Software Simulation of Depth of Field Effects in Video from Small Aperture Cameras

by

Jordan Sorensen

B.S., Electrical Engineering, M.I.T., 2009

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

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2010

Author .. ..
Department of Electrical Engineering and Computer Science
May 21, 2010

Certified by .....
Ramesh Raskar
Associate Professor, MIT Media Laboratory
Thesis Supervisor

Accepted by .....
Dr. Christopher J. Terman
Chairman, Department Committee on Graduate Theses

Software Simulation of Depth of Field Effects in Video from Small Aperture Cameras

by

Jordan Sorensen

Submitted to the Department of Electrical Engineering and Computer Science
on May 21, 2010, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

This thesis proposes a technique for post processing digital video to introduce a simulated depth of field effect. Because the technique is implemented in software, it affords the user greater control over the parameters of the effect (such as the amount of defocus, aperture shape, and defocus plane) and allows the effect to be used even on hardware which would not typically allow for depth of field. In addition, because it is a completely post processing technique and requires no change in capture method or hardware, it can be used on any video and introduces no new costs. This thesis describes the technique, evaluates its performance on example videos, and proposes further work to improve the technique's reliability.

Thesis Supervisor: Ramesh Raskar
Title: Associate Professor, MIT Media Laboratory

Acknowledgements

Thanks to Ramesh Raskar for his guidance and suggestions as my thesis advisor, and for his help in brainstorming research ideas. Thanks to the rest of the Camera Culture group for allowing me a glimpse of the research and publication process, and for everything I have learned from them over the past year and a half.

I would also like to thank Sharmeen Browarek for her help formatting this document and for her assistance in navigating the world of M. Eng theses at MIT.

Thanks to Erik, Jeff, and all my friends for their constant support throughout my MIT career.

Finally, thanks to my family for helping me survive graduate life. Curt and Laura, your visit helped me relax when I was most worried. Mom and Dad, your care packages kept me well fed while I wrote this thesis and your weekly calls reminded me that someone was thinking of me.

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation for Simulating Depth of Field

The past decade has witnessed an explosion of pro-sumer digital cameras, cameras which bridge the gap between affordable point and shoot cameras, designed for casual use, and expensive professional devices. Digital SLRs, by far the most popular of these cameras, can be purchased for as little as a few hundred dollars and offer their owners quality which approximates the quality of devices costing thousands of dollars. As a result, the digital camera market offers a continuous scale of increasing quality for increasing price.

The market for digital camcorders, on the other hand, offers no such smooth scale. Buyers trying to purchase one of these will find themselves deciding between very cheap and very expensive devices with very little middle ground. And the separation in quality is equally vast. While consumer level still cameras frequently capture professional quality shots, consumer level video is never mistaken for video from a professional device.

In addition, a recent boom in the popularity of smartphones has placed digital cameras and camcorders into the hands of millions of consumers. These integrated devices typically have very limited hardware capabilities. As a result, the quantity of digital video captured has exploded, while the quality remains relatively low.

New software techniques can bridge the divide between low quality video captured

Figure 1-1: Can a cheap consumer camera, like the one on the left, capture video featuring advanced effects like the expensive professional camera on the right?

by these cheap integrated camcorders and higher quality video captured by expensive professional devices. By simulating effects normally achieved only by more expensive hardware and by eliminating noise introduced by low quality hardware, these techniques can significantly increase the utility of these cheap capture devices.

1.2 Proposal

This thesis proposes a method for applying an artificial depth-of-field effect to captured video. That is, given a video in which the entire scene is in focus, the proposed technique allows the user to specify a focal plane (to control what parts of the scene are in focus), an aperture shape, and an aperture size (to control the degree of defocus applied to out of focus objects), and renders a new video which estimates the video which might have been captured by a device with the given parameters.

Such a system can improve consumer video capture in two important ways. First, it would make a feature usually restricted to professional camcorders available to consumer devices. Consumer devices are restricted to using small image sensors and small apertures due to both price and size constraints. As a result, these devices typically exhibit infinite depth of field, and are unable to focus on only part of a scene. Professional devices, on the other hand, allow for larger apertures, and as a result, can capture depth of field effects. By simulating these effects in software, they

would be made available to video captured on any hardware.

Second, an artificial depth of field effect would help to mask the quality difference between capture devices. Because of their smaller sensor size and lower sensor quality, cheaper camcorders tend to produce grainy or otherwise noisy frames. By keeping only the scene's point of interest in focus and blurring the rest of the frame, a depth of field effect could help to mask this noise.

The ability to simulate depth of field effects in software would benefit all video, including video captured on more expensive hardware for which hardware capabilities are not a problem. A software solution allows the user to refocus the image at will. If the depth of field were captured through regular means at the time of video capture, the effect would be static. Once it was captured, the user would be unable to adjust its parameters. With an artificial depth of field, on the other hand, the user would be free to try many different effects using the same captured video. In addition, it may allow them to capture an effect which would otherwise be too difficult or require complicated hardware setups. For example, if the user's goal is to keep a given object in focus while the camera moves towards or away from that object, they may find it difficult to change the camera's focal plane at exactly the right rate to keep this object in focus.

1.3 Goals of Method

In order to fully realize the benefits outlined above, the proposed process must meet a number of requirements. Since a primary aim of the project is to make high quality video available to consumers, a process which requires hardware modifications is undesirable. Such modifications would be unlikely to be implemented (at least in the short term) by manufacturers, and would drive the cost of the devices up. Consumers would be hesitant to make modifications to hardware themselves, decreasing the likelihood that the technique achieves wide spread adoption. A technique which aims to capitalize on the widespread availability of cheap capture devices must require no modifications to hardware.

In addition, any technique which requires significant effort on the part of the user or significant deviation from their normal routine is unlikely to be adopted. In other words, an effective technique must not only require no hardware modifications, but no changes to the capture process at all. In addition, the post processing steps should require little intervention on the user's part. An effective technique will take as input only the required inputs the desired focal plane and desired aperture size (or some other measure of the degree of defocus) and will automate all other processing.

Finally, in order to be useful in any field of video editing, be it consumer level or professional, a proposed technique must be robust. If a technique works only on a very narrow subset of videos, or only works sporadically, it is unlikely to be useful for most applications.

1.4 Structure of Thesis

- Chapter 1 provides motivation for the technique, describing its potential applications and outlining goals by which to judge the technique's success.
- Chapter 2 introduces the simple camera and lens models which provide a theoretical underpinning for the technique. It goes on to describe research in similar problem areas.
- Chapter 3 describes the method itself.
- Chapter 4 presents and analyzes the results of applying the technique to sample videos.
- Chapter 5 discusses potential improvements, modifications, and extensions to the technique.
- Chapter 6 concludes the thesis, summarizing the method and evaluating its success.

Chapter 2

Background on Depth of Field and View Synthesis

The following chapter explains two simple models for modeling a camera. It discusses how the two models are related and how their relationship is relevant to the proposed technique. It goes on to explore approaches other researchers have taken to similar or related problems.

2.1 Modeling a Camera

To physically justify a technique for simulating depth of field, it's necessary to understand what causes such an effect. The sections that follow first introduce a model (the pinhole camera model) which does not exhibit depth of field, then explains what extensions are necessary to allow for such effects. It then describes how a pinhole camera model can be used to approximate cameras which do exhibit depth of field.

2.1.1 Pinhole Camera Model

In a pinhole camera model, light travels along a straight line and enters the camera through an infinitely small hole. The entire image captured by the camera is from light which passes through this hole. In a pinhole camera model, a single point in the

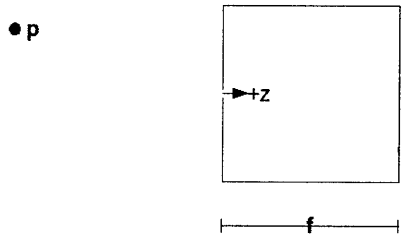

Figure 2-1: Pinhole camera capturing point.

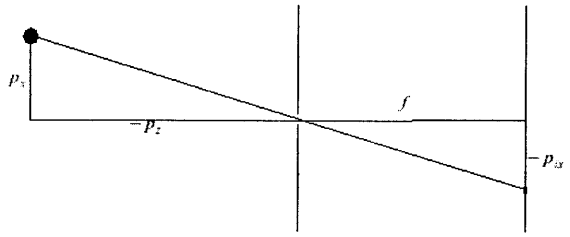

Figure 2-2: Finding point's image location using similar triangles.

input scene maps to a single point on the image. In contrast, a camera exhibiting depth of field maps a point to a finitely large area on the image (the area over which that point is blurred). Therefore, a pinhole camera model cannot model the effects of depth of field. The image captured by a pinhole camera is a simple projection of the scene onto the image plane.

Say we have a pinhole camera capturing a scene, as shown in figure 2-1. The camera's pinhole is located at the origin, and the camera is looking down the $-\overrightarrow{z}$ axis. The plane onto which the camera is capturing an image is located at $z = f$, and we would like to model how a point $\overrightarrow{p}$ falls on the image.

Since all light entering a pinhole camera enters through the pinhole and travels in a straight line, the line connecting the point $\overrightarrow{p}$ to its location on the image must pass through the origin. We know the location of the point in the scene $\begin{bmatrix} p_x \\ p_z \end{bmatrix}$, and we know the distance from the pinhole to the image plane, f. We can use similar triangles to find the location of the point on the image plane p_{ix}.

$$\frac{-p_{ix}}{f} = \frac{p_x}{-p_z}$$

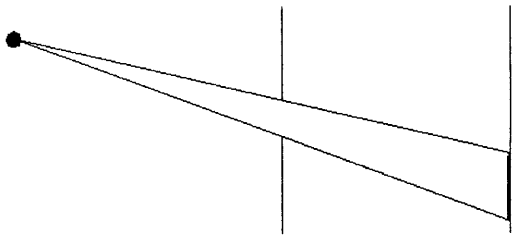

Figure 2-3: Increasing the size of the aperture to a finite area.

$$p_{ix} = f\frac{p_x}{p_z}$$

In the more general case, there is a pinhole camera at some known location in the world with some known orientation. We can reduce such a situation to the one just described by applying a simple transformation to the coordinates of the scene to put them in a coordinate system in which the pinhole is at the origin and the camera looks down the $-\overrightarrow{z}$ axis.

2.1.2 Finite Aperture Model

In reality, cameras are not pinhole cameras. Even if it were possible to build a camera with an infinitely small opening, such a camera would suffer from diffraction effects not exhibited by our pinhole model. In addition, a pinhole camera would not allow enough light through to capture good images. Because of these factors, real cameras have a finitely large hole. This finite area is known as the aperture.

Increasing the size of the aperture, however, allows light rays from a single point in the scene to fall on many points in the image, resulting in a blurry image as shown in Figure 2-3. To alleviate this effect, a lens is placed in the aperture to focus light rays to a single point. With this lens in place, a point at a distance d from the lens emits rays in all directions. All these rays converge at a distance i on the other side of the lens, as shown in Figure 2-4. The focal length f is a property of the lens itself. Given a lens of focal length f, light rays emanating from a distance d from the lens will converge at a distance i on the opposite side of the lens, as governed by the thin

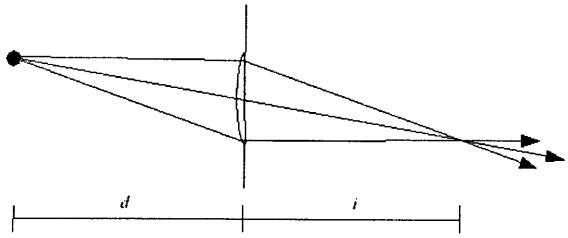

Figure 2-4: A lens focuses light rays at a point.

lens formula:

$$\frac{1}{d} + \frac{1}{i} = \frac{1}{f}$$

Therefore, if we are taking a picture of an object a distance d from the lens and we have a camera with focal length f, we should place the image sensor a distance i behind the lens. If we can achieve this, our object will be captured in focus. If, however, our image sensor were at a different depth, our point would map not to a point on the image, but to an area of finite size. This area is known as the circle of confusion (when the lens aperture is circular, the area formed on the image by a point in the scene is also a circle). The further the image plane is from its desired distance, the larger the circle of confusion. This is the principle behind the depth of field effect an object that is at exactly the right distance from the lens is captured perfectly in focus, while objects further from or closer to the lens are out of focus and appear blurry in the image. Figure 2-5 demonstrates this effect.

2.1.3 Modeling a Finite Aperture Camera Using a Pinhole Camera Model

This model is a much more complex camera model than the pinhole model first described. However, it is possible to simulate the effects of a camera with finite aperture using many pinhole cameras.

Notice that the circle of confusion arises from the fact that a ray of light may travel many different paths from the source to the image, passing through any point on the lens. However, if we pick a given point on the lens, there exists only one path

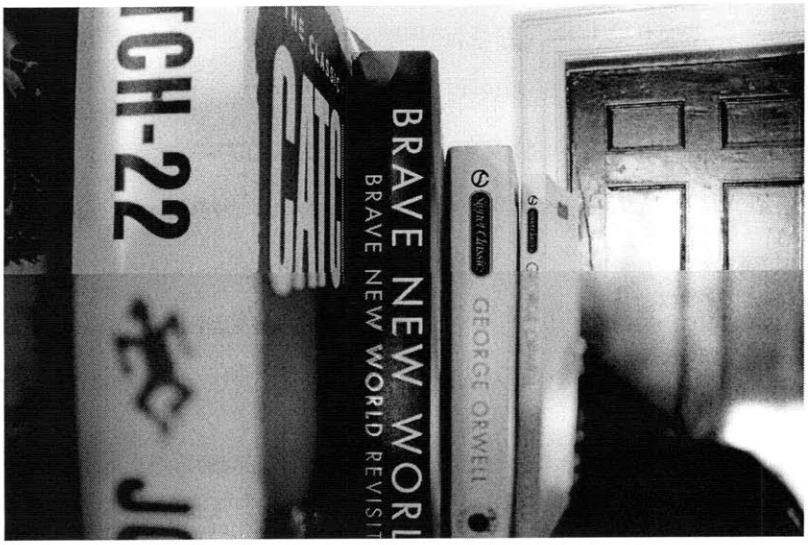

Figure 2-5: An image captured using a large aperture (bottom) together with the same image captured using a tiny aperture.

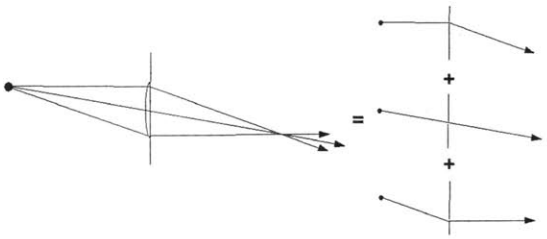

Figure 2-6: A lens can be modeled as a sum of pinhole cameras.

from the source to the image through this point. That means that each point on the lens maps a point on the source object to a point on the image plane (even when the image is out of focus) so we can model a particular point on the lens as a pinhole camera. The image captured by a camera with finite aperture size can be simulated by summing the images captured by a pinhole camera at every point on the aperture, as shown in Figure 2-6.

In the original pinhole camera model, light passed straight through the aperture. In the lens model, however, light rays passing through a single infintely small point on the lens are bent by a fixed amount as they pass through the lens. Therefore, the image capture by a pinhole camera at a point on the lens can simply be translated and scaled to reproduce the image captured by that same point on the lens.

Video captured with a consumer camcorder rarely exhibits depth of field effects. This is because the apertures of these cameras tend to be very small. In fact, they're so small that we can model these cameras fairly accurately as pinhole cameras.

Our original video gives us the image captured by a pinhole camera from that camera's position in space $\overrightarrow{p_c}$. We'd like to simulate the image captured by a camera with a finite aperture centered at $\overrightarrow{p_c}$ with some given radius. If we had not only the image captured by a pinhole camera at $\overrightarrow{p_c}$, but also the images captured by pinhole cameras at all points within the aperture area, we could easily simulate the finite aperture image by summing these pinhole images.

As a result, the problem of depth of field simulation can be reduced to the problem of generating the images captured by cameras which never actually existed. These cameras are pinhole cameras with very slight offsets from the actual camera. This is a problem which has already received significant attention, and is commonly referred to as view synthesis.

This brief introduction to simple camera and lens models has transformed the problem of depth of field simulation into the problem of view synthesis. The next section will discuss prior research into the field of view synthesis.

2.2 Video Stabilization

Video stabilization, like depth of field simulation, aims to narrow the gap between consumer level video and professional video. Casual video often includes a high degree of camera shake – high frequency movements in the camera's path. Professionally captured video, on the other hand, eliminates this shake through a variety of means – the camera is often mounted on a fixed track, or on a mobile device called a Steadicam designed specifically to reduce camera shake. As the authors of "Light Field Video Stabilization" point out, "camera shake is one of the biggest components of the large quality difference between home movies and professional videos." [18]

The connection between video stabilization and depth of field runs deeper than the common desire to improve consumer video. Video stabilization techniques must

also solve the problem of view synthesis. In video stabilization, the input is a video captured along some jittery path. The desired output is another video, but this one following a smooth path. For each frame of the input video, a video stabilization technique aims to generate a new frame as would have been captured by a camera at a slightly different position – the camera following the smooth path. We will now discuss three video stabilization papers and describe the approach each takes to the view synthesis problem.

2.2.1 Re-Cinematography

In "Re-Cinematography: Improving the Camera Dynamics of Casual Video" [6], Gleicher and Liu propose a method which breaks the input video into segments. Each segment is classified as a static segment, in which the camera moves very little, a moving segment, in which the camera moves a large amount, or a bad segment, in which not enough information can be extracted to perform good reconstruction.

In static segments, the image will be reconstructed to simulate a perfectly still camera (any camera movement will be removed) and in moving segments, the camera will be constrained to move along a smooth, goal directed path (bad segments are left as-is). Both of these tasks require view synthesis. Gleicher and Liu propose to synthesize new views from existing frames by applying a simple homography to those frames. An image transformed by a homography will accurately represent a new camera view if the new camera view is a rotation of the old view about the camera's optical axis or if the scene viewed by the camera is planar.

This notion of view synthesis, while good enough in many situations, will not be enough for the purpose of depth of field simulation. The cameras we want to synthesize involve translation from the original camera position (a translation to each point on the aperture), so our new view is not a rotation about the camera's optical axis. In addition, our problem is to model the effects of depth of field, an effect which is only visible if the scene contains multiple planes at different depths, so it is unreasonable for us to assume that the scene viewed by the camera is planar.

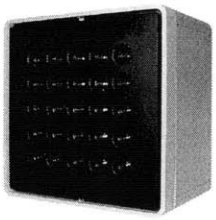

Figure 2-7: Profusion 25, a light field camera. Image courtesy Point Grey, [7]

2.2.2 Light Field Video Stabilization

Recall that if we could capture images from many pinhole cameras in a small area, we could simulate the image captured by a camera with a finite aperture covering that area. One device, a light field camera, comes very close to capturing exactly the data we are looking for. It is capable of capturing images from several tightly clustered cameras simultaneously. A light field camera, as shown in Figure 2-7, is a grid of cameras in a dense array.

In "Light Field Video Stabilization" [18], Smith, Zhang, Jin and Agarwala approach the same problem as the previous paper – view stabilization – using a light field camera. In their approach, the original video must be captured using a light field camera. They then detect edge features in each frame and compute a camera transformation for each frame that minimizes the acceleration of these features. Finally, using a depth map computed from the image array for each frame, they synthesize a new view of the image using the computed camera transformation.

By using a light field camera, this technique is able to capture all information for a single frame simultaneously, avoiding many of the difficulties associated with videos of dynamic scenes. However, the fact that a light field camera is required violates our goal of using unmodified hardware to capture video, so we cannot use this exact technique for view synthesis in our application. Although other papers have proposed methods for capture light field data by inserting a lens array into a regular camera [15], these techniques still require hardware modifications.

2.2.3 Content Preserving Warps

Content-Preserving Warps for 3D Video Stabilization by Liu, Gleicher, Jin and Agarwala [10] proposes a technique for video stabilization which does not have the scene limitations of the first technique (planar scene or camera which undergoes rotation only), and does not have the hardware requirements of the second technique (light field camera). In this technique, feature points are found in each frame, and a structure from motion algorithm estimates the 3D locations of these points and the position, rotation, and parameters of the camera which captured each frame. Then, a desired camera path is calculated (the authors describe three different possible ways to generate this path). At each from, the 3D feature points are then projected back into 2D using the position and orientation of the new desired camera for the generated path. This computes the position each of these 3D feature points would have fallen on in the image if the scene had been captured from the new camera. Finally, a warp is applied to the input frame which moves each feature point from its location in the input image to its final projected location in the new camera view.

Instead of triangulating the image and applying a straightforward warp to the image, Content-Preserving Warps breaks the input frame into a regular grid, and finds new locations for the grid points which minimize an energy function. This energy function involves not only the new locations of the projected points, but also a "content preservation" term which attempts to make the transformation for each cell as similar to a similarity transform as possible. The goal of this content preservation term is to maintain the shape of objects in the scene at the expense of absolute physical correctness.

2.3 Depth of Field Simulation for Still Images

Kusumoto, Hiura, and Sato use a similar technique to achieve artificial depth of field in images in their paper "Uncalibrated Synthetic Aperture for Defocus Control" [8]. Their goals are very similar to ours – they wish to achieve a post-processing depth of field effect using unmodified camera hardware – but their technique is designed to

work on individual images instead of video. In addition, they require the user to snap several shots of the scene from slightly different locations (but they do not require that the shots have any structure – they can be collected randomly).

Like our technique, their process simulates the view that would have been seen by several cameras within the virtual aperture and sums these images together to generate a defocused image. In their technique, cameras are first brought into a common imaging plane using a simple homography. This step corrects for the fact that the captured images may not have parallel optical axes (the cameras were not simply translated but also rotated from shot to shot). Then, feature points are detected across all images, and the image is triangulated using these points. The camera locations from which the actual images were taken are placed at their estimated locations. Finally, all desired viewpoints are generated by morphing images captured by the cameras at the actual viewpoints. For example, if a feature point is on image j at location p_j and on image k at p_k, then a viewpoint halfway between the camera for image j and the camera for image k would have that feature point at $\frac{1}{2}p_j + \frac{1}{2}p_k$, and would be colored the average of the pixel value from images j and k.

This technique is better described as view interpolation, since it attempts to generate images "between" existing images. An image morphing technique similar to the one described here was first proposed by [3], and [5] explored using similar view interpolation techniques to achieve refocusing effects. Other view interpoloation approaches include [17].

[14] explores a method which allows defocus control of both images and video. However, their system requires a specific pattern to be projected onto the scene at the time of capture. "Image Destabilization" [13] proposes a method in which the lens and image sensor are moved during capture which allows them to simulate larger aperture sizes. Since this method requires hardware changes and is designed for capturing still images, it is not suitable for our purposes. [1] proposes a method for defocus control of images which occurs completely in software. Their technique, however, is only able to magnify existing defocus; it cannot change the plane of defocus or introduce defocus in a completely sharp image.

2.4 Unstructured Lumigraph Rendering

In "Unstructured Lumigraph Rendering" [2], Buehler, Bosse, McMillan, Gortler and Cohen combine two existing view synthesis techniques into a more general framework. One view synthesis technique, light field rendering, relies on having images from regularly spaced and planar camera positions. The other technique, view dependent texture mapping, does not rely on the cameras being in a regular pattern, but instead relies on having good scene geometry. Their generalized technique can handle either case.

In their technique, they reproject the image from each camera into the new synthesized view using whatever geometry is known (or a simple plane if there is no known geometry). Then, they calculate how well each actual camera approximates the desired camera using a goal function for each pixel. This goal function has terms to account for a number of factors. For example, cameras which capture a particular scene point with higher resolution should be preferred, and cameras which capture a particular scene point at an angle very close to the desired angle should be preferred. The resultant image is a blended version of every input image, where at each pixel, each image is weighted by how well it satisfies the specified goals.

2.5 Feature Point Detection

Three of the papers described above – "Light Field Video Stabilization", "Content Preserving Warps", and "Synthetic Aperture for Defocus Control" – include a common step, feature point detection. Feature point detection is the process of identifying "interesting" points on an image. These are typically corners, edges, or points. In the papers described here (as in many applications of feature points) they're used to find correspondences between images. A feature point detector can detect that the feature point at pixel (i, j) in one image is the same as the feature point at pixel (k, l) in another image. This problem is very hard to solve – even a human might have a difficult time comparing two images and determining if they are seeing the

same scene from two different angles. Feature points detectors try to achieve this using detectors which are invariant to the types of transformations cameras undergo as they move about a scene. One common detector, the Harris affine invariant feature point detector [12], is invariant to any affine transformation (combinations of rotations, scaling, and translation) and to changes in illumination. Other methods, such as Scale Invariant Feature Transforms (commonly referred to as SIFT) have similar invariance properties [11].

Note that some view synthesis methods have been proposed [9] which do not require feature point correspondences between images. However, these typically require complex hardware or capture setups, so would be unsuitable for casual video.

2.6 Structure from Motion

Of the three papers which use feature point detectors, one of them – "Content Preserving Warps" – goes on to use those feature points to perform structure from motion. Structure from motion analyzes the correspondences between points in images – how the locations of feature points change relative to one another – to calculate 3D positions of each of the points. Obviously, this process can determine the feature point locations in an absolute coordinate system - the exact same images could be generated by a scene which was translated, scaled, or rotated in an arbitrary way – but structure from motion at least aims to detect the 3D locations of the points relative to one another.

Structure from motion is a well studied problem, and a variety of solutions have been proposed. The basic steps of most of these solutions are:

- Detect feature points
- Guess a 3D location for each feature point, and a set of parameters (location, orientation, focal length) for each camera
- Calculate error between actual 2D location of each feature point and the 2D location predicted by feature point's 3D location and camera parameters

- Adjust the location of each point in a way that lowers the error (perhaps using gradient descent or a more advanced technique)
- Iterate until error term is sufficiently low

A number of structure from motion toolkits are available. One of the better known ones is Bundler, used in Microsoft's Photo Tourism project [19]. Other tools, such as the Voodoo camera tracker by the Laboratorium fur Informationstechnologie [4], are designed for more specific purposes, such as analyis of frames from video sequences.

Chapter 3

Method for Simulating Depth of Field

Our process takes as input a video, a set of feature points which should be in focus, and the desired amount of defocus. For each frame, it uses the actual camera position as the center of the lens and synthesizes the views seen by several cameras randomly scattered over the aperture (a small circle centered on the actual camera position whose size grows with the amount of desired defocus). These synthesized views are then averaged together to produce the final version of that frame. The crucial step of view synthesis is performed using a technique which borrows heavily from both "Content-Preserving Warps" and "Uncalibrated Synthetic Aperture".

It will be useful to standardize our terminolgy to facilitate discussion of the proposed technique. Throughout this section, we will refer to the real camera which captured the original scene as the "actual camera", and the cameras which generate scattered over the aperturess "virtual cameras". The entire aperture will be referred to as the "virtual aperture". The image captured by a single virtual camera will be referred to as a "synthesized view".

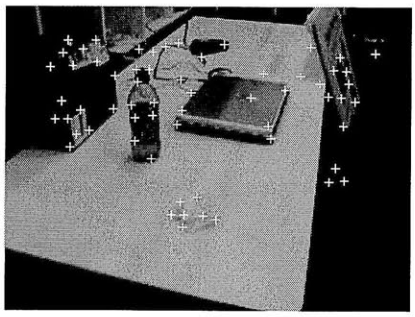

Figure 3-1: Feature points detected in one frame of test video.

3.1 Overview

First, we split the video into individual frames and perform feature point detection on each frame. After matching feature points across frames, we use off the shelf structure from motion software to estimate the 3D coordinates of each feature point. In addition, this software gives us the position, rotation, and parameters of the actual camera. Next, we ask the user to select a focal plane by selecting feature points in the first frame of the video. If the specified feature point disappears partway through the video, we ask the user to continue selecting feature points until we have at least two feature points in each frame. Next, we generate virtual camera positions in the aperture for each frame. For each virtual camera in each frame, we use that virtual camera's parameters to reproject the 3D feature points to their new locations in the synthesized view. We then triangulate the original image using its feature points, and warp each feature point to its new location in the synthesized view. Finally, we sum the resultant images to get the final image with a depth of field effect applied. We'll now explain each step in more detail.

3.2 Feature Point Detection

A variety of feature point detection tools are widely available, so we chose to use an existing solution (instead of implementing our own). Using the open source Voodoo camera tracking tool, we tried both SIFT and Harris feature point detection. We also

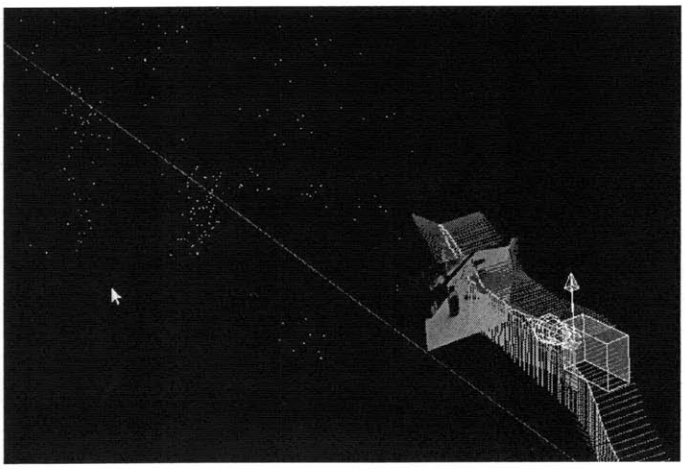

Figure 3-2: Structure from motion provides 3D feature point locations and camera information.

used the SIFT detector in the Bundler tool.

Our input is not simply a collection of photos - it is more constrained. We know that each input frame is a frame from the same video, so the camera position in each frame must be very near the camera position in the previous and next frames. We can use this information in our reconstruction. The Voodoo package takes advantage of this fact – it eliminates feature points which move too much between frames. The Bundler package, on the other hand, is designed to be used on any unsorted collection of images, so it does not take advantage of this fact.

3.3 Structure from Motion

We use the same two off the shelf packages to perform structure from motion calculation. Both use bundle adjustment to detect the 3D coordinates of the provided feature points and to estimate the parameters of the camera which captured the image. Both Bundler and Voodoo provide a means to save the 3D feature point coordinates and actual camera parameters they calculate. We have written simple MATLAB scripts to parse these files, and do the rest of our processing in MATLAB.

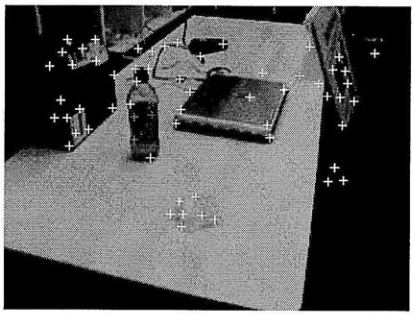

Figure 3-3: The user selectes four feature points on an object they would like to stay in focus.

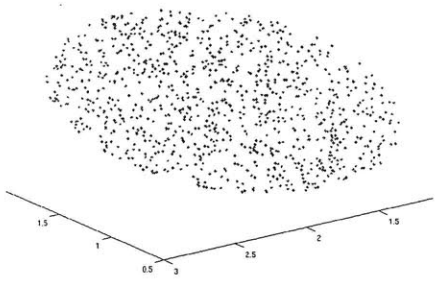

Figure 3-4: Generated virtual camera positions (dots) surrounding actual camera position (plus sign).

3.4 Focal Plane Selection

Next, we display the first frame to the user with the locations of feature points super imposed. We ask the user to select four feature points which lie in their desired focal plane. Although two points would be sufficient, we use four to average out any noise that may result from errors in the estimation of point's 3D coordinates.

3.5 Camera Generation

In addition to feature points in the focal plane, we also ask the user to specify the amount of defocus they would like to apply. Currently, a user provides this information by specifying the radius of virtual aperture they would like to simulate. This radius is defined in terms of the world coordinates of the generated feature points.

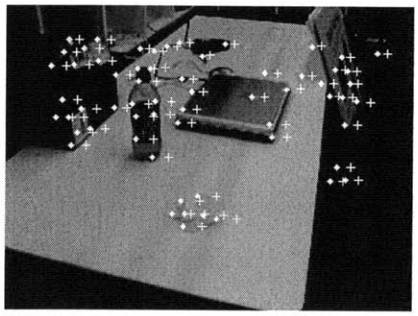

Figure 3-5: Original feature points marked with dots; projected points with crosses. Projected points are points as seen by a virtual camera slightly to left of actual camera.

While this solution is not ideal – the user has no knowledge of the size of the world the feature points were generated in – the current implementation is only intended as a test of the concept, not as a final product.

Once the user has specified the desired radius of the virtual aperture, r, we use the $\overrightarrow{x}$ and $\overrightarrow{y}$ axes of the actual camera's coordinate system to determine the plane in which the aperture should lie. Using this information, we generate some number of positions randomly distributed across a circle of radius r in the plane containing the actual camera's $\overrightarrow{x}$ and $\overrightarrow{y}$ axes, centered at the actual camera position. We allow the user to vary the number of virtual cameras they'd like to use.

Note that the figure above contains almost a thousand virtual camera positions. When actually using technique, we found as few as 20 or 30 virtual cameras was almost always sufficient. The high number of virtual cameras was used here for demonstration purposes only to make the shape of the aperture obvious from the point cloud alone.

3.6 Feature Point Projection

The next step in our technique is to project the feature points into their 2D locations in the image frame of each virtual camera in the aperture. Since our virtual cameras have parallel optical axes, they share the same $\overrightarrow{x}$, $\overrightarrow{y}$, and $\overrightarrow{z}$ axes as the original camera. These three axes are given to us by the structure from motion software. The

position of the virtual camera, $\overrightarrow{t}$, was generated in the last step. We can transform a feature point into camera space using the following transformation, where $\overrightarrow{x}$, $\overrightarrow{y}$, and $\overrightarrow{z}$ give the actual (and virtual) camera's coordinate axes, $\overrightarrow{f_w}$ gives the feature point's 3D coordinates in world space, and $\overrightarrow{f_c}$ gives the feature point's coordinates in the virtual camera space.

$$\overrightarrow{f_c} = [\overrightarrow{x}\,\overrightarrow{y}\,\overrightarrow{z}] * \overrightarrow{f_w} - \overrightarrow{t}$$

Next, we convert these camera space coordinates to image coordinates by breaking the $\overrightarrow{f_c}$ into its components and dividing by $f_{c,z}$:

$$i_x = \frac{f_{c,x}}{f_{c,z}}$$

$$i_y = \frac{f_{c,y}}{f_{c,z}}$$

If $\overrightarrow{x}$, $\overrightarrow{y}$, and $\overrightarrow{z}$ were normalized vectors, the next step would be to scale the given points according to the virtual camera's parameters, giving us normalized coordinates which range from -1 to 1 in each dimension. Finally, we would scale these coordinates back into pixel space according to the size of the original image in pixels.

However, Bundler and Voodoo both provide their output in a convenient format in which the camera parameters are baked into the lengths of the $\overrightarrow{x}$, $\overrightarrow{y}$, and $\overrightarrow{z}$ axes. We chose to use this approach rather than computing the effects of the camera parameters ourselves.

3.7 Image Warping

We now have a set of 2D feature point locations for each virtual camera in each frame. Using a Delaunay triangulation, we triangulate the input image, using the positions of the original feature points (as seen by the actual camera) as the vertices of our triangles. Next, we simply warp the image by moving each feature point to its projected location (from the last step), stretching the image as we do. This generates our synthesized view of the scene from the virtual camera.

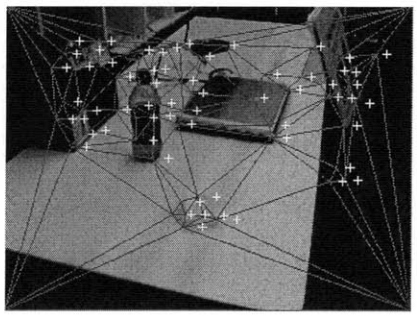

Figure 3-6: Image is triangulated. The points will then be warped to their projected locations, marked by crosses.

The feature point projection step will not give us enough points for a complete triangulation of the image. Since the feature points do not necessarily extend to every edge of the image, the triangulation will not cover the entire image. We simply add points which keep the corners of the image fixed. This solution is clearly not ideal, since the edges of the image will probably move when the camera moves. We will explore other solutions to this problem in the section on potential further work.

There is one important feature of this step worth considering. Notice that while the 3D locations of the various feature points and cameras are calculated using data from all video frames, the actual pixel data for one output frame draws only from the pixel data for the same input frame. This will hopefully allow us to gracefully handle scenes with moving objects - they will be ignored by structure from motion as outliers, so they will simply blend into the static background behind them. The fact that pixel data comes from one and only one image avoids problems such as ghosting or blurring that can be problematic in other techniques [10].

3.8 Image Matching

We now have a set of images captured by virtual pinhole cameras scattered about the virtual aperture. Recall from our discussion in the background section that an image captured by a pinhole camera at a point on the lens is the same as the image captured through that point on the lens with an additional scaling and translation.

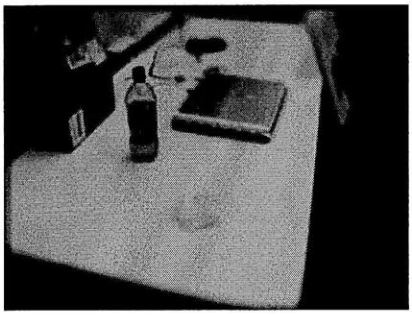

Figure 3-7: Final averaged result of one frame of test video.

In addition, recall that our original image was at the center of the lens, which allows light to pass straight through (the center of the lens does not bend incoming light). This implies that the image captured by our original image (captured by a pinhole camera) is the same as the image captured through the center of the lens. Also recall that any point source imaged on the focal plane maps to exactly one point on the focal plane. Therefore, all feature points on the focal plane must map to the same position that they do in the original image. For each generated image, we will generate a scaling and translation which exactly maps the feature points selected by the user to the original locations of those feature points in the image as seen by the actual camera.

3.9 Image Averaging

After scaling and translating the synthesized views, we simply sum them together and divide by the number of images to generate the final output image.

Chapter 4

Results of Simulation Method

4.1 Hardware Used

Input data was captured using two different devices. One, a Canon Powershot A40, was chosen to represent a low end video capture device. This digital camera can capture video at 20 fps with a resolution of 320x240. Note that this is an older device; most consumer devices now offer better resolution and image quality. The second device was a Canon Digital Rebel XS with EF-S 18-55 mm lens. This device does not capture video, but was used to capture still images at a resolution of 3888x2592. These images were shrunk to 972x648 before being processed in order to speed processing. Shots taken with this camera were taken with very slight camera movements between each shot to simulate the effect an actual video. This device was used for "ground truth" comparisons – the ability to take both an image with pinhole like aperture, and the same image with a larger aperture, and compare the actual large aperture image with the generated image.

4.2 Test Videos

A number of video sequences were used to test and analyze the system. Two of them are used throughout this analysis. One, a short ten second walkthrough of our group's lab space, was captured using the Powershot camera. It is a handheld video,

and the primary motion is orthogonal to the image plane (the camera is moving into the scene). It totals 255 frames. The other, a sequence of images of two cake boxes on a table, was captured using the Rebel XS. It is not a video, but a series of images with only small camera movements in between. It too features mostly movement orthogonal to the image plane. The camera was resting on the table for capture (it was not hand held). It totals 37 frames. The 37 frame sequence was captured using a very small aperture to simulate a pinhole camera. For a number of frames, "ground truth" depth of field images were also captured using larger apertures. The exact dimensions of this scene were measured and recorded.

4.3 Feature Point Detection

Two existing tools were used for feature point detection. One, Voodoo, is a camera tracking system, designed to find 3D camera and feature point locations from videos. The other, Bundler, is the tool developed in conjunction with Microsoft's PhotoTourism application.

4.3.1 Voodoo

The first tool, Voodoo, provides a number of options for feature point detection. It allows us to choose between different options for both the feature point detection algorithm and the feature point correspondence analysis (which affects which feature points are kept and which are eliminated as outliers). We sampled three different combinations:

- Harris feature point detection with KLT correspondence analysis
- SIFT feature point detection with KLT correspondence analysis
- SIFT feature point detection with SIFT correspondence analysis

Each combination provided us a set of customizable parameters. We used the default values provided by Voodoo for all parameters.

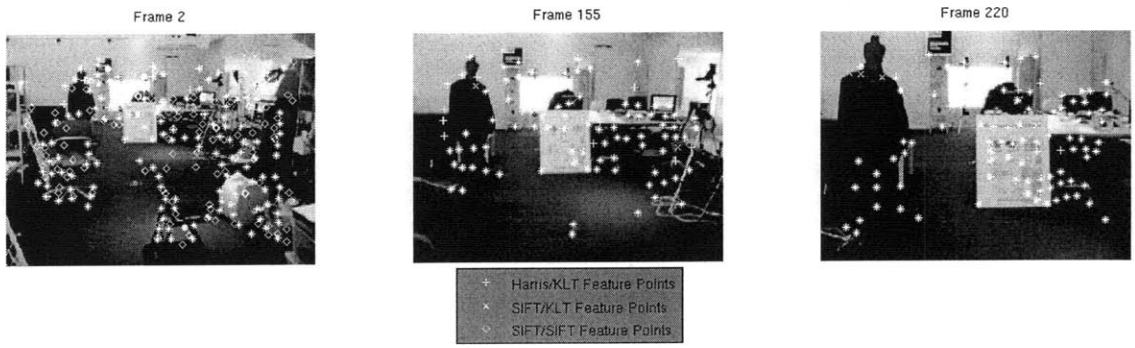

Figure 4-1: Feature points for three sample frames of lab space video using three detection schemes with Voodoo.

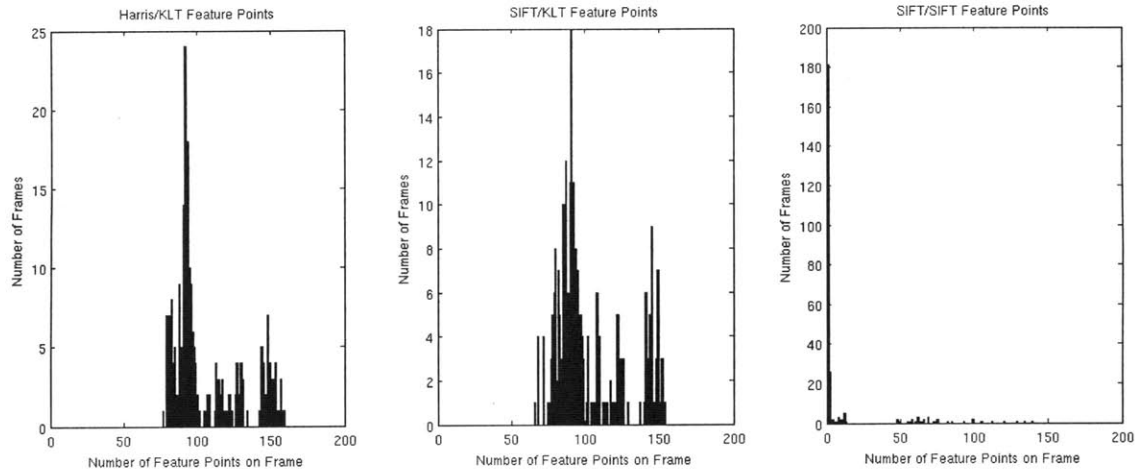

Figure 4-2: Histograms showing how many frames registered x feature points for each of the three methods.

Harris/KLT v SIFT/KLT v SIFT/SIFT

We compared the three combinations to determine whether any of them could provide us with a set of feature points for each frame which met our needs.

Figure 4-1 shows three frames of the lab space video. In each frame, the feature points detected by each of the three methods is marked. Notice that the feature points detected by Harris/KLT and by SIFT/KLT are almost identical in every frame. Also notice that for the last two frames, SIFT/SIFT failed to register any feature points.

The histograms in Figure 4-2 give us an idea of how many feature points were

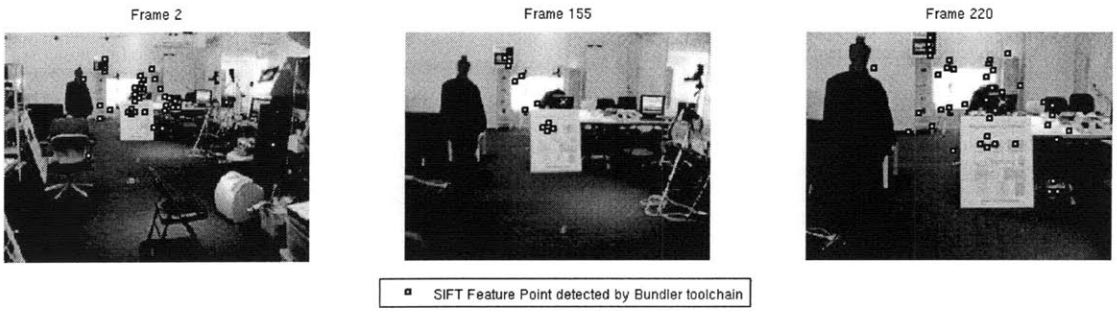

Figure 4-3: Feature points for three sample frames of lab space video using Bundler.

registered in each frame of video. Notice again that the results for Harris/KLT and SIFT/KLT are nearly identical. Also notice that for a large percentage of the frames in the video (180+/255) SIFT/SIFT registered no feature points.

This data suggests two general trends. First, it suggests that SIFT/SIFT returns no feature points for many frames in a video sequence. As a result, we will abandon the SIFT/SIFT combination. It also suggests that Harris/KLT and SIFT/KLT return nearly identical results. Therefore, we'll concentrate on Harris/KLT (a mostly arbitrary choice, loosely based on the observation that when one method returned more results, it was usually Harris/KLT).

4.3.2 Voodoo v Bundler

We now compare the results of the Voodoo tool with those of Bundler. Bundler also uses SIFT feature point detection. However, the details of its execution are unknown, and appear to return very different results than the Voodoo tool using SIFT/KLT or SIFT/SIFT.

Lab Space Video

Figures 4-3 and 4-4 show the feature points registered in three frames of the lab space video using Bundler and Voodoo Harris/KLT, respectively. The Bundler tool seems to produce points almost exclusively in the area the camera eventually closes in on; it has almost no feature points on the outer portions of the image. Notice, however,

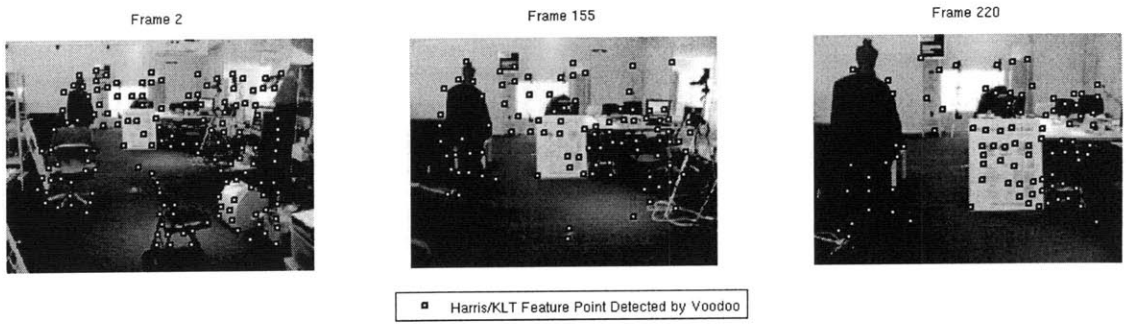

Figure 4-4: Feature points for three sample frames of lab space video using Voodoo.

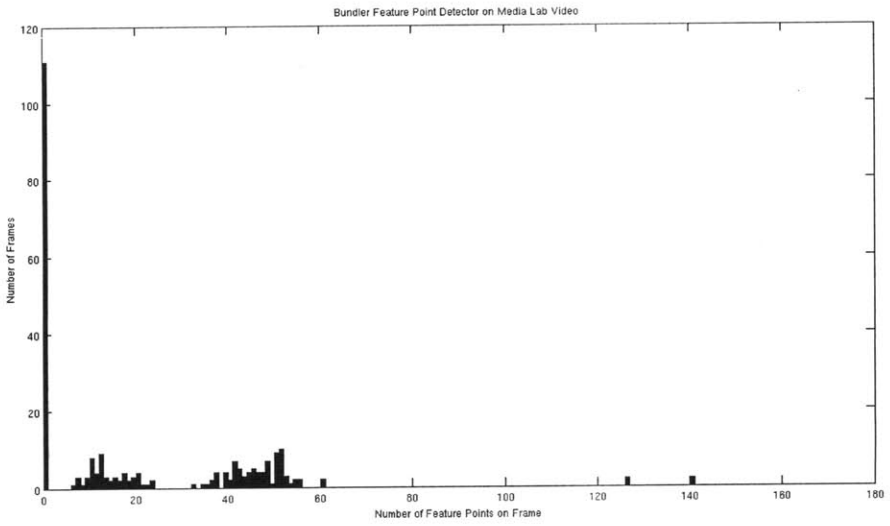

Figure 4-5: Histogram showing how many frames registered X feature points using Bundler.

that Voodoo also fails to produce many feature points near the outer extremes of the image. This will prove to be problematic later when attempt to warp the image. Recall that our technique uses feature points as grid points and warps each feature point to its new projected location in the view of the virtual camera. We would like if our feature points provided a good estimation of the scene geometry.

These figures also show that (at least for this video) Voodoo produces feature points which are relatively well scattered throughout the image (with the exception, as noted earlier, of the outer extremities). This will prove to be important later when we attempt to warp the image to produce novel views.

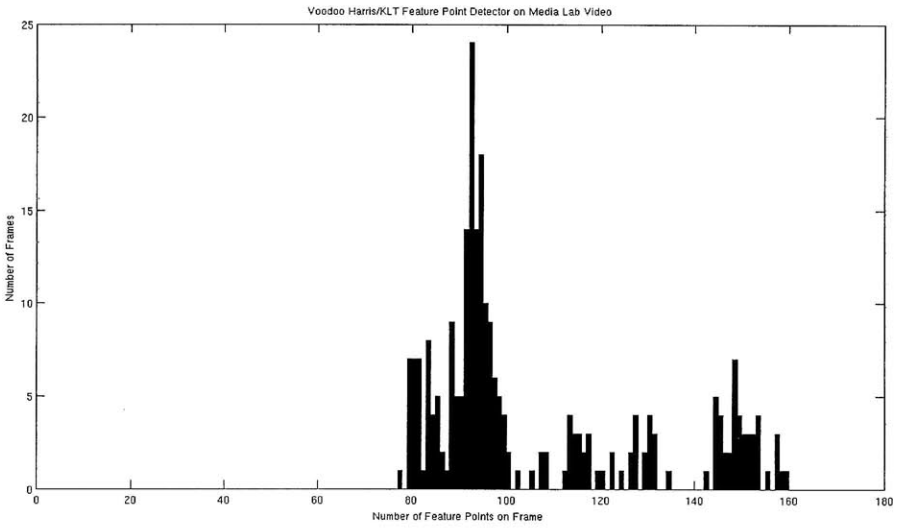

Figure 4-6: Histogram showing how many frames registered X feature points using Voodoo.

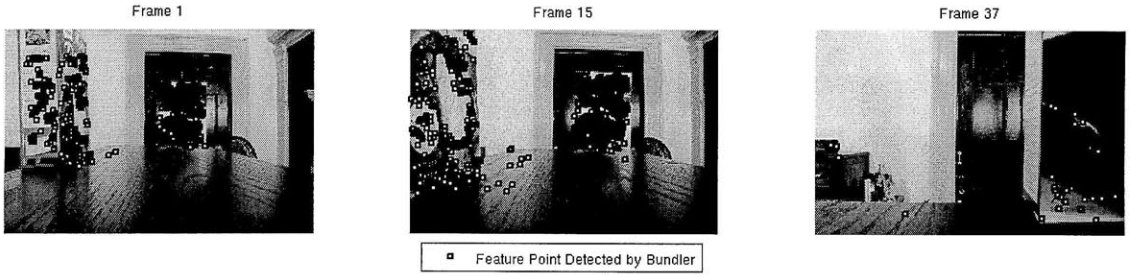

Figure 4-7: Feature points for three sample frames of cake box sequence using Bundler.

The histograms in figures 4-5 and 4-6 show the number of feature points registered in each frame using Bundler and Voodoo, respectively. As with the SIFT/SIFT variant of the Voodoo tool studied earlier, notice that Bundler has many frames (100+/255) for which it produces no feature points. Voodoo, on the other hand, consistently produces at least 80 feature points per frame.

Cake Boxes Sequence

Looking only at data from the lab space video might lead us to abandon Bundler as our feature point detector of choice, since Voodo consistently produced more feature points which more completely covered the image. The cake box sequence of images

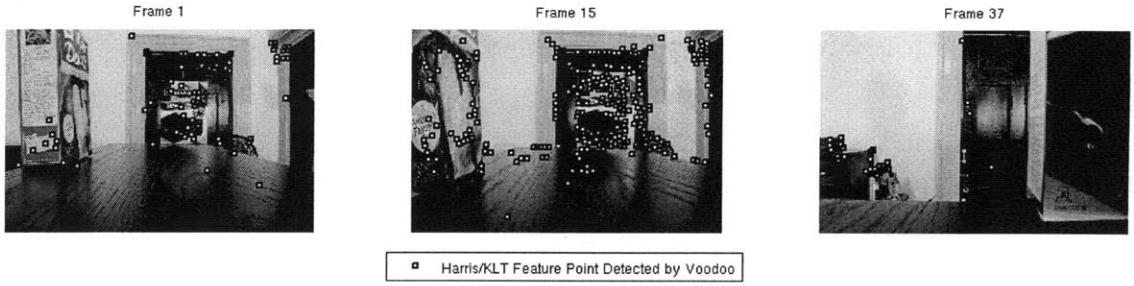

Figure 4-8: Feature points for three sample frames of cake box sequence using Voodoo.

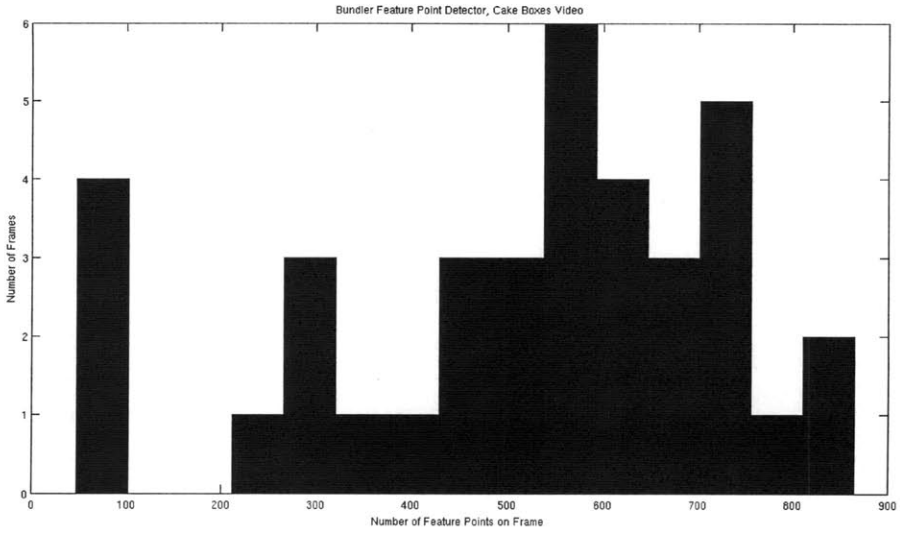

Figure 4-9: Histogram showing how many frames registered X feature points using Bundler.

demonstrates that the quality of feature points generated by either tool is much more nuanced than the lab space example suggests, and gives the first indication that our method, in its simplest form, will not be robust enough to handle general input. Notice in figures 4-7 and 4-8 that neither method reliably produces feature points that are well spread across the images. Bundler fails to register any feature points in the background of the scene, and Voodoo completely loses sight of the brownie box in the last frame. In a later section, we will see that this failure to produce feature points across the image will cause view synthesis to fail.

These histograms (figures 4-9 and 4-10) show us that both methods reliably produce a large number of feature points for all frames in this sequence of images. Since

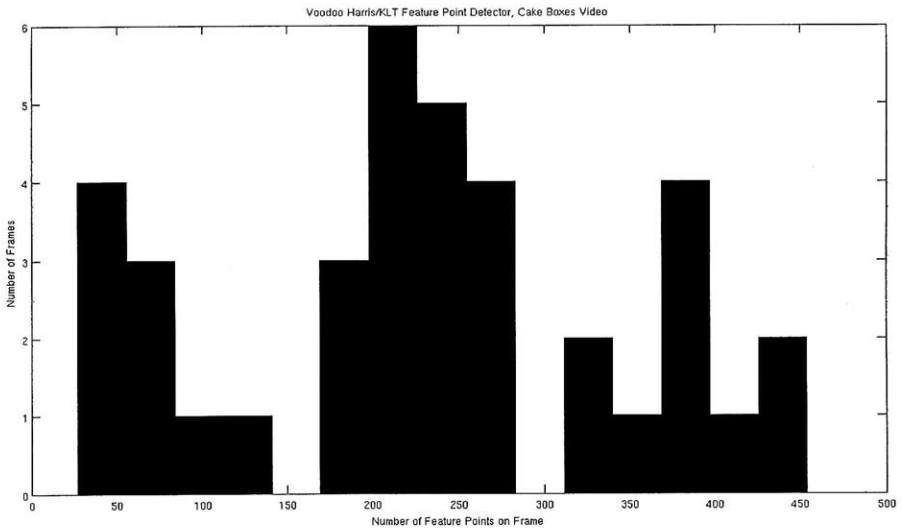

Figure 4-10: Histogram showing how many frames registered X feature points using Voodoo.

Bundler failed to produce features points for many of the lab space frames, it seems likely that Bundler does best with images of more reasonable resolution (the lab space video is 320x240, while the cake box sequence is 972x648).

4.4 Structure from Motion

After performing feature point detection, we perform structure from motion analysis to determine the 3D positon of each feature point and the position, rotation, and parameters of each camera. Again, we used the built in capabilities of the Voodoo and Bundler software packages to perform this step. In order to test its accuracy, the cake box sequence was measured for "ground truth" measurements. In this section, we compare the results of the structure from motion estimation produced by Voodoo and Bundler with the real-world measured results. Figure 4-11 shows the setup of the scene and its measurements.

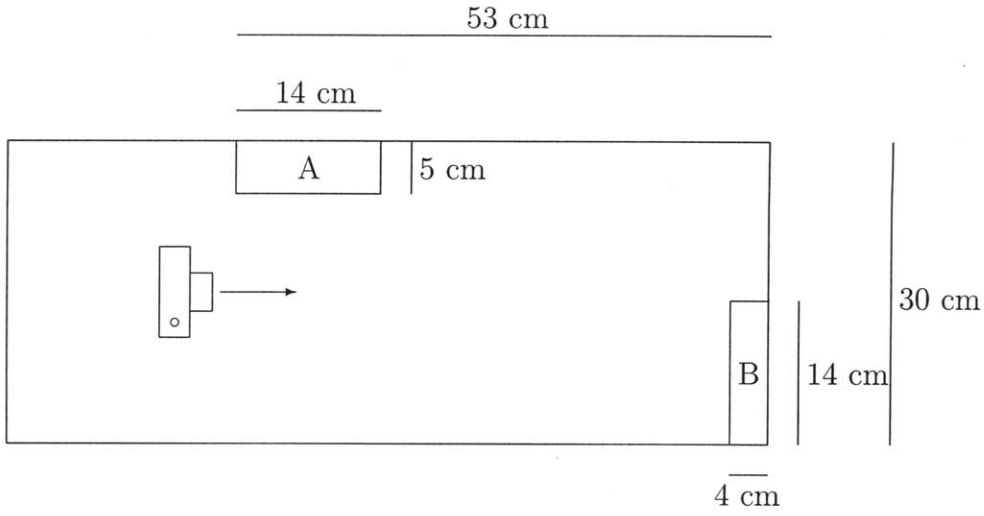

Figure 4-11: Cake box sequence setup. The box labeled "A" is a red and yellow cake box, and is 18 cm tall. The box labeled "B" is a brown and yellow brownie box, and is 19 cm tall.

Figure 4-12: Single frame from cake box sequence. Feature points registered by Voodoo are marked with white dots. Selected feature points used for measurement are marked with white squares with black edges.

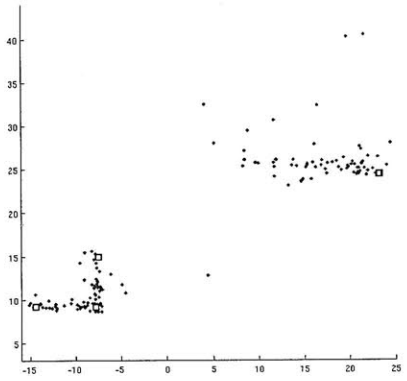

Figure 4-13: Estimated 3D coordinates of one frame's feature points using Voodoo structure from motion technique. Scene is viewed from overhead. Selected feature points are again marked with white squares with black edges. Feature points from scene background are cropped from view.

4.4.1 Voodoo

Figure 4-12 shows the feature points captured by Voodoo for one frame of the cake box sequence. Figure 4-13 shows the estimated 3D coordinates of the feature points, as viewed from the top of the table. This figure suggests that while Voodoo correctly estimated the approximate shape of the scene, the scales of the dimensions are distorted relative to one another. The long edge of the cake box is very short compared to its short edge. In other words, Voodoo seems to have compressed one axis relative to the others.

We would like to get a more mathematical sense of whether this is true. We will compare the ratios of the lengths of each edge to their actual values. This will allow for the scene to be arbitrarily translated, rotated, and uniformly scaled. However, it will register errors in which one axis was scaled relative to another.

Ratio	Actual	Estimated
$\frac{\lvert\overline{BC}\rvert}{\lvert\overline{AB}\rvert}$	2.8	.9
$\frac{\lvert\overline{DE}\rvert}{\lvert\overline{AB}\rvert}$	3.8	2.9

As Figure 4-12 suggested, the data in this table confirms that our structure from motion algorithm failed to correctly estimate the proportions of scene geometry. This was most likely a result of the type of camera movement used. Since the camera

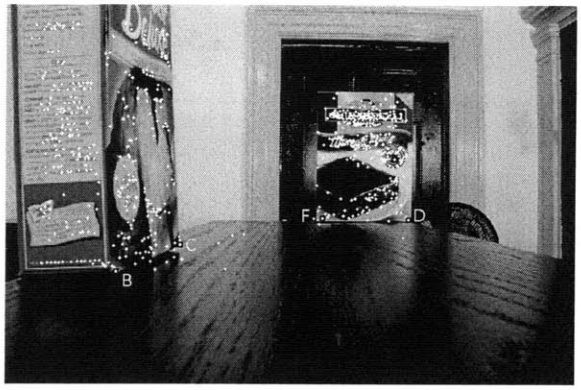

Figure 4-14: Single frame from cake box sequence. Feature points registered by Bundler are marked with white dots. Selected feature points used for measurement are marked with white squares with black edges.

moved straight into the scene (in a straight line parallel to the optical axis) the structure from motion algorithm was unable to distinguish between a longer/shorter focal length and compressed/extended scene along that axis.

As a result of this error, the depth of each feature point in our images will be correctly ordered, but not scaled correctly. This will allow us to accurately reconstruct a believable depth of field effect, but not a physically accurate one, since the depth is incorrect.

4.4.2 Bundler

Let's now explore how Bundler performed at the same task. Bundler registered different corners than Voodoo, as can be seen in Figure 4-14. Notice that we chose to use F as a reference point even though it does not fall exactly on the corner of the box as we would like.

The overhead view tells a similar story to Voodoo - while the overall structure of the scene is correct, the axes are scaled disproportionately. Comparing the lengths of the two line segments in the scene confirms this.

Ratio	Actual	Estimated
$\frac{\overline{\lvert DF \rvert}}{\overline{\lvert BC \rvert}}$	1	.28

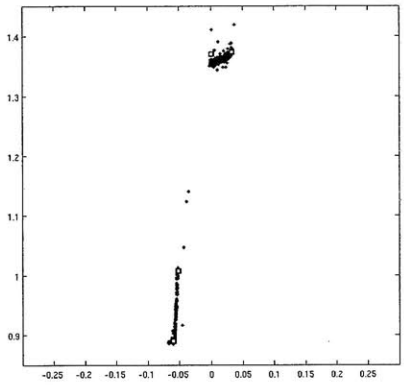

Figure 4-15: Estimated 3D coordinates of one frame's feature points using Bundler structure from motion technique. Scene is viewed from overhead. Selected feature points are again marked with white squares with black edges. Feature points from scene background are cropped from view.

4.5 Feature Point Projection

After extracting feature points, estimating their 3D coordinates, estimating the camera parameters, and generating virtual camera positions, the next step in the process is to use the virtual cameras to reproject the feature points to their new locations in the image. In this section, we show the results of this step and qualitatively compare the results to our expectations.

Figures 4-16 and 4-17 compare the actual locations of the detected feature points and their "projected" positions after estimating their 3D coordinates, estimating the camera coordinates, and reprojecting the points onto the image using the camera. The projection seems quite accurate - all projected points are within a few pixels of their original locations, and almost all are within one or two pixels.

Figures 4-16 and 4-17 compared the actual feature point locations and their predicted locations using the parameters of the actual camera. The more interesting challenge, however, is virtual camera views. Next, we will calculate camera positions slightly to the left and right of the actual camera and slightly above and below the actual camera. We will then project the 3D coordinates of the feature points using these cameras.

To move the camera up or left, we started at the (estimated) actual camera posi-

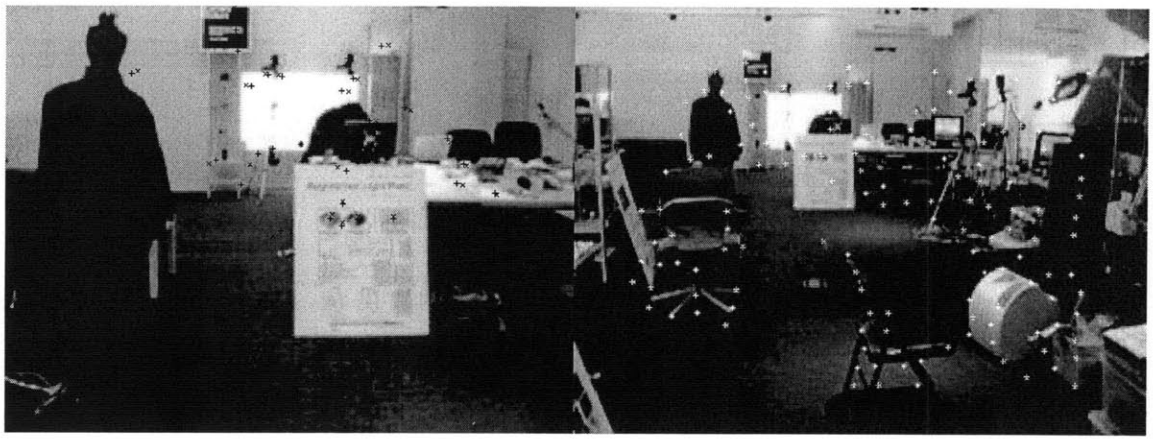

Figure 4-16: One frame from media lab video marked with actual feature point locations and locations projected from 3D coordinates of each feature point. The original feature point is marked with a '+', the projected 3D coordinates are marked with 'x'. Data in left image captured using Bundler, data in right image captured using Voodoo. Different frames were used since Bundler captured very few feature points in many frames.

Figure 4-17: One image from cake box sequence marked with actual feature point locations and locations projected from 3D coordinates of each feature point.

Figure 4-18: One frame from media lab video marked with actual feature point locations and locations projected from cameras slightly above, below, and to the left and right of the actual camera position. Original points are marked in white, the camera to the left is marked with black, right is marked with blue, up is marked with green, down is marked with yellow. Left image used data from Blender, right image used data from Voodoo.

tion and moved along the (estimated) direction of the camera's $\overrightarrow{y}$ or $\overrightarrow{x}$ axis, respectively. For the media lab video, shown in figure 4-18, we wanted to have the camera move a few inches in each direction. We accomplished this by measuring the distance between two points whose actual distance was approximately known, and scaling the 'up' and 'right' vectors appropriately. The same technique was used for the cake box sequence, shown in figure 4-19. In this scene, the camera was moved only a couple (estimated) centimeters. It should be noted, however, that this technique is probably error prone since, as we determined in the last section, our estimated feature point locations are scaled disproportionately along different axes.

In both scenes, we see the qualitative features we expect - the camera to the right of the original camera registers the same feature point to the left of the original feature point (since the image plane moves right with respect to the world, the object moves left with respect to the image plane). In addition, we see that feature points further back in the image are closer to their original location, and feature points closer to the camera move a great deal when the camera moves. This parallax effect is important - the fact that objects at different depths move different amounts when the camera

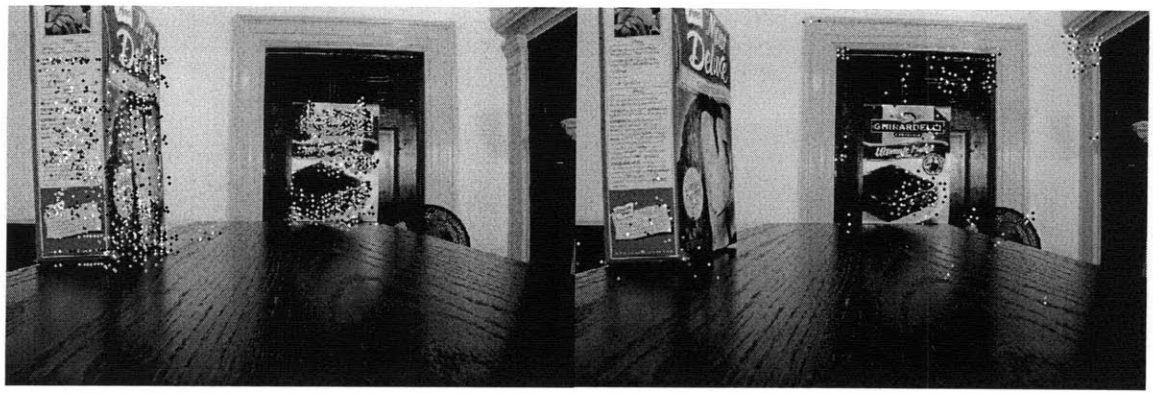

Figure 4-19: One image from cake box sequence marked with actual feature point locations and locations projected from cameras slightly above, below, and to the left and right of the actual camera position.

moves will eventually cause depth-dependent blur (depth of field) when we average all images together.

4.6 Image Warping

Next, we triangulate the image using the original feature point locations, then warp the image, dragging each feature point to its new calculated location for a given virtual camera.

Note that all images in this section are generated using data from Voodoo.

Figure 4-20 shows the triangulation of the first frame of the media lab scene. Because of the lack of feature points around the edges, which was discussed earlier, we see that the interior of the image is well tesselated, but the borders are poorly triangulated (remember that in order to deal with the lack of feature points on the sides of the image, we always mapped the image corners to themselves).

With this triangulation, we can project every feature point to its new location for the virtual camera, and warp the image to generate the image seen by this new virtual camera. Figure 4-21 shows the result of this operation, which is the final step in generating novel views of the scene. In this figure, we see images from virtual cameras about one foot and about two feet to the left and to the right of the original

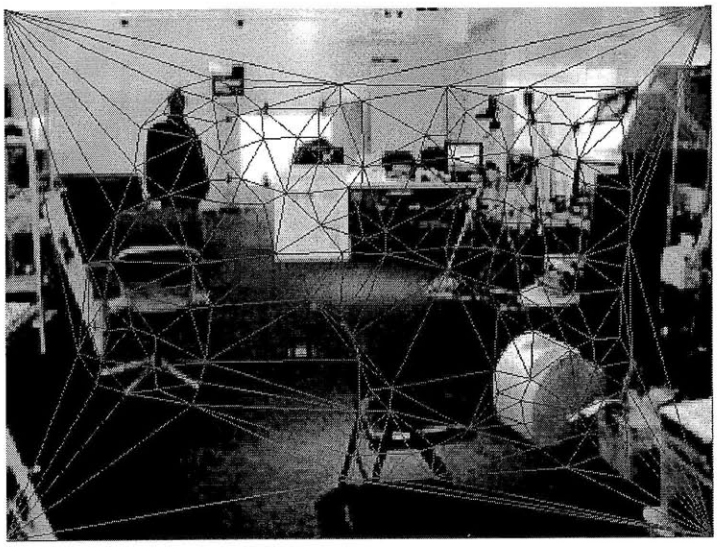

Figure 4-20: Delaunay triangulation of one frame of media lab scene.

camera (remember, distance are estimated).

It's difficult to capture the movement in these images in still image form, and in a small printable area. The results are relatively good - the center of the image is quite believable, and, as Figure 4-21 demonstrates, we don't see noticable distortions until the camera has been moved on the order of two feet from its original location. However, the edges of the image are a big problem - the fact that the corners stay fixed (and do not move as they should when the camera moves) is quite noticable.

Figure 4-22 shows the delaunay triangulation of the cake box sequence. This triangulation is much more problematic. In this triangulation, we see large parts of key scene objects, such as the cake boxes, which do not have feature points. Triangles span across the edges of both cake boxes.

Figure 4-23 shows the results of these inaccurate triangulations. The boxes do not move at all convincingly - they are distorted and warped.

4.7 Image Matching and Averaging

Now that we have generated novel views of the scene, we are ready to generate a depth of field image for a single frame. This involves simply generating novel views from

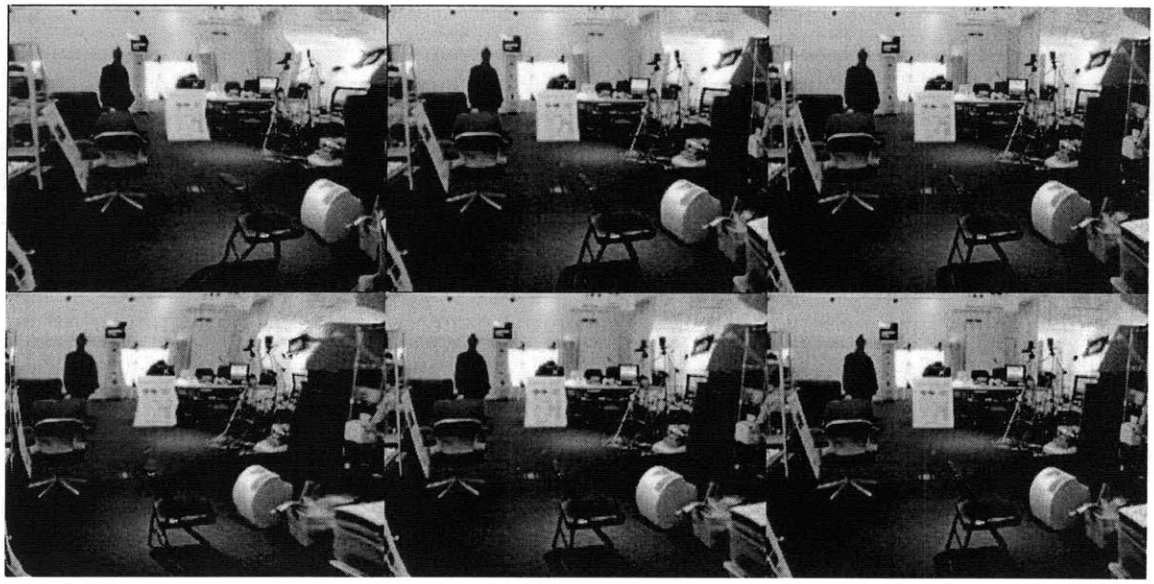

Figure 4-21: Novel views of first frame of media lab video. Image on top left is an image from a camera far to the left and bottom left is from a camera far to the right. Top middle image is from a camera slightly to the left, and bottom middle is slightly to the right. The rightmost column shows the original image.

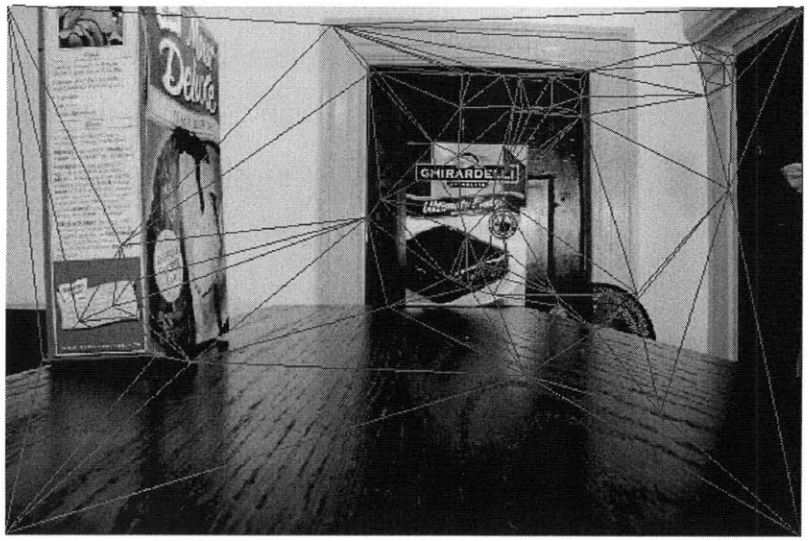

Figure 4-22: Delaunay triangulation of one frame of cake box sequence.

Figure 4-23: Novel views of first frame of media lab video. Middle image is original image, images further left are from a camera further to the left, and images further right are from a camera further to the right.

virtual cameras scattered about the virtual aperture area. To decide the aperture radius for the images in this section, we simply tried different values, increasing or lowering the radius until achieving the blur we desired. We chose this method (rather than trying to pick an actual reasonable aperture value based on a real camera and the scale of the scene) because of the disproportionate distances mentioned earlier in feature point data.

Figure 4-24 shows the generated depth of field images for the first frame of the media lab video. The left column is focused on the gray chair, while the right column is focused on the white poster near the back of the image. Each row has a slightly larger image aperture than the image above it. All images were generated using 20-40 virtual cameras in the aperture, and all took less than a minute to generate. Note that these images are mostly convincing, except for the edges, where our lack of feature points causes problems. Notice also that there's a slight "ringing" noticable in the most blurred images. This ringing could be reduced by simply increasing the number of virtual cameras.

We had hoped to offer a more objective analysis of the accuracy of our results using the cake box sequence. Since we know the exact dimensions of this scene and captured actual images with a large aperture, we had hoped to compare the "generated" depth of field image with the actual, captured image. However, neither of our structure from motion packages provided feature points dense enough for use in this scene. In addition, the inconsistent proportions along different axes would have prevented us

from accurately estimating what aperture size we should use in the coordinate system of the generated feature points.

4.8 Final Video

To produce a final video, we simply performed the above step for each frame. The results are mostly as might be expected from the images shown in the last section - while the frames are not perfect, they provide a convincing estimation. There was one new issue which became apparent in the video - the problem of "jitter". As feature points leave or enter the scene, the triangulation shifts, causing the entire video to act "twitchy". We will discuss possible means of alleviating this issue in the next section.

Overall, our technique seems to be a solid foundation, but still has much work to be done before it can be used by consumers. On some videos, such as the media lab example used in these results, the technique works relatively well, producing results that could use improvement, but are reasonable. On others, such as the cake example, it produces completely useless results. The technique must be modified to be more robust before it can be used by average consumers.

Figure 4-24: Generated DoF images from the first frame of the media lab video.

Chapter 5

Further Work

Our results in the last section demonstrated that while our process sometimes works well, it needs improvement before it can be widely applied. In this chapter, we propose a number of ways that the technique might be improved.

5.1 Using Features Points from Other Frames

One of the core issues with the existing technique is its reliance on having a large number of feature points which are well scattered across the image. Almost all of our problems came from having large areas of the image with no feature points. We could alleviate this by using feature points detected in other frames. For example, consider the video sequence shown in Figure 5-1. The video is a short clip in which the camera walks past three black blocks. This figure shows two frames from the sequence - one frame from early in the sequence and another from very late in the sequence - as well as the feature points registered in each.

Notice that in the first frame, some key feature points are missing, which will probably cause problems when we attempt to perform view synthesis. For example, there are no feature points registered on the top of the last block. As a result, this block will probably be flattened into the floor, and will not be at the appropriate depth in the reconstruction. However, these feature points are registered in the later frame of the video.

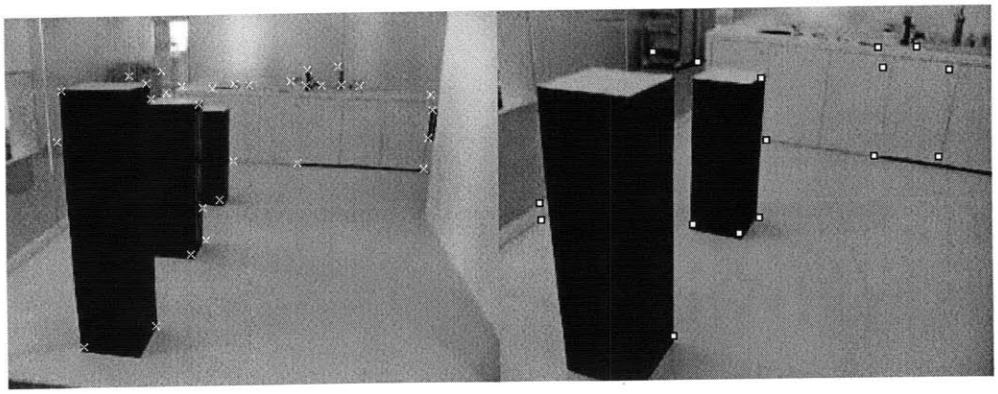

Figure 5-1: Two frames from a short video sequence. Each frame shows the feature points registered in that image.

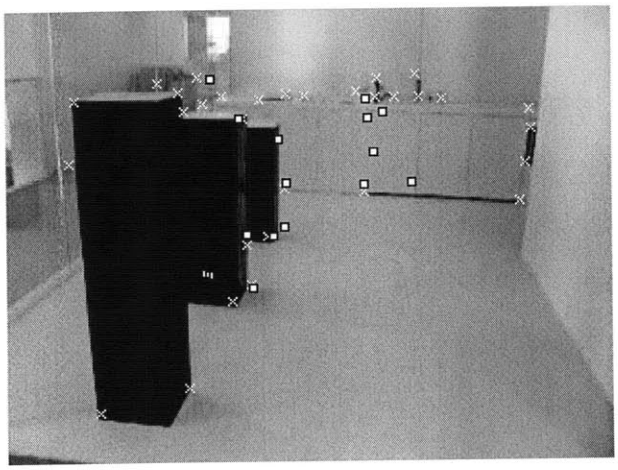

Figure 5-2: First frame of three block sequence with both its own feature points and feature points from later frame.

Figure 5-2 shows the results of projecting the 3D coordinates of the feature points from the last frame into the first frame. The original frame's feature points are marked with crosses, while the feature points from the later frame are marked with white squares. The later frame provides critical depth info about the top of the block, which could significantly improve view synthesis for the first frame.

However, this combination of feature points also poses some problems. These problems are made evident in the above image. First, errors in 3D feature point reconstruction are exaggerated by the large camera change. Notice that while the feature point for the top of the block is near its actual location, it does have significant error.

More troubling is the fact that feature points from other frames might not have actually been visible in the current frame. For example, there are two points near the bottom of the black box which are actually not visible in the first frame; they are occluded by this block. This will likely lead to strange view synthesis errors near these points, since they will have a depth inconsistent with the rest of the block.

In order to effectively use feature points from other frames, we must estimate when to use feature points and when not to in order to avoid situations in which occluded points are projected onto a frame in which they are not visible.

5.2 Better Triangulation Method

Delaunay triangulation, the triangulation method we used to test our technique, is one of the most widely used triangulations. It picks a triangulation which maximizes the minimum angle in any triangle - in other words, it tries to make the skinniest triangle in the triangulation as thick as possible.

However, this notion of triangulation does not map well to our problem. We essentially use triangulation to estimate the 3D geometry of the scene, breaking the image into object polygons which are then slid around the image based on the feature points at their edges. In the ideal case, we would like each distinct object in our scene to be triangulated completely separate from all other objects. This does not

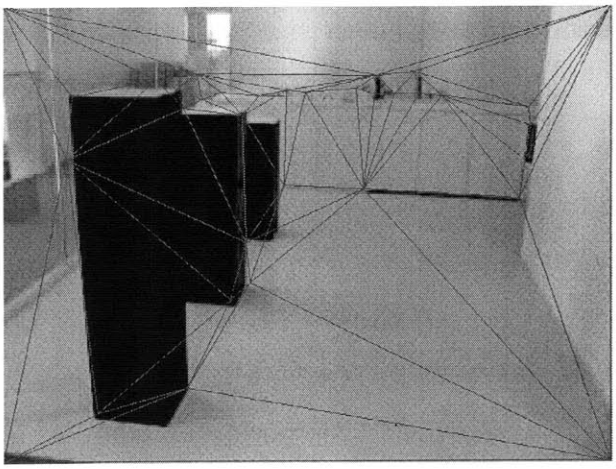

Figure 5-3: Delaunay triangulation of first frame of three blocks scene.

necessarily map to the goal of minimizing skinny triangles, as Delaunay triangulation does.

Figure 5-3 shows the Delaunay triangulation of the three blocks scene. Notice that the blocks are not triangulated at all how we would like them - triangles stretch across the first two blocks, and across the first block into points in the background. However, it appears as though a "good" triangulation probably exists - there are feature points on each of the four corners of the front block, which would ideally become a quad completely separate from the rest of the scene.

Other authors using a technique similar to ours have noted the same problems. In "Uncalibrated Synthetic Aperture for Defocus Control", Kusumoto, Hiura and Sato note that "Delaunay triangulation is problematic because the edges of an in-focus object are often blurred by the triangles which step over the occluding contour." [8]

We would like to use a triangulation technique which closely approximates the actual geometry of the scene. For example, we might want to use a method which takes as input not only the points to be triangulated, but also information about intensity edges in the scene. The triangulation method could then triangulate the points while trying to respect strong edges from the image, hopefully resulting in a triangulation which approximates the scene's structure.

5.3 Content Preserving Warps

Recall that "Content Preserving Warps for 3D Video Stabilization" [10] used a technique similar to ours in which feature points were reprojected to generate a novel view of a frame from a new camera position. Unlike our technique, however, they did not rigidly move each feature point to its new, projected location. Instead, they placed a regular mesh over the top of the image, and used these points as their control points when warping.

Since each point in the control mesh no longer maps directly to a feature point, the control points do not move rigidly with the feature points. Instead, each control point is guided by all feature points in its grid cell. In each grid cell, this technique computes the squared error – the difference between where a point should project to in the new camera view and where it actually ends up after the control points are moved. By minimizing this error, they respect the constraints imposed by the projected feature points, though not as rigidly as our scheme does. This allows them more flexibility – for example, they introduce a "shape preservation" term to the error, in which the shapes of objects must be preserved.

It also allows them to weight each feature point, allowing a feature point to smoothly fade in and out of the scene rather than quickly jumping in and disappearing. This would help our technique by eliminating the jitter we saw when we compiled the results of each frame into a video.

This modification would also help our technique by reducing the detrimental effects of too few feature points. For example, when an object registers only a few feature points, or the feature points are not on the edges of the object, the shape preservation term would encourage the object to move rigidly, while the feature points that DID register would encourage the object to move to its correct location. It would especially help with the lack of feature points around image edges. Currently, we arbitrarily choose to map the corner of the image to itself, which produced very poor results. Using the content preserving warp modification, however, the control points on the edges of the image would have no feature points to guide them, and would

instead be encouraged to move in the same manner as their neighbors by the shape preservation term.

There is, however, one detrimental effect of using content preserving warps in our technique. Content preserving warps were originally inspired by the realization that physical accuracy does not matter as long as the image looks good to a viewer. For our purposes, however, physical accuracy matters more than in the original paper – we rely on objects at different depths moving different distances to achieve accurate blur. Even with this downside, however, we feel that modifying our technique to use content preserving warps could achieve excellent results.

5.4 Generalized View Synthesis Technique

It may also be helpful to explore more general view synthesis techniques proposed by other authors such as the technique proposed in "Unstructured Lumigraph Rendering". [2] This technique finds the value for each pixel by weighting the colors from several projected images based on their fitness for the given view. We avoided this technique initially because we wanted each frame to be based on only a single input frame so that our technique could better handle scenes with moving objects. However, perhaps a "time" term could be added to the fitness function to allow but discourage pixels from other frames.

5.5 Recinematography

Besides improving the performance or robustness of our technique, we could also improve its usefulness. Many of the papers we reviewed while developing our technique proposed techniques for view stabilization. We could make our technique more useful by generalizing it. Currently, the user is constrained to simulating an aperture with the actual camera location at its center. However, we currently don't use the actual camera position in any special way - we randomly generate virtual camera positions scattered throughout the aperture, and its unlikely that any will land exactly on the

actual camera position. With this in mind, we could allow the user to specify any camera position for the center of the aperture. For example, we could provide view stabilization, generating apertures centered at positions along a smoothed camera path instead of the actual, shaky camera path. In fact, a combined technique could have an advantage over techniques which perform only view stabilization, since much of the scene will have a slight blur applied, hiding reconstruction errors.

5.6 Bokeh Effects

When an image is captured using an aperture large enough to demonstrate depth of field effects, the shape of the aperture affects the kind of blur achieved. A point light source will be blurred to the exact shape of the aperture. Different effects can be achieved by varying the shape of the aperture. This effect is called the bokeh of the image. In "Modeling and Synthesis of Aperture Effects in Cameras", Lanman, Raskar and Taubin develop the "Bokeh Brush" – a method to control the shape of defocused points by capturing the scene with a set of particularly shaped apertures [16] and applying post-processing to achieve any desired aperture shape.

Since we are already simulating aperture effects in our technique, it would be trivial to give the user control over the shape of the aperture, simply by generating virtual cameras over different areas. Users could control not only the shape, but also the weight given to the image generated at each point in the aperture. Currently, the images are all weighted equally, but interesting effects could be achieved by allowing control over this. For example, by specifying that the weights should fall off towards the edge of the aperture, a softer blur could be achieved. The authors of "Uncalibrated Synthetic Aperture for Defocus Control" also explore this effect in the context of defocusing still images [8].

Chapter 6

Conclusion

This thesis proposed a method for generating depth of field effects in video via post processing. The technique requires no changes to the capture process itself, so could be used on any digital video. The crucial step in this process was view synthesis - the generation of an image taken from a virtual pinhole camera, based only on information captured by the actual camera. Once the problem of view synthesis was solved, we simply needed to average the images captured by pinhole cameras scattered over the aperture in order to simulate a lens with a finite aperture size.

Our technique for view synthesis used information from other frames only for geometry estimation - the actual color values of each pixel were drawn only from a single frame of video. This allowed us to ignore the effects of moving objects in the scene, and prevented common pitfalls like ghosting. In our technique, we overlaid a mesh onto the image based on a triangulation of the feature points. This mesh was then warped by dragging each feature point to its new image location, based on the projection of the feature points' 3D coordinates into the new camera system.

A number of trial videos demonstrated that while our technique lays a solid foundation for future work, it is not currently robust enough for widespread use. The primary problem was the distribution of feature points across the image - when feature points were not detected in critical locations, our technique made large errors while synthesizing new views of the scene.

We proposed a number of possible extensions or modifications to our technique

which we believe could lead to a much more robust system. Most of these modifications attacked the problem of too few feature points, but other improvements were explored as well.

Artificial depth of field could greatly improve the quality of consumer video and could allow much better control over depth of field effects at both the consumer and professional level. We hope that the technique we proposed and the modifications we envisioned will help others achieve this effect with robust and convincing results.

Bibliography

[1] Soonmin Bae and Fredo Durand. Defocus magnification. In *Proceedings of Eurographics 2007*, 2007.

[2] Chris Buehler, Michael Bosse, Leonard McMillan, Steven Gortler, and Michael Cohen. Unstructured lumigraph rendering. pages 425–432. ACM, 2001.

[3] Shenchang Eric Chen and Lance Williams. View interpolation for image synthesis. In *SIGGRAPH '93: Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 279–288. ACM, 1993.

[4] Laboratorium fur Informationstechnologie. Voodoo camera tracking system. `http://www.digilab.uni-hannover.de/docs/manual.html`.

[5] Todor Georgiev, Ke Colin Zheng, Brian Curless, David Salesin, Shree Nayar, and Chintan Intwala. Spatio-angular resolution tradeoff in integral photography. In *Proceedings of Eurographics Symposium on Rendering*, 2006.

[6] Michael L. Gleicher and Feng Liu. Re-cinematography: Improving the camerawork of casual video. *ACM Transactions on Multimedia Computing, Communications, and Applications*, 5(1):1–28, 2008.

[7] Point Grey. Profusion 25 product information page. `http://www.ptgrey.com/products/profusion25/`.

[8] Natsumi Kusumoto, Shinsaku Hiura, and Kosuke Sato. Uncalibrated synthetic aperture photography for defocus control. *Information and Media Technologies*, 4(4):913–921, 2009.

[9] Marc Levoy and Pat Hanrahan. Light field rendering. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 31–42, 1996.

[10] Feng Liu, Michael Gleicher, Hailin Jin, and Aseem Agarwala. Content-preserving warps for 3d video stabilization. *ACM Transactions on Graphics*, 28(3), 2009.

[11] David G. Lowe. Object recognition from local scale-invariant features. In *ICCV '99: Proceedings of the International Conference on Computer Vision-Volume 2.* IEEE Computer Society, 1999.

[12] Krystian Mikolajczyk and Cordelia Schmid. An affine invariant interest point detector. In *Proceedings of the Seventh European Conference on Computer Vision*, pages 128–142, 2002.

[13] Ankit Mohan, Douglas Lanman, Shinsaku Hiura, and Ramesh Raskar. Image destabilization: Programmable defocus using lens and sensor motion. In *IEEE International Conference on Computational Photography (ICCP)*, 2009.

[14] Francesc Moreno-Noguer, Peter N. Belhumeur, and Shree K. Nayar. Active refocusing of images and videos. In *SIGGRAPH '07: ACM SIGGRAPH 2007 papers*, 2007.

[15] Ren Ng, Marc Levoy, Mathieu Bredif, Gene Duval, Mark Horowitz, and Pat Hanrahan. Light field photography with a hand-held plenoptic camera. Technical report, Stanford University, April 2005.

[16] Ramesh Raskar, Douglas Lanman, and Gabriel Taubin. Modeling and synthesis of aperture effects in cameras. *Computational Aesthetics in Graphics, Visualization and Imaging*, 2008.

[17] Steven M. Seitz and Charles R. Dyer. Physically-valid view synthesis by image interpolation. In *Proc. IEEE Workshop on Representations of Visual Scenes*, pages 18–25, 1995.

[18] Brandon Smith, Li Zhang, Hailin Jin, and Aseem Agarwala. Light field video stabilization. *IEEE International Conference on Computer Vision*, 2009.

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