corresponding 2D constraints planes may still intersect in 4D, as is shown in Figure 6-7, where two correspondences from two horizontal scanlines might intersect on the vertical scanline. These intersections are quite usual in practice. In other words, there does not exist a valid triangulation (simplex-tiling) of the 4D space where all simplex faces align well with correspondences without introducing additional vertexes.
One simple solution to the problem is to calculate intersections between all pairs of constraint planes and insert intersection points back to the 4D space. However, it turned out to be quite impractical for the following reasons. First, notice any two planes can intersect in 4D space unless they are parallel, therefore it usually leads to a huge number of intersections from a small set of correspondences. It also leads to tremendous computational cost for calculating these intersections. Second, it is not clear how to determine the color of these intersection points and how to insert them back to the 4D space while maintaining the simplex-tiled structure. Finally, the simplex-based barycentric coordinate interpolation is independent of the quality of correspondences and it therefore may give equal importance to low quality correspondences as to high quality ones.

The scam rendering, however, is an implicit but more general alternative to the simplex-based 4D continuous light field rendering. First of all, our scam representation uses a generalized form of the constraint planes. Since we continuously interpolate the corresponding scam images, we maintain the continuity on all constraint planes. Secondly, our scam-rendering algorithm first projects all scams onto the image plane and then blend them. Notice if two constraint planes intersect in 4D space, projections of the rays nearby the intersection point should be close to each other on the image plane. By collecting and blending scams in certain neighborhood, our rendering algorithm maintains a continuous interpolation between them. In particular, in the scam metric, if we only take the projection metric and ignore the smoothness and disparity, we are exactly implementing barycentric coordinate interpolation. Finally, our scam-rendering algorithm takes advantage of the knowledge of the quality and the type of the scams and it is more robust than the simplex-based rendering in presence of low-quality correspondences. The caching methods discussed in the previous chapter also make our rendering speed comparable with simplex-based rendering.
Chapter 7

Results

We have developed a user-guided scam rendering system where the users are able to specify image regions to be correlated. Because local correspondences are faster to generate and are more reliable, 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 user interface is shown as Figure 7-1. The system starts with a conventional light field rendering system, where the user can specify the focal plane by disparity and the system renders the new view in real time. Users can choose any pair of images from the light field and any pair of regions to be correlated. A dynamic programming engine then generates all correspondences and users can view scam images of them to decide whether they want to keep or remove them. The new view is then rendered in real-time using forward-mapped scam rendering and backward-mapped light field rendering. Users can then decide whether more correspondences needed to be provided to improve rendering quality.

The pumpkin dataset shown in Figure 7-2 is constructed from a 4x4 sparse light field. Figure 7-2(a) renders the new image using standard light field 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 pumpkin, as is shown in 7-2(b), the head in 7-2(c), the fish and part of the red stuffed animal in 7-2(d) by
correlating rectangular regions respectively using the dynamic programming algorithm. As a result, reconstructions in the yellow regions of the synthesized images are significantly improved in all images. Boundaries between the light field rendered parts and scam rendered parts maintains smoothness.

The office scene light field in Figure 7-3 illustrates the gradual improvements using scam rendering with more and more correspondences. Figure 7-3(a) is a synthesized view using light field rendering with the optimal disparity. Figure 7-3(b) improves the reconstruction of the red stuffed animal with scam rendering associating its correspondences. Figure 7-3(c) renders the scene with additional correspondences from different parts of the scene. Figure 7-3(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.
Figure 7-1: Scam rendering system allows the user to change focal plane distance, assign regions to be correlated, display scam images and it renders the new view in real-time.
Figure 7-2: Light field and scam rendering: (a) light field rendering with optimal disparity; (b), (c), (d) scam rendering with correspondences of objects in the yellow regions below each image while rest of the scene rendered by light field as (a).
Figure 7-3: Light field and scam rendering with view dependencies: (a) light field rendering with optimal disparity; (b) scam rendering with correspondences of the red stuffed animal and the screen; (c) scam rendering with correspondences of multiple objects; (d) scam rendering as (c) with additional correspondences of the reflected lights on the back wall.
Chapter 8

Conclusions and Future Work

In this thesis 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. As is shown in previous chapters, most scam images have almost constant colors and hence can be used to compress the light field. We want to study how we can efficiently represent the light field with compressed scam images.

If we plot the image-based rendering algorithms with an axis indicating the amount of geometry information used for rendering, as is shown in Figure 8-1, light field rendering uses unknown geometry, surface light field uses known geometries and our scam-rendering uses partial geometries in form of correspondences. In this thesis, we study the relationship between light field rendering and scam rendering. It will be very interesting to extend our scam rendering to surface light field and study efficient ways to use hardware to accelerate our rendering.

Figure 8-1: Relationship between light field, scam and surface light field rendering.
References


