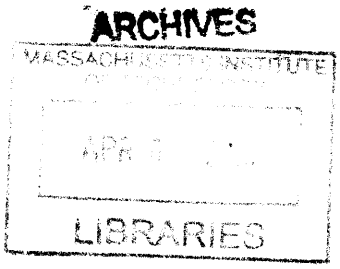


Unstructured Light Fields

by

Myers Abraham Davis (Abe Davis)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2013

Author .....
Department of Electrical Engineering and Computer Science
August 31, 2012

Certified by .....
Fredo Durand
Professor
Thesis Supervisor

Accepted by .....
Leslie A. Kolodziejski
Chair, Department Committee on Graduate Students

Unstructured Light Fields

by

Myers Abraham Davis (Abe Davis)

Submitted to the Department of Electrical Engineering and Computer Science
on August 31, 2012, in partial fulfillment of the
requirements for the degree of
Master of Science

Abstract

We present a system for interactively acquiring and rendering light fields using a hand-held commodity camera. The main challenge we address is assisting a user in achieving good coverage of the 4D domain despite the challenges of hand-held acquisition. We define coverage by bounding reprojection error between viewpoints, which accounts for all 4 dimensions of the light field. We use this criterion together with a recent Simultaneous Localization and Mapping technique to compute a coverage map on the space of viewpoints. We provide users with real-time feedback and direct them toward under-sampled parts of the light field. Our system is lightweight and has allowed us to capture hundreds of light fields. We further present a new rendering algorithm that is tailored to the unstructured yet dense data we capture. Our method can achieve piecewise-bicubic reconstruction using a triangulation of the captured viewpoints and subdivision rules applied to reconstruction weights.

Thesis Supervisor: Fredo Durand
Title: Professor

Acknowledgments

First I'd like to thank both Fredo Durand and Marc Levoy, who advised me through the various stages of this work. I'd like to thank the Stanford and MIT computer graphics groups for providing me with an excellent environment to learn and explore research. There are many researchers who have given me great advice and guidance - and many more who have taught me a great deal through their example. Unfortunately, I don't have room to list them all here. I would like to add a special thanks to everyone who modeled for my light fields. Your unflinching patience was both impressive and greatly appreciated.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Light fields and lumigraphs provide a faithful reproduction of 3D scenes by densely sampling the plenoptic function, but their acquisition often requires camera arrays or robotic arms, e.g. [16, 10, 32, 2]. We present an interactive system for capturing dense light fields with a hand-held consumer camera and a laptop computer. First, the user points the camera at the scene and selects a subject to capture. The system then records new images whenever it determines that the camera is viewing an under-sampled region of the light field. We guide the user with a viewpoint coverage map to help them achieve dense coverage. To render our light fields we use a piecewise-bicubic reconstruction across viewpoints that leverages the specific attributes of our captured data. The resulting system is cheap, portable, and robust.

We derive a geometric criterion for when to record new images using a bound on the reprojection error between captured views. For each new image, we first compute pose using a real-time Simultaneous Localization And Mapping (SLAM) library [14]. We check if this view can be reconstructed without violating our reprojection error bound. If it can, we discard it; if it cannot, we add this view to the set of captured views. During acquisition, we display a coverage map that shows the views captured so far to help the user achieve dense coverage.

We describe two implementations of our system: one that uses a laptop with a webcam and relies on feature-based pose-estimation; and a reduced prototype that runs on a mid-range cell phone (the Nokia N95) using the device camera, but requires

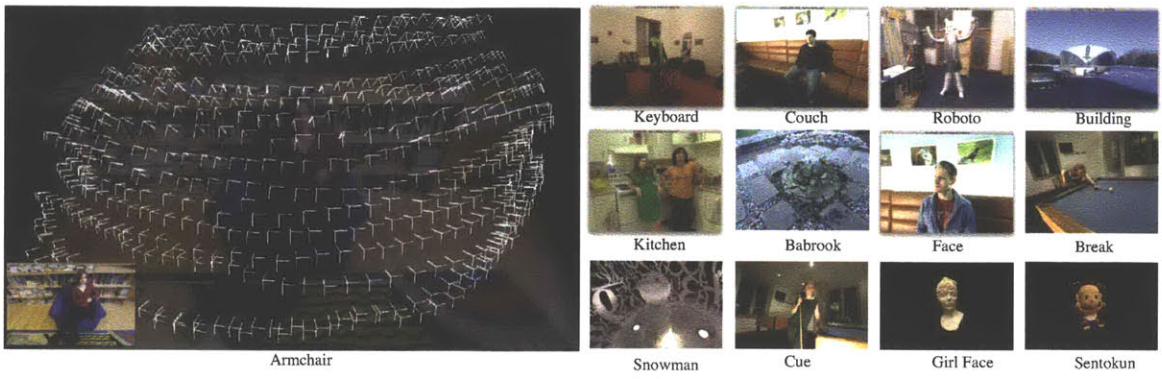


Figure 1-1: A subset of the light fields captured using a simple hand-held camera and our system. Left: Visualization of the pose of the images in the armchair light field. Right: Some of the light fields captured with our system.

a fiducial marker to estimate pose.

Our system is fast, easy to use, and portable. A typical capture session takes between 1 and 11 minutes. We have tested it by acquiring hundreds of light fields of indoor and outdoor scenes at a variety of scales.

1.1 Related work

A number of approaches have addressed light field capture, including robotic arms (e.g. [16]) and camera arrays (e.g. [32]). A single-camera light field capture can be achieved with a microlens array [1, 20, 8] but the extent of the light field is restricted to the camera's aperture. Similar systems use additional lenses [9], masks [29], or mirror arrays [28].

The capture approach most similar to ours is Gortler et al.'s Lumigraph system [10], which uses a hand held camera with a specially-designed stage for pose estimation. Like us, they provide a visualization of the set of viewpoints acquired so far, but the user must gauge the density required for good coverage. We derive a formal criterion for viewpoint coverage and provide a visualization that displays from which viewpoints the subject can be reconstructed.

Buehler et al. [4] also sought to alleviate the requirement of dense uniform light

field sampling, but focused on the rendering algorithm, whereas we focus on acquisition. They briefly discuss different strategies they used for acquisition, including robotic arms, heuristics, and structure from motion. They note, however, that without available scene geometry their hand-held input results in reconstructions that exhibit parallax only along the one-dimensional trajectory defined by a camera's path. Good coverage is difficult to achieve for the hand-held capture of light fields and lumigraphs. This is the main problem our system addresses.

Koch et al. [15] and Heigl et al. [11] perform camera calibration as a batch process after hand-held light field capture, which rules out interactive capture. We use a real-time alternative: the PTAM library developed for augmented reality applications [14].

The rendering algorithm used for unstructured lumigraphs [4] selects views based on a variety of criteria, such as angular distance, and then uses a k-nearest-neighbor reconstruction. They ensure that their interpolation weights fall off smoothly to zero for the k-th nearest neighbor.This approach comes with some limitations. The first problem is scalability. For each sample on the image plane the penalties for every input image must be evaluated and sorted in order to find the k nearest neighbors. The second problem is that the k nearest neighbors might exhibit a poor angular distribution around a given location. For example, the nearest neighbors may not surround the reconstructed view; they may all be on one side, and may suddenly switch to the other side as the virtual camera is moved (Fig. 5-1(a)). Furthermore, the blending field (Fig. 5-1(b)) may have discontinuities and is not always monotonic as a function of the distance to a viewpoint projection because of the normalization term. Despite these limitations, we have found unstructured lumigraph rendering to be robust for sparse datasets and use it when our number of viewpoints is low.

For rendering, [17] use a triangulation of the input cameras similar to ours. However, they use simple bilinear interpolation over the entire output image, restricting the sampling of reconstructed views to linear combinations of just a few input views. This is analogous to restricting the virtual camera to move along the manifold of input views in our method and using bilinear interpolation over the blending field

instead of bicubic.

Plenoptic sampling [5] and light field reparameterization [12] also address the issue of sampling rate in light field acquisition using analysis in the Fourier domain. In contrast, our technique handles unstructured inputs, and uses a geometric bound in the primal domain to drive sampling.

Snavely et al. [26, 25] combine large photo collections and provide a broad but sparse coverage of the plenoptic function. The strength of their approach is the ability to leverage photos that have already been taken. They can, for example, acquire a miniature object by rotating it in front of a camera [25]. However, their approach does not assist the user during the capture process. Our work is complementary and seeks to achieve a dense capture by guiding the user.

View planning has also been recognized as an important problem for 3D scanning, but it seeks to plan a small number of discrete views, e.g. [19, 21]. However, Rusinkiewicz and Hall-Holt demonstrated that real-time feedback can dramatically help a user achieve good coverage for interactive 3D scanning [23]. We similarly found that feedback is critical, but focus on light fields and avoid the use of custom hardware.

Chapter 2

System Design

Our system's design was motivated by the following goals:

Inexpensive commodity hardware The method should work with a single commodity camera and no specialized hardware. This means that camera movement must be performed by the user. This also means that camera orientation and location must be computed from image features.

Horizontal and vertical parallax We want dense light fields that enable both horizontal and vertical parallax. While one-dimensional parallax is easy to achieve by recording a video while the camera is moving, it is much harder for users to move the camera in a way that captures parallax in both directions.

Conservative coverage for complex geometry We should record enough images to bound the reprojection error within some volume of interest even if we have no explicit geometric proxy. Doing so ensures that even difficult subjects like transparent objects or objects with complex occlusions can be reconstructed faithfully.

Speed and real-time feedback Given the coverage requirement, the system must provide real-time feedback to tell the user how to move their camera. To keep capture sessions short, our bound on reconstruction error should be set so that the user achieves good coverage without requiring a prohibitive number of images.

2.1 Representation

Our light field is stored as a collection of photographs with associated camera poses. We also store information about the subject extent (a rough bounding sphere), which we use to compute our coverage criterion during capture and to assist with focus and navigation during rendering.

We separate the computation of a camera blending field from the representation of scene geometry, making it easy to use an arbitrary geometric proxy when it is available. We have tested several types of proxies including a screenspace triangulation of SLAM feature points as in the Unstructured Lumigraph [4], multiview stereo [6] paired with Poisson surface reconstruction as in [7], and different variations on planar proxies including those used in [26, 25] and [16]. We discuss this more in Chapter 5.

2.2 Navigation challenges

Capturing a light field with a hand held camera is difficult primarily because it is hard to tell where previously captured images are and how much of the light field they cover. The user must control all six degrees of freedom of the camera's pose to provide good coverage.

Rotation Rotation of the camera mostly serves to keep the subject centered in the frame. This is not a hard task for users. We assist by alerting them when the subject has left the camera's field of view.

Translation Translation is the main challenge addressed by our approach. In order to reproduce parallax in multiple dimensions, the user must cover a 2D set of views around their subject. One dimension, typically horizontal, corresponds to the trajectory of the camera, and is easy to cover. This is why some existing systems only provide horizontal parallax. The second direction, typically vertical, is the one that is hard to cover and is usually achieved with a back-and-forth scanning motion. This is where the user needs most guidance to make sure that the camera stays at

the correct offset from the previous "scanline" to ensure good coverage.

Subject Distance Ideally the subject should be recorded at the same scale across the different input images. That is, the camera should stay roughly the same distance from the subject as it moves. We assist by providing visual feedback to show the user how much their distance to the subject varies. Our coverage criterion also considers distance to the subject, automatically adapting for images that see the subject at different scales.

Chapter 3

The capture process

Our main implementation relies on a camera connected to a laptop. Feedback to the user is provided via the laptop's screen and superimposed on the current view coming from the camera. We have also implemented a reduced prototype that runs a mobile phone but requires a fiducial marker to enable camera pose estimation. The user needs to first initialize pose estimation, then select a subject of interest before capture begins. The full capture process typically takes between 1 and 11 minutes depending on the size of the subject, the desired density of views, and the expertise of the user.

Pose estimation initialization The user first initializes pose estimation by moving the camera around the scene. This step is required by PTAM [14] and takes between ten seconds for small scenes and two minutes when the user needs to walk farther around a large scene.

Subject specification (bounding sphere) The user then selects their subject by placing a virtual bounding sphere around it, which we visualize with a wireframe mesh. For each input view, the sphere represents a range in two of the four dimensions of the light field. The set of painted views on the coverage map represents a range in the remaining two dimensions of the light field. The intersection of these ranges defines the part of the light field with bounded reprojection error. The sphere does

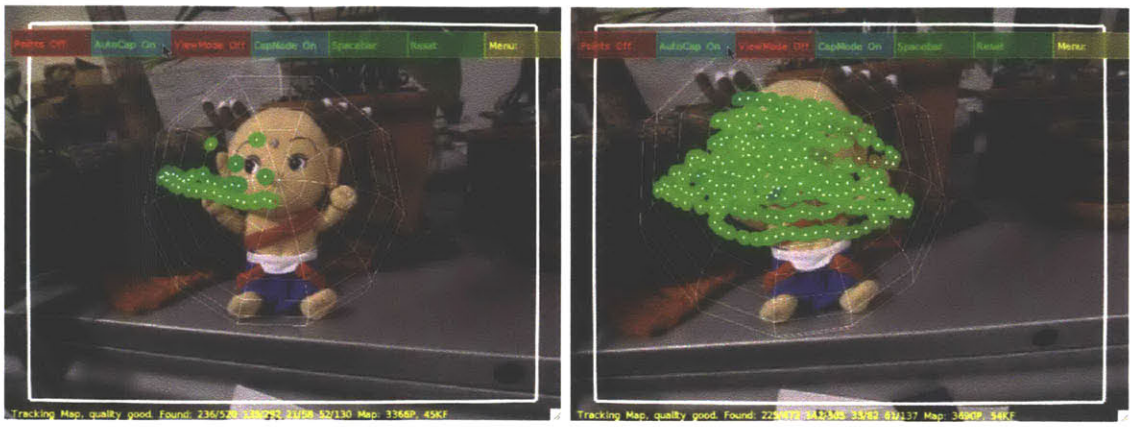


Figure 3-1: Screen capture of our visualization. The virtual meshed sphere serves both as a bound on the scene to be captured and as a coverage map showing the range of viewpoints already covered. At the beginning (left), only a limited range is covered, and the user moves the camera to "paint" the sphere (right).

not need to precisely bound the subject, but our bound on reprojection error does not necessarily hold outside of the sphere.

To specify the sphere, the user centers the subject in the view and the system initializes its distance with the average depth of feature points used in SLAM. This distance estimate can be refined through a keyboard or mouse interaction, enabled by two feedback visualizations.

First, the user can move the camera and view the scene augmented with the sphere mesh. Second, we offer a new *augmented aperture* mode where we augment the current live view of the camera with a synthetic aperture computed from the current sparse unstructured light field. This creates a live view with shallow depth of field focused on the center of the sphere. The view is refreshed as the user moves the camera. This makes it easy to find a good depth for the object by refocusing until the object's silhouette becomes sharp in the augmented aperture view. It is particularly useful for large scenes where the user would need to walk far before getting enough parallax to assess the position of the sphere.

Capture The user then starts the actual light field acquisition. We display a coverage map on top of the bounding sphere overlaid on the current view of the camera. The current location of the camera is projected onto the surface of the sphere as a

small red dot. Every previously recorded image of the scene is also projected onto the coverage map; it occupies a circle corresponding to the range of views that it covers under our sampling criterion (Fig. 3-1). The user's goal is to control the movement of the red dot with the camera and to "paint" the surface of the sphere. When the sphere is well painted, the user has captured enough data to generate a high quality rendering of their subject.

To keep the camera oriented towards the subject we display a safety rectangle in white (Fig. 3-1) within which the user should keep the sphere. If the sphere is not kept inside the rectangle then the rectangle turns red and the system stops capturing images. Only views that see the subject are recorded.

To help the user capture images at a consistent scale we use a color code to represent the distance to the subject in the stored views relative to the current view: green means that the recorded view is the same distance as the current view. Red means that it is further from the subject, and blue means that it is closer. Ideally, the map should always be green, but deviations are tolerable as long as distance does not vary too discontinuously.

Review At any point during capture, the user can review the light field by controlling the movement of a virtual camera. This can be done using a keyboard and mouse interface, or by using the current pose of the capture camera to control the virtual one. In this case, the system renders the scene from the current pose of the real camera. If coverage is good, this visualization becomes difficult to distinguish from the camera's live view. This pre-visualization of the final light field is useful to assess the coverage of the scene and decide if a wider set of viewpoints is needed.

Mobile phone prototype Our mobile phone prototype relies on fiducial markers rather than scene features for two reasons. First, it is computationally cheaper. Second, the current version of PTAM does not handle well the limited optical quality and field of view of most cell phones. Instead, we rely on fiducial markers and the AR Toolkit [13]. The capture interface is also significantly reduced. The user can

review their light field, using the marker as a 3D mouse. Rendering is simplified and only uses nearest neighbor reconstruction for increased speed.

Chapter 4

Coverage criterion

Traditional approaches to light field sampling [5, 12] rely on Fourier analysis and require uniform sampling of a two-plane parameterization. In contrast, our acquisition results in scattered viewpoints, often arranged on a sphere, and a two-plane parameterization of the domain is not appropriate. We seek to determine if a new image would provide significant improvement over an acquired light field. For this, we characterize the reprojection error that is incurred when using already acquired views to render the new view. If the error is above a given threshold, we store the new view. Our analysis is related to error analysis for stereo and other computer vision tasks, e.g. [3, 22, 30], but to our knowledge, it is the first time it is used to derive a sampling criterion for light fields.

Consider a captured view, s, and a new view, n, being tested. We want to derive a bound on reprojection error from the geometric bounds given by our subject sphere. We will compare the bound we compute against our threshold to determine whether n is covered by s.

To compute the worst case reprojection error we consider a cone of ambiguity associated with a pixel V_s of s (Fig. 4-1). This cone represents points in our scene that might contribute to V_s. We represent the cone as the projection of a circle centered at V_s into our scene. We assume that the input images were taken with a small aperture and use a circle with radius equal to the distance between adjacent pixels. The intersection of this cone with our subject sphere contains all subject

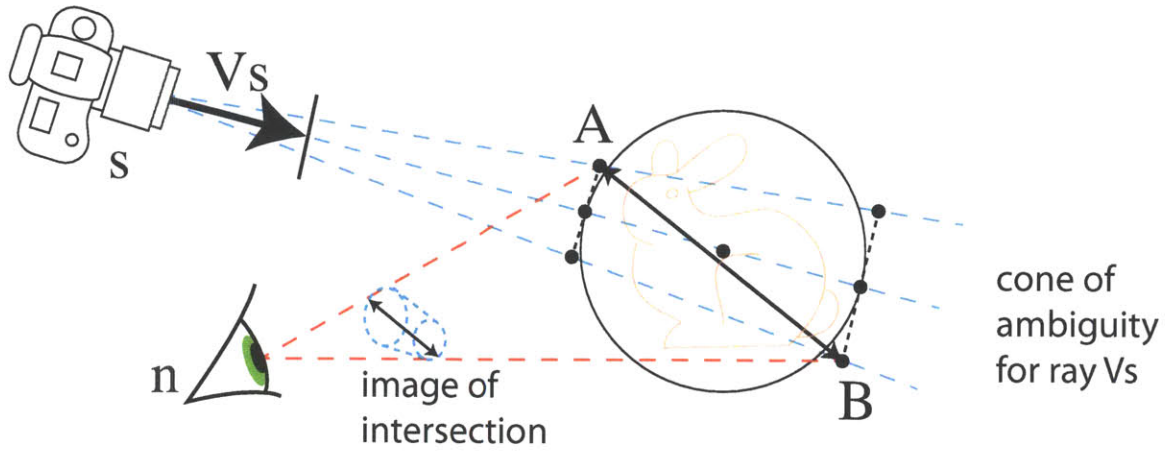


Figure 4-1: Our coverage criterion computes a bound on the reprojection error from view s to view n associated with pixel V_s. This bound is sensitive to both parallax error and the ambiguity resulting from differences in resolution, covering all 4 degrees of freedom of the light field.

points that might contribute to V_s. The image of this intersection in n represents the range of possible reprojections for these points. Our bound on reprojection error is then the longest distance between two points inside the imaged intersection. This distance is the projection of AB in Fig. 4-1 where AB lies on the epipolar plane defined by s, n, and the center of our subject sphere.

Our bound on reprojection error is sensitive to both parallax error and changes in resolution, accounting for all 4 dimensions of the light field. Another way to interpret this coverage threshold is as an effective resolution at which the reconstructed image will not have ghosting or magnification artifacts.

The computation of this criterion is straightforward. For a derivation of the precise formula please refer to the supplementary material.

Chapter 5

Viewpoint-Subdivision Rendering

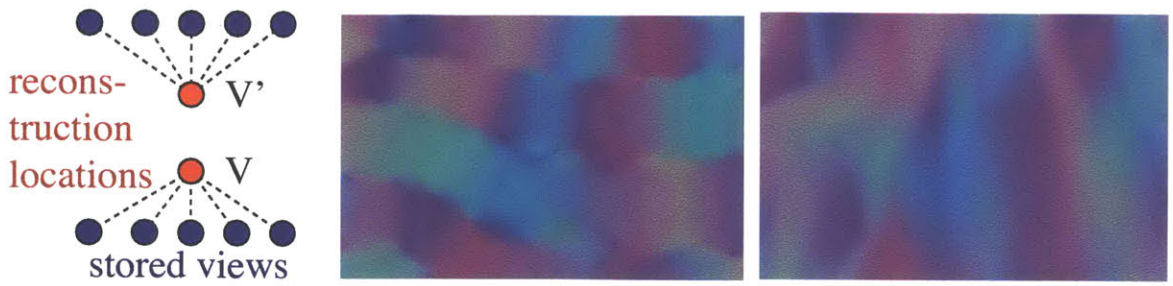
IBR techniques differ primarily in how they compute the *camera blending field*, or the weights used to blend each input image at each pixel of rendered output. In this section we describe how our rendering method computes a blending field.

While the strength of unstructured lumigraph rendering [4] for sparse datasets is that it does not seek to extract a structure over the captured views, this limits its ability to achieve a smooth reconstruction and to scale to larger datasets. At a given output pixel, the algorithm only knows the relative penalties of different views and it is hard to design a blending field that has higher-order smoothness and takes into account the relative locations of the neighboring viewpoints (Fig. 5-1(a)). In contrast, we extract structure from our set of viewpoints by building their Delaunay triangulation. This allows us to design blending weights that smoothly conform to the nearby captured views. In particular, we apply subdivision rules over the triangulated viewpoints to compute smooth reconstruction weights.

We start by describing simple piecewise-linear reconstruction which forms the basis of our slightly more costly but smoother subdivision approach.

5.1 Piecewise-linear reconstruction

Consider the reconstruction of a pixel P from a viewpoint V (Fig. 5-2(a)). We can reconstruct the appropriate ray color using linear interpolation across the image



(a) (b) (c)

Figure 5-1: (a) With k-nearest-neighbor interpolation, all neighbors can suddenly switch when moving from V to V'. (b) Unstructured lumigraph rendering blending weights. Each image is assigned a random color for visualization. (c) blending weights with our technique.

and viewpoint dimensions. We focus on the viewpoint dimensions and use standard projective texture mapping with a geometric proxy for the image ones. We can perform piecewise-linear reconstruction on the camera blending field by intersecting the ray with a triangulation of the set of stored viewpoints (Fig. 5-2(a)). The weight of each of the corresponding three images is given by the barycentric coordinates of the intersection.

For efficiency, we do not reconstruct each ray independently and instead project the vertices of a viewpoint triangle onto the image plane (Fig. 5-2(a)). At the projected vertex corresponding to a stored view W, its weight is 1 while the weight of the other two images is zero. This means that, for a given stored image W, its impact on the rendered view corresponds to the projection of a triangle fan: its 1-ring in a viewpoint triangulation (Fig. 5-2(b)). The weight of W is 1 in the center of the ring and 0 at the boundary, leading to a classical piecewise-linear tent.

We render the fans efficiently using texture mapping and alpha blending, where alpha is one at the center vertex and zero at the boundary of the 1-ring. This means that each triangle is rendered three times, once for each view corresponding to one of its vertices. This could be optimized by binding the triangle to three textures but we have not found overdraw to be a speed limitation. This algorithm provides piecewise

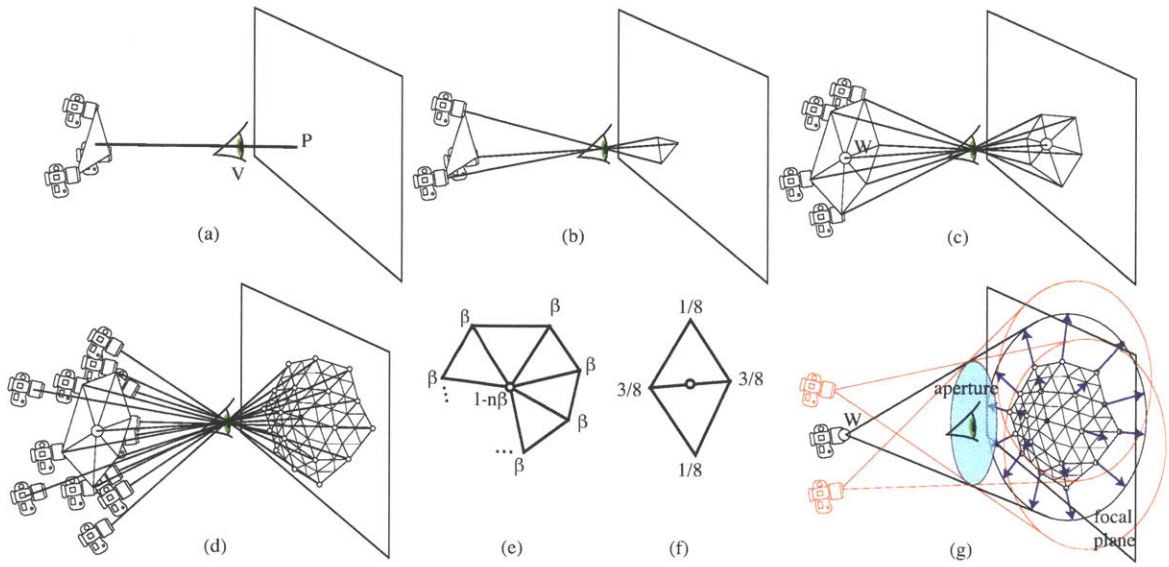


Figure 5-2: Piecewise-linear and viewpoint-subdivision rendering for a new viewpoint V. (a) The color at a ray can be linearly reconstructed by barycentric interpolation of three adjacent views. (b) This is equivalent to projecting the triangles corresponding to the captured viewpoints onto the image plane. (c) For piecewise-linear reconstruction, each captured view W projects onto a triangle fan with weight 1 in the center and 0 at the periphery. The fan is texture-mapped with the image captured from W. (d) Our viewpoint-subdivision reconstruction renders each image onto a subdivision of its projected 2-ring. (e-f) Loop subdivision rules are applied to the weights of a given image inside its 2-ring. (g) Wide-aperture rendering is achieved by warping the 2-ring onto a disk corresponding to the projection of the aperture with respect to the corresponding viewpoint onto the focal plane. The projection of the various views through the aperture (in red) overlap, which leads to a shallow depth of field.

linear reconstruction. Later we extend it to piecewise bicubic reconstruction.

5.2 Viewpoint triangulation

Because we guided the user to capture images roughly on a spherical shell around the subject, we use spherical coordinates to create a 2D triangulation (Fig. 5-3). Our implementation uses the Euclidean library by Shewchuck [24][1]. For this, we transform the input viewpoints W_I into the spherical coordinate system centered at the subject

[1]http://www.cs.cmu.edu/ quake/triangle.html

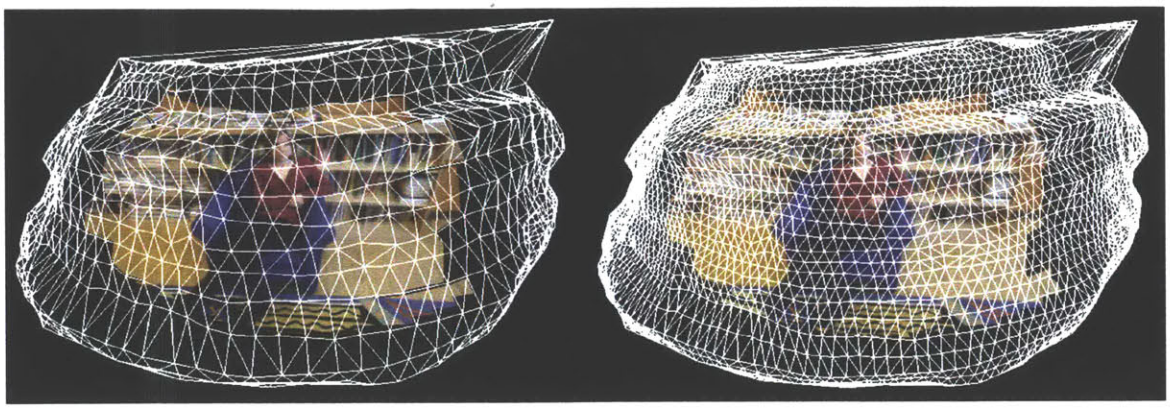


Figure 5-3: Triangulation of the captured viewpoints before and after subdivision.

selected by the user during capture. We then compute a Delaunay triangulation over the θ and ϕ dimensions of our projected points and apply the resulting graph to the original W_I. [2]

In some light fields captured in environments with obstacles or captured by inexperienced users we found that the distances between input views and the subject could vary discontinuously. When rendering views very close to the input images this may cause self occlusions in the projection of the viewpoint manifold, creating sharp discontinuities in the blending field. In these cases it is useful to smooth the radial coordinate of each viewpoint until the self occlusions are gone. We iteratively change the distance of a vertex from the subject center to be 0.75 times its current distance and 0.25 times the average subject distance of its neighbors. In practice most light fields needed no smoothing at all, and even challenging light fields can be fixed with minor smoothing (3-5 iterations). Note that this smoothing only affects the computation of the blending field and not the computation of texture coordinates, so that perspective remains geometrically accurate.

[2] In cases with full 180 degrees of coverage around a subject this can create a seam in the viewpoint manifold. This issue can be resolved by replacing Shewchuck's code with an implementation of spherical Delaunay triangulation.

5.3 Subdivision rendering

We can extend our piecewise-linear reconstruction and use subdivision over the triangulation of viewpoints to achieve a smooth reconstruction of the camera blending field that is piecewise-bicubic at the limit. For this, we extend the influence region of a given stored view W to the projection of its 2-ring. To compute the weights of W across this influence region we first assign the vertex corresponding to W with a weight of 1 and the other vertices with 0. We then apply the Loop subdivision rules [18] as modified by Warren [31] to the topology of the 2-ring and these weight values. Note that, unlike traditional subdivision surfaces, we do not modify the geometry of the subdivided mesh. We instead use the subdivision rules on the vertex weights to compute a smooth blending field that converges to a bicubic function if iterated until convergence [27].

We perform this process independently for the 2-ring of each image. We then render all the subdivided triangles with their weights as alpha values. This means that a given triangle is rendered multiple times, but this does not noticeably affect performance. In practice our implementation applies only one subdivision step, but more could be performed, or, better, closed-form formulas could be used within each triangle after one step [27]

5.3.1 Rendering with geometry

Our algorithm separates the computation of a camera blending field from the choice of a geometric proxy. This is done by rendering two passes. The first pass renders a depth map corresponding to a geometric proxy to determine which pixels to sample from each input image. The second pass computes the blending field as described above and renders the final image. This separation makes it easy to use different kinds of geometric proxies.

We tested three types of proxies. The first is a simple plane of focus as used in [16]. While this choice of proxy tends to exhibit ghosting and blurring away from the depth of focus, this blurring appears relatively natural compared to other types of

artifacts as it resembles a shallow depth of field.

The second type of proxy consists of a screenspace triangulation of the SLAM feature points. This is a view dependent proxy similar to the one used in [4]. When this proxy works well it produces much better results than a plane of focus. However, such proxies sometimes produce temporal artifacts when the topology of the screenspace triangulation changes, and since there is no notion of occluding surfaces in the triangulated point cloud there are often highly objectionable artifacts in scenes with large occlusions.

The third type of proxy is a world space triangle mesh. We compute this using multiview stereo ([6]) paired with Poisson surface reconstruction as in [7]. This is probably the most popular type of proxy in related work. When it fails it sometimes produces artifacts similar to a screenspace triangulation, but when it works well it often produces the best looking results.

In general, geometry produces better results than a simple plane of focus only if the geometry is very accurate. Even if such accurate geometry is not available a plane of focus provides a basic standard of quality for all light fields.

5.4 Discussion and comparison to the unstructured lumigraph

The algorithm described above focuses on smooth and scalable reconstruction, while unstructured lumigraph rendering [4] deals better with low viewpoint density and, in particular, 1D datasets. We require a triangulation of the set of input viewpoints, which implies dense coverage. In contrast, unstructured lumigraph rendering makes no such assumption and relies on k-nearest-neighbor reconstruction. Unstructured lumigraph degrades gracefully when reconstructing outside the convex hull of the input viewpoints, while our method may have poor-aspect-ratio triangles at the boundary of the set of views. As future work, we consider the insertion of virtual points outside the convex hull of our triangulation to better leverage the information around the

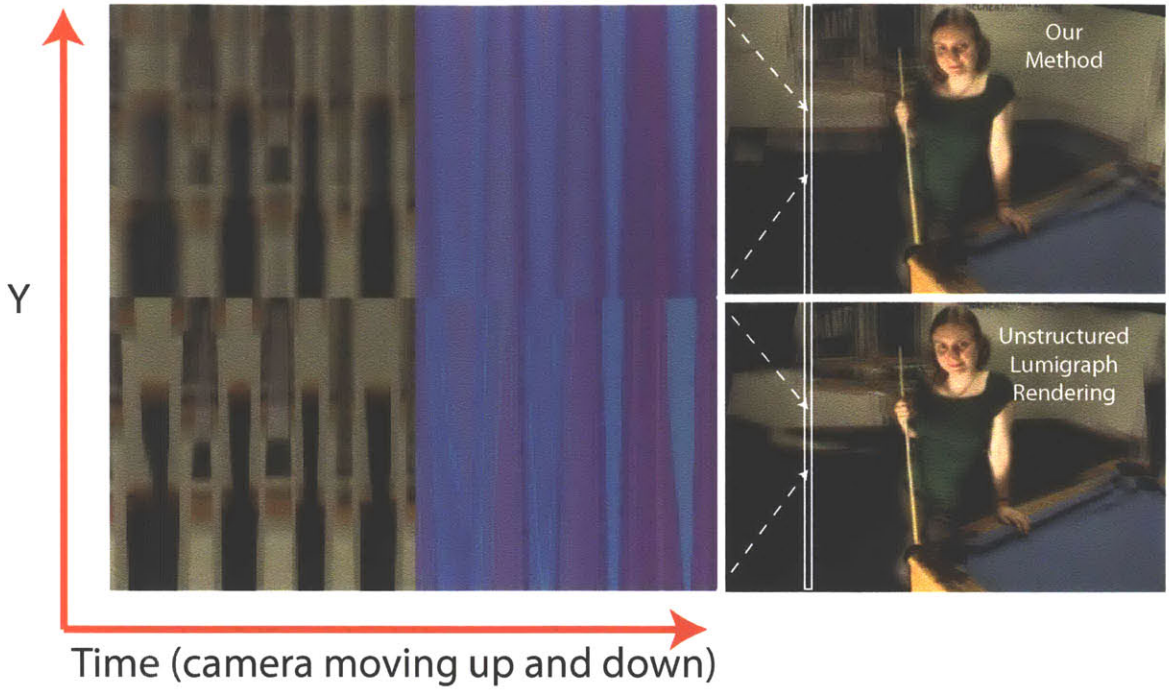


Figure 5-4: A comparison of our rendering method (top) to the unstructured lumigraph (bottom) over time. Each column of the images on the left is taken from a different frame of the cue example in our video. Since the camera motion is up and down, we expect the images on the left to look like sine waves. The stair-stepping artifacts on the left correspond to temporal artifacts in the video. We can see that these artifacts are more pronounced in the unstructured lumigraph.

boundary.

Unstructured lumigraph rendering has problems when the set of viewpoints is large. First, the computation of k nearest neighbors may become prohibitive. For example, we found that our implementation of unstructured lumigraph rendering is 50 times slower (8.4Hz vs. 430Hz) than our subdivision technique for a medium-density light field of 183 viewpoints and using 2,000 triangles for the acceleration of the blending weights for the unstructured lumigraph. For the same scene, our piecewise-linear rendering is 20% faster than the subdivision method.

Second, k-nearest-neighbor reconstruction results in more discontinuities than our piecewise-bicubic approach. In particular, consider a set of viewpoints such as those shown in Fig. 5-1(a). The density of samples in the horizontal dimension is higher

than in the vertical dimension. This is common in light fields captured by hand because it is easier for a user to cover a dense set of views along the path of a camera than it is to connect paths for coverage in a second dimension. Unstructured lumigraph rendering computes the blending field at point V as a weighted average of the k-nearest neighbors at V. Since sampling in the horizontal direction is denser, there will be more nearest neighbors in this direction. As a result, a point in the blending field uses a wide footprint in the direction of dense sampling and a narrow one in the direction of sparse sampling. This leads to excessive blur in the direction of dense sampling and discontinuities in the direction of sparse sampling (see Fig. 5-1(a)). The blurring is not as noticeable because it happens where sampling is already dense. The discontinuities, however, lead to serious temporal artifacts (Fig. 5-4). In contrast, our reconstruction leverages the topology of the input samples via the triangulation and can smoothly interpolate in both the vertical and horizontal direction. The resulting blending field (Fig. 5-1(c)) is smoother than that of unstructured lumigraph rendering (Fig. 5-1(b)). The improvement is best seen under camera motion, as demonstrated by our video.

While unstructured lumigraph rendering also relies on a triangulation, it is used only as a way to reduce the cost of k-nearest-neighbor computation and not to exploit the topology of the set of viewpoints for smoother interpolation.

In a nutshell, the limitations of our method are due to the need for a triangulation, but its strength comes from the triangulation, which allows us to use a spatially-smoother reconstruction. We use unstructured lumigraph rendering for sparse datasets, in particular 1D sets of views, and our viewpoint-subdivision rendering for denser datasets such as those produced by our capture approach.

5.5 Wide-aperture rendering

Our piecewise-linear and subdivision rendering methods seek to achieve the sharpest possible reconstruction. We can, however, extend them to perform wide aperture rendering and create depth of field effects, which is well know to address aliasing in

the background of light fields [16, 5, 12]. (Fig. 5-2(d)) shows the geometry of wide aperture rendering. A given input image contributes all the rays that originate from it and go through the aperture. This corresponds to projecting the image from a stored viewpoint W onto the new focal plane from W and through the aperture. Note that the projection is now with respect to each captured viewpoint, and that the location of the focal plane determines the part of the scene in focus.

We observe that when this projection is smaller than the 1-ring of W, the requested aperture is too small for the viewpoint sampling rate. In a sense, our triangulation-based rendering uses, for each stored image, the smallest possible aperture given the viewpoints samples. On the other hand, when the aperture is larger, we need to increase the region of influence of W in the image to cover the full projection of the aperture. Our method simply warps the 2-ring radially in a vertex shader so that the boundary vertices are warped to the projection of the outer circle of the aperture (Fig. 5-2(d)). An advantage of this method is that vertices are warped individually, handling even the case where some vertices are inside the projection of the aperture while others are outside. Inner vertices are warped according to their relative topological distance to W.

In the case of wide-aperture rendering, the weights at a pixel are no longer normalized, so we perform an additional rendering pass for normalization.

Chapter 6

Results

We have used our system to capture light fields with a webcam and with a DSLR. The number of views varies from tens to about a thousand, which compares favorably to a camera array (e.g. 100 cameras for the Stanford array [32]), and typical capture time varies from under a minute to 11 minutes (Table 6.1 and Fig. 6-1). To assess the range of available camera motions, we compute the angular range as the maximum horizontal and vertical angle between the center of the subject and the camera locations used for capture. We also compute the average length of the edges in our viewpoint triangulation, expressed in degrees from the object center, as an indication of viewpoint density.

Pose estimation initialization takes between a few seconds to about a third of the capture time for difficult scenes. Most of the time, we use the augmented aperture visualization when specifying the distance to the subject because it is more precise and does not require the user to move as much. We typically review the light field once or twice per capture to assess the range of motion available. The light fields that were captured in under 2 minutes came from users who were asked to capture their scene in that amount of time. All other light fields were captured at the user's leisure under no instruction concerning speed.

We use 7 pixels as the coverage reprojection threshold for high-resolution capture such as Keyboard, Face, Couch, Armchair and 12 pixels for faster capture. For a scene like the Keyboard, we place the sphere to bound the torso of the musician, and

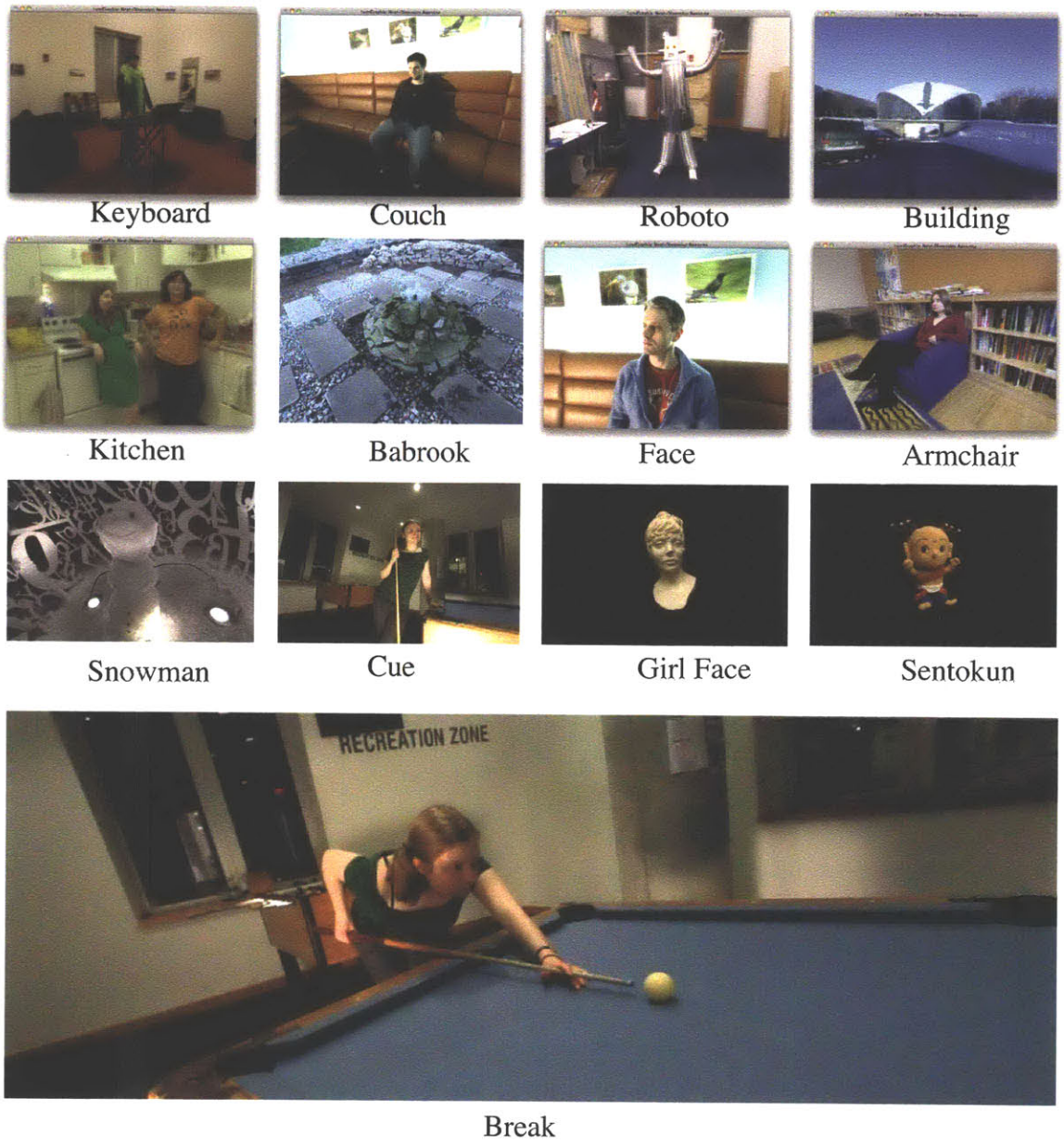


Figure 6-1: Images from the light fields described in table 6.1.

the depth range of the character is conservatively over-estimated, which means that reprojection error is usually less than the bound.

Most light fields were captured by the authors. However, novices have also found the system easy to use. The Couch scene was acquired by a novice user who had no problem achieving a large angular range (95x60 degrees) with good coverage. Moving

Scene name	# views	Capture time (min)	angular range (degrees)	average angle
Keyboard	800	8	77x31	2.02
Couch	842	10	95x60	2.19
Roboto	549	4	77x45	2.51
Building	48	3	50x2	3.23
Kitchen	240	6	90x47	4.67
Babrook	166	2	64x36	4.21
Face	691	5	80x53	2.67
Armchair	573	11	127x 77	3.39
Snowman	423	9	82x31	2.67
Cue	183	<1	64x46	4.5
Girl Face	261	2	123x65	6.10
Sentokun	133	<1	95x48	6.73
Break	190	<1	61x31	3.64

Table 6.1: Statistics for a small subset of light fields captured with our system.

the camera while looking at our visualization is easy because our visual feedback is overlaid on the live view and the user does not need to look at the scene directly.

The view-dependent effects afforded by light fields can be best assessed in video and so we refer the reader to our supplementary material. We demonstrate the range of viewpoint motion enabled by our captured data, including rotations around the center of the object and forward translation. We also implemented a "Vertigo" effect where the camera moves forward as it zooms out. We demonstrate translations that reduce the distance to the center of the object by a factor of 2 to 2.5, for e.g. the girl face, Babrook, and bush scenes.

Limitations and discussion Pose estimation needs distinct image features that are stable across a range of viewpoints. This sometimes requires the capture of a wider field of view than desired to include enough such features. In fact, stable features often come from the background rather than the object itself - especially in light fields with human subjects, transparent subjects, or highly specular subjects.

Errors in pose estimation can result in popping artifacts during rendering. A

final global bundle adjustment might help fix this, but would be expensive given the number of images in larger light fields.

Our sampling criterion relies on a Lambertian assumption and specular objects might be undersampled. We have nonetheless captured highly specular objects such as the refractive glass sphere example in our video, but the highlights do not move smoothly, as their apparent depth is outside of the bounding sphere. Nevertheless, our light fields give a strong impression of high-frequency view-dependent effects such as reflections and refraction.

The mobile phone prototype is limited in scale by the need for a fiducial in the scene similar to the original lumigraph. Larger scenes would require fiducials of a size that is not compatible with a lightweight approach. Furthermore, because the fiducial is flat, it limits the precision and range of pose estimation. The limited memory and bandwidth of the device also reduce the amount of data that can be recorded.

User mobility can limit the range of captured viewpoints. This can be due to the large scale of a scene, which might limit relative vertical mobility. This can also be due to clutter in the scene and occlusion and is inherent to hand-held capture.

Many of the current problems with our system have more to do with the challenges associated with visual SLAM than with our system's design. Improvements in real time pose estimation over time will make our system even more robust.

We have focused on the capture of dense light fields of static scenes and our subject needs to remain sill during capture. We have nevertheless captured light fields of humans (in comfortable poses). A camera array solution [32] has the advantage that all views can be taken at the same time, a critical aspect when capturing dynamic scene. In comparison, our approach is lightweight and can be run on portable commodity hardware. Furthermore, the range of viewpoints enabled by a camera array is often limited, even with reconfigurable solutions. Our range of viewpoints is limited only by the user.

Chapter 7

Conclusions

Our system allows a user to capture and render light fields with a commodity hand-held camera and a laptop. We address the key challenge of achieving dense coverage over two dimensions of parallax. For this, we derive a geometric error criterion that accounts for all four dimensions of the light field in terms of reprojection error. We use this criterion to decide which viewpoints are well covered and visualize, in real-time, a coverage map that helps the user assess where they need to move the camera. Real-time feedback is critical to enabling hand-held capture of a complex function such as a light field. We introduce a new rendering algorithm that is scalable and leverages the dense yet unstructured datasets generated by our capture process. Using a triangulation of the set of captured viewpoints, we show how subdivision rules can be applied to achieve smooth reconstruction. Our system is portable and can be used to capture light fields of real scenes that range from miniature size to buildings. We hope that this approach will make it easy for a wide audience to capture light fields.

Bibliography

[1] E. H. Adelson and J. Y. A. Wang. Single lens stereo with a plenoptic camera. *IEEE PAMI*, 14(2):99–106, February 1992.

[2] Daniel G. Aliaga, Thomas Funkhouser, Dimah Yanovsky, and Ingrid Carlbom. Sea of images. In *Proc IEEE VIS-02*, pages 331–338, 2002.

[3] A. Blake, P. H. S. Torr, I. J. Cox, and A. Criminisi. Estimating uncertainty in dense stereo disparity maps. Technical Report MSR-TR-2003-93, Microsoft Research (MSR), March 2003.

[4] Chris Buehler, Michael Bosse, Leonard McMillan, Steven Gortler, and Michael Cohen. Unstructured lumigraph rendering. In *SIGGRAPH '01*, pages 425–432. ACM, 2001.

[5] Jin-Xiang Chai, Xin Tong, Shing-Chow Chan, and Heung-Yeung Shum. Plenoptic sampling. In *SIGGRAPH 2000*, pages 307–318, 2000.

[6] Yasutaka Furukawa and Jean Ponce. Patch-based multi-view stereo software. http://grail.cs.washington.edu/software/pmvs.

[7] Yasutaka Furukawa and Jean Ponce. Accurate, dense, and robust multi-view stereopsis. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 2009.

[8] Todor Georgiev and Andrew Lumsdaine. Focused plenoptic camera and rendering. *J. Electronic Imaging*, 19(2):021106, 2010.

[9] Todor Georgiev, Ke Colin Zheng, Brian Curless, David Salesin, Shree Nayar, and Chintan Intwala. Spatio-angular resolution tradeoffs in integral photography. In

Tomas Akenine-Möller and Wolfgang Heidrich, editors, *Proc. Eurographics Symposium on Rendering Techniques, 2006*, pages 263–272. Eurographics Association, 2006.

[10] Steven J. Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F. Cohen. The lumigraph. In *SIGGRAPH '96*, pages 43–54, New York, NY, USA, 1996. ACM.

[11] B. Heigl, R. Koch, M. Pollefeys, J. Denzler, and L. Van Gool. Plenoptic modeling and rendering from image sequences taken by a hand-held camera. In *DAGM*, pages 94–101, 1999.

[12] Aaron Isaksen, Leonard McMillan, and Steven J. Gortler. Dynamically reparameterized light fields. In *SIGGRAPH 2000*, Annual Conference Series, pages 297–306, 2000.

[13] H. Kato, M. Billinghurst, B.Blanding, and R. May. ARToolkit. Technical report, Hiroshima City University, December 1999.

[14] Georg Klein and David Murray. Parallel tracking and mapping for small AR workspaces. In *Proc. (ISMAR'07)*, Nara, Japan, November 2007.

[15] R. Koch, M. Pollefeys, B. Heigl, L. Van Gool, and H. Niemann. Calibration of hand-held camera sequences for plenoptic modeling. In *ICCV*, pages 585–591, 1999.

[16] Marc Levoy and Pat Hanrahan. Light field rendering. In *SIGGRAPH '96*, pages 31–42, New York, NY, USA, 1996. ACM.

[17] Christian Lipski, Christian Linz, Kai Berger, Anita Sellent, and Marcus Magnor. Virtual video camera: Image-based viewpoint navigation through space and time. *Computer Graphics Forum*, 29(8):2555–2568, December 2010.

[18] Charles Loop. Smooth subdivision surfaces based on triangles. Master's thesis, University of Utah, Department of Mathematics, 1987.

[19] J. Maver and R. Bajcsy. Occlusions as a guide for planning the next view. *IEEE Trans. Pattern Anal. Mach. Intell.*, 15:417–433, May 1993.

[20] Ren Ng, Marc Levoy, Mathieu Bredif, Gene Duval, Mark Horowitz, and Pat Hanrahan. Light field photography with a hand-held plenoptic camera. Technical Report CSTR 2005-02, Stanford University Computer Science, 2005.

[21] R. Pito. A sensor-based solution to the next best view problem. In *Proc. ICPR '96*, ICPR '96, pages 941–, Washington, DC, USA, 1996. IEEE Computer Society.

[22] M. Pollefeys and S. N. Sinha. Iso-disparity surfaces for general stereo configurations. In *ECCV*, pages Vol III: 509–520, 2004.

[23] Szymon Rusinkiewicz, Olaf Hall-Holt, and Marc Levoy. Real-time 3D model acquisition. In John Hughes, editor, *SIGGRAPH 2002*, pages 438–446. ACM Press/ACM SIGGRAPH, 2002.

[24] Jonathan Richard Shewchuk. *Delaunay Refinement Mesh Generation.* PhD thesis, CMU, 1997. T. Report CMU-CS-97-137.

[25] Noah Snavely, Rahul Garg, Steven M. Seitz, and Richard Szeliski. Finding paths through the world's photos. In *SIGGRAPH '08*, pages 1–11, New York, NY, USA, 2008. ACM.

[26] Noah Snavely, Steven M. Seitz, and Richard Szeliski. Photo tourism: exploring photo collections in 3d. In *SIGGRAPH '06*, pages 835–846, New York, NY, USA, 2006. ACM.

[27] Jos Stam. Exact evaluation of catmull-clark subdivision surfaces at arbitrary parameter values. In *SIGGRAPH*, pages 395–404, 1998.

[28] Yuichi Taguchi, Amit Agrawal, Ashok Veeraraghavan, Srikumar Ramalingam, and Ramesh Raskar. Axial-cones: modeling spherical catadioptric cameras for wide-angle light field rendering. *ACM Trans. Graph*, 29(6):172, 2010.

[29] Ashok Veeraraghavan, Ramesh Raskar, Amit K. Agrawal, Ankit Mohan, and Jack Tumblin. Dappled photography: mask enhanced cameras for heterodyned light fields and coded aperture refocusing. *ACM Trans. Graph*, 26(3):69, 2007.

[30] B. Volpel and W.M. Theiner. Localization uncertainty in area-based stereo algorithms. *Systems, Man and Cybernetics, IEEE Trans.*, 25(12):1628 –1634, December 1995.

[31] Joe Warren. Subdivision methods for geometric design, December 06 1995.

[32] Bennett Wilburn, Neel Joshi, Vaibhav Vaish, Eino-Ville Talvala, Emilio Antunez, Adam Barth, Andrew Adams, Mark Horowitz, and Marc Levoy. High performance imaging using large camera arrays. *ACM Trans. Graph.*, 24(3):765–776, 2005.