2011

# Single View Modeling and View Synthesis

Miao Liao
*University of Kentucky*, miao.liao@gmail.com

ABSTRACT OF DISSERTATION

Miao Liao

The Graduate School
University of Kentucky
2011

Single View Modeling and View Synthesis

ABSTRACT OF DISSERTATION

A dissertation submitted in partial
fulfillment of the requirements for
the degree of Doctor of Philosophy in
the Department of Computer Science
at the University of Kentucky

By
Miao Liao
Lexington, Kentucky

Director: Dr. Ruigang Yang, Associate Professor of Computer Science
Lexington, Kentucky 2011

ABSTRACT OF DISSERTATION

Single View Modeling and View Synthesis

This thesis develops new algorithms to produce 3D content from a single camera. Today, amateurs can use hand-held camcorders to capture and display the 3D world in 2D, using mature technologies. However, there is always a strong desire to record and re-explore the 3D world in 3D. To achieve this goal, current approaches usually make use of a camera array, which suffers from tedious setup and calibration processes, as well as lack of portability, limiting its application to lab experiments.

In this thesis, I try to produce the 3D contents using a single camera, making it as simple as shooting pictures. It requires a new front end capturing device rather than a regular camcorder, as well as more sophisticated algorithms. First, in order to capture the highly detailed object surfaces, I designed and developed a depth camera based on a novel technique called light fall-off stereo (LFS). The LFS depth camera outputs color+depth image sequences and achieves 30 fps, which is necessary for capturing dynamic scenes. Based on the output color+depth images, I developed a new approach that builds 3D models of dynamic and deformable objects. While the camera can only capture part of a whole object at any instance, partial surfaces are assembled together to form a complete 3D model by a novel warping algorithm.

Inspired by the success of single view 3D modeling, I extended my exploration into 2D-3D video conversion that does not utilize a depth camera. I developed a semi-automatic system that converts monocular videos into stereoscopic videos, via view synthesis. It combines motion analysis with user interaction, aiming to transfer as much depth inferring work from the user to the computer. I developed two new methods that analyze the optical flow in order to provide additional qualitative depth constraints. The automatically extracted depth information is presented in the user interface to assist with user labeling work.

In this thesis, I developed new algorithms to produce 3D contents from a single camera. Depending on the input data, my algorithm can build high fidelity 3D models for dynamic and deformable objects if depth maps are provided. Otherwise, it can turn the video clips into stereoscopic video.

KEYWORDS: 3D Reconstruction, Single View, Depth Camera, Depth Recovery, 2D-3D Video Conversion

Author's signature:_____

Date:_____

Single View Modeling and View Synthesis

By
Miao Liao

Director of Dissertation:_____

Director of Graduate Studies:_____

Date:_____

RULES FOR THE USE OF DISSERTATIONS

Unpublished dissertations submitted for the Doctor's degree and deposited in the University of Kentucky Library are as a rule open for inspection, but are to be used only with due regard to the rights of the authors. Bibliographical references may be noted, but quotations or summaries of parts may be published only with the permission of the author, and with the usual scholarly acknowledgments.

Extensive copying or publication of the dissertation in whole or in part also requires the consent of the Dean of the Graduate School of the University of Kentucky.

A library that borrows this dissertation for use by its patrons is expected to secure the signature of each user.

| Name | Date |
| --- | --- |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |

DISSERTATION

Miao Liao

The Graduate School
University of Kentucky
2011

Single View Modeling and View Synthesis

DISSERTATION

A dissertation submitted in partial
fulfillment of the requirements for
the degree of Doctor of Philosophy in
the Department of Computer Science
at the University of Kentucky

By
Miao Liao
Lexington, Kentucky

Director: Dr. Ruigang Yang, Associate Professor of Computer Science
Lexington, Kentucky 2011

# ACKNOWLEDGMENTS

I would like to thank my advisor Dr. Ruigang Yang. He has guided me into this field, inspired me to pursue my PhD on this topic and helped me on this thesis in too many ways to enumerate. I could not have reached this point without him.

I also want to give many thanks to my committee members, Dr. Fuhua Zheng, Dr. Melody Carswell and Dr. Brent Seales, for the time they spent on my thesis review. I appreciate all of the suggestions and comments. I also want to acknowledge my co-workers, Dr. Minglun Gong, Jiejie Zhu, Liang Wang, Qing Zhang and Jizhou Gao, who have contributed to my thesis work.

My deepest gratitude goes to the people who have had such a significant impact on my life. My parents encouraged me to enter a Ph.D. program in the United States. They have always given me unconditional support and love. My mother even hid her rectal cancer from me until the last minutes of her life, just so as not to distract me from my studies. She must be very happy knowing that I finally earned my degree.

Last but not least, I would like to express my infinite appreciation for my wife and life mate, Yingxue. She married me when I was a struggling student and accompanied me through this journey. I am so lucky to have married such an understanding, supportive and beautiful woman.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

# Chapter 1 Introduction

## 1.1 Motivation and Goals

This thesis develops new algorithms to produce 3D content from a single camera. Presently, imaging technology has been so well developed that taking a picture is as simple as point-and-shoot. Nevertheless, the recorded images are only 2-dimensional, due to the loss of depth information during the capturing process. The 2D information prevents the users from re-exploring the world in 3D on their display devices. Thus, there is always a strong desire to record and re-explore the 3D world in 3D. To achieve this goal, current approaches usually make use of a camera array.

However, multiple-camera setup has some obvious disadvantages compared to a single camera. First, the calibration and synchronization process for multiple cameras is tedious work and needs to be done every time the cameras are moved to a new location. Second, the lack of portability makes multiple cameras not practical to carry around for outdoor capturing activities. Last, purchasing and maintaining multiple cameras could cost much more money than a single camera.

In other words, a camera array is not the right tool for amateur users to explore the world in 3D, limiting its applications to lab experiments. This is my motivation to investigate the new methods that use a single camera to record and build the world in 3D. My ultimate goal is to enable the amateur users to capture and re-explore the world in 3D using a hand-held camera. Imagine a future in which we can bring a

camcorder to a baseball game and, from the output footage, we should be able to watch the game from any perspective we want, truly re-experience the ball game in interactive 3D (figure 1.1).

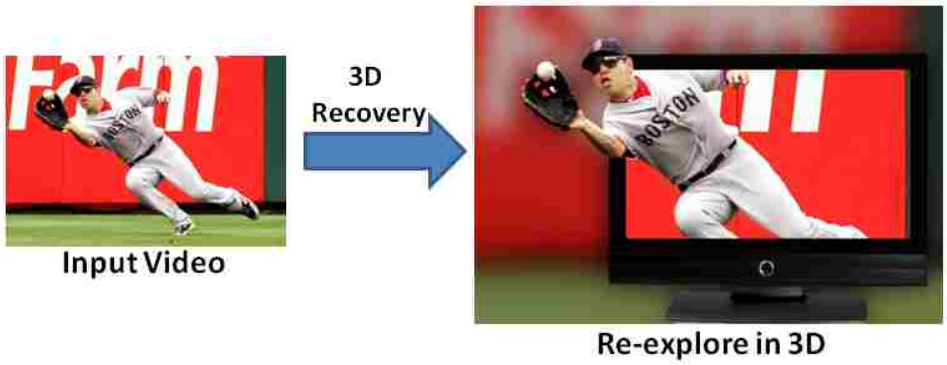


Figure 1.1: Baseball game re-exploration in 3D.

This is an extremely difficult problem to solve since it is inherently ill-posed. In this thesis, I am not trying to give a complete solution to the problem. Instead, I have made a few steps towards the ultimate goal, each of which has its own applications.

The process of image formation (perspective projection) leads to the loss of depth. If the camera can provide us with color images, as well as depth maps, it will be much improved. Therefore, in the first step, I designed and developed a depth camera based on a novel technique called light fall-off stereo (LFS). An LFS depth camera takes a number of images from a stationary camera as the illumination source moves away from the scene. Based on the inverse square law for light intensity, the ratio images are directly related to scene depth from the perspective of the light source. The LFS depth camera needs as few as two images, does not require calibrated cameras or light sources, or reference objects in the scene. The LFS depth camera outputs color+depth images sequences and achieves 30 fps, which is necessary for capturing

dynamic scenes.

In the second step, based on the output from the depth camera, I seek to produce a visually plausible model that deforms naturally, and which is consistent with the input data. The core idea is based on the assumption that the deformation is continuous and predictable in a short temporal interval. While the camera can only capture part of a whole surface at any time instance, partial surfaces reconstructed from different times are assembled to form a complete 3D surface for each time instance, even when the shape is under severe deformation. A mesh warping algorithm, based on linear mesh deformation, is used to align different partial surfaces.

Lastly, I remove the requirement of depth on the camera, which means I can take any video clips from regular camcorders as my input. The depth information needs to be inferred with the assistance of user interaction. I developed a semi-automatic system that converts conventional videos into stereoscopic videos by combining motion analysis with user interaction, aiming to transfer as much as possible work from the user to the computer. In addition to the widely-used structure from motion (SFM) techniques, I develop two new methods that analyze the optical flow, in order to provide additional qualitative depth constraints. With these algorithms, the user's labeling task is significantly simplified. A quadratic programming is further formulated to incorporate both quantitative depth and qualitative depth (such as those from user scribbling) to recover dense depth maps for all frames, from which stereoscopic views can be synthesized.

The proposed approach in each step has wide applications in the real world. For instance, the LFS depth camera provides one more option for the users in addition

to the existing depth cameras. The system of step two would enable modeling of dynamic objects in an outdoor environment. Imagine that you see an interesting street performer, and use a depth camera to make a surrounding shot. When you are back home and put the shot into computer, the 3D dynamic models will be built automatically by the proposed method. Then you can watch the performance from whichever viewpoints you want.

Given the recent success of 3D movies in Hollywood, there has been a renewed public interest in stereoscopic videos. Although new contents can be captured by stereoscopic cameras, or CG contents can be re-rendered in the stereo mode, converting the large collection of legacy monocular movies and videos to stereo remains labor intensive. The semi-automatic conversion system in the final step points to a smarter way to convert conventional videos into stereoscopic videos.

## 1.2 A Brief Historical Note

Before I introduce my dissertation work, let me first present a brief historical note on how I arrived at my interests.

In 2007, numerous real-time depth cameras were emerging on the commercial market, thanks to the advancement in both supporting hardware and software. Inspired by the popularity of the depth cameras, I developed my own real-time depth camera based on my previous work "Light Fall-off Stereo" (LFS). LFS takes a number of images from a stationary camera as the illumination source moves away from the scene. Based on the inverse square law for light intensity, the ratio images are directly related to scene depth from the perspective of the light source. Compared to

previous reconstruction methods for non-lamebrain scenes, LFS needs as few as two images, does not require calibrated camera or light sources, or reference objects in the scene. As the ratio of two images is enough to estimate the depth, the computational simplicity makes it possible to build a real-time depth sensor using the commodity hardware. The built depth sensor generates VGA (640 x 480) resolution depth maps as well as color images at 30Hz.

With existing off-the-shelf depth cameras, many applications became possible. Compared to the laser scanners, depth cameras capture a surface instead of a scan line at each instance, making it possible to capture those dynamic objects which cannot be kept static for a long period of time e.g., human faces. In 2008, by observing the advantages of the depth cameras, under the guidance of my advisor, I started working on a project called Self-Completion of 4D (Space+Time) Models, which is supported by NSF awards. The project aims to building a complete 4D model by capturing a dynamic scene over time from different locations. The recovered models can be used in many applications, such as simulations to create realistic virtual environments, to render special effects, and perhaps simply to allow everyone to enjoy their cherished moments, such as a baby's first step, in interactive 3D.

In my first attempt to solve this problem, I tried to extend the traditional methods of completing rigid objects to non-rigid objects. The trick is to segment a non-rigid object into piece-wise rigid parts, and apply the rigid methods to the individual parts. The problem with this method is that not all the non-rigid objects are articulated, so it is impossible to reasonably segment those objects into smaller rigid parts. Plus, the intermediate region between two separate rigid parts is not well defined, and it is

usually opt to artifacts. Therefore, I proposed a method that deals with deformable objects and avoids the assumptions on the object's motion. The result is the first part of my thesis work.

The success of the 4D reconstruction project inspires me to explore the methods that could achieve the same results with the output video clips of a regular camcorder. However, after reading the literature, I found that it is extremely hard, if not impossible, to build high fidelity 4D models from videos. The main reason is that we cannot obtain high quality depth information from monocular videos by existing state-of-the-art techniques. After realizing this, I dropped my attempt of building 4D models, and started to investigate the possibilities of a relatively easier task: turning a monocular video into a stereo video.

As cognitive studies by Koenderink and his colleges [1] show, the human visual system is more tuned in to depth order and precedence than to absolute depth. Therefore, accurate disparity values are not needed in order to create convincing 3D viewing experience. In the meantime, Guttmann [2] had developed a system that semi-automatically converts a monocular video into a stereo video. The system merely relies on users to provide the depth ordering information of the objects in the video and propagate the information to every pixel on every frame. In fact, although high quality depth maps cannot be obtained, some sparse absolute or relative depth information can be acquired by visual cues that are exhibited by the video. By observing this, I developed a more sophisticated system that will first automatically extract any 3D information before asking for user input; thus, the second part of my thesis work.

Since all three parts of this work fall into 3D recovery and reconstruction using a single camera (depth camera/regular camera), I have coined these collective processed as single view modeling and view synthesis.

## 1.3 Single View Modeling and View Synthesis

In this section, I will first introduce the basic ideas behind the LFS depth camera, and the efforts I made to create a real-time depth sensor. Then, I will briefly provide a high level overview of the other two parts of my thesis: modeling of dynamic and deformable objects, and stereo view synthesis from monocular video clips.

### 1.3.1 Light Fall-off Stereo Depth Camera

Recovering depth information from 2D images is one of the central problems in computer vision. Many different cues in the images have been used, such as stereoscopic disparity, shading, textures, focus and defocus. The vast majority of these methods makes a strong assumption that objects in the scene reflect light equally in all directions. Such a diffuse or Lambertian surface assumption is violated by almost all real-world objects, leading to incorrect depth estimates. Although techniques that go beyond Lambertian surfaces have been proposed, they typically require precise calibration of cameras and/or light sources, sufficient surface textures, or reference objects in the scene.

I exploit a different cue in the image formulation process, namely the *inverse square law* for light intensity. As illustrated in Figure 1.2 (left), the intensity of light observed from a source of constant intrinsic luminosity falls off as the square of the

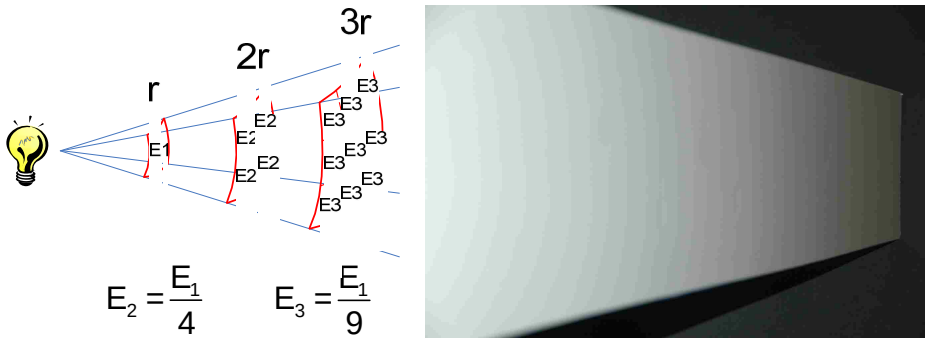


Figure 1.2: An illustration of the light fall-off effect. Left: light energy falls off as the square of the distance, e.g., the irradiance of a unit area at distance $2r$ is one fourth of that at distance $r$. Right: a real photograph showing the effect.

distance from the object. A real example is provided in Figure 1.2 (right), the scene is illuminated with a flash light, objects further away appear to be darker due to the fall-off of light intensity over distance. It can be derived that

$$I(p) = \frac{k_p}{r_p^2}, \text{ or } \quad r_p \propto 1/\sqrt{I(p)}, \tag{1.1}$$

where $I(p)$ is the pixel value of a scene point $p$, $r_p$ is the distance between the light source and $p$, and $k_p$ is a constant related to the intensity of the light source and the reflectance and orientation of point $p$. In fact, in this ideal setup in which the object is flat with a uniform color, we can almost assume that $k_p$ is identical for all surface points; therefore the pixel values directly provide a qualitative measure of depth.

However, real-world objects are usually not flat and typically have spatially varying textures. To deal with these practical issues, we can take an additional image under a different lighting configuration. The new pixel intensity then is

$$I'(p) = \frac{k_p}{r'^2_p} \tag{1.2}$$

If we compute the ratio of $I(p)$ and $I'(p)$, the impact of reflectance cancels out, making the ratio only related to the depth. Furthermore, if we measure the distance between the two light positions (i.e., $\Delta r = r'_p - r_p$), which is very easy to do, we can compute the scene depth to the light source.

The above formulation, which I called *light fall-off stereo* (LFS), holds for scenes with arbitrary bidirectional texture function (BTF), and therefore can be used to design a practical depth acquisition system with a single camera and a moving light source. As I will show later, LFS is easier to implement compared to existing depth acquisition methods for arbitrary BRDF/BTFs. It only needs as few as two images, making it possible to capture dynamic scenes.

I developed a *real-time* depth recovery system using Light Fall-off Stereo (LFS). Based on the formulation above, my system uses a single camera to capture a scene under two different lighting conditions: one illuminated by a near point light source and the other by a far one. Per-pixel depth is solved based on the pixel intensity ratio and the distance between the two lights, without the need for matching pixels.

My system can generate a VGA (640x480) resolution depth map at 30Hz. Quantitative accuracy evaluation shows that my system compares favorably to other commercial 3D range sensors, particularly in textured areas. In addition, my system is made of commodity off-the-shelf components, offering an inexpensive solution to real-time, high-resolution, video-rate range sensing.

### 1.3.2 Modeling Dynamic and Deformable Objects

Once a real-time depth camera is available in hand, I will use it to build 4D models of dynamic and deformable objects. Here I don't stick to a certain kind of depth sensor, as long as it outputs synchronized color and depth image sequences. Therefore, in this part of my work, how to generate the best depth maps is not my focus. In this part, I only focus on how to stitch and merge the existing piecewise surfaces.

Recent advances in camera self-calibration and stereoscopic vision have made it possible to create high-quality 3D models using a single hand-held camera (e.g., [3]) or even a community photo collections [4]. However, most of these techniques are limited to static objects. Typical treatment for dynamic scenes has been widely studied using an array of surrounding cameras, (e.g., [5,6]). Compared to a single camera, a camera array is cumbersome, less affordable, and not practical to carry around for outdoor capturing activities. Using a single depth camera or stereo camera pair to capture a dynamic scene is a challenging problem since it can capture only the visible part of a dynamic object at each time instance. Fortunately, the underlying dynamic nature of the scene can be used to help provide more samples over time. In the simplest case, if the object is rigid or articulated, this *model-completion* task becomes the well-studied Structure-from-Motion (e.g., [7,8]) problem using 3D point registration. Here, I would like to develop a similar technique for time-varying objects deforming arbitrarily but predictably, so that visible partial surfaces can be assembled together to complete a water-tight object surface.

Existing non-rigid Structure from Motion (SFM) techniques (e.g., [9,10]) can only

handle small deformation or viewpoint changes. Encouraged by the recent development of full-frame range sensors and the rapid progress in stereo matching research, I expect that color+depth maps captured in the video rate will be practically available soon. The focus of this part is how to fuse partial deformable surfaces over time to form a complete model. In general, this *deformable model completion* task is an ill-posed problem [10] — the occluded part can be in any shape at any instance. Fortunately most dynamic cases behave continuously in a short temporal interval as we observe in the real world, even though this may not be valid in rare cases when extreme deformation happens under an sudden impulse (such as the popping of a balloon). Under this assumption, I seek to produce a *visually* plausible model that is deforming naturally and consistently with the input.

My entire modeling pipeline can be separated into three steps. In the first step, an image sequence is captured using a depth camera (or a stereo camera). Each captured depth map defines a partial surface of a deforming object at each time instance, and I use the image sequence to locate temporal point correspondences. Those correspondences are then used as anchor points in the second step to warp partial surfaces, so they become part of the same object surface at the same time instance. Extended from variational linear mesh deformation approaches [11], I developed a global deformation algorithm, in order to warp all partial surfaces together to their destination positions in a single step. After that, partial surfaces are assembled together into a complete watertight surface using a volumetric method in the third step. All surfaces are also optimized in order to complete missing regions and remove remaining errors at the same time. Compared with the ground truth deformation data, the experiment

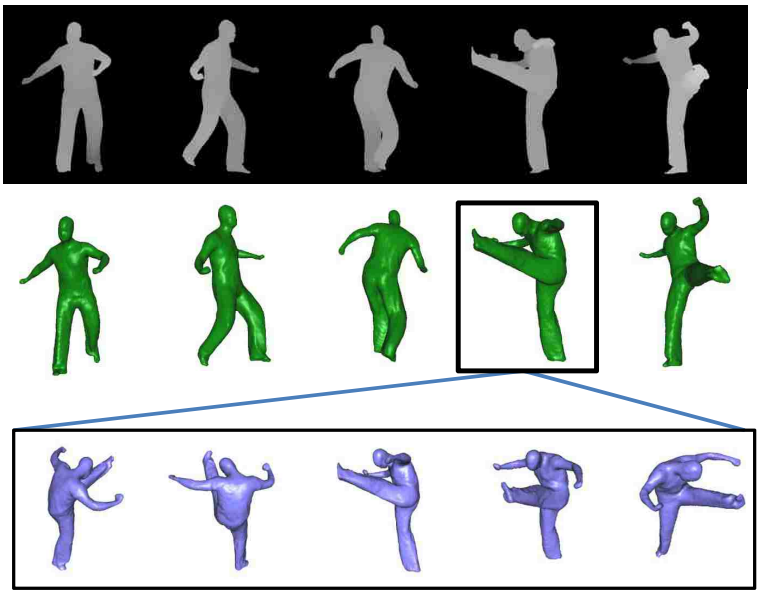

Figure 1.3: The input to my system is range data ($1^{st}$ row) captured by a single camera at different times, which is simulated with the data shared by [12]. And the output is a sequence of watertight 4D models ($2^{nd}$ row) reconstructed from a dynamic object. The $3^{rd}$ row shows one 3D model in different views.

shows that my approach can accurately recover the time-varying 3D shape sequence of a deforming object (as shown in Figure 1.3).

### 1.3.3 2D-3D Video Conversion

On the other hand, if we don't have a depth camera but a regular camcorder, this means we are given a video sequence without any depth information. I explore the possibilities to turn these sequences into stereoscopic video pairs, enabling 3D perception and lifelike viewer experiences (Figure 1.4). As cognitive studies by Koenderink and his colleges [1] show, the human visual system is more tuned in to depth order and precedence than to absolute depth. Therefore, accurate disparity values are not needed in order to create an immersive 3D viewing experience.

The production of filmed (as opposed to CG) 3D content requires either the cap-

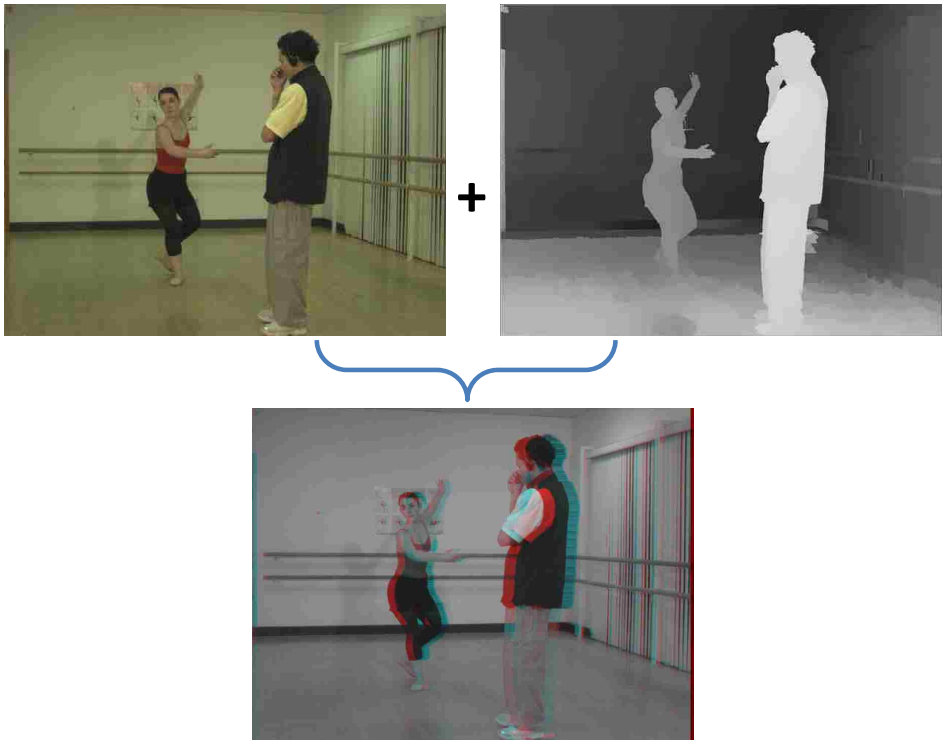

Figure 1.4: Turning an image into an stereo image by depth map. Image copyright of Zitnick [6].

ture of stereoscopic or multiple-camera content, or the conversion of 2D to 3D content in post-production. The former has several disadvantages, including specialized equipment and production pipeline. Conversion technologies, on the other hand, can be applied to any existing conventional content, inducing older material. However, despite the conversion advantages, most 3D content today is created by specialized cameras and not by conversion technologies. A notable exception is the June 2006 release of the movie "Superman Returns", which included 20 minutes of 3D images converted from the 2D original digital footage. It was recently declared that a company called "In-Three" may convert all six "Star Wars" movies to 3D, in a process that seems to be mostly manual.

Although there exists commercial software (e.g. Tri-Def $DDD^{©}$) that automatically turns images into stereo ones, the 3D effect is lacking. Understandably, converting a single image or a monocular video into stereo requires knowing the depth information for every pixel in the scene. Recovering 3D range from a single 2D image is an ill-posed problem. A universal approach that requires no user interaction will be extremely difficult, if not impossible, to achieve. The goal of my research is to reduce the amount of user interaction as much as possible by *taking advantage of the movement in the scene.*

Although structure-from-motion techniques (SFM) is an obvious choice and has been explored in several prior papers [13–15], I introduce two more methods to estimate scene depth. The first is based on the observation that in many *follow* shots, the foreground remains in the center of the screen and changes little while the background changes frequently. The second is based on the observation that as an object moves towards or far away from the camera, its image size varies accordingly. By analyzing the optical flow between frames, we can perform automatic foreground segmentation and determine the relative depth changes of moving objects.

All the analysis based on movement, caused by object motion, camera motion or both, leads to constraints on the scene depth. For certain videos, these constraints are enough to automatically obtain a scene depth map from which stereo view synthesis is possible. For more complex videos, or videos with little or no movement, user interaction is only needed in places where depth cues cannot be extracted automatically. The user interaction only requires sparse scribbles indicating relative depth (inequality and equality). *Combining the prior analysis through motion and*

*the depth ordering by user interaction*, my system can generate dense depth maps that produce visually plausible 3D stereo images for a variety of scenes and shots.

## 1.4 Innovations

The overall contribution of this thesis is the accomplishment of those multiple-camera tasks using a single camera, overcoming the limitations of a camera array. It makes several solid steps towards the goal of allowing the amateur users to build and re-explore the world in 3D, using a hand-held camera. It provides some new insight into these challenging research areas.

In addition to the overall contributions to the level of applications, there are a few technical innovations that can lead to several publications in the related conferences or journals.

**Light Fall-off Stereo** I developed a novel way to estimate depth information from scenes beyond Lambertian reflectance model. I also developed a global optimization-based method that uses multiple light variations to further improve the accuracy and robustness. The effectiveness of LFS is demonstrated by a variety of real-world scenes exhibiting complex reflectance and geometries.

**Real-time LFS Camera** I developed a novel depth range system that can generate a VGA (640x480) resolution depth map at 30Hz. In order to toggle between two LED lights in a fast and accurate way, I designed a dedicated circuit to control the state of the LED lights and receive synchronization signals to the camera. I also immigrated

the whole computation of the depth map to GPU, achieving real-time performance. In terms of quantitative accuracy, my system compares favorably to other commercial 3D range sensors, particularly in textured areas. In addition, my system is made of commodity off-the-shelf components, offering an inexpensive solution to real-time, high-resolution, video-rate range sensing.

**Single View Reconstruction of Deformable Model** To the best of my knowledge, I present the first method to generate a complete deformable model using a single depth camera. This is made possible by two main technical contributions: a global linear method to fuse all deformable meshes into a complete model, and a volumetric method to refine the 4D model for hole-filling and smoothing. With the wider availability of depth sensors, I hope that my approach can eventually push the continued digitalization of our world toward dynamic scenes.

**Global mesh deformation and alignment** The alignment between surfaces of the deformable and dynamic object captured at different time needs not only the rigid transformation [16], but also laplacian surface deformation [17] controlled by feature points. Laplacian surface deformation ensures that the surface shape is unchanged as much as possible, while at the same time, the control points continue to coincide with their destination positions. However, previous mesh deformation techniques only address the issue of deforming one mesh at a time, which means that we can only sequentially stitch different pieces of surface to reconstruct a complete 3D model. In most cases, sequential alignment may lead to misalignment between the first and

the last surface, due to accumulative errors. I proposed to modify the traditional mesh deformation to align all pieces of surfaces globally and simultaneously, so that a complete model can be obtained. This research work is already published in ICCV 2009 [18].

**Motion analysis** I developed two novel techniques that automatically estimate the 3D information from video sequences. Unlike SFM that requires non-axis camera movement (e.g., dolly, crane), my techniques can work with arbitrary camera/object movement, such as camera pan or zoom, which are frequently used in both everyday video and professional shots.

**Intuitive user interface** I provide a user-friendly interface that requires users to label depth relationships other than depth value on the images. My UI design benefits from the already defined 3D cues by the pre-processing of movement, providing users with a more intuitive and less labor intensive UI environment. In the case that none of the 3D cues can be inferred in the pre-processing step, my labeling can still simulate the direct depth labeling under the same amount of manual work.

**Quadratic programming formulation** I formulate the sparse to dense depth propagation as a quadratic programming problem, which could elegantly integrate both relative (such as layer orders) and absolute (such as that from SFM) depth constraints.

## 1.5 Dissertation Outline

The remainder of my dissertation is organized as follows:

In chapter 2, I will review the related work in detail. I divide the existing methods of 3D reconstruction into two categories. The first category attempts to recover the depth/range information from a single view, which is reviewed in section 2.1. The output of those methods are piecewise surfaces, or 2.5D models. My LFS reconstruction method falls into this category. The second category builds the complete 3D models of real objects directly from the 2D images or from the captured piecewise surfaces. Those methods that fall into the second category are reviewed in section 2.2. The proposed method of building 4D models for dynamic objects also belongs to this category. In section 2.3, I will review the previous work on 2D-3D video conversion, and discuss the advantages and disadvantages of each method.

In chapter 3, I will introduce the LFS reconstruction method. The basic idea of LFS will be discussed in section 3.1. Since the practical LFS system is based on the approximation of the ideal case, an error analysis is conducted in section 3.2. The results of the error analysis provide us with a guidance to build the real-time depth sensor. Although a minimum of 2 images are enough to estimate scene depth, multiple images could be used to generate better depth estimation using the global approach, which is introduced in section 3.3. The issues of building the real-time depth sensor from the LFS theory is discussed and solved in section 3.4. And some of the results are shown in section 3.5.

In chapter 4, I will discuss how to remove the matching outliers in section 4.1

before getting into the details of the surface alignment algorithm in section 4.2. Most of the time, the alignment algorithm will not give us a 100% correct and complete 3D model. There are some more issues at hand. For example, the occlusion handling and missing feature correspondences need inclusion. I will discuss these issues in section 4.3. The results and conclusion are in section 4.4 and section 4.5, respectively.

I will introduce my 2D-3D video conversion algorithm in chapter 5. The innovations of my work, automatic motion analysis, intuitive user interface and new formulation of depth propagation, are discussed in section 5.1, section 5.2 and section 5.3, respectively. Some of the results generated by my system are shown in section 5.4. Since I build a new 2D-3D video conversion system that aims at reducing manual work, I conducted a user study to rate its success. The results of the user study show that my system saves much of their time, while also generating comparable results as previous methods. The methodology and results of the user study are presented in section 5.5.

Finally, I will conclude my dissertation work and discuss the future directions in chapter 6.

# Chapter 2 Background and Related Work

My proposed framework was motivated by the demonstrated success of previous works on a variety of computer vision and computer graphics topics. In this chapter, I will set the context of this dissertation and introduce several related previous approaches.

## 2.1 Shape Recovery

Depth recovery of an unknown scene from a set of images is one of the oldest problems in computer vision. One major driving application is autonomous robots, which need to understand the shape of the scene to navigate throughout. Other applications include surveillance, reverse engineering, and human-computer interactions. depth recovery has been and continues to be one of the most active research areas in computer vision; many methods and techniques have been tried and tested. These methods vary vastly as to the image features used and the underlying scene representation. I review several categories of methods, which are the most widely-used methods and are most relevant to this dissertation.

### 2.1.1 Stereo Vision Methods

Stereo vision attempts to infer depth information from images. Although a single image contains a lot of information about an observed scene, it loses the depth information. This is due to the nature of the image formation process, which consists of a projection from a 3D scene onto a 2D image plane.

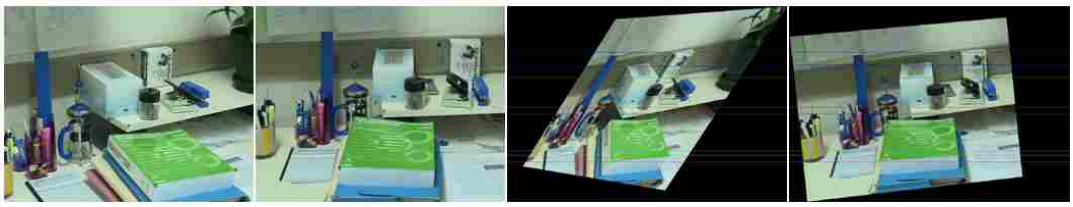

Figure 2.1: A pair of stereo images before and after rectification. The first two are the original images, while the last two are the rectified images. Note that the corresponding features are on the same horizontal scan line after rectification.

Figure 2.3 shows this. With a single 2D image, we only know that a 3D point lies on the ray that connects the camera center and the the pixel of the point on the image plane; it is impossible to know which point on this ray corresponds to the image pixel. If two (or more) images are available, then, the intersection of the two lines of sight can uniquely determine a 3D point. This process is called triangulation. In order to reconstruct a 3D point, we must know the corresponding image pixels between the two images, relative positions and orientations of the cameras (extrinsic camera parameters) and the relation between image pixels and the corresponding lines of sight (intrinsic camera parameters).

Although there are several techniques for recovering depth with unknown, or partially unknown, camera parameters [7,19–21], I will limit the scope of this dissertation, as in many existing reconstruction techniques, to calibrated cases. That is, both the intrinsic and extrinsic camera parameters are known a priori. The central problem for stereo vision thus becomes finding the correspondences between two images.

The earliest attempt to solve the stereo problem, by Marr and Poggio, dates back to 1976 [22]. Since then, stereo matching has been one of the most active research areas in computer vision. Compiling a complete survey of existing correspondence-

based stereo methods would be a formidable task. A large number of new methods are published every year. However, there are a few excellent survey papers collectively covering the history of stereo vision. There are two very good reviews of early vision; the first is by Barnard and Fischler [23] and covers the early 70s and 80s. The second is by Dhond and Aggarwal [24] and covers the late 80s. More recently, Scharstein and Szeliski [25] updated us on the current state of art. They also introduced a taxonomy of two-view, or binocular, stereo algorithms that allows the dissection and comparison of individual algorithm component design decisions. First I will provide a more rigorous definition of the binocular stereo problem , and then I will use the taxonomy from Scharstein and Szeliski to review a number of binocular stereo algorithms. I will further examine several multi-view stereo methods that use more than two images.

**Binocular Stereo Representation**

Most binocular stereo correspondence methods compute an univalued disparity function d(u, v) with respect to one of two reference images. The term disparity was first introduced in the human vision literature to describe the difference in location of corresponding features seen by the left and right eyes [26]. For a 2D feature s in one reference image, say the left image, its corresponding 3D point is constrained to be on the line of sight. This lines projection in the other image is called the epipolar line. Thus the corresponding feature s' in the right image must be on the epipolar line. This is the important epipolar constraint which reduces the search space of corresponding features to one dimension. In computer vision, input images are usually

transformed so that the epipolar lines are aligned horizontally. This process is called rectification [27–30]. A pair of stereo images before and after rectification is shown in Figure 2.3. For a pair of rectified images, disparity can be treated as synonymous with inverse depth, i.e., a large disparity value means that the 3D point is close, a small disparity value means that the 3D point is further away, and a zero disparity value means that the 3D point is at infinity.

Given a pair of rectified images, let (u, v) be the pixel coordinates in a reference image chosen from the pair, say the left image. The correspondence between a pixel (u, v) in a reference image and a pixel (u', v') in matching image is then given as

$$u' = u + d(u, v), v' = v \tag{2.1}$$

The goal of a stereo algorithm is then to produce a univalued function d(u, v) that best describes the shape of the surfaces in the scene [25].

### A Taxonomy of Stereo Algorithms

Scharstein and Szeliski [25] proposed a taxonomy based on the observation that stereo algorithms generally perform (subsets of) the following four steps:

- matching cost computation.
- cost aggregation.
- disparity computation / optimization.
- disparity refinement.

**Matching cost computation** A matching cost is a value indicating how likely two pixels are to correspond to the same scene point. The three most common pixel-based matching costs include squared intensity differences (SD) [31–33], absolute intensity differences (AD) [34], and normalized cross-correlation [31, 35, 36]. They all behave similarly in terms of disambiguating power [25].

Besides the above three, there are many other cost criteria that are designed for specific needs. Some costs are insensitive to differences in camera gain or bias; these include, for example, gradient-based measures [37, 38] and non-parametric measures such as rank and census transforms [39]. More recently, robust measures such as truncated quadratics and contaminated Gaussians [40–42] have been introduced to limit the influence of mismatches during cost aggregation (the next step). Other cost criteria include phase and filter-bank responses and the sample-insensitive cost measure developed by Birchfield and Tomasi [43].

All these cost measures assume the scene surfaces are Lambertian, i.e., that their appearance does not vary with viewpoint. This poses a significant restriction on the type of scenes a stereo algorithm is able to reconstruct. Later in this dissertation, I will introduce a novel matching cost that is valid for both specular and Lambertian surfaces.

**Matching cost aggregation** The matching cost for each pixel is usually ambiguous and noisy. To reduce ambiguity, many stereo algorithms use a local and window-based approach: matching costs are aggregated by summing or averaging over a support region. Most stereo algorithms use a 2D support region at a fixed dispar-

ity, as in [44–50]. Such approaches favor front-parallel surfaces. By contrast, a 3D support region in x-y-d space, as in [51, 52], does not have this bias, thus supporting both slanted and front-parallel surfaces.

A different method of aggregation is iterative diffusion, i.e., an aggregation (or averaging) operation that is implemented by repeatedly adding to each pixels cost the weighted values of its neighboring pixels costs [42, 53, 54]. The work presented in this dissertation uses a similar iterative strategy in applying a view-dependent smoothness constraint.

**Disparity computation and optimization** There are two broad classes of methods used at this stage, local methods and global methods. In local methods, the disparity computation at a given pixel depends only on the intensity values within a finite window. Computing the final disparities is trivial: one simply chooses for each pixel, the disparity associated with the minimum cost value. Thus, these methods perform a local "winner-take-all" (WTA) optimization for each pixel. A major limitation is that the uniqueness of matches is enforced for only one image (the reference image), while pixels in the other image might have multiple matches.

Global methods, by contrast, are usually formulated within an energy-minimization framework [55]. The objective is to find a disparity function d that minimizes the global energy, such as:

$$E(d) = E_{data}(d) + \alpha E_{smooth}(d) + \beta E_{visibility}(d) \tag{2.2}$$

The data term, $E_{data}(d)$, measures how well the disparity function d agrees with

the input image pair. The smoothness term, $E_{smooth}(d)$, encodes the smoothness assumptions made by the algorithm. To make the optimization computationally tractable, the smoothness term is often restricted to measuring only the differences between neighboring pixels disparities. The last visibility term is often a bi-valued termzero if the visibility constraint is satisfied, infinity otherwise.

Once the global energy has been defined, a variety of algorithms can be used to find a (local) minimum. Traditional approaches associated with regularization and Markov Random Fields include continuation [56], simulated annealing [57–59], highest confidence first [60], and mean-field annealing [61]. More recently, max-flow and graph-cut methods have been proposed to solve a special class of global optimization problems [62–67]. Each of these methods constructs a graph such that the maximum flow or minimum cut on the graph also minimizes the energy. These approaches are more efficient than simulated annealing and have produced good results. Kolmogorov and Zabih have characterized the energy functions that can be minimized by graph-cut [68].

Although the optimization of Equation 2.2 can be shown to be NP-hard for common classes of smoothness functions [65], dynamic programming can be used to find the global minimum for each scanline independently in polynomial time [45, 69–72]. A major limitation of this approach is the difficulty of enforcing inter-scanline consistency.

Although global methods tends to produce better results than local methods, they are typically sensitive to the precise definition of the global energy. Usually these parameters are tuned empirically, and parameters that work well for one data

set may not necessarily be good for others. In contrast, local methods typically have fewer parameters and generate consistent, albeit not optimal, results. Local methods are also computationally feasible for real-time implementations.

**Disparities refinement** Most stereo algorithms compute a disparity map using only integer values. To improve quality, the disparity values can be interpolated for sub-pixel accuracy. Usually, the profile of matching costs for a pixel is fitted into a parabolic curve, then the local maximum is found as the refined disparity value [33, 73–76]. This increases the resolution of the disparity map with little computation.

Other types of disparity post-processing are also possible, such as applying a median filter to remove spurious mismatches or filling holes due to occlusion using surface fitting or distributing neighboring disparity estimates [43, 77].

### Multi-view Stereo

Although there has been significant progress on stereo algorithms during the last two decades, from simplistic but fast local methods to sophisticated global methods based on energy minimization, the problem of computing depth from two images has not been entirely solved. Indeed, some even deem it an ill-posed problem in general [55, 58, 78]. There simply is not enough information to distinguish correct correspondences from false positives in many practical cases. Various constraints and assumptions have to be imposed to make the problem tractable.

To ameliorate the above problems, Okutomi and Kanade in 1993 proposed the use of more than two images in stereo [79]. Using one of the input images as the

reference image, the matching costs from all the other images are summed up to a final matching cost. This new cost is more salient to image noise and false positives. Since the search for matches is still performed within the disparity space, there is a major restriction on the camera arrangement: the cameras have to be co-planar or even co-linear.

In 1996, Collins [80] proposed a plane-sweep algorithm, which projects all images onto a series of planes in 3D space that correspond to different disparity values. Matching is performed in 3D space. This allows more flexible camera configurations. But an important problemocclusionis still not addressed. This is particular important in the multi-view case, since as more and more cameras are added, it becomes less and less likely that all the cameras will see the same surface. In the next section, I will introduce several reconstruction methods based on a volumetric representation of the scene. These methods can deal elegantly with the multi-view reconstruction problem.

### 2.1.2 Shape from X

We have an image formation model, and we have methods for processing images and extracting structure. We would now like to "invert" this and use the image properties that we measure to infer 3D scene properties. In computer vision, this process of recovering shape are called shape-from-X techniques, where X can be shading, stereo, motion, texture, etc. since I have introduced stereo matching algorithms in previous section, I will discuss other approaches that have been explored to infer depth.

### Shape from Texture

Shape from texture recovers a surface model from a projection of a texture field that is assumed to lie on that surface. Global methods attempt to recover an entire surface model, using assumptions about the distribution of texture elements. Appropriate assumptions are isotropy [81] (the disadvantage of this method is that there are relatively few natural isotropic textures) or homogeneity [82]. Methods based around homogeneity assume that texels are the result of a homogeneous Poisson point process on a plane; the gradient of the density of the texel centers then yields the plane's parameters. However, deformation of individual texture elements is not accounted for.

Figure 2.2: Illustration of shape from texture. Image is from [83].

Local methods recover some differential geometric parameters at a point on a surface (typically, normal and curvatures). This class of methods, which is due to Garding [84], has been successfully demonstrated for a variety of surfaces by Malik and Rosenholtz [85]; a reformulation in terms of wavelets is due to Clerc [86]. The method

has a crucial flaw; it is necessary either to know that texture element coordinate frames form a frame field that is locally parallel around the point in question, or to know the differential rotation of the frame field. For example, if one were to use these methods to recover the curvature of a doughnut dipped in chocolate sprinkles, it would be necessary to ensure that the sprinkles were all parallel on the surface (or that the field of angles from sprinkle to sprinkle was known). As a result, the method can be demonstrated to work only on quite a small class of textured surfaces.

### Shape from Shading

Shading plays an important role in human perception of surface shape. Researchers in human vision have attempted to understand and simulate the mechanisms by which our eyes and brains actually use the shading information to recover the 3-D shapes. Ramachandran [87] demonstrated that the brain recovers the shape information not only by the shading, but also by the outlines, elementary features, and the visual system's knowledge of objects. The extraction of SFS by visual system is also strongly affected by stereoscopic processing. Barrow and Tenenbaum discovered that it is the line drawing of the shading pattern that seems to play a central role in the interpretation of shaded patterns [88]. Mingolla and Todd's study of human visual system based on the perception of solid shape [89] indicated that the traditional assumptions in SFS-Lambertian reflectance, known light source direction, and local shape recovery-are not valid from psychology point of view. One can observe from the above discussion that human visual system uses SFS differently than computer vision normally does.

Recently, Horn, Szeliski and Yuille [90] discovered that some impossibly shaded images exist, which could not be shading images of any smooth surface under the assumption of uniform reflectance properties and lighting. For this kind of image, SFS will not provide a correct solution, so it is necessary to detect impossibly shaded images.

SFS techniques can be divided into four groups: minimization approaches [91–97], propagation approaches [98–104], local approaches [105,106] and linear approaches [107, 108]. Minimization approaches obtain the solution by minimizing an energy function. Propagation approaches propagate the shape information from a set of surface points (e.g., singular points) to the whole image. Local approaches derive shape based on the assumption of surface type. Linear approaches compute the solution based on the linearization of the reflectance map.

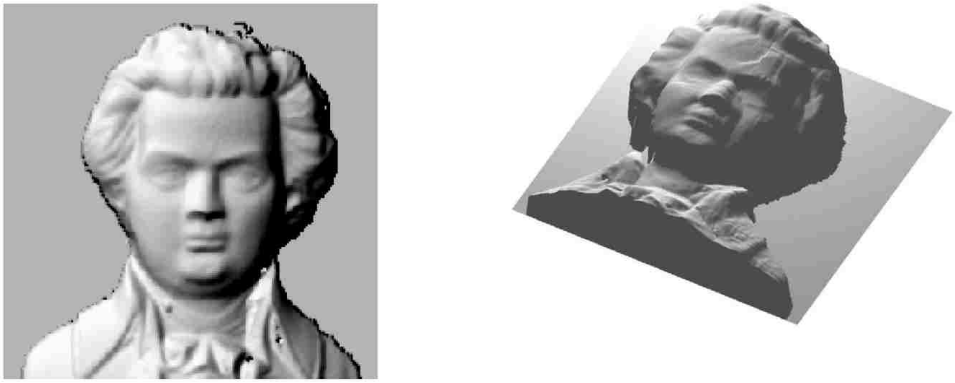

Figure 2.3: Illustration of shape from shading. Image is from [109].

### Shape from Focus/Defocus

It is well-known that the focus setting that yields the sharpest image is useful not only to take good photos, but also to infer the depth map of the scene [110, 111]. In

commercial cameras, such focus setting is found with relative ease by the autofocus function. One may then wonder whether the search for the best in focus position amounts to finding the unique minimum of a smooth and convex curve as shown in Figure 2.4.

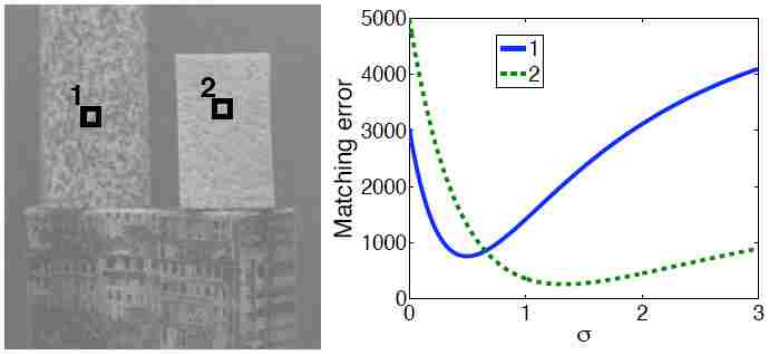


Figure 2.4: Quasiconvexity of shape from focus/defocus.

Favaro [112] extended the analysis to the more general problems of shape from de-focus [113–115] and image restoration [57, 116, 117]. The analysis is divided into two parts: one that studies the convexity of shape from focus/defocus and image restoration, and the other that studies the sensitivity of shape estimation as a function of the input data. In the first part, to keep the analysis as general as possible, Favaro [112] analyzed cost functionals that can be written in the form of Bregmans divergences [118]. In particular, this family of divergences includes two important and common cases: least-squares and the Kullback-Leibler divergence. The conclusion is that shape from defocus enjoys strict quasiconvexity (as shown in Figure 2.4), and therefore simple local methods, such as gradient-flow algorithms, can be used to find global optima. This conclusion is verified by showing that a simple gradient descent method and a method based on graph cuts [119] yield approximately the

same solution.

### Structure from Motion

Structure from motion (SFM) recovers the 3D geometry of the scene points as well as the camera positions by analyzing the motion of the tracked feature points across multiple images. To find correspondence between images, features such as corner points (edges with gradients in multiple directions) need to be tracked from one image to the next. The feature trajectories over time are then used to reconstruct their 3D positions and the camera's motion. The SFM methods fall into two major categories: sequential method and factorization method.

**Sequential Methods** Sequential algorithms are the most popular. They work by incorporating successive views one at a time. As each view is registered, a partial reconstruction is extended by computing the positions of all 3D points that are visible in two or more views using triangulation. A suitable initialization is typically obtained by decomposing the fundamental matrix relating the first two views of the sequence. There exist several strategies for registering successive views:

- **Epipolar constraints**. One possibility is to exploit the two-view epipolar geometry that relates each view to its predecessor. For example, where camera intrinsic parameters are known, essential matrices can be used. Essential matrices are estimated linearly using 8 or more point correspondences and decomposed to give relative camera orientation and the direction of camera translation. The magnitude of the translation can be fixed using the image in the new view of a

single known 3D point, i.e. a point that has already been reconstructed from its image in earlier views.

- **Resection**. An alternative is to determine the pose of each additional view using already-reconstructed 3D points [120–123]. 6 or more 3D to 2D correspondences allow linear solution for the 12 elements of a projection matrix.

- **Merging partial reconstructions**. Another alternative is to merge partial reconstructions using corresponding 3D points [124,125]. Typically, two- or three-view reconstructions are obtained using adjacent image pairs or triplets; then they are merged using corresponding 3D points. In [125], longer reconstructions are built up hierarchically by merging progressively longer subsequences.

**Factorization Methods** Unlike sequential methods, batch methods work by computing camera pose and scene geometry using all image measurements simultaneously. One advantage is that reconstruction errors can be distributed meaningfully across all measurements; thus, gross errors associated with sequence closure can be avoided.

One family of batch structure from motion algorithms are called factorization methods (after Tomasi and Kanade [7]). Fast and robust linear methods based on direct SVD factorization of the image point measurements have been developed for a variety of simplified linear (affine) camera models, e.g. orthographic (Tomasi and Kanade [7]), weak perspective (Weinshall and Tomasi [126]), and para-perspective (Poelman and Kanade [127]). Unfortunately, none of these methods is generally applicable to real-world scenes because real camera lenses are too wide-angle to be

approximated as linear.

More recently, a number of researchers have described "factorization-like" algorithms for perspective cameras too (Sturm and Triggs [128], Heyden [129], Schaffalitzky et al. [130]). However, these methods are iterative and there is no guarantee that they will converge to the optimal solution.

Again, one limitation of all of these algorithms [7, 126–130] is that there exist degenerate structure and motion configurations for which they will fail. Another is that they cannot cope with missing data, i.e. every 3D points must be visible in every view. Hence, they are not applicable to sparse modeling problems (except perhaps as a means of initializing sequential algorithms like in [130]). Jacobs [131] overcomes this limitation, but only for affine cameras (and at the expense of the optimality of the solution).

**Non-rigid structure from motion**

Non-rigid structure from motion (NRSFM) studies how to extract 3D shape and motion of dynamic objects from tracked feature points. The motion of non-rigid moving object can be decomposed into a rigid transformation and a non-rigid deformation. Since NRSFM problem is ill-posed, additional assumptions about the deforming shape are needed to make it solvable. Linear subspace model is used for handling objects with small deformations such as human faces [9, 132]. Torresani et al. [10] assume that the object shape variation follows Gaussian distribution and simultaneously estimate the 3D shapes, the rigid motion, and the parameters of the Gaussian using an Expectation-Maximization (EM) algorithm. In their later work,

a Probabilistic Principal Components Analysis model is used instead of the linear subspace model [133].

Different from existing NRSFM algorithms [10, 133], which estimate the 3D position and motion of sparse feature points only, the presented algorithm tries to generate complete 3D model of the dynamic objects. In addition, by assuming the object consists of rigid moving parts and non-rigid moving joints, the presented algorithm can handle articulated objects with large deformation.

## 2.2 Shape Completion

Previously introduced methods focus on recovering the depth/range information of a given scene. The output are dense or sparse depth maps corresponding to the input images. We call this kind of output 2.5D model. In the following subsections, I will introduce the existing approaches that build complete 3D models given the image sequences and the recovered depth maps.

### 2.2.1 Volumetric Curving

Instead of searching in image space as in stereo algorithms, an alternative approach to scene reconstruction is based on computations in 3D scene space, in which a volumetric representation of the scene can be inferred from input images. Volumetric methods usually assume there is a known, bounded volume in which the objects of interest lie. The most common approach to representing this volume is as a regular tessellation of cubes, called voxels, in Euclidean 3D space. The task of a reconstruction algorithm is to decide which voxels belong to the objects of interest and which

do not. Most methods also assign a color to each voxel based on its projections into input images.

Compared to the disparity space representation used in most stereo algorithms, a volumetric representation is much more flexible and general. In particular, it allows a much wider range of camera arrangements, and more elegant modeling of visibility in 3D scene space.

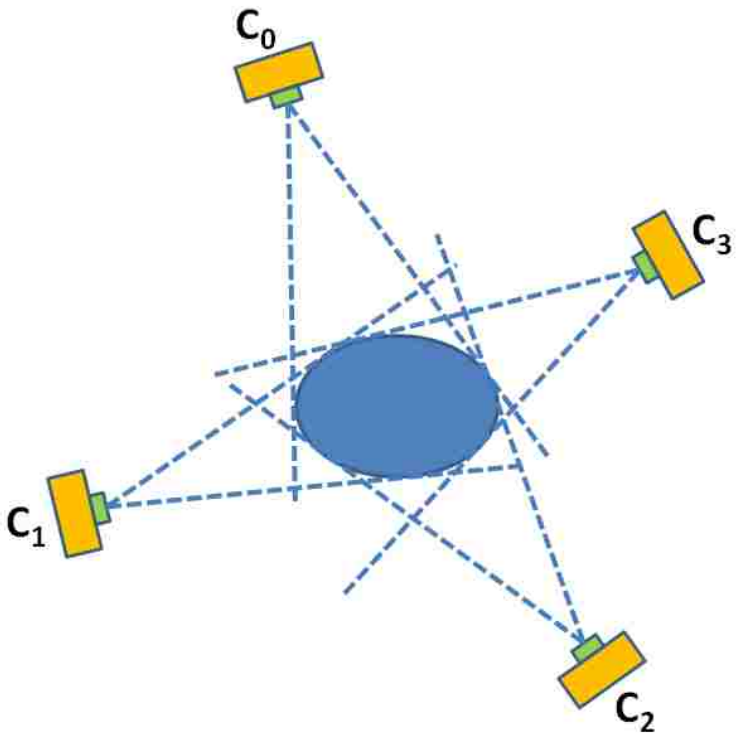


Figure 2.5: Visual hull illustration by 2D examples.

Recently there has been considerable progress in developing techniques that build volumetric scene models. There are two broad classes of volumetric reconstruction methods; one uses only silhouette information to compute the visual hull of the original shape; while the other uses photo-consistency measures to compute the photo hull of the original shape. I will review these two classes in the sections that follow. Inter-

ested readers are also encouraged to read two recent reviews of different volumetric scene reconstruction methods by Slabaugh [134] and Dyer [135], respectively.

### Visual Hull Reconstruction

An object's contour (or profile) provides important clues about the object shape. Suppose a 3D object is viewed by a camera. The object's silhouette image, which can be obtained using segmentation algorithms or blue-screen techniques, contains values that distinguish regions where the object is or is not present. Combined with calibration information for the camera, each pixel in a silhouette defines a ray in scene space that intersects the object at some unknown depth. The union of these visual rays for all pixels in the silhouette defines a generalized cone within which the 3D object must lie in, as showed in Figure 2.5. If we are presented with multiple views of the object, the intersection of these generalized cones from all views defines a volume of the scene space that must contain the original object. As the number of the reference views goes to infinity to include all the views possible from all locations, the intersection volume converges to the shape known as the object's visual hull, a term defined by Laurentini [136]. The visual hull is guaranteed to contain the object. In 2D, the visual hull is equal to the convex hull of the object. For 3D scenes, the visual hull is a tighter fit than the convex hull. In a visual hull, hyperbolic regions are removed, but concavities are not.

In practice, since one has access to only a finite number of views, one can only construct approximate visual hulls. Given a set of n silhouette images from different views, the approximate visual hull is the best conservative geometric description one

can achieve based on silhouette information alone. From now on, I will use the term "visual hull" to mean the approximate volume computed from n views.

Central to visual hull reconstruction is the intersection test. If the silhouettes are described using a polygonal mesh, the visual hull representation can be constructed using a series of 3D constructive solid geometry (CSG) intersections [137]. But it is well known that polyhedral CSG operations are very hard to perform in a robust manner due to numerical inaccuracy [74].

A more common approach is to reconstruct a quantized representation of the visual hull [136, 138–149]. Starting from a bounding volume that is known to enclose the entire scene, the volume is discretized into voxels. Voxels falling outside of the back-projected silhouette cone of any given view are eliminated from the volume. This can be done efficiently by projecting each voxel into 2D images and testing whether it is contained in every silhouette. In the end, only voxels that are in the intersections of back-projected silhouette cones from all views are retained.

To make the voxel traversal more efficient, most methods use an oct-tree representation and test voxels in a coarse-to-fine hierarchy [143, 145, 146, 150]. The volume enclosing the entire scene space is initialized to a single voxel. The current voxel is projected into all the views and tested to determine whether it is inside the silhouette in each view. If the projected voxel is outside the silhouette in at least one view, the voxel is removed (marked transparent). If the projected voxel is inside the silhouette in every view, the voxel is retained. Otherwise, the voxel intersects both background and silhouette points in some views, so it is subdivided into octants and each sub-voxel is processed recursively. This process terminates when no subdivision

is necessary or when the size of the subdivided voxels reaches a user-defined threshold.

### Photo Hull Reconstruction

Shape-from-silhouettes methods avoid the difficult correspondence problem associated with stereo vision, and are thus quite robust if the segmented silhouette images are accurate. However, concavity cannot be preserved by the visual hull presentation. In practice, image segmentation is not always possible. In addition, the color information about the objects is not used in any shape-from-silhouettes method, except to assign color to a voxel model already reconstructed.

In 1998 Seitz presented a volumetric reconstruction method, voxel coloring, that makes full use of the photometric information contained in input images [151]. The basic idea is to reconstruct the 3D scene model that best reproduces the input images when rendered from the perspectives that correspond to the input images. Starting from a regular 3D voxel grid that encloses the scene, the goal is to assign colors and binary transparency values to voxels so as to achieve photo-consistency with a set of input images (shown in Figure 2.6). That is, rendering the colored voxels from each input viewpoint should reproduce the original image as closely as possible. Assuming a Lambertian scene, a voxel on the scene surface is photo-consistent with a set of images if, for each image in which it is visible, the voxel's color is equal to the color of its corresponding image pixel.

Conversely, in order to determine a voxel's photo-consistency, we can project it onto un-occluded images and test whether or not the pixels in its projection have equal color values. In the presence of image noise or quantization effects, we can evaluate

the correlation of the pixel colors to measure the likelihood of voxel consistency. Let s be the standard deviation of the pixel colors. One possibility is to threshold the color space error. If s is smaller than a user-defined threshold $\lambda$, the voxel is considered to be photoconsistent and is assigned a colorthe mean of the pixel colors. Otherwise, the voxel is considered to belong to the free space and is marked as transparent. Alternatively, a statistical measure of voxel consistency can be used [151]. This has been done using an F test, where a photo-consistency score is computed by the ratio of the variances of pixels' colors and the colors of pixels associated with a known homogeneous surface. A threshold on this score determines the photo-consistency of the voxel. Experiments using other definitions of photo-consistency have also been conducted [134, 152–154].

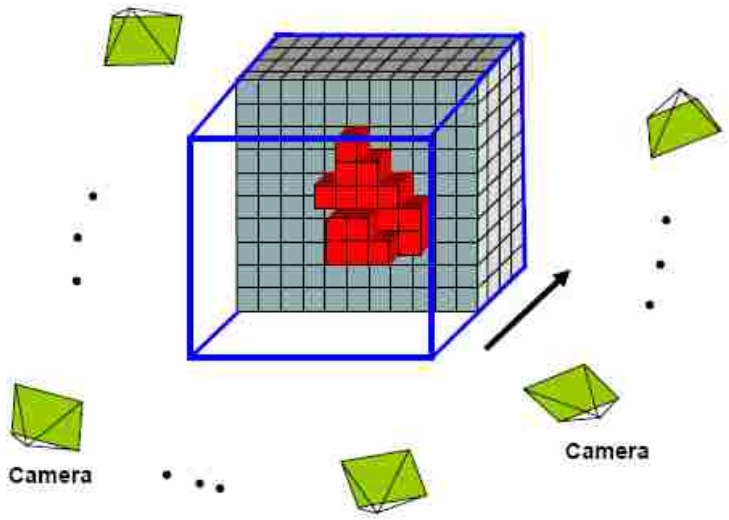


Figure 2.6: Volumetric Reconstruction Using Photo Consistency. Image is from [155].

Note that the photo-consistency test should only be applied to these visible pixels. To avoid a combinational search of all visibility configurations, efficient methods for

determining the visibility of each voxel are essential. A voxel y can occlude a voxel x if and only if y intersects at least one of the line segments connecting x to the optical centers associated with input views. If there exists a topological sort of voxels in which occluders must be before the voxels that are being occluded, i.e., if y occurs before x in the ordering, then the visibility test becomes tractable. We can traverse voxels in that order and guarantee that when a voxel is visited, all possible occluders for this voxel have been visited. Seitz pointed out that such an order, which is referred to as the occlusion-compatible order, does exist whenever no scene point is contained within the convex hull of the camera centers [151]. For instance, if the cameras and the scene are separated by a plane, voxels can be sorted by increasing distance from the plane, resulting in a sequence of voxel planes. We can traverse these planes from near to far, and only voxels in the previous planes, which have been processed and labeled, can occlude the voxels in the current plane, enabling a single-pass algorithm.

The voxel coloring algorithm is most closely related to the plane-sweeping algorithm first proposed by Collins [80]. However, with its explicit modeling and handling of visibility, the voxel coloring algorithm allows more flexible camera configurations and typically generates superior reconstruction results, especially for highly complex scenes in which occlusions and dis-occlusions frequently occur.

In 1999, Kutulakos and Seitz [153] extended the voxel color algorithm to allow even more flexible camera configurations. In essence, all views are partitioned into subgroups, so that within each subgroup, an occlusion-compatible order exists. The basic voxel coloring algorithm is applied successively for each subgroup to refine the voxel model from previous passes. They called this method the Space Carving

algorithm.

Culbertson et al. [156] presented another variation of the basic voxel coloring algorithm that also allows more flexible camera placement. At the expense of computer memory, a visible list of surface voxels for every pixel in every image is maintained. Thus visibility for a voxel can be quickly determined using all input images without the need for an occlusion- compatible order.

To deal with practical issues such as inaccurate calibration, Kutulakos [157] defined an Approximate Space Carving algorithm using a weakened photo-consistency test. Each voxel v is projected onto every input image at $p_1, \ldots, p_n$, respectively. If there is a common color for pixels within a small radius of $p_i$ in all input images, that voxel v is declared to be photo-consistent. Although this approach was designed to recover shapes from images with inaccurate calibration information, it can also be used to build a series of coarse-to-fine scene models from multiple views.

As always, voxel reconstruction can also be formulated as an energy minimization problem. Slabaugh et al. [158] used an iterative method, which they presented as a post-processing step, to add or remove surface voxels until the sum of the squared differences between the input images and the scene model rendered in each camera was minimized. Simulated annealing and greedy methods were used for optimization. This refinement effectively produces a spatially varying consistency threshold. More recently, Kolmogorov and Zabih [159] used graph-cut to optimize volume reconstruction directly. In order to make optimization tractable, the visibility test was approximated in their formulation. Although strong results have been obtained for a few standard data sets with very small baselines (originally used for stereo), the

effectiveness of their method is yet to be evaluated using other multi-view data sets in which visibility changes are substantial.

### 2.2.2 Volumetrical Merging

Reconstructing a complete 3D model for an stationary object has been extensively studied from both computer graphics and computer vision perspectives [160, 161]. These approaches start from 2D depth maps of the object under different view directions and merge them into a single 3D model using volumetric approach. The 2D depth maps can be obtained from either laser scanner [160] or stereo matching technique [161]. My work, inspired by the method in [160], can be considered as extending the merging of partial depth/range maps to dynamic scenes.

### 2.2.3 3D Surface Registration (ICP)

The ICP (originally Iterative Closest Point, though Iterative Corresponding Point is perhaps a better expansion for the abbreviation) algorithm has become the dominant method for aligning 3D models based purely on the geometry, and sometimes color, of the meshes. The algorithm is widely used for registering the outputs of 3D scanners, which typically only scan an object from one direction at a time. ICP starts with two meshes and an initial guess for their relative rigid-body transform, and iteratively refines the transform by repeatedly generating pairs of corresponding points on the meshes and minimizing an error metric. Generating the initial alignment may be done by a variety of methods, such as tracking scanner position, identification and indexing of surface features [162, 163], "spin-image" surface signatures [164], com-

puting principal axes of scans [165]. Since the introduction of ICP by Chen and Medioni [166] and Besl and McKay [167], many variants have been introduced on the basic ICP concept. We may classify these variants as affecting one of six stages of the algorithm:

- Selection of some set of points in one or both meshes.
- Matching these points to samples in the other mesh.
- Weighting the corresponding pairs appropriately.
- Rejecting certain pairs based on looking at each pair individually or considering the entire set of pairs.
- Assigning an error metric based on the point pairs.
- Minimizing the error metric.

I now examine ICP variants for each of the stages listed above.

**Selection of Points**

I begin by examining the effect of the selection of point pairs on the convergence of ICP. The following strategies have been proposed:

- Always using all available points [167].
- Uniform subsampling of the available points [168].
- Random sampling (with a different sample of points at each iteration) [169].

- Selection of points with high intensity gradient, in variants that use per-sample color or intensity to aid in alignment [170].

- Each of the preceding schemes may select points on only one mesh, or select source points from both meshes [171].

A strategy such as random sampling will often select only a few samples in these features, which leads to an inability to determine certain components of the correct rigid-body transformation. Thus, one way to improve the chances that enough constraints are present to determine all the components of the transformation is to bucket the points according to the position of the normals in angular space, then sample as uniformly as possible across the buckets. Normal-space sampling is therefore a very simple example of using surface features for alignment; it has lower computational cost, but lower robustness, than traditional feature-based methods [162–164].

### Matching Points

The next stage of ICP that we will examine is correspondence finding. Algorithms have been proposed that, for each sample point selected:

- Find the closest point in the other mesh [167]. This computation may be accelerated using a k-d tree [172] and/or closest point caching [173].

- Find the intersection of the ray originating at the source point in the direction of the source point's normal with the destination surface [166]. We will refer to this as "normal shooting".

- Project the source point onto the destination mesh, from the point of view of the destination mesh's range camera [174, 175]. This has also been called "reverse calibration".

- Project the source point onto the destination mesh, then perform a search in the destination range image. The search might use a metric based on point-to-point distance [176], point-to-ray distance [177], or compatibility of intensity [170] or color [178].

- Any of the above methods, restricted to only matching points compatible with the source point according to a given metric. Compatibility metrics based on color [171] and angle between normals [179] have been explored.

The first four of these algorithms are accelerated using a k-d tree. For the last algorithm, the search is actually implemented as a steepest descent neighbor walk in the destination mesh that attempts to find the closest point.

### Weighting of Pairs

I now introduce the effect of assigning different weights to the corresponding point pairs found by the previous two steps. I consider four different algorithms for assigning these weights:

- Constant weight

- Assigning lower weights to pairs with greater point-to-point distances. This is similar in intent to dropping pairs with point-to-point distance greater than a

threshold, but avoids the discontinuity of the latter approach.

- Weighting based on compatibility of normals. Weighting on compatibility of colors has also been used [171], though we do not consider it here.

- Weighting based on the expected effect of scanner noise on the uncertainty in the error metric. For the point-to-plane error metric, this depends on both uncertainty in the position of range points and uncertainty in surface normals. The result for a typical laser range scanner is that the uncertainty is lower, hence higher weight should be assigned, for surfaces tilted away from the range camera.

### Rejecting Pairs

Closely related to assigning weights to corresponding pairs is rejecting certain pairs entirely. The purpose of this is usually to eliminate outliers, which may have a large effect when performing least-squares minimization. The following rejection strategies have been proposed:

- Rejection of corresponding points more than a given (user-specified) distance apart.

- Rejection of the worst n% of pairs based on some metric, usually point-to-point distance. As suggested by [179], they reject 10% of pairs.

- Rejection of pairs whose point-to-point distance is larger than some multiple of the standard deviation of distances. In [169], they reject pairs with distances

more than 2.5 times the standard deviation.

- Rejection of pairs that are not consistent with neighboring pairs, assuming surfaces move rigidly [177]. This scheme classifies two correspondences ($p_1$, $q_1$) and ($p_2$, $q_2$) as inconsistent iff

$$|Dist(p_1 - p_2) - Dist(q_1 - q_2)| \tag{2.3}$$

  is greater than some threshold. The algorithm then rejects those correspondences that are inconsistent with most others. Note that the algorithm as originally presented has running time $O(n^2)$ at each iteration of ICP.

- Rejection of pairs containing points on mesh boundaries [168].

The latter strategy, of excluding pairs that include points on mesh boundaries, is especially useful for avoiding erroneous pairings (that cause a systematic bias in the estimated transform) in cases when the overlap between scans is not complete. Since its cost is usually low and in most applications its use has few drawbacks, using this strategy is recommended by [180].

### High-Speed ICP

The ability to have ICP execute in real time (e.g., at video rates) would permit significant new applications in computer vision and graphics. For example, [181] describes an inexpensive structured-light range scanning system capable of returning range images at 60 Hz. If it were possible to align those scans as they are generated, the user could be presented with an up-to-date model in real time, making it easy to

see and fill "holes" in the model. Allowing the user to be involved in the scanning process in this way is a powerful alternative to solving the computationally difficult "next best view" problem [182], at least for small, handheld objects. As described by [173], another application for real-time ICP is model-based tracking of a rigid object.

All of the performance measurements presented so far have been made using a generic ICP implementation that includes all of the variants described. It is, however, possible to make an optimized implementation of the recommended high-speed algorithm, incorporating only the features of the particular variants used. David Simon, in his Ph. D. dissertation [173], demonstrated a system capable of aligning meshes in 100-300 ms. for 256 point pairs. His system used closest-point matching and a point-to-point error metric, and obtained much of its speed from a closest-point cache that reduced the number of necessary k-d tree lookups.

### Global Registration

An important issue to consider when discussing registration algorithms is the concept of global registration verses pair-wise registration. As the name suggests, pair-wise registration techniques involve matching of pairs of images or data sets. However, global registration techniques perform the registration across multiple images or data sets simultaneously. A pair-wise technique can be used to solve a global problem. This can be accomplished by performing the registration of the multiple data sets two at a time. However, this is not necessarily the most appropriate solution as an uneven distribution of inaccuracy can result. Although individual pairs may have high

registration accuracy, the entire system may not be in the most optimal registered state. Thus, a truly global system will evenly distribute registration errors throughout the entire data sets.

Another important issue to consider is the treatment of errors and uncertainty in the point features of 3D images. There are a number of factors that lead to the generation of noise in the 3D point sets. These include the inherent noise that is generated during the imaging process, and also the noise due to the actual feature extraction methods. If the statistical properties of this noise can be modeled, then it is beneficial to incorporate this information to improve the registration accuracy.

Many interesting and useful approaches to the global registration problem have been proposed in the recent past. Chen and Medioni, and Masuda and Yokoya both incrementally register views against a growing global union of view points [166, 183]. Pulli also performs incremental registration against a growing set, but includes a backtracking step when global error becomes unacceptable [179]. Pennec describes a method that alternates between computing an average shape for the set of images and registration of the scans against the mean shape [184]. Bergevin et al. place all views into a global frame of reference, and then repeatedly select a view and register it against all others [185]. Blais and Levine use simulated annealing to simultaneously minimize a cost metric based on the total distance between all matches in all views [174]. Stoddart and Hilton find pairwise correspondences between points in all views, and then iteratively minimize correspondence errors over all views using a descent algorithm [186]. This basic technique is extended using a multiresolution framework, surface normal processing, and boundary point processing by Neugebauer [187]

and Eggert et al. [188]. Williams and Bennamoun suggested a further refinement by including individual covariance weights for each point [189]. Sawhney et al. and Shum and Szeliski perform the global alignment of 2D mosaics by nonlinear minimization of distance between point correspondences [190,191]. Benjemaa and Schmitt proposed a nonlinear solution based on the quaternion representation [192]. Their formulation is a multiview extension of the pairwise solution proposed by Horn [193] using distance between pairwise correspondences as the optimization criterion.

When there are large numbers of views, or when information such as odometry is used in conjunction with point correspondences, the global registration parameters can be solved as the parameters that minimize error with respect to the estimates of the relative motion between view pairs. Lu and Milios solve this problem by linearizing the rotational component [194]. This formulation is useful when the total rotational error is small. They proposed an analytic method for solving the global registration parameters using the relative motion between view pairs as the error criterion [195,196]. This criterion does not require linearization and, therefore, can be used even when the accumulated rotational error is large. Furthermore, this criterion does not require point correspondences and can therefore be used together with robot odometry or any pairwise registration method.

### 2.2.4 Hole Filling

Hole filling is a common problem in geometric modeling. Many methods have been developed (e.g. [197–199]). They deal with high-quality laser-scanned models with relatively small missing part. The problem we try to address is significantly more

difficult. My inputs are models acquired by vision methods. They are dynamic and guaranteed to be at most 50% complete (only one disparity map for each instance $t$).

## 2.3 2D-3D Video Conversion

Stereoscopic media augments traditional video technologies with 3D perception, thus resulting in a more lifelike viewer experience. However, the adoption of such media is hindered by two main factors: the cumbersome nature of 3D (stereoscopic) displays, and the extra effort required to produce 3D content.

The production of filmed (as opposed to CG) 3D content requires either the capture of stereoscopic or multiple camera content, or the conversion of 2D to 3D content in post-production. The former has several disadvantages including specialized equipment and production pipeline. Conversion technologies, on the other hand, can be employed to any existing conventional content, inducing the usage of old material. However, despite the conversion advantages, most 3D content today is created by specialized cameras, and not by conversion technologies. A notable exception is the June 2006 release of the movie "Superman Returns", which included 20 minutes of 3D images converted from the 2D original digital footage. It was recently declared that a company called "In-Three" may convert all six "Star Wars" movies to 3D, in a process that seems to be mostly-manual. Other players in the market include DDD and Philips.

2D to 3D video conversion can boil down to the problem of inferring the relative or absolute depth for the underlying scene. Some recent work [200–202] proposed to use machine learning approaches to recover the 3D structure from a single image.

Although they have achieved some excellent results given this very difficult problem, the outcome depends largely on the training data set. Given the complexity and varieties of everyday videos, it is not clear if this type of data-driven approach can be made tractable for videos.

Automatic stereoscopic extraction can be achieved by generating dense depth map for every frame in a monocular video of static scenes [13]. Structure from motion algorithm is employed to compute the 3D positions of the tracked feature points as well as the camera poses. With the calculated camera poses, multiple view stereo is applied on each frame to produce the dense depth maps [203]. [14] and [15] can deal with dynamic scenes by segmenting rigid objects into layers and SFM [204] is applied on each layer respectively to reconstruct 3D structure. [205], [206] and [207] avoid the generation of dense depth map by synthesizing the other view from the other frames in the input video. Their methods involve less computational resources. However, they are designed to handle static scenes, and certain assumption has to be made on the camera paths so there would be an image in the sequence that can be borrowed for stereo-pair view synthesis. Finally it should be noted that SFM [204] can only deal with certain camera motions, such as dolly and crane shots [208]. For camera movement that does not change the center of projection, such as these commonly used pan/tilt/zoom shots [208], SFM [204] will fail.

In general, full automatic methods do not produce satisfactory results in complex scenes exhibiting both complex camera movement and non-rigid object motion. Some semi-automatic approaches resort to the assistance of user interaction. [209] developed a software (DeepSee Studio) that enables the lasso operation on an image to segment

objects from the background. [210] requires the users to specify depth to some sparse points in certain key frames, followed by the spatial and temporal depth propagation. The propagation is implemented by classification, where the classifier is trained by the user input information. The most recent work [2] is similar to [210]. They also ask the users to assign some absolute depth value, which is color encoded, to a few key frames by scribbling. The user scribbling is used to train a classifier that classifies the rest pixels in the image sequence. Only those pixels with high confidence are assigned a depth value by the classifier. The depth is propagated to the entire video by a similar approach of colorization [211] that encourages same depth values for neighboring pixels, except for those edges separated ones. Instead of using a linear least square formulation as in [2, 211], my depth propagation formulation is based on quadratic programming since it allows both equality and non-equality assignment [212] (expressing the fact that one layer is in front of the other).

All these user-scribble based approaches rely solely on the user to provide vital depth information. The depth cues that can be automatically estimated from motion are ignored. In my system, I combine motion analysis with user interactions so that we can get the best of both worlds: motion analysis can provide depth constraints automatically, reducing the amount of user input required while user interaction can guarantee the final quality of the synthesized view.

Since I employed optical flow estimation and foreground extraction in the motion analysis preprocessing. I will review the previous work on these two topics.

### 2.3.1 Optical Flow Estimation

Estimation of optical flow and its derivatives is an important task in the area of computer vision. [213] studied the role of differential invariants of optical flow with respect to 3D-interpretation of image sequences. Specific 3D-tasks like obstacle detection ( [214]) and computation of bounds for time to collision ( [215, 216]) may be solved based only on $0^t h$ and $1^s t$ order properties of optical flow. Furthermore, first-order properties [217, 218], can be used as features for the classification of image patches into regions corresponding to independently moving objects.

[219] proposed an approach to estimate spatiotemporal derivatives of the optical flow, whereas [220] limited their approach to compute only spatial ones. All of them use at least second order derivatives of the gray-value pattern in order to capture the variation of optical flow in the neighborhood of the point under consideration. These differential approaches are to be distinguished from 'neighborhood-sampling' approaches which use the actual values of the gray-value gradient at every point of the observed neighborhood like [221, 222]. Regarding the above mentioned approaches to estimate optical flow and its derivatives, [223] built a common framework to derive all local differential methods based on the brightness change constraint and to present a method which combines differential with neighborhood-sampling techniques. Furthermore, within this framework [223] show that if one refers strictly to the assumptions of [220], it will turn out that their approach is equal to the optical flow estimation technique presented by [224].

Optical flow is defined as the apparent velocity of gray-value structures. Assuming

temporal constancy of a moving gray-value structure $g(x, y, t)$ results in the well known Optical Flow Constraint Equation (OFCE) postulated by [225]:

$$\frac{d}{dt}g(x, y, t) = \nabla g^T u = g_x u_1 + g_y u_2 + g_t = 0 \tag{2.4}$$

with $u = (u_1, u_2, 1)^T$. This equation does only allow to estimate a linear combination of the components $u_1$ and $u_2$ of the optical flow. It has to be supplemented, therefore, by additional assumptions.

[226–228] estimate the gray-value gradient with a set of spatiotemporal filters to obtain two or more constraint equations. Unfortunately, this kind of estimating optical flow must fail the more, the better the estimated partial derivatives approximate the real derivatives, because in this case, the equations tends to become linearly dependent.

[229–231] use a generalized form of the OFCE by assuming intensity changes due to shading or due to changes of the surface orientation with respect to light sources. [232] argue that different biological visual systems do compute different optical flows. In biological visual systems it suffices to comply with the qualitative properties of the motion field as good candidates for subsequent analyzing cells. In this connection it must be allowed to define different "optical flows", since they have to be considered as an approximation of the true displacement rate field.

Most publications presenting an optical flow estimator discuss their results only qualitatively. A remarkably broad comparison has been presented by [233], who implemented various optical flow estimation techniques and tested them quantitatively

on several synthetic and quasi-synthetic (i.e. one real image with simulated camera motion) image sequences. Their comparison with real image sequences as input data has been limited to a qualitative judgement, since the true displacement rate fields of their image sequences are unknown.

### 2.3.2 Foreground Extraction

As digital cameras become ubiquitous to capture our real world, how to distill useful information from the vast image collection becomes a problem that is of not only intellectual curiosity but also urgent utility. Among many aspects of image understanding, the automatic identification and segmentation of a frequently recurring object is a fundamental topic that serves as the foundations for high-level tasks such as data compression, pattern recognition, and visual information retrieval. Recent techniques from data mining have been introduced to automatically detect frequent visual patterns from a large collection of images [234] [235]. In these cases, the authors develop algorithms that take advantage of the repeated occurrences of visual patterns and automatically identify these patterns without any training data set or user interaction.

Interactive image segmentation brings human's prior knowledge of the location, size, colors and/or boundaries to segment a target object from an image, for example via user-provided bounding box [236] [237] and strokes [238] [239]. Cui [240] proposed to first manually label one image and then propagate the segmentation labels to other related images. Liu [241] developed a semantic-level segmentation algorithm to transfer the segmentation labels from a user-annotated image database using SIFT

flow [242]. However, manual labeling is often a labor intensive task. Even simple tasks such as the selection of a bounding box may be daunting when dealing with large sets of images. Therefore fully automatic segmentation remains one of the most active topics in computer vision.

By providing just one additional image, co-segmentation aims to simultaneously segment the similar objects embedded in a pair of images with different backgrounds. Co-segmentation typically utilizes a global appearance model for the foreground object, such as appearance (color and/or filter bank) histograms [243] [244] [245]. It is further extended to incorporate human interaction in [246] and handle more than two images in [247].

Another related line of research is object recognition which focuses on detecting, classifying and even segmenting objects in images. The idea of concurrently recognizing and segmenting objects has been exploited in [248] [249] [250] [251] [252] [253] [254]. Most object recognition algorithms need a relatively large training dataset and require that the object constitutes a dominant portion of the input image, which often implicitly brings the need for pre-segmentation.

Video segmentation is another active research topic in computer vision, which includes but not limited to interactive segmentation [255], background subtraction from a stationary camera [256] and extracting moving objects taken by a camera with free motion [257].

## 2.4 Discussion

After a thorough review of the existing 3D recovery/reconstruction methods, it is not surprising to find that none of them meets the requirements for my goal set in section 1.1. This is the reason I dedicated my Ph.D. work in this topic, in hope of making tomorrow's 3D capture and display as simple as today's photo shooting.

The LFS is a new concept that falls into the category of shape from X. While existing shape from X approaches rely on intensity matching, shading, or focusing information, my approach uses a completely different cue: the light fall-off property. The derived algorithm makes no assumption about the surfaces in the scene and can handle surfaces beyond lambertian reflectance properties and textures. Due to the low computational cost of LFS, it is very easy to perform the depth reconstruction process in real-time, thus the real-time depth sensor based on LFS.

As can be seen, when we talk about building the 3D model from a single camera, structure from motion only deals with static scenes. Although it has been extended to non-rigid Structure from Motion (SFM), it can only handle small deformation or viewpoint changes. Hole filling is also known as a common problem in the geometric modeling community. Typically, they are focused on high-quality static models that are acquired using laser range scanner with relatively small missing parts. The problem I am trying to solve here is significantly more challenging. I allow 3D models acquired by depth sensors as my input, since a laser range scanner can hardly capture dynamic scenes. Compared with those from range scanners, depth sensors contain more noise, and they are only 50% complete at most (one depth map for each instance

t).

The most relevant work is by Pekelny and Gotsman [258]. They assume the deformation is articulated and piecewise rigid, and estimate each rigid transformation component and use them to merge partial surfaces over time using the Iterative Closest Point (ICP) method. My method is intended to deal with both rigid and non-rigid smooth deformations and does not require manual segmentation of different components. To the best of my knowledge, I present the first method to generate a complete deformable model using a single depth camera.

Converting 2D videos into 3D has attracted more and more attention from the research community when the 3D displays get more popular. Although there exists commercial software that automatically turns images into stereo ones, there still exists large space to improve the 3D effect. Understandably, converting a single image or a monocular video into stereo requires knowing the depth information for every pixel in the scene. Recovering 3D range from a single 2D image is an ill-posed problem, a universal approach that requires no user interaction will be extremely difficult, if not impossible, to achieve. Therefore, the most popular and reliable approaches still mainly rely on the user interaction to provide vital depth information. The depth cues that can be automatically estimated from motion are ignored. In my system, I combine motion analysis with user interactions so that we can get the best of both worlds: motion analysis can provide depth constraints automatically, reducing the amount of user input required while user interaction can guarantee the final quality of the synthesized view.

## Chapter 3 Light Fall-off Stereo Depth Camera

In this chapter, I will present my novel depth recovery method, light fall-off stereo and a real-time depth sensor based on this theory. LFS estimates depth from scenes beyond lambertian reflectance and texture. LFS takes a number of images from a stationary camera as the illumination source moves away from the scene. Based on the inverse square law for light intensity, the ratio images are directly related to scene depth from the perspective of the light source. Using this as the invariant, I developed both local and global methods for depth recovery. Compared to previous reconstruction methods for non-lamebrain scenes, LFS needs as few as two images, does not require calibrated camera or light sources, or reference objects in the scene. I demonstrated the effectiveness of LFS with a variety of real-world scenes.

The real-time depth sensor contains two co-axial point light sources (LEDs) synchronized with a video camera. The video camera captures the scene under these two LEDs in complementary states (e.g., one on, one off). Based on the inverse square law for light intensity, the depth can be directly solved using the pixel ratio from two consecutive frames. I demonstrate the effectiveness of my approach with a number of real world scenes. Quantitative evaluation shows that my system compares favorably to other commercial real-time 3D range sensors, particularly in textured areas. I believe my system offers a low-cost high-resolution alternative for depth sensing under controlled lighting.

## 3.1 Method

In this section I first present the radiometric model used in LFS and show how it can be used to provide depth estimate. I discuss the assumptions made and the associated issues in designing a practical range sensor system.

### 3.1.1 Depth Recovery for a Pivot Point

Here we define pivot point as the intersection between the line connecting the two lighting positions and the surface in the scene. We first discuss how to recover depth for the pivot point. Let us recall the image formation process. As shown in Figure 3.1 (left), the scene is illuminated by a single point light source $L$. The irradiance of the pivot point $p$ in the scene is

$$E(p) = W\frac{\cos(\omega_L)}{r_p^2} \tag{3.1}$$

where $W$ is the light radiance, $r_p$ is the distance between point $p$ and the light, and $\omega_L$ is the incident angle. $p$ will reflect light toward the observation camera $C$. The reflected radiance value is defined as

$$L(p) = R(p, \omega_L, \omega_C)E(p), \tag{3.2}$$

where $\omega_C$ is the viewing direction, and $R(p, \omega_L, \omega_C)$ is the spatially varying bidirectional reflectance function (a.k.a., bidirectional texture function–BTF). It takes into consideration of surface albedo variations. Finally, the imaging system measures the irradiance value on the camera's sensor, which is

$$I(p) = \rho L(p) \tag{3.3}$$

where $\rho$ is a constant parameter determined by the imaging optics (details can be found in [259]). For the scope of this thesis, we assume that the camera has a linear response, in other word, the camera is measuring relative irradiance directly. To deal with cameras with non-linear responses, standard radiometric calibration procedures (e.g. [260, 261]) should be applied to correct the pixel values.

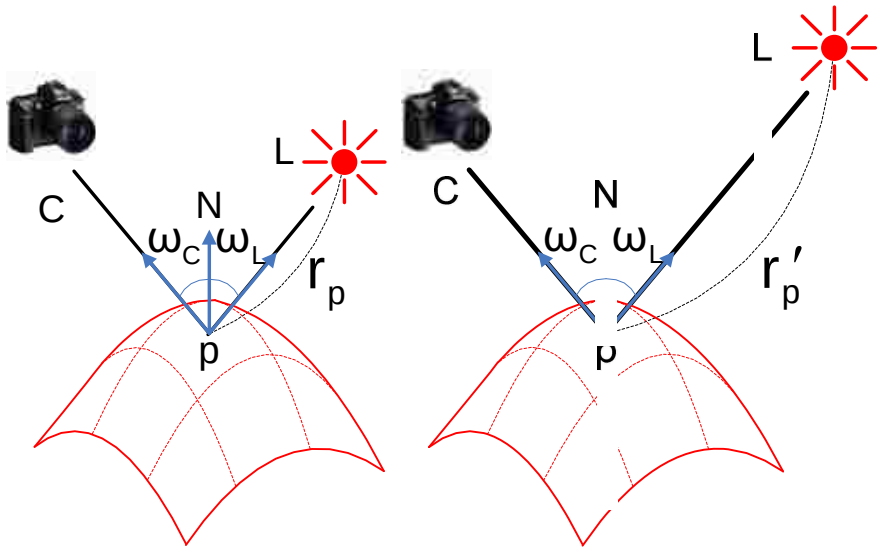


Figure 3.1: The pivot point illuminated by a point light source at the first (left) and the second (right) positions.

Combing equations 3.1, 3.2 and 3.3, we get

$$I(p) = \rho W \frac{\cos(\omega_L)}{r_p^2} R(p, \omega_L, \omega_C). \tag{3.4}$$

This is how equation 1.1 is derived.

As shown in Figure 3.1 (right), With both camera position and light intensity fixed, we move the point light along the direction of $\omega_L$ to a new position. In this process, every parameter in equation 3.4 remains the same except the distance to the

light changes to $r'_p$. The new observed intensity $I'(p)$ measured by the camera is:

$$I'(p) = \rho W \frac{\cos(\omega_L)}{r'^2_p} R(p, \omega_L, \omega_C) \tag{3.5}$$

Computing the ratio of $I'(p)$ and $I(p)$ cancels out all the terms but the distance to the light source, that is

$$\frac{I'(p)}{I(p)} = \frac{r_p^2}{r'^2_p} \tag{3.6}$$

With $\Delta r = r'_p - r_p$ measured using a ruler, we can compute $r_p$ as

$$r_p = \frac{\Delta r}{\sqrt{I(p)/I'(p)} - 1} \tag{3.7}$$

It should be noted that the depth is actually from the perspective of the light source, not from that of the camera.

In the above LFS formulation, the camera position is fixed, both the BTF term and the light intensity term are canceled out. Therefore, the depth measured by this method is invariant to lighting intensity and surface property.

### 3.1.2 Estimate a Depth Map for the Whole Scene

The above process can be extended for recovering depth for other points in the scene as well. As shown in Figure 3.2, the depth of all points in the scene are measured with respect to a depth reference plane, which passes through the first lighting position $S$ and is perpendicular to the lighting direction for the pivot point $p$. For an arbitrary point $q$ in the scene, here we use $I(q)$ to denote the observed intensity of the point under the lighting position $S$. It is worthy noting that $I(p)$ and $I(q)$ can be measured at same time as long as both points $p$ and $q$ are within the camera's field of view.

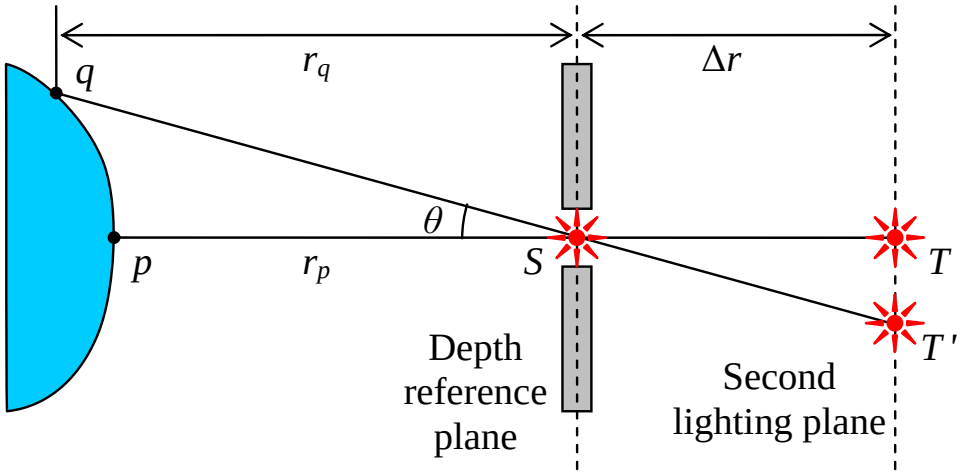


Figure 3.2: The setup for recovering a depth map for the whole scene.

In order to maintain the same incident lighting direction, when capture the intensity of point $q$ under the second lighting position, we need to move the light along the ray $qS$. Here we place the light at location $T'$, the intersection between ray $qS$ and the second lighting plane, and refer the new observed intensity as $I'(q)$. According to the lighting distances shown in the figure, we have:

$$\frac{I'(q)}{I(q)} = \frac{(r_q/\cos\theta)^2}{((r_q + \Delta r)/\cos\theta)^2} \tag{3.8}$$

After simplification, we get:

$$r_q = \frac{\Delta r}{\sqrt{I(q)/I'(q)} - 1} \tag{3.9}$$

Comparison between equation 3.7 and 3.9 suggests that, once the light is positioned at proper locations on the second lighting plane, the same equation can be used to estimate the depth of different points in the scene.

Calibrating the light position for different points in the scene can be a tedious task. Fortunately, we can simplify the problem by placing an optical occluder along the depth reference plane such that light can only go through a small hole at location $S$.

Hence, when the light is positioned at the second lighting plane, only a small portion of the scene is illuminated and the incident lighting direction for this small portion is the same as the case when light is placed at location $S$. Multiple images of the scene can be captured either by moving the light along the second light plane or by building a light array at the second light plane and turning on one light at a time. Through finding the maximum intensity at each pixel location from multiple captured images, we can effectively extract the illuminated portions of the scene and merge them into a single image. The resulting image, referred as Multi-Lighting-Direction-Image, shows the appearance of the scene under different lighting directions for different portions and guarantees that the incident light direction on every surface point remains the same. As a result, it can be used to recover the depth of the entire scene using equation 3.9.

### 3.1.3 Practical Approximation

Although the approach discussed in above subsection gives more accurate depth estimation results in theory, the experiments show that reasonable good estimations can be obtained through a much simplified procedure. By assuming the observed intensity difference of point $q$ under lighting position $T$ and $T'$ is negligible, we simply capture a single image of the scene under lighting position $T$ and use it to approximate the required Multi-Lighting-Direction-Image. In fact, this is similar as assuming that the incident lighting directions for different points in the scene are parallel. When the object size is relatively small compared to the distance to the light, such an approximation is valid and has been used in almost all photometric stereo and shape-

from-shading algorithms.

### 3.2 Error Analysis

I try to analyze the error introduced by the approximation on the Lambertian surface and use the analysis results to guide us setting up the real-time system.

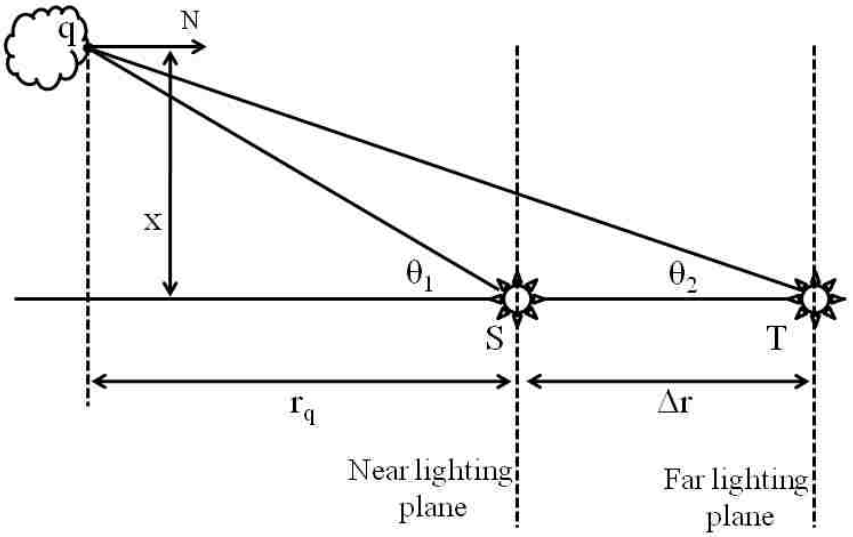


Figure 3.3: Estimation error introduced by incident lighting direction changes for surface point $q$, whose distance to line $ST$ (offcenterness) is $x$ and normal is $N$.

As shown in Figure 3.3, a 3D point $q$ is illuminated by a near light and a far light. Without loss of generality, we assume the normal at $q$ is parallel to line $ST$. Under the two lighting conditions, the observed intensities of $q$ are:

$$\begin{aligned} I_q &= \rho \frac{L(\theta_1)}{(r_q/\cos\theta_1)^2} \cdot R(q, \theta_1, \lambda) \\ I'_q &= \rho \frac{L(\theta_2)}{(r_q + \Delta r)/\cos\theta_2)^2} \cdot R(q, \theta_2, \lambda) \end{aligned} \tag{3.10}$$

Based on the assumption that the variation in incident lighting direction can be ignored, i.e., $\theta_1 \approx \theta_2$, we have $L(\theta_1)R(q,\theta_1,\lambda)\cos^2\theta_1 \approx L(\theta_2)R(q,\theta_2,\lambda)\cos^2\theta_2$. The depth estimated accordingly, $\tilde{r}_q$, can then be calculated using:

$$\tilde{r}_q = \frac{\Delta r}{\sqrt{I_q/I'_q} - 1} \tag{3.11}$$

To evaluate the accuracy of the above estimation, we now model the error caused by incident lighting direction changes using an error term $e$. That is $L(\theta_1)R(q,\theta_1,\lambda)\cos^2\theta_1 = e \cdot L(\theta_2)R(q,\theta_2,\lambda)\cos^2\theta_2$. As a result, the following equation holds:

$$\frac{I_q}{I'_q} = \frac{(r_q + \Delta r)^2}{r_q^2} e \tag{3.12}$$

Substituting Equation 3.12 into Equation 3.11 gives us:

$$\begin{aligned} \tilde{r}_q &= \frac{\Delta r}{\frac{r_q+\Delta r}{r_q}\sqrt{e} - 1} \\ &= \frac{r\Delta r}{(\sqrt{e}-1)r + \sqrt{e}\Delta r} \end{aligned} \tag{3.13}$$

Consequently, the relative range error $E$ of an estimation $\tilde{r}$ can then be expressed as:

$$\begin{aligned} E &= |r - \tilde{r}|\,/r \\ &= \left|1 - \frac{\Delta r}{(\sqrt{e}-1)r + \sqrt{e}\Delta r}\right| \\ &= \left|\frac{(\sqrt{e}-1)(r+\Delta r)}{(\sqrt{e}-1)r + \sqrt{e}\Delta r}\right| \end{aligned} \tag{3.14}$$

In Figure 3.4 we plot $E$ with respect to different $\Delta r$ under a fixed near lighting distance ($r$). Note that the unit of $r$ is not important since we are dealing with relative errors only. From the plot, we can observe that: 1) the overall estimation error ($E$) is zero when there is no incident direction error ($e = 1$), but increases quickly when $e$ departs from 1; and 2) under the same amount of incident direction error, the smaller the lighting distance change ($\Delta r$), the larger the estimation error.

It is noteworthy that the second observation does not contradict the fact that, with a small $\Delta r$, the change in the incident direction is also small. Instead it suggests that, when $\Delta r$ is smaller, the estimation result is more sensitive to incident direction changes. We conclude that, this is because, with a small difference in the observed scene radiance, the denominator in Equation 3.11 is close to zero, so any perturbation in the incident direction can be "magnified".

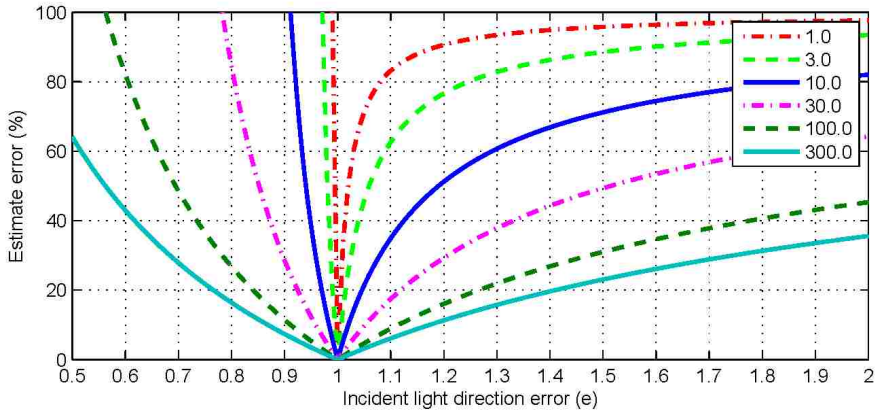


Figure 3.4: Estimation error under different $e$ and $\Delta r$ settings. $r$ is set to 100. In general the larger $\Delta r$, the smaller estimation error $E$ is.

The above analysis over a generic surface model tells us that the depth estimation is error-prone when $\Delta r$ is small, but it cannot help us to select the optimal setting since the incident direction error ($e$) is also a function of $\Delta r$. The relation between

$e$ and $\Delta r$ depends on the following factors: the light radiance function ($L(\theta)$), the surface BTF ($R(q, \theta, \lambda)$), as well as the offcenterness ($x$) and the normal ($N$) of the surface point $q$. It is therefore very difficult, if not impossible, to do a mathematical analysis for generic surface models. To address this problem, I perform the analysis through simulation.

A simulator is designed for estimating the depth of a 3D point under the setting shown in Figure 3.3. For simplicity, the light radiance function is assumed to be uniform over different directions and the surface BTF is assumed to be Lambertian. The surface normal is assumed to be uniformly distributed and, for each setting, the estimation is repeated 10000 times with a random forward facing surface normal assigned to point $q$ each time. The mean relative estimation errors under different $\Delta r$ and $x$ values are calculated and plotted in Figure 3.5.

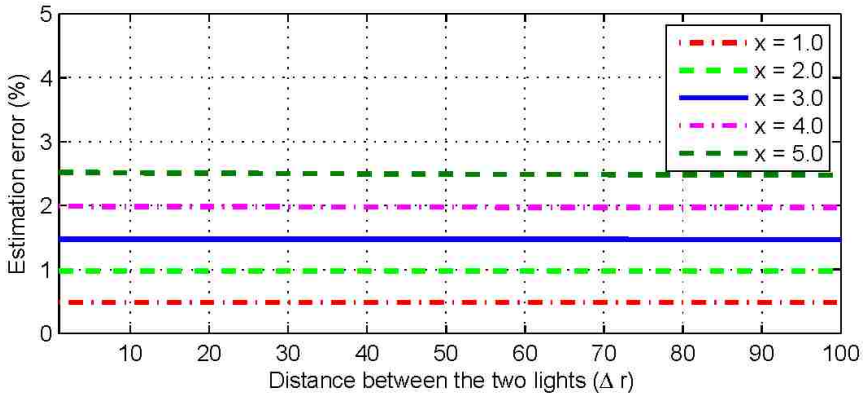


Figure 3.5: Estimation errors under different $\Delta r$ and $x$ settings. $r$ is fixed at 100.

Figure 3.5 suggests that, under the Lambertian model and when $r$ is large enough, adjusting $\Delta r$ has little effect on the estimation error. This suggests that, while increasing $\Delta r$ increases the incident lighting direction difference between the two

lighting conditions, it also makes the estimation less sensitive to lighting direction changes (as shown in previous mathematical analysis). In the end, the two effects cancel out.

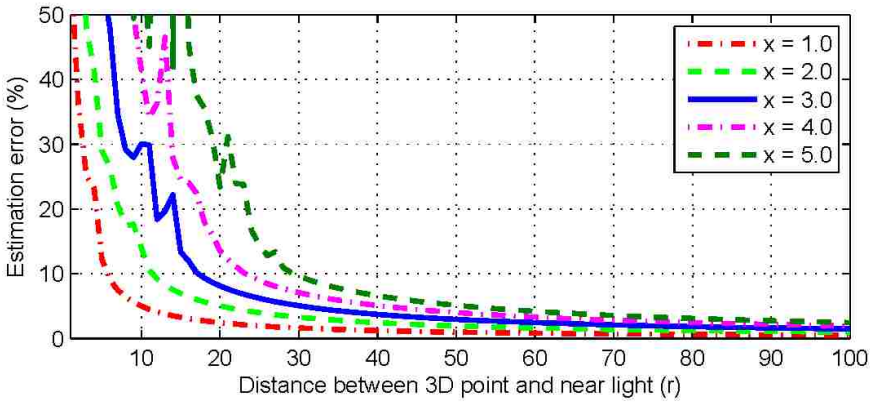


Figure 3.6: Estimation errors under differen $r$ and $x$ settings. $\Delta r$ is fixed at 100.

In the second simulation, $\Delta r$ and the estimation error are fixed as a function of $r$ and $x$ (shown in Figure 3.6). This time, the estimation error decreases quickly as $r$ increases. The result shows that, when $r > 10 \times x$, the estimation error is less than 5%. As a result, the setup of the LFS system needs to ensure the lighting distance is larger than 10 times of the object size.

Finally, the third simulation studies the effect of surface normal on estimation error. Here the angle between surface normal $N$ and $Z$-axis is referred as $\alpha$. In the simulation, both $\Delta r$ and $r$ are fixed and the estimation errors under different $\alpha$ values are plotted in Figure 3.7. As expected, the error is small when the surface is facing the light and is larger otherwise. This suggests that the LFS algorithm performs well for front-facing surfaces ($-60 < \alpha < 60$), but not as good for surfaces facing the sides. A possible solution to this problem is to use additional pairs of lights to illuminate

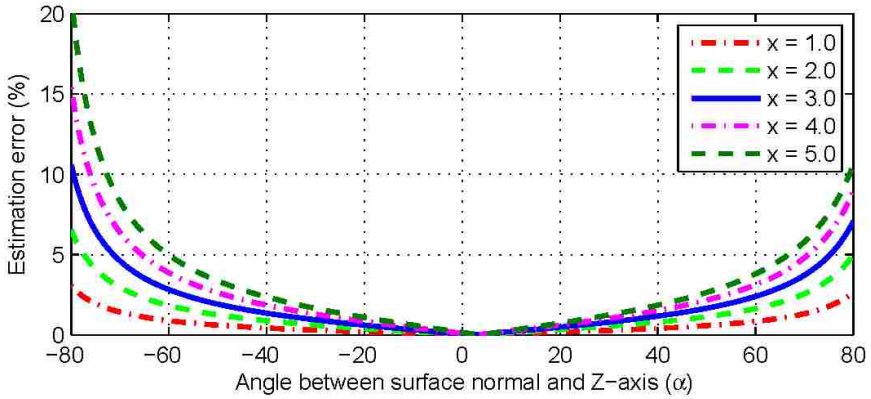


Figure 3.7: Estimation errors under different $\alpha$ and $x$ settings. Both $r$ and $\Delta r$ are set to 100.

these surfaces.

## 3.3 Global Method

So far I have shown that a depth map can be generated using two images of the scene illuminated by a point light source positioned at two different locations. The derived local LFS method is both efficient and simple to implement. However since the image acquisition process can be corrupted with noise, the depth maps obtained using local approach are sometimes noisy. In this section I re-formulate the LFS problem under an energy minimization framework. The derived global optimization-based method can produce smooth and accurate depth maps since: 1) it can make use of images captured under more than two lighting configurations; and 2) it enforces spatial consistency among adjacent pixels in the depth map.

### 3.3.1 Formulation under Energy Minimization

As illustrated in Figure 3.8, the scene is illuminated by a point light source located at different positions. A sequence of images of the scene, referred as $I$, $I^1$, $\ldots I^N$, are thereby acquired. Please note that the point light source is again carefully positioned so that, for a pivot point $p$ in the scene, the incident light direction does not change. I also assume that the lighting direction changes for the remaining points on the scene are negligible.

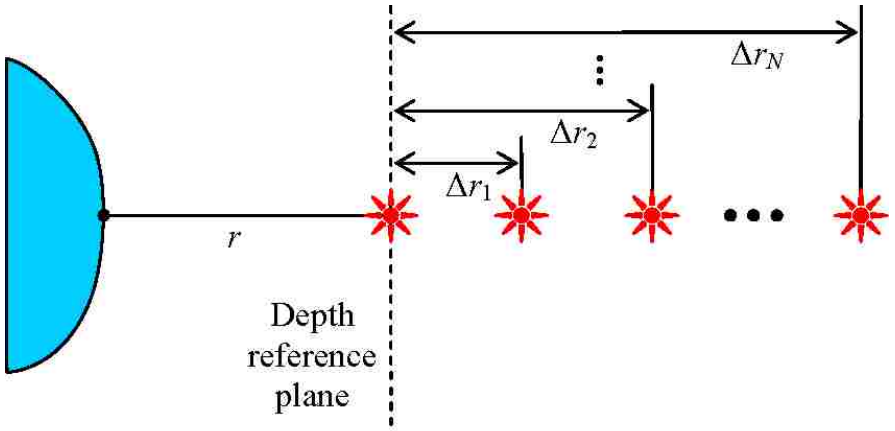


Figure 3.8: Images are captured under multiple lighting conditions. The light movement is carefully controlled to minimize the change of incident directions on the scene surface.

Now if we set the depth reference plane at the first light position, according to equation 3.6, the following holds regardless of surface BTF and light intensity.

$$\sqrt{I_{x,y}} r_{x,y} = \sqrt{I^1_{x,y}}(r_{x,y} + \Delta r_1) = \ldots = \sqrt{I^N_{x,y}}(r_{x,y} + \Delta r_N) \tag{3.15}$$

where $I_{x,y}$ is the intensity of pixel $(x, y)$ in the captured image $I$. $r_{x,y}$ is value of pixel $(x, y)$ in the estimated depth map, i.e., the distance between the depth reference plane and the corresponding 3D point of pixel $(x, y)$.

Due to the noise in the image acquisition process, when more than two images are used, for a given pixel $(x, y)$, we may not be able to find a value $r_{x,y}$ that satisfies the above equation. Our objective is therefore to find a depth value that minimizes the variance among different terms. Here we use $K_{0,\ldots,N}$ to represent the value of the above terms:

$$\begin{aligned} K_0 &= \sqrt{I_{x,y}} r_{x,y} \\ K_i &= \sqrt{I^i_{x,y}}(r_{x,y} + \Delta r_i), \quad 1 \leq i \leq N \end{aligned} \tag{3.16}$$

The energy minimization objective function can then be defined accordingly using the following equation:

$$E = (1 - \lambda) \sum_{x,y} \sum_{i=0}^{N} (K_i - \overline{K})^2 + \lambda \sum_{x,y} (u^2_{x,y} + v^2_{x,y}) \tag{3.17}$$

where $u$, $v$, defined in equation 3.18, are the symmetric second finite differences of the variables $r_{x,y}$ and $\overline{K}$ is the mean of all K's. The second term is used to enforce the smoothness of the solution. $\lambda$ $(0 \leq \lambda \leq 1)$ represents a user defined parameter that adjusts the relative importance between the error term and the smoothness term.

$$\begin{aligned} u_{x,y} &= r_{x+1,y} - 2r_{x,y} + r_{x-1,y} \\ v_{x,y} &= r_{x,y+1} - 2r_{x,y} + r_{x,y-1} \end{aligned} \tag{3.18}$$

### 3.3.2 Optimization Approach

Our goal is to find a depth map that satisfies the inverse-square law at all pixel locations and in all captured images. When there is no noise, it is equivalent to finding a surface that incurs zero measurement error. However in the real world

where sensor measurement is almost always corrupted with noise, we approximate the objective surface by finding a smooth surface that minimizes our cost function defined in equation 3.17.

The global LFS method implements the standard Conjugate Gradient (CG) algorithm together with the line search algorithm $DBRENT$ as an iterative minimization tool [262]. Given the objective function expressed in equation 3.17, this minimization process simply requires the construction of two functions: one that computes the objective cost value $E$ and the other calculates the gradient of $E$ with respect to the vector $\vec{r}$. We also impose boundary conditions by defining the gradient to be zero at boundary pixels. Although we cannot guarantee that the solution converges to the exact correct surface, experiment shows that the recovered depth map is indeed smooth and incurs small measurement error.

## 3.4 Prototype System Setup

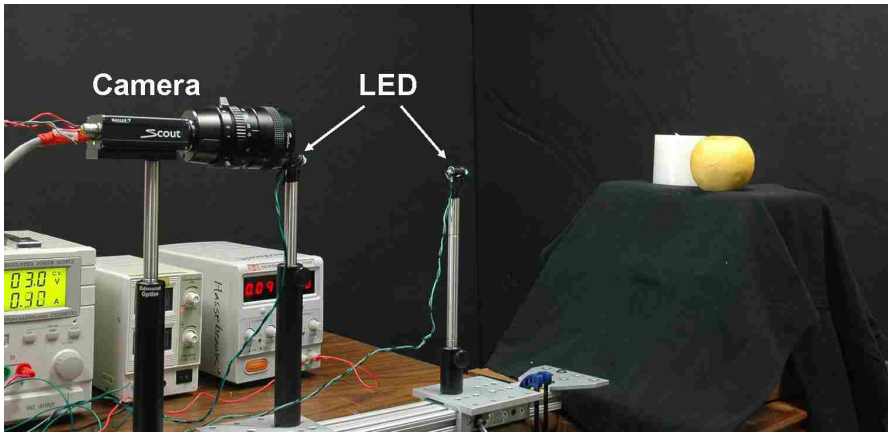


Figure 3.9: experimental setup.

I now describe the prototype system implementation. It consists of two 3W LED

light sources and a video camera on a linear translation stage. The LEDs and video cameras are co-axial. The LEDs occlude a small part of the scene in the camera image, for which I mask out. The camera can capture $640 \times 480$ gray-scale images at 60Hz with progressive scan. Although the camera can produce pixels at 12 bits per pixel, I have found that the 8-bit image has a much higher signal to noise ratio. Therefore I always operate with 8-bit images. The camera responds linearly to light intensity.

The two LEDs are synchronized with the camera's shutter. The camera generates a TTL signal when it opens its shutter. With each shutter pulse, the LEDs toggle their on/off state. LEDs can be switched on or off in the order of 100 nanoseconds. The LEDs heat up when powered. During the "start-up" process, the device's temperature rapidly increases, and the LED's forward current decreases until it reaches a steady state, at which point I start the capture. I have verified that the light output is very stable once the LED reaches its steady state.

Although an LED is an excellent point light source, its spatial distribution of radiance is not uniform. As a result, I must radiometrically calibrate the two LEDs. The procedure is straightforward. Before running the system, I turn on the two lights in turn to illuminate a piece of white paper covering the entire field of view of the camera. The calibration object is captured by the camera under the two lighting conditions. From the two images $I^n$ and $I^f$, we measure the ratio ($Q$) of the corresponding pixel values, that is

$$Q(x, y) = L_n(\theta)/L_f(\theta) = (d_n/d_f)^2 * I^n(x, y)/I^f(x, y), \tag{3.19}$$

where $x, y$ are the pixel coordinates,$\theta$ is the incident angle at $(x, y)$, $d_n, d_f$ are distance between calibration object and the near and far light respectively. Referring to equation 3.4, under my assumption that the incident lighting direction change is small enough, intensity variation that cannot be explained by the inverse square law is attributed to the light radiance distribution.

**Run-time algorithm** The run-time system consists of two parallel threads, one is for image capture and the other is for depth computation and display.

The capture thread waits for camera images transferred via the IEEE1394 bus and alternatively stores them into the near and the far image buffers.

In the depth computation thread, image ($I^f$) from far light is first corrected by calibration ratio $Q$,

$$I^c(x, y) = I^f(x, y) * Q(x, y), \tag{3.20}$$

where $I^c$ is corrected image. Then we plug $I_{x,y} = I^n(x, y)$ and $I'_{x,y} = I^c(x, y)$ into equation 3.7 to compute the depth of pixel $(x, y)$.

Before the computation, I exclude those pixels that will potentially give bad results. Those pixels include saturated ones in either the near image or the far image. Saturated pixels (especially highlight areas) are usually not real measurements of light intensity; they are likely to result in inaccurate depth estimates. The pixels with intensities below a certain threshold are also excluded, because these pixels are either background or reside in shadow areas. Furthermore, low intensity values are more sensitive to noise. For those bad pixels, I simply set them to black in the depth map. This is why black holes are occasionally present in the depth map, likely the

results of surface highlights and shadows. Finally, the depth map is smoothed by a mean filter.

Since graphics cards are excellent for parallel image processing, the entire depth computation pipeline is implemented on the graphics processing unit (GPU). In this case, the captured images are directly transferred to two textures on the graphics board. To further improve speed, the mean filter is implemented by a two-pass process. One pass is averaging on a horizontal line, and the other is averaging on a vertical line.

**System Issues** One problem introduced by using two sources is that they do not have the same power spectrum. With a different power spectrum in the illumination source, the reflected radiance of objects with a wave-length dependent BTF, after converted to gray-scale by the camera, cannot be canceled out even under identical incident lighting direction with the proper radiometric correction term ($Q$). To solve this problem, I add a narrow band (green) filter in front of the camera lens, which limits the camera to measure only a single wavelength light. This is preferred over the alternative solution to use color LEDs at a single wavelength (instead of white ones), since the power spectrum varies from one LED to another.

A better optic design could use some beam-splitters, with which the light/camera occlusion problem can be avoided. I do not use this design in my current prototype because the light emitted from the LED is not bright enough when transmitted through multiple beam-splitters. It is not easy to achieve a fairly uniform radiance distribution with multiple LEDs, while keeping them as a point light source. I also tried to use two commodity projectors, but they have a significant amount of leaked

light that biases the depth estimation.

### 3.5 Experiments and Results

I show some of the results here. Figure 3.10 and 3.11 show the depth map and the 3D rendering result for two Lambertian objects. Most of the fine details are recovered. Figure 3.12 and Figure 3.13 show the depth maps of two scenes. Since my method does not deal with areas in shadow, these areas are masked out in the resulting depth map. Figure 3.14 illustrates the global approach presented in section 4. The scene consists of two very specular objects: a glossy book cover and a piece of silk. Note that this kind of specular surfaces is very sensitive to the light's incident angle. As shown in the lower row, my global method demonstrates significantly improvement for this challenging scene.

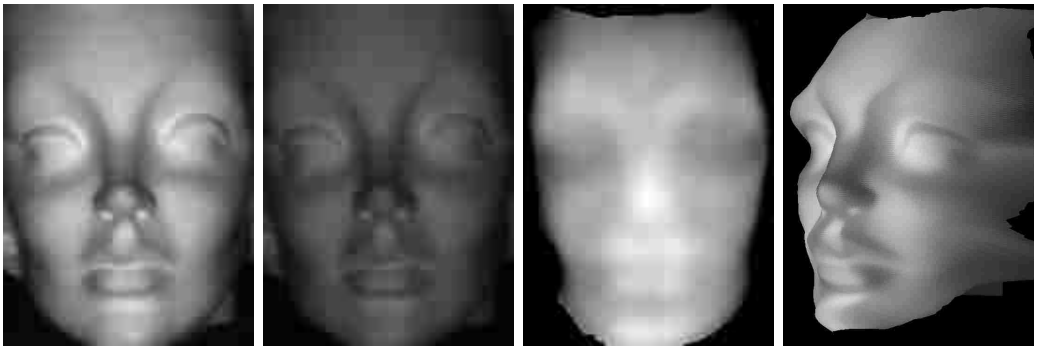

Figure 3.10: Left 2 images: images taken with light at two positions. $3^{rd}$ image is the depth map, and $4^{th}$ image is a view of the recovered 3D model.

**Quantitative evaluation** I also evaluate the quality performance of my system by comparing it with two other commercially available live range sensors. The first one is Canesta range sensor which is able to generate low resolution $(64 \times 64)$ range maps at

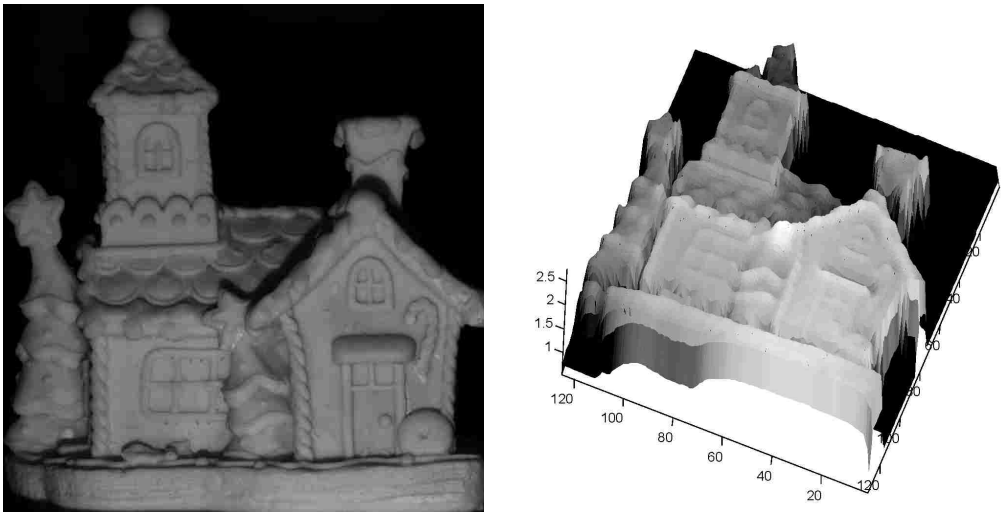


Figure 3.11: A toy house with very fine details. I show its depth in 3D view.

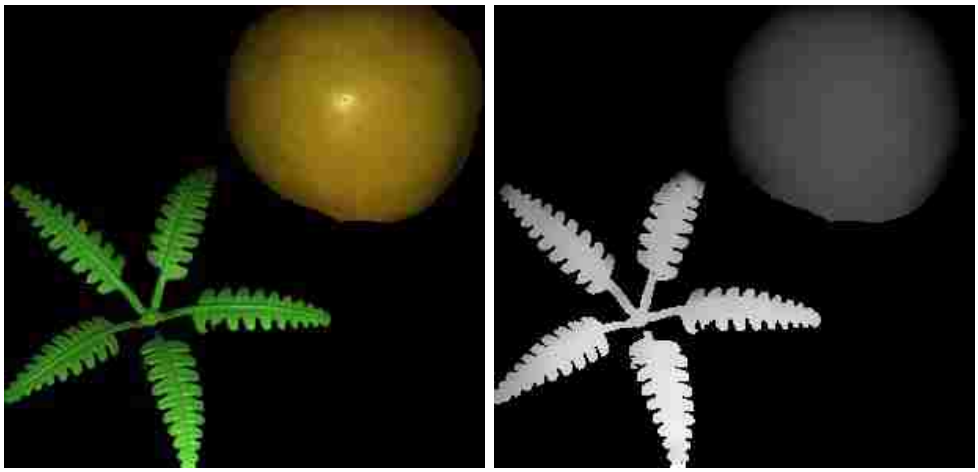

Figure 3.12: left image is a simple scene with plastic leaves and an apple in it. Right image is its depth map. Both the leaves and the apple are non-lambertian.

video frame rate. The other is the Z-mini from 3DV Systems, Ltd. that can provide high resolution (maximum $640 \times 480$) live range maps.

As shown in figure 3.15 (first row) the target object in the first experiment is a piece of white paper glued onto a planar surface. This paper can be regarded as a perfect Lambertian reflector with constant surface albedo. In the second experiment the white paper is replaced by a piece of paper containing rich textures. The object

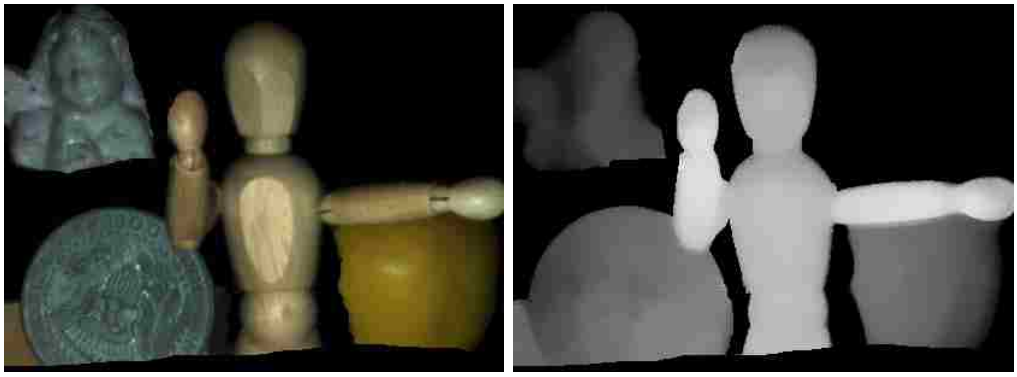

Figure 3.13: A more complex scene. There are wood, metal,plastic in this scene. Since I don't deal with shadow areas, I ignore the pixels below a certain threshold.

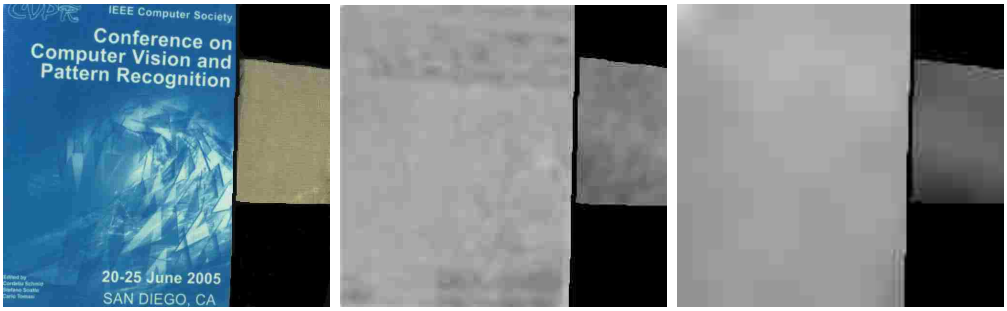


Figure 3.14: Comparison between local method and global method. Left image: there are two objects in the scene at different depth. The surface of these two objects are very specular making the result quite sensitive to incident angle change. Middle image: depth map obtained by per-pixel calculation from two shaded images only. Right image: depth map processed by global method from six shaded images. (The parameter $\lambda$ is set to 0.15 in the experiment)

is carefully placed so that the principle axis of these sensors are perpendicular to the plane and go though the plane's geometric center.

One thing worth noting is that these three sensors have different fields of view, resolutions and their recovered range maps are not within the same coordinate system. These limitations make a metric comparison with ground truth very difficult. To warrant a fair evaluation, I first normalize their output depth values to a uniform space. It is done by calculating a scale factor so that the sample mean of the depth values is normalized to 0.5. After this step I apply a plane fitting algorithm to each

sample data and compute their mean square deviation. Clearly smaller variance implies better reconstruction quality.

The recovered 3D shapes and error rates of these sensors are presented in Figure 3.15 and Table 3.1 respectively. The raw range map is processed with a $5 \times 5$ smoothing filter to reduce high frequency noise resulting primarily from the CCD camera. In general, when the sample target is textureless, all three sensors yield satisfactory results (the Canesta senor returns a single-colored depth map). However, when the target contains non-uniform surface albedo, my system outperforms the other two. It is not surprising given the fact that my depth values are recovered from the ratio of two images, effectively cancel out the surface albedo's influence. Conversely, the reconstruction model adopted by SLP sensors suffers from bias as a function of object intensity [263].

| | Canesta | Z-mini | LFS |
|---|---|---|---|
| newspaper | 0.0741 | 0.0260 | 0.0101 |
| white paper | 0 | 0.0166 | 0.0021 |

Table 3.1: Mean square deviation of depth recovered by different range sensors.

I also reconstructed depth maps of the white paper under different orientations. The error is listed in Table 3.2, in which degree 0 means frontal parallel (the ideal case). As the paper turns away from the camera, the error grows larger, which concurs with the prediction in Figure 3.7. Nevertheless, it is still superior than the Z-mini sensor which has the same resolution.

**Live system** Figure 3.16 shows some live images from my system. Note that since my algorithm does not handle shadow areas, those regions are automatically detected

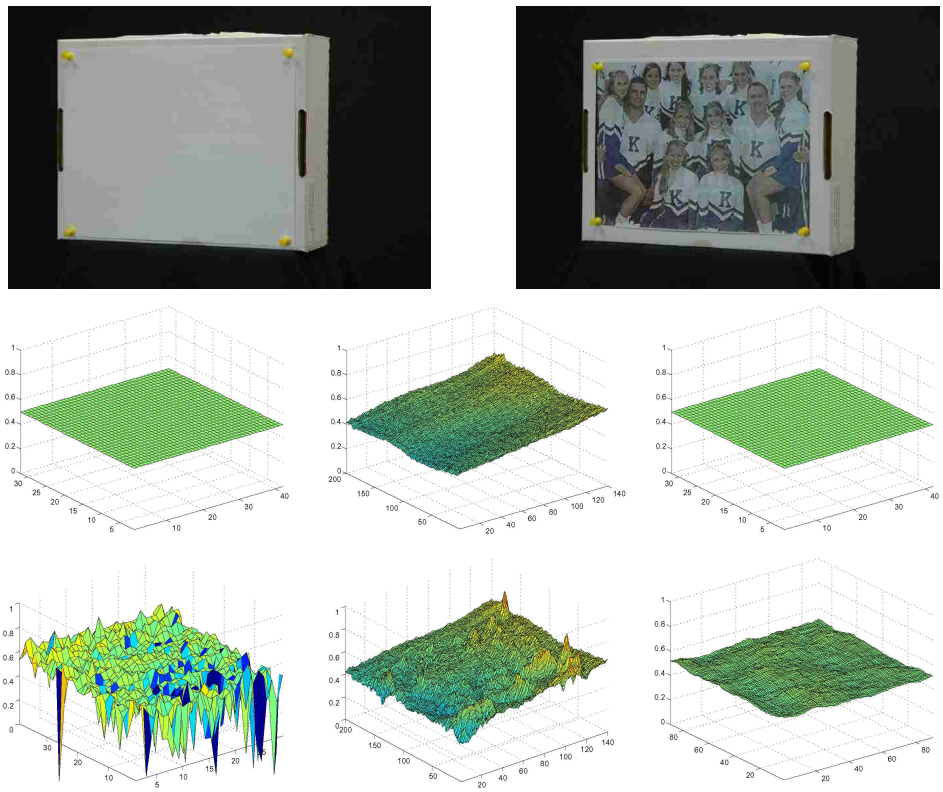

Figure 3.15: Depth recovered by different sensors. (first row) two sample scenes; (second row) 3D plots of the recovered white paper, from left to right, showing results from Canesta, Z-mini and LFS; (third row) 3D plots of the recovered news paper. The mean depth is normalized to 0.5.

| 0° | 10° | 20° | 30° | 40° | 50° |
|---|---|---|---|---|---|
| 0.0021 | 0.0022 | 0.0024 | 0.0028 | 0.0034 | 0.0039 |

Table 3.2: Mean square deviation becomes larger when angle between incident light and surface normal increases.

and masked out during the depth map computation process. The first scene (top two images) contains objects with different shapes and reflectance properties. The resulted depth map is fairly accurate despite the slightly non-Lambertian surface reflectance. In the second scene (bottom two images), the depths map of two, more challenging, metal objects are computed.

My camera captures at 60fps and the off-line depth computation can achieve

60fps on a Geforce 8800 graphics card from NVIDIA. But given that two images are required to generate one depth map, my system's overall speed performance is 30fps.

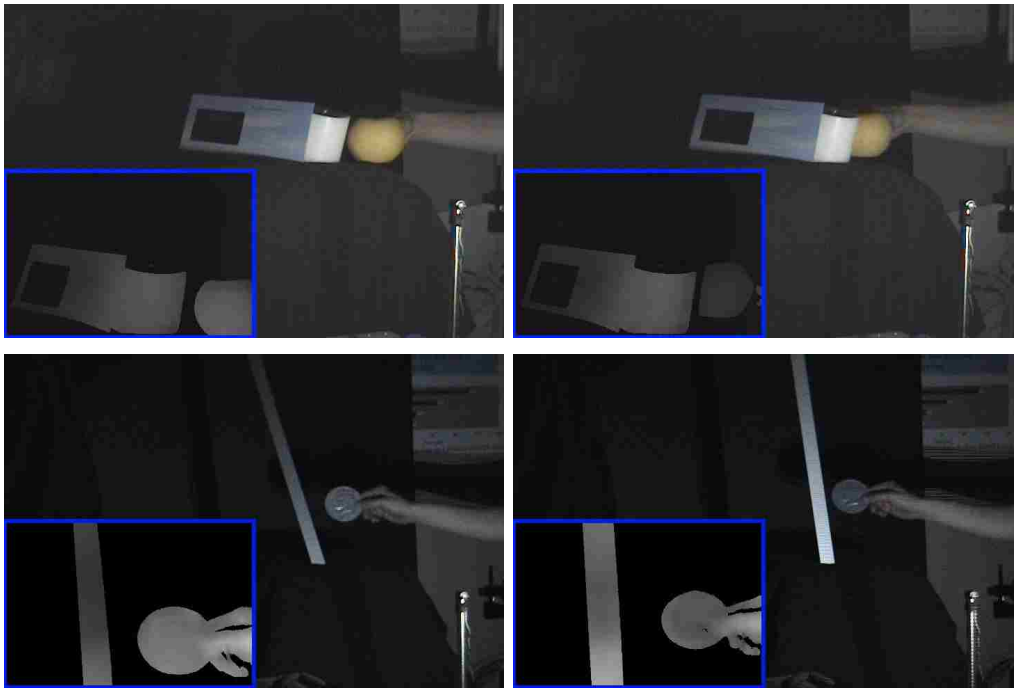

Figure 3.16: Some snapshots of my real-time system results. The insets show the depth maps.

**Discussions** Based on the analysis and experimental results, I believe my system generates superior results than existing 3D commercial sensors based on shuttered light-pulse. Common to these approaches is that they do not require matching and can measure a 2D depth map at once. However, my system has a smaller field of view than the other sensors because it has to maintain the approximately incident lighting direction invariance.

One may notice that the scenes in the live experiments show a lack of high contrast texture. Small motion between two consecutive frames will introduce large error to the image ratio computation, especially on pixels around texture boundaries. The

motion problem is common to many real-time sensors, both the Z-mini and Canesta sensor take at least two measurements of the scene. It can be solved by better engineering, e.g., high-speed cameras. I believe this is also an interesting venue for future research. I can probably compute the optical flow field between even (or odd) frames and interpolate the middle images.

## 3.6 Conclusion

In this chapter I presented a novel system that can generate real-time depth maps. My system, based on the theory of LFS, takes two images under different lighting conditions to estimate the range for each pixel, without the need for matching. Compared to commercial 3D range sensors, it is more robust to textured areas (when the object remains static or the object motion between two frames is smaller than one pixel). My prototype is made from off-the-shelf, low cost components with a simple computation model. It can be used in low-cost embedded systems. I believe my system provides a viable alternative for 3D range sensing under controlled lighting.

## Chapter 4 Modeling Dynamic and Deformable Object

Given the LFS real-time depth camera as well as other commercial depth cameras, I can turn my focus from depth recovery to 3D model building and completion. To be clear, Although I use a SwissRanger 3000 depth camera here, I do not have to stick to a certain kind of depth sensor, as long as it outputs synchronized color and depth image sequences.

In this chapter, I present a novel approach to reconstruct complete 3D deformable models over time by a single depth camera, provided that most parts of the models are observed by the camera at least once. The core of this algorithm is based on the assumption that the deformation is continuous and predictable in a short temporal interval. While the camera can only capture part of a whole surface at any time instance, partial surfaces from different time instants are assembled together to form a complete 3D surface for each time instance, even when the shape is under severe deformation. Since Iterative Closest Point (ICP) algorithm does not work for deformable surfaces, I employ a two-step algorithm to align the partial surfaces. In the first step, the dominating rigid transformations between neighboring frames are estimated and accumulated to approximate the transformation between arbitrary pairs of frames. All the captured partial surfaces are therefore transformed into the coordinate of the destination frame. In the second step, a global mesh warping algorithm based on linear mesh deformation is used to stitch all the piecewise surfaces together. The alignment in both steps are guided by vertex correspondences derived from the

feature tracking on the corresponding images. A volumetric method is then used to combine partial surfaces, fix missing holes, and smooth alignment errors. The experiment shows that this approach is able to reconstruct visually plausible 3D surface deformation results with a single depth camera.

This work focuses on how to assemble surface patches captured at different time instants for the same dynamic object into a complete 4D space-time model. I assume that the individual surface patches have already been acquired using existing vision techniques [264, 265] or any commercial video-rate depth cameras.

Figure 5.4 shows the major steps of my algorithm. The inputs to my system are multiple color-depth image pairs captured at different time instants. In the initialization step, correspondences among salient features extracted from different frames are established by any tracking algorithms, typically SIFT in the experiment. In addition, the depth maps are triangulated into 3D meshes. The second step estimates a rigid transformation for each frame to map surface patches from all frames into a reference global coordinate, where they roughly align with each other. A linear mesh deformation method is then applied in the next step to warp one mesh to another, while preserving local details, which is the core of my algorithm. I break down the presentation into several subsections, first introducing the basics for pairwise warping, then extending it to a global alignment scheme. The issue of severe occlusion is also discussed. Finally in the smoothing and refinement step, meshes are merged into a single dynamic 3D model by volumetric methods, with temporal coherence among models from different frames enforced.

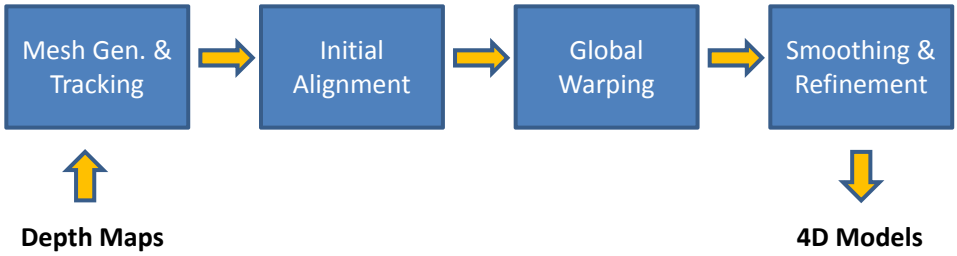


Figure 4.1: The flow chart of my overall algorithm.

## 4.1 Matching Outlier Removal

Since both the initial alignment and global warping are guided by feature correspondences, the surface alignment tends to collapse if the incorrect correspondences are not excluded from the computation. For example in figure 4.2, if the incorrect matching is used as anchor point in the mesh deformation step, the corresponding surface point will be pulled towards a false position, causing large distortion. Therefore, those feature mis-matchings that could cause large distortion should be detected and removed in the first place.

My matching outlier detection algorithm is based on the assumption that the feature points in one frame should be close to their matching points in another frame after a dominating rigid transformation. This assumption is actually consistent with the one I made for the whole thesis, that is, the deformation is continuous and predictable in a short temporal interval.

Thus, the basic idea of outlier detection is to transform the feature points with the dominating rigid transformation, and classify them by the distance to their matching points. The mean and variance are computed for the resulting distance values and those points that have extremely large deviation will be taken as outliers. Note that

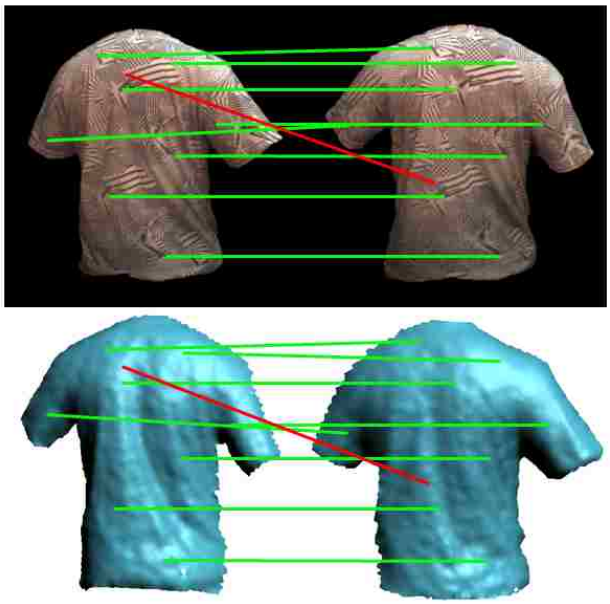

Figure 4.2: Illustration of feature mismatching. Green lines show the correct featuring matching and red line is the mismatching. The feature matchings are mapped to vertex matchings on the partial surfaces.

the mean and variance may suffer from the masking effect [266] if we include the distance of the outliers in the computation. In other words, the outlier may skew the mean and variance toward it, making its deviation small. To avoid the masking effect, I adopt the modified Z-score suggested by [267] to compute the mean and variance values and measure the deviation.

$$M_i = \frac{0.6745|x_i - \widetilde{x}|}{MAD} \tag{4.1}$$

$\widetilde{x}$ is the median value and MAD denotes the median absolute deviation:

$$MAD = median(|x_i - \widetilde{x}|) \tag{4.2}$$

In my experiments, if the Z-score is larger than 10, its corresponding point is marked as outliers.

One issue of estimating the dominating transformation from all of the matching pairs is that the estimated transformation may suffer from the masking effect. To address this issue, a strategy similar to RANSAC [268] is employed to sample 5 pairs of points from the feature matchings. The dominating transformation is estimated from the sampled 5 pairs and applied on all the pairs of matching points. Since the number of outliers is small compared to the total number of matchings, the 5 sampled pairs are unlikely to include any outliers, meaning that the estimated transformation is close to the true one. After the transformation, the feature points should be close to their correspondences except for the outliers. Statistically, if we repeat this process multiple times, the outliers will be identified most of the times, even though some times certain non-outlier might be identified as outlier by swamping effect [266].

## 4.2 Surface Alignment

In this section, I will talk about how to align all the piecewise surfaces together. I will first introduce the alignment between 2 neighboring surfaces, and it is followed by the global alignment of all the surfaces. Prior to these, an initial rigid alignment is applied to compensate for the large rotation introduced by the camera movement.

### 4.2.1 Initial Alignment

Here I decompose the motion of a deformable object into a rigid part and non-rigid part. The goal is to separate a potentially large rigid translation and rotation from a relatively small surface deformation, preventing the later deformation estimation process being biased by the large rigid motion.

Since we do not need to precisely align the surface patches into the reference frame, and all the detailed warping will be handled by the next global alignment step, a rough rigid transformation is enough. The feature correspondences between frames are mapped to 3D point correspondences which could be used to estimate a rigid transformation by absolute orientation ( [269]). Although absolute orientation is used to estimate the transformation of a rigidly moving object, it can still give a rough estimate of the dominating rigid motion of the object when combined with RANSAC [268]. The transformation between two non-overlapping frames will be estimated by accumulating the pairwise ones between consecutive frames.

**Minimum Number of Points** The transformation between two Cartesian coordinate systems can be thought of as the result of a rigid-body motion and can thus be decomposed into a rotation and a translation. In stereo photogrammetry, in addition, the scale may not be known. There are obviously three degrees of freedom to translation. Rotation has another three (direction of the axis about which the rotation takes place plus the angle of rotation about this axis). Scaling adds one more degree of freedom. Three points known in both coordinate systems provide nine constraints (three coordinates each), more than enough to permit determination of the seven unknowns.By discarding two of the constraints, seven equations in seven unknowns can be developed that allow one to recover the parameters. Two points clearly do not provide enough constraint.

**Sum of Squares of Errors** In practice, measurements are not exact, and so greater accuracy in determining the transformation parameters will be sought for by using more than three points. We no longer expect to be able to find a transformation that maps the measured coordinates of points in one system exactly into the measured coordinates of these points in the other. Instead, we minimize the sum of squares of residual errors. Finding the best set of transformation parameters is not easy. In practice, various empirical, graphical, and numerical procedures are in use. These are iterative in nature. That is, given an approximate solution, such a method leads to a better, but still imperfect, answer. The iterative method is applied repeatedly until the remaining error is negligible.

At times, information is available that permits one to obtain so good an initial guess of the transformation parameters that a single step of the iteration brings one close enough to the true solution of the least-squares problem to eliminate the need for further iteration in a practical situation.

As we shall see, the translation and the scale factor are easy to determine once the rotation is known. The difficult part of the problem is finding the rotation. Given three non-collinear points, we can easily construct a useful triad in each of the left and the right coordinate systems (figure 4.3). Let the origin be at the first point. Take the line from the first to the second point to be the direction of the new x axis. Place the new y axis at right angles to the new x axis in the plane formed by the three points. The new 2 axis is then made to be orthogonal to the x and y axes, with orientation chosen to satisfy the right-hand rule. This construction is carried out in both left and right systems. The rotation that takes one of these constructed

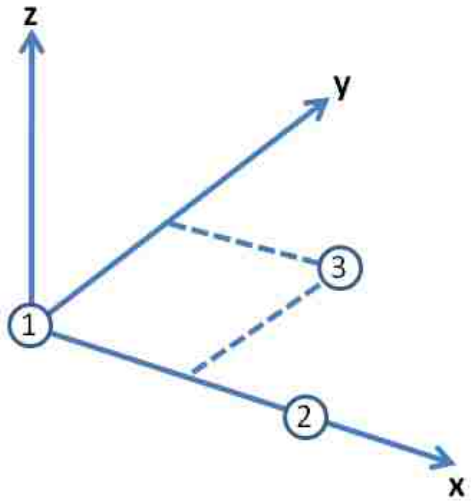


Figure 4.3: Three points define a triad.

triads into the other is also the rotation that relates the two underlying Cartesian coordinate systems. This rotation is easy to find, as I show below.

**Selective Discarding Constraints** Let the coordinates of the three points in each of the two coordinate systems $r_{l,1}$, $r_{l,2}$, $r_{l,3}$ and $r_{r,1}$, $r_{r,2}$, $r_{r,3}$ respectively. Construct

$$x_l = r_{l,2} - r_{l,1} \tag{4.3}$$

Then

$$\hat{x}_l = x_l / \|x_l\| \tag{4.4}$$

is a unit vector in the direction of the new x axis in the lefthand system. Now let

$$y_l = (r_{l,3} - r_{l,1}) - [(r_{l,3} - r_{l,1}) \cdot \hat{x}_l]\, \hat{x}_l \tag{4.5}$$

be the component of $r_{l,3} - r_{l,1}$ perpendicular to x. The unit vector

$$\hat{y}_l = y_l / \|y_l\| \tag{4.6}$$

is in the direction of the new y axis in the left-hand system. To complete the triad, we use the cross product

$$\hat{z}_l = \hat{x}_l \times \hat{y}_l \tag{4.7}$$

This construction is now repeated in the right-hand system to obtain $\hat{x}_r$, $\hat{y}_r$ and $\hat{z}_r$. The rotation that we are looking for takes $\hat{x}_l$ into $\hat{x}_r$, $\hat{y}_l$ into $\hat{y}_r$, and $\hat{z}_l$ into $\hat{z}_r$.

Now adjoin column vectors to form the matrices $M_l$ and $M_r$ as follows:

$$M_l = |\hat{x}_l \hat{y}_l \hat{z}_l|, \qquad M_r = |\hat{x}_r \hat{y}_r \hat{z}_r| \tag{4.8}$$

Given a vector $r_l$ in the left coordinate system, we see that

$$M_l^T r_l \tag{4.9}$$

gives us the components of the vector $r_l$ along the axes of the constructed triad. Multiplication by $M_r$ then maps these into the right-hand coordinate system, so

$$r_r = M_r M_l^T r_l \tag{4.10}$$

The sought-after rotation is given by

$$R = M_r M_l^T \tag{4.11}$$

The result is orthonormal since $M_r$ and $M_l$ are orthonormal, by construction. The above constitutes a closed-form solution for finding the rotation, given three points. Note that it uses the information from the three points selectively. Indeed, if we renumber the points, we get a different rotation matrix, unless the data happen to be perfect. Also note that the method cannot be extended to deal with more than three points.

Even with just three points we should really attack this problem by using a least-squares method, since there are more constraints than unknown parameters.

**Finding the Translation** Let there be n points. The measured coordinates in the left and right coordinate system will be denoted by

$$\{r_{l,i}\} \quad and \quad \{r_{r,i}\} \tag{4.12}$$

respectively, where i ranges from 1 to n. We are looking for a transformation of the form

$$r_r = sR(r_l) + r_0 \tag{4.13}$$

from the left to the right coordinate system. Here s is a scale factor, $r_0$ is the translational offset, and $R(r_l)$ denotes the rotated version of the vector $r_l$. We do not, for the moment, use any particular notation for rotation. We use only the facts that rotation is a linear operation and that it preserves lengths so that

$$\|R(r_l)\|^2 = \|r_l\|^2 \tag{4.14}$$

where $\|r\| = r \cdot r$ is the square of the length of the vector r.

Unless the data are perfect, we will not be able to find a scale factor, a translation, and a rotation such that the transformation equation above is satisfied for each point. Instead there will be a residual error

$$e_i = r_{r,i} - sR(r_{l,i}) - r_0 \tag{4.15}$$

We will minimize the sum of squares of these errors

$$\sum_{i=1}^{n} \|e_i\|^2 \tag{4.16}$$

We consider the variation of the total error first with translation, then with scale, and finally with respect to rotation.

**Centroids of the Sets of Measurements** It turns out to be useful to refer all measurements to the centroids defined by

$$\bar{r}_l = \frac{1}{n} \sum_{i=1}^{n} r_{l,i} \quad \bar{r}_r = \tfrac{1}{n} \sum_{i=1}^{n} r_{r,i} \tag{4.17}$$

Let us denote the new coordinates by

$$r'_{l,i} = r_{l,i} - \bar{r}_l \quad r'_{r,i} = r_{r,i} - \bar{r}_r \tag{4.18}$$

Note that

$$\sum_{i=1}^{n} r'_{l,i} = 0 \quad \sum_{i=1}^{n} r'_{r,i} = 0 \tag{4.19}$$

Now the error term can be rewritten as

$$e_i = r'_{r,i} - sR(r'_{l,i}) - r'_0 \tag{4.20}$$

where

$$r'_0 = r_0 - \bar{r}_r + sR(\bar{r}_l) \tag{4.21}$$

The sum of squares of errors becomes

$$\sum_{i=1}^{n} \left\| r'_{r,i} - sR(r'_{l,i}) - r'_0 \right\|^2 \tag{4.22}$$

or

$$\sum_{i=1}^{n} \left\| r'_{r,i} - sR(r'_{l,i}) \right\|^2 - 2r'_0 \cdot \sum_{i=1}^{n} \left[ r'_{r,i} - sR(r'_{l,i}) \right] + n \left\| r'_0 \right\|^2 \tag{4.23}$$

Now the sum in the middle of this expression is zero since the measurements are referred to the centroid. So we are left with the first and third terms. The first does not depend on $r'_0$, and the last cannot be negative. The total error is obviously minimized with $r'_0 = 0$ or

$$r_0 = \bar{r}_r - sR(\bar{r}_l) \tag{4.24}$$

That is, the translation is just the difference of the right centroid and the scaled and rotated left centroid. We return to this equation to find the translational offset once we have found the scale and rotation.

This method, based on all available information, is to be preferred to one that uses only measurements of one or a few selected points to estimate the translation.

At this point we note that the error term can be written as

$$e_i = r'_{r,i} - sR(r'_{l,i}) \tag{4.25}$$

since $r'_0 = 0$. So the total error to be minimized is just

$$\sum_{i=1}^{n} \left\| r'_{r,i} - sR(r'_{l,i}) \right\|^2 \tag{4.26}$$

**Finding the Scale** Although in my cases, we do not need to estimate the scale factor, I include it here just for algorithm integration.

Expanding the total error and noting that

$$\left\| R(r'_{l,i}) \right\|^2 = \left\| r'_{l,i} \right\|^2 \tag{4.27}$$

which can be written in the form

$$S_r - 2sD + s^2 S_l \tag{4.28}$$

where $S_r$ and $S_l$ are the sums of the squares of the measurement vectors (relative to their centroids), while D is the sum of the dot products of corresponding coordinates in the right system with the rotated coordinates in the left system. Completing the square in s, we get

$$(s\sqrt{S_l} - D/\sqrt{S_l})^2 + (S_r S_l - D^2)/S_l \tag{4.29}$$

This is minimized with respect to scale s when the first term is zero or $s = D/S_l$, that is,

$$s = \sum_{i=1}^{n} r'_{r,i} \cdot R(r'_{l,i}) / \sum_{i=1}^{n} \left\| r'_{l,i} \right\|^2 \tag{4.30}$$

**Symmetry in Scale** If, instead of finding the best fit to the transformation,

$$r_r = sR(r_l) + r_0 \tag{4.31}$$

we try to find the best fit to the inverse transformation,

$$r_l = \bar{s}\bar{R}(r_r) + \bar{r}_0 \tag{4.32}$$

we might hope to get the exact inverse:

$$\bar{s} = 1/s, \quad \bar{r}_0 = -\tfrac{1}{s} R^{-1}(r_0) \qquad \bar{R} \ = R^{-1} \tag{4.33}$$

This does not happen with the above formulation. By exchanging left and right, we find instead that $\bar{s} = \bar{D}/S_r$ or

$$\bar{s} = \sum_{i=1}^{n} r'_{l,i} \cdot \bar{R}(r'_{r,i}) / \sum_{i=1}^{n} \left\| r'_{r,i} \right\|^2 \tag{4.34}$$

which in general will not equal $1/s$, as determined above.

One of the two asymmetrical results shown above may be appropriate when the coordinates in one of the two systems are known with much greater precision than

those in the other. If the errors in both sets of measurements are similar, it is more reasonable to use a symmetrical expression for the error term:

$$e_i = \frac{1}{\sqrt{s}} r'_{r,i} - \sqrt{s} R(r'_{l,i}) \tag{4.35}$$

Then the total error becomes

$$\frac{1}{s} \sum_{i=1}^{n} \left\| r'_{r,i} \right\|^2 - 2 \sum_{i=1}^{n} r'_{r,i} \cdot R(r'_{l,i}) + s \sum_{i=1}^{n} \left\| r'_{l,i} \right\|^2 \tag{4.36}$$

or

$$\frac{1}{s} S_r + 2D + s S_l \tag{4.37}$$

Completing the square in s, we get

$$\left( \sqrt{s} S_l - \frac{1}{\sqrt{s}} S_r \right)^2 + 2(S_l S_r - D) \tag{4.38}$$

This is minimized with respect to scale s when the first term is zero or $s = S_r / S_l$, that is,

$$s = \left( \sum_{i=1}^{n} \left\| r'_{r,i} \right\|^2 / \sum_{i=1}^{n} \left\| r'_{l,i} \right\|^2 \right)^{1/2} \tag{4.39}$$

One advantage of this symmetrical result is that it allows one to determine the scale without the need to know the rotation. Importantly, the determination of the rotation is not affected by the choice of one of the three values of the scale factor. In each case the remaining error is minimized when D is as large as possible. That is, we have to choose the rotation that makes

$$\sum_{i=1}^{n} r'_{r,i} \cdot R(r'_{l,i}) \tag{4.40}$$

as large as possible.

**Representation of Rotation** There are many ways to represent rotation, including the following: Euler angles, Gibbs vector, Cayley-Klein parameters, Pauli spin matrices, axis and angle, orthonormal matrices, and Hamilton's quaternions [270,271]. Of these representations, orthonormal matrices have been used most often in photogrammetry and robotics. There are a number of advantages, however, to the unit-quaternion notation. One of these is that it is much simpler to enforce the constraint that a quaternion have unit magnitude than it is to ensure that a matrix is orthonormal. Also, unit quaternions are closely allied to the geometrically intuitive axis and angle notation.

Here I solve the problem of finding the rotation that maximizes

$$\sum_{i=1}^{n} r'_{r,i} \cdot R(r'_{l,i}) \tag{4.41}$$

by using unit quaternions. If desired, an orthonormal matrix can be constructed from the components of the resulting unit quaternion.

### 4.2.2 Warping Between Two Consecutive Frames

Different from constraining the problem by traditional epipolar geometry as in [272], we assume the object is under an arbitrary, non-linear deformation in long term, but linearly continuous locally in a short time. Therefore, surface patches can be warped

to shapes in neighboring frames using linear mesh deformation, given sufficient feature point correspondences.

Let $M$ be a polygon mesh defined by a pair $(V, K)$ of vertices $V = \{\vec{v}_1, ..., \vec{v}_n\}$ and edges $K$, the 1-ring neighborhood of a vertex $\vec{v}_i$ is the set of its adjacent vertices $N_i = \{j|(i,j) \in K\}$ and the degree $d_i$ denotes the number of vertices in $N_i$.

Given $\vec{u}_i$ on $M_0$ and $\vec{v}_i$ on $M_1$ be a correspondence pair representing the same feature point over a deforming object, the warping process on mesh $M_1$ from $t_1$ to $t_0$ changes $\vec{v}_i$ to $\vec{v}_i'$, which should be close to $\vec{u}_i$, implying that $M_0$ and $M_1$ will represent part of the same object. On the other hand, the warping process should minimize the mesh deformation as much as possible in order to maintain shape details. This can also be considered as a constraint to calculate warping over uncontrolled vertices in $M_1$.

In the mesh deformation community, surface shapes are usually described locally by Laplacian coordinates for vertices. The Laplacian coordinate $L(\vec{v}_i)$ for vertex $\vec{v}_i$ is defined by applying the Laplacian-Beltrami operator over the vertex coordinate:

$$L(\vec{v}_i) = \frac{1}{\sum_{j \in N_i} w_{ij}} \sum_{j \in N_i} w_{ij}(\vec{v}_i - \vec{v}_j) \tag{4.42}$$

$$w_{ij} = \tfrac{1}{2}(\cot \alpha_{ij} + \cot \beta_{ij}) \tag{4.43}$$

in which $\alpha_{ij}$ and $\beta_{ij}$ are two angles opposing to the edge $(i, j)$ as in [273]. When the mesh is regular and nearly uniformly defined, Equation 4.42 can be simplified as:

$$L(v_i) = \vec{v}_i - \frac{1}{d_i} \sum_{j \in N_i} \vec{v}_j \tag{4.44}$$

Since the goal of a warping procedure is to move specified control points closer to their target positions and still maintain the mesh shape as much as possible, mathe-

matically, this can formulated as a quadratic energy functional minimization problem as in [274]:

$$E(V') = \sum_{i \in V} \|L(\vec{v}_i) - L(\vec{v}_i')\|^2 + \sum_{i \in F} \|\vec{v}_i' - \vec{u}_i\|^2 \tag{4.45}$$

in which $V'$ is the vertex position after warping, and $F$ is the correspondence subset ($F \subseteq V$). The first sum measures the shape similarity before and after warping using Laplacian coordinates, whose least square solution is a linear system:

$$M_L V' = L, \quad V' = \begin{bmatrix} \vec{v}_1' \\ \vec{v}_2' \\ \cdots \\ \vec{v}_V' \end{bmatrix}, \quad L = \begin{bmatrix} L(\vec{v}_1) \\ L(\vec{v}_2) \\ \cdots \\ L(\vec{v}_V) \end{bmatrix} \tag{4.46}$$

$M_L$ is the laplacian matrix of the mesh. The second term gives the sum of squared differences over all control points, whose solution is given by:

$$M_I V' = U, \quad U = \begin{bmatrix} u_{i0} \\ u_{i1} \\ \cdots \\ \cdots \end{bmatrix} \tag{4.47}$$

Similar to an identity matrix, $M_I$ is a non-square matrix composed of zeros and ones, in which a row stands for a control vertex and a column stands for a mesh vertex. Each row has exactly one non-zero entry if and only if that control vertex is the corresponding mesh vertex.

Stacking $M_L$ and $M_I$ together, we obtain an over-determined linear system in order to find the overall least square solution to Equation 4.45:

$$\begin{bmatrix} M_L \\ M_I \end{bmatrix} V' = \begin{bmatrix} L \\ U \end{bmatrix} \tag{4.48}$$

Since X, Y and Z coordinates are independent in Equation 4.45, they can either be solved separately in three matrix systems, or simultaneously as a single system, in which case the matrix system will be three times larger.

Equation 4.48 produces plausible applaudable results when the deformation is small, but if the shape undergoes large rotation or scaling, the Laplacian coordinate is not a good descriptor since it is well known as affine-variant. The Laplacian coordinate of a vertex is actually a vector in 3D space that originates from the centroid of its neighbors and ends at the vertex. For example, if a mesh performs rotation, the Laplacian coordinates of its vertices should also rotate with the same angle along the same axis. Unfortunately, this cannot be properly handled by Laplacian coordinates in Equation 4.45. In order to address this issue, we apply an explicit affine transformation together with the warping procedure to account for any large affine transformation; thus the first term in Equation 4.45 becomes:

$$\sum_{i \in V} \|T_i L(\vec{v}_i) - L(\vec{v}_i')\|^2 \tag{4.49}$$

in which $T_i$ is an estimated local affine transformation for $\vec{v}_i$. Since the Laplacian coordinate is determined by the 1-ring neighborhood of $\vec{v}_i'$, $T_i$ can be calculated from the 1-ring neighborhood as well. By describing $T_i$ as a function of $V'$, the estimation of $T_i$ can be implicitly contained into a single system with $V'$ as the only unknowns. Unfortunately, this becomes a non-linear optimization problem since $T_i$ is nonlinearly determined by $V'$, which is known to be difficult to solve. Instead of using an exact solution, we adopted the method proposed in [274] to simply approximate $T_i$ as a linear function of $V'$ if the rotation angle is small. Specifically, we first define $T_i$ in the homogeneous coordinates:

$$T_i = \begin{bmatrix} s & -h_3 & h_2 & t_x \\ h_3 & s & -h_1 & t_y \\ -h_2 & h_1 & s & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.50}$$

By definition, an optimal $T_i$ should minimize the following functional:

$$\sum_{k \in \{i\} \cup N_i} \|T_i \vec{v}_k - \vec{v}'_k\|^2 \tag{4.51}$$

Let $t_i = (s, h_1, h_2, h_3, t_x, t_y, t_z)^T$ be the vector of the unknowns in $T_i$, Equation 4.51 can be rewritten as:

$$\|A_i t_i - V_{N_i}\|^2 \tag{4.52}$$

in which

$$A_i = \begin{bmatrix} v_{k_x} & 0 & v_{k_z} & -v_{k_y} & 1 & 0 & 0 \\ v_{k_y} & -v_{k_z} & 0 & v_{k_x} & 0 & 1 & 0 \\ v_{k_z} & v_{k_y} & -v_{k_x} & 0 & 0 & 0 & 1 \\ \vdots & & & & & & \end{bmatrix} \tag{4.53}$$

and

$$V_{N_i} = \begin{bmatrix} v'_{k_x} \\ v'_{k_y} \\ v'_{k_z} \\ \vdots \end{bmatrix} \tag{4.54}$$

is a vertex vector of $\vec{v}'$ and its neighborhood. $t_i$ can then be calculated from $V_{N_i}$:

$$t_i = (A_i^T A_i)^{-1} A_i^T V_{N_i} \tag{4.55}$$

By creating a new matrix $\Delta_i$ using the coordinates in $L_(v_i)$ as follows:

$$\Delta_i = \begin{bmatrix} L_x(\vec{v}_i) & 0 & L_z(\vec{v}_i) & -L_y(\vec{v}_i) & 1 & 0 & 0 \\ L_y(\vec{v}_i) & -L_z(\vec{v}_i) & 0 & L_x(\vec{v}_i) & 0 & 1 & 0 \\ L_z(\vec{v}_i) & L_y(\vec{v}_i) & -L_x(\vec{v}_i) & 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.56}$$

we can then calculate $T_i L(\vec{v}_i)$ as:

$$T_i L(\vec{v}_i) = \Delta_i t_i = \Delta_i (A_i^T A_i)^{-1} A_i^T V_{N_i} \tag{4.57}$$

$D_i = \Delta_i (A_i^T A_i)^{-1} A_i^T$ is solely defined on $V$, so it can be computed in advance, meaning that $T_i L(\vec{v}_i)$ is linear function of $V_{N_i}$. By stacking all $T_i L(\vec{v}_i)$ together into a

large vector $T_L$, a large sparse matrix $M_T$ can be constructed using submatrices $\{D_i\}$ such that:

$$T_L = M_T V' \tag{4.58}$$

Replace the right hand side of equation 4.46 with $T_L$:

$$M_L V' = M_T V' \tag{4.59}$$

or,

$$(M_L - M_T)V' = 0 \tag{4.60}$$

Replacing the corresponding part in equation 4.48 with equation 4.61, we get the linear equation that can stitch two pieces of mesh with rotation and scaling between them.

$$\begin{bmatrix} (M_L - M_T) \\ M_I \end{bmatrix} V' = \begin{bmatrix} 0 \\ U \end{bmatrix} \tag{4.61}$$

### 4.2.3 Warping All Frames Simultaneously

A naive approach to obtain a complete 3D shape is to simply assemble two surface patches each time. For example, in order to create the shape surface at frame $i$, the surface patch at frame 1 is first combined with frame 2 into a new surface $1-2$ which is combined with frame 3, so on and so forth. By keeping doing so we combine all surface patches together into the desired shape at frame $i$. Since local temporal correspondences between two adjacent frames determine the warping procedure in each step, errors can be easily accumulated from frame to frame, causing misalignment between surface patches.

Another reason we need a global method is that sequential warping cannot deal with occlusion. I will discuss this in detail in the occlusion handling section.

I develop a global warping algorithm in order to warp surface patches in all frames altogether to the destination frame in a single step. This gives a single linear system with unknowns as the final positions of vertices in the destination frame. As an extension from the local warping algorithm, a global warping matrix system lists all Laplacian constraints as Diagonal sub-matrices as shown in Equation 4.62.

$$\begin{bmatrix} Q_1 & & & & & & & \\ & Q_2 & & & & & & \\ & & \ddots & & & & & \\ & & & Q_{d-1} & & & & \\ & & & & Q_{d+1} & & & \\ & & & & & \ddots & & \\ & & & & & & Q_n & \\ & & & & = 0 & & & \end{bmatrix} \begin{bmatrix} V'_1 \\ V'_2 \\ \vdots \\ V'_{d-1} \\ V'_{d+1} \\ \vdots \\ V'_n \end{bmatrix} \tag{4.62}$$

in which $Q = M_L - M_T$ is the Laplacian constraint, $n$ is the number of frames, and $d$ is the destination frame index. Feature correspondence constraints are imposed by adding more rows into equation 4.62. For example, if a correspondence exists from frame $i$ to frame $d$, the following row will be added to the matrix system in equation 4.62 as a correspondence constraint:

$$\begin{bmatrix} 0 & \dots & 0 & \underset{k^{th}}{1} & 0 & \dots & 0 \end{bmatrix} \tag{4.63}$$

$k$ is the index of the vertex into the vector of unknowns in equation 4.62. Accordingly, the position of the corresponding vertex in frame d should be added to the end of the vector on the right hand side of equation 4.62. If frame $i$ has a correspondence point in frame $j$, which is also unknown, a different row will need to be added to the

matrix:

$$\begin{bmatrix} 0 & \dots & \underset{k^{th}}{1} & \dots & \underset{h^{th}}{-1} & \dots & 0 \end{bmatrix} \tag{4.64}$$

Here, $k$ and $h$ indicate the matching vertices' position in the vector of unknowns, and a 0 should be added to the end of the right hand side vector ensuring that these two vertices be at the same 3D position after warping.

This global method is in spirit similar to *bundle adjustment*. However my formulation is linear while typical bundle adjustment is formulated as non-linear optimization that requires iterative methods.

## 4.3 Exception Handling

Most of the time, the above discussed method will not give us 100% correct and complete 3D models. There are some more issues we need to take care of. For example, the occlusion handling, missing feature correspondences. I will discuss these issues in the following subsections and introduce my solutions.

### 4.3.1 Occlusion Handling

Since both the camera viewpoint is moving and the object is deforming, the least square doesn't necessarily yield the correct position. As illustrated in figure 5.7, to complete the 3D model of frame 17, the occluded leg needs to be recovered with the information from its neighboring frames 16 and 18. One possibility is that this leg is topologically connected to the visible surfaces, and will be pulled to a certain position under the laplacian constraint. This approach will result in incorrect warping

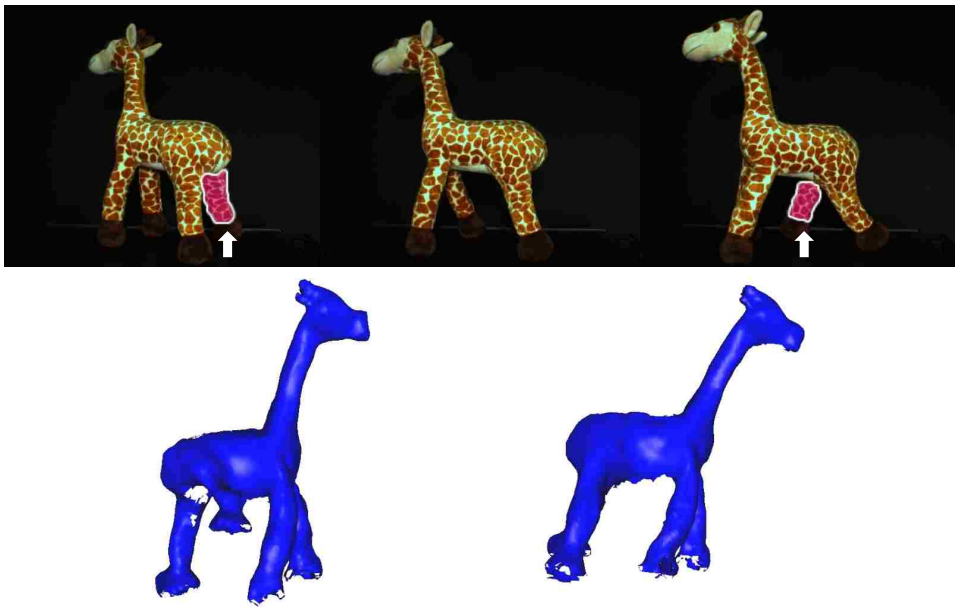

Figure 4.4: The need for occluded feature interpolation: $1^{st}$ row from left to right: frame 16, 17 and 18 of a walking giraffe toy. Note that one leg is completely occluded in frame 17. $2^{nd}$ row shows the reconstructed results of frame 17 without and with the occlusion handling. As can be seen in the left image, the occluded leg is largely distorted. After features are interpolated, it is corrected in the right image.

(figure 5.7) when the leg itself is moving during these frames.

A more sophisticated approach is to use the tracked features to predict their occluded positions. The features are first extracted and stored for each frame. In the next step, I establish a global feature pool by searching the feature set of each frame for those that are visible in multiple frames. The global features are recorded along with their frame numbers and corresponding 3D positions. With those globally tracked features, the occluded part can be interpolated or extrapolated under the continuous motion assumption. Figure 5.7 shows the correct result by this method.

Figure 4.5: Mis-alignment is shown in the leftmost image. I manually add some feature correspondences on the mis-aligned region in the corresponding images (middle 2 images). The rightmost image shows that the surfaces are well aligned with the additional feature matchings.

### 4.3.2 User Interaction

Since the alignment of the non-rigid deforming surfaces completely relies on the guidance of vertex correspondences, it could happen that a certain region of object cannot be stitched together due to the lack of correspondences. As shown in figure 4.5, two neighboring surfaces are well aligned except for the left arm, because the automatic feature tracking algorithm fails to find feature matching in this area. In this case, I manually add some matchings between the corresponding images (figure 4.5 middle two images). The added feature matchings will be appended to those existing matchings, and the global warping algorithm will be initiated again to incorporate the additional matchings. The deformed surfaces output from the global warping are inspected for mis-alignment in other frames/regions, which need to be fixed by some more manual input of feature matchings. This procedure is repeated until satisfactory alignment is achieved.

### 4.3.3 Smoothing and Refinement

After the warping procedure, surface patches are aligned together to cover the shape of an object at the same time instance. In this section, they will be merged together to form a single surface. Instead of manipulating meshes directly, I choose to use an Eulerian approach by volumetric representation for several major reasons. First of all, a volumetric representation can easily handle topological changes among different meshes. Secondly, a volumetric representation can fix missing holes and misalignments, which are often caused by image noise, correspondence errors, occlusions or other errors. Last but not least, an Eulerian approach is straightforward to implement and it does not require the complicated re-meshing process.

We first define a distance function $d(\vec{x})$, which gives the minimum absolute distance from $\vec{x}$ to any surface patches. Let $\phi$ be a signed distance function representing the final steady shape we would like to achieve, the level set formulation is:

$$\frac{\partial \phi}{\partial t} = \left( \nabla d \cdot \frac{\nabla \phi}{|\nabla \phi|} + d \nabla \cdot \frac{\nabla \phi}{|\nabla \phi|} \right) |\nabla \phi| \tag{4.65}$$

Intuitively, $\phi$ will first be smoothed by a mean curvature flow. Once it gets close to surface patches $d$, the smoothing effect will be gradually reduced and it will cover all surface patches. As an example, in Figure 4.8, the holes by invisibility are filled up with this method. Details of this technique can be found in [275].

## 4.4 Experiments and Results

I have tested my algorithms on both synthetic data and real data. The synthetic data is generated with the 4D models shared by [12]. As in figure 4.7, for each frame,

the renderer outputs the depth map and tracked 3D points specified in advance. 400 out of 20000 vertices are specified as tracking points, which are uniformly distributed on the object surface. Since this data set has no color features to track, we generate correspondences directly.

Figure 4.7 shows results with correct correspondences. The complete 3D model is recovered. I further evaluated the performance of my algorithms under imperfect tracking by perturbing the original matching by some amount. Specifically, 1-pixel perturbation means matching a pixel to one that is randomly selected from the 1-ring neighbor pixels of its true correspondence. 3-pixel, 5-pixel, and 10-pixel perturbations are performed in the similar way. One pixel distance is approximately 10mm in the real world. So, 1-pixel perturbation is roughly a 1% perturbation in the real 3D space. Figure 4.6 shows the reconstructed results under the perturbations. It can be seen that when the amount of perturbation increases, the fine details are lost, nevertheless the overall shape is always well recovered. Table 4.1 shows the errors between the reconstructed model of frame 1 and the ground truth.

| | DimX | DimY | DimZ | Max Dist | Avg Dist |
|---|---|---|---|---|---|
| 1-p | 844.4 | 1273.7 | 1814.0 | 13.60 | 1.927 |
| 3-p | 844.4 | 1273.7 | 1814.0 | 14.78 | 2.278 |
| 5-p | 844.4 | 1273.7 | 1814.0 | 16.95 | 2.646 |
| 10-p | 844.4 | 1273.7 | 1814.0 | 20.02 | 3.487 |

Table 4.1: The errors between reconstructed model of frame 1 and the ground truth. From row 2 to row 5: 1-pixel, 3-pixel, 5-pixel and 10-pixel perturbation. DimX, DimY, and DimZ are the size of the model in x,y,z dimensions. Max Dist and Avg Dist are the maximum and average distance between the result and ground truth. Details are lost when noise increases.

The real data is captured by a SwissRanger depth camera combined with a point

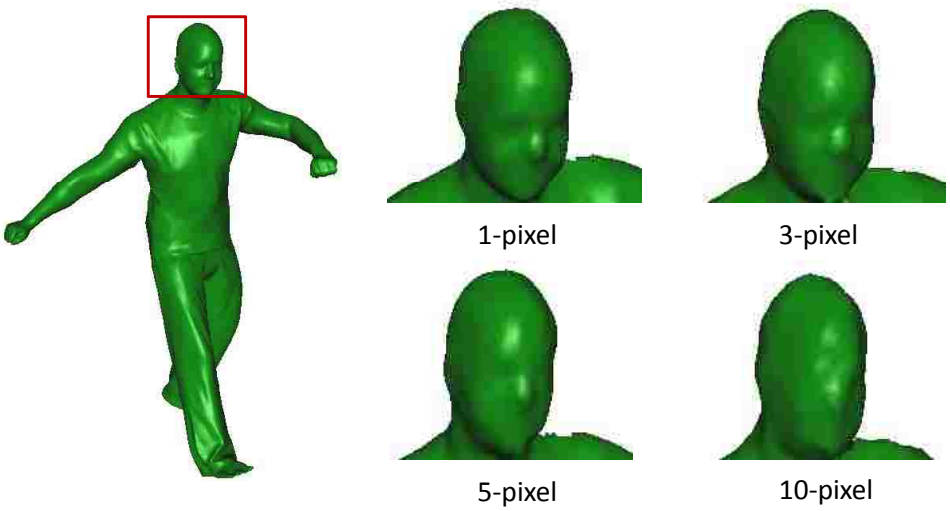


Figure 4.6: The comparison of results from different perturbations. As perturbation amount increases, details are lost.

grey flea color camera that provides texture information. The depth camera can produce $176 \times 144$ depth map at video rate. However, the quality of the depth map drops quite significantly for dynamic scenes. So I manually animate the toy giraffe and use temporal averaging to improve the signal-to-noise ratio of the depth map. SIFT features are extracted for each frame and features are tracked across different frames by searching in the pool of the extracted ones. The depth camera and color camera are almost coaxial so the mapping between their images can be approximated by a homography. There are about 200 features that are reliably tracked. Results in figure 4.8 show that the occlusion can be well handled by my algorithms. Figure 4.9 is a comparison of the models from frame 5 before and after hole-filling and smoothing. Figure 4.10 shows the reconstruction of a T-shirt worn by a person who turned around 360 degrees in front of the capture device, while moving his hands up and down.

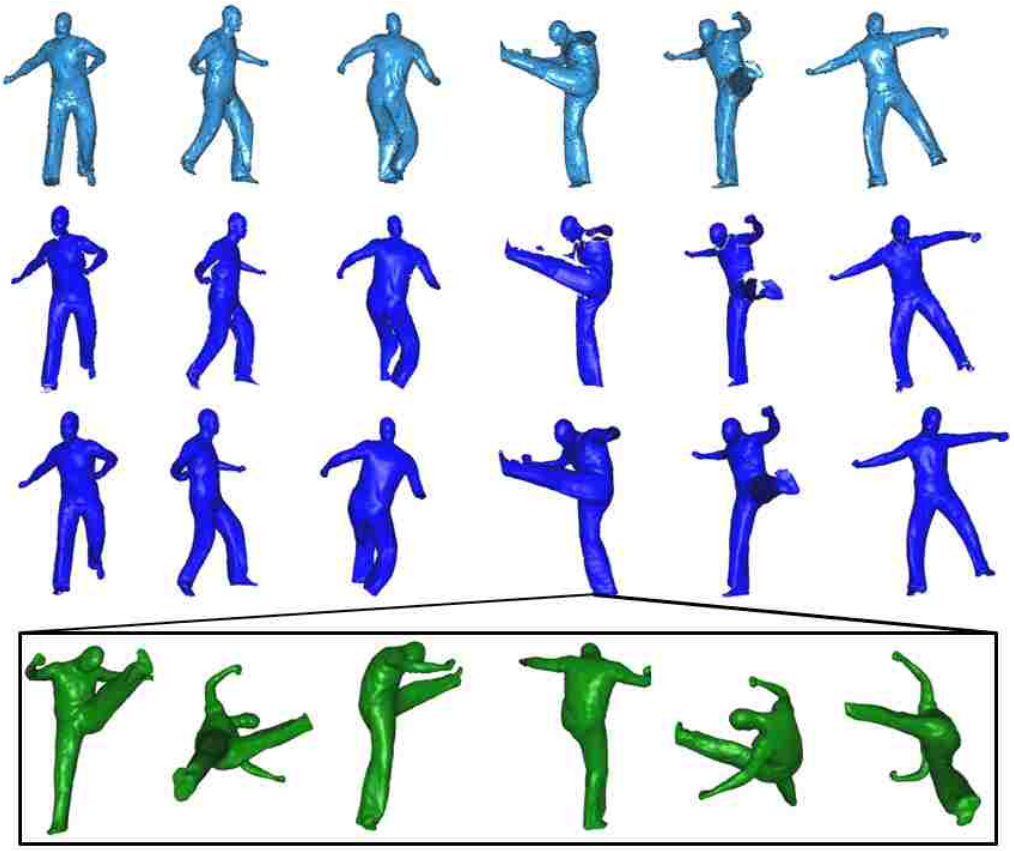

Figure 4.7: Six frames (frame 1, 10, 17, 24, 29 and 35 out of all 38 frames) are shown in this figure. $1^{st}$ row shows the rendered models. The black dots indicate the tracked features. $2^{nd}$ row is the partial meshes constructed from depth maps. The $3^{rd}$ row shows the reconstructed 3D models by my algorithm. And the $4^{th}$ row shows different views of the 3D model of frame 24.

## 4.5 Conclusion

I have developed a novel approach to reconstruct complete 3D surface deformation over time by a single camera. The deformable surface patches are stitched together by mesh deformation in a global manner, and merged into a complete model by a volumetric method. Test on both synthetic and real data demonstrated that my approach works well with even large deformation. I believe my approach will help to simplify the difficult task of creating time-varying models for dynamic objects.

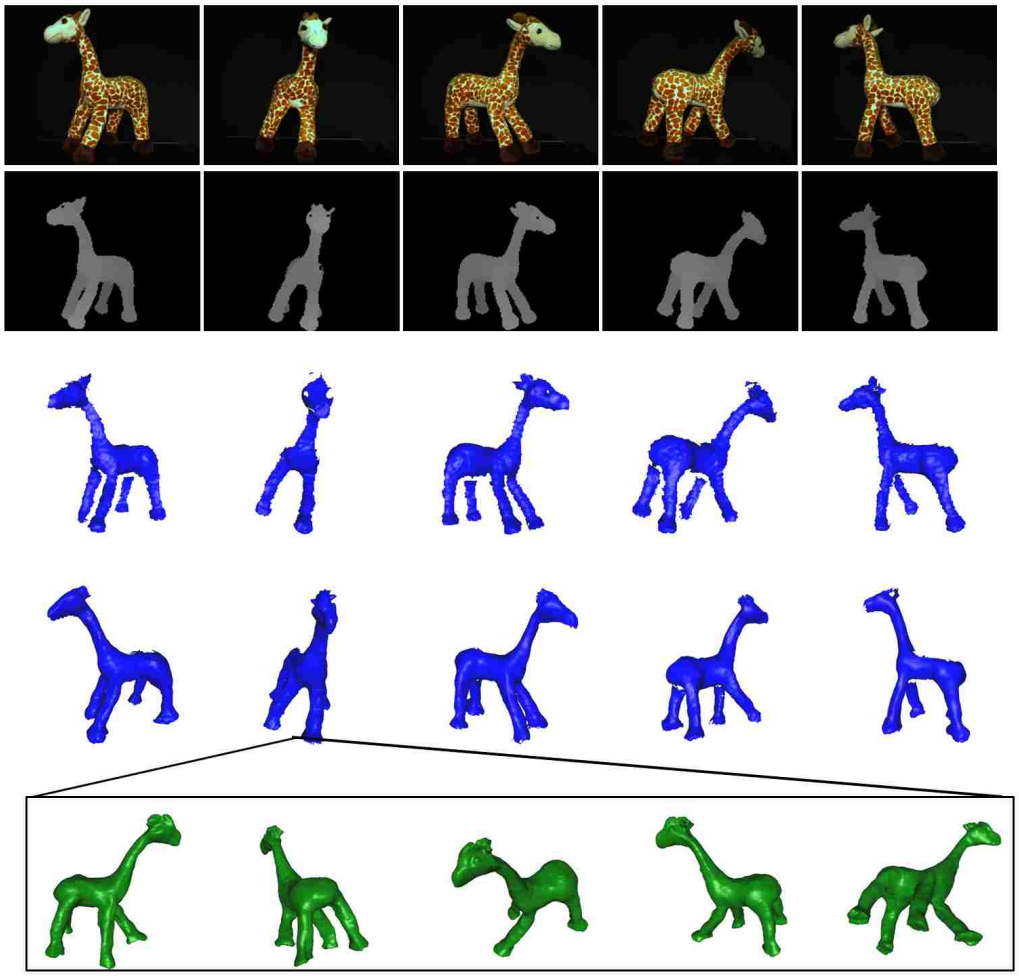

Figure 4.8: Frame 3, 5, 7, 11 and 17 out of total 18 frames are shown as an example here. $1^{st}$ row shows the color images. $2^{nd}$ row are the captured depth maps. The $3^{rd}$ row shows the partial meshes constructed from depth maps. The $4^{th}$ row shows the reconstructed 3D models by my algorithm. And the $5^{th}$ row shows different views of the 3D model of frame 5.

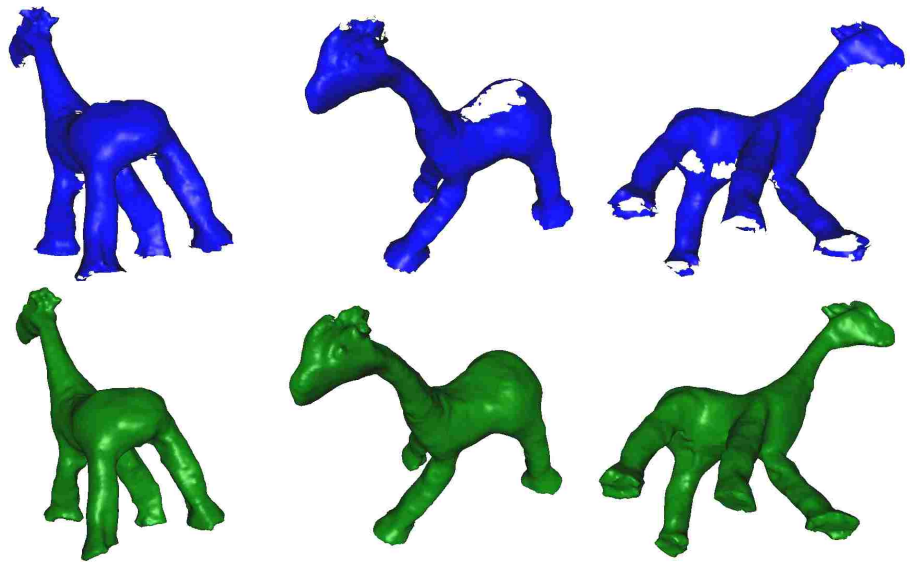

Figure 4.9: Three different views of the reconstructed model from frame 5 of the real data. The watertight model after smoothing is shown on the $2^{nd}$ row.

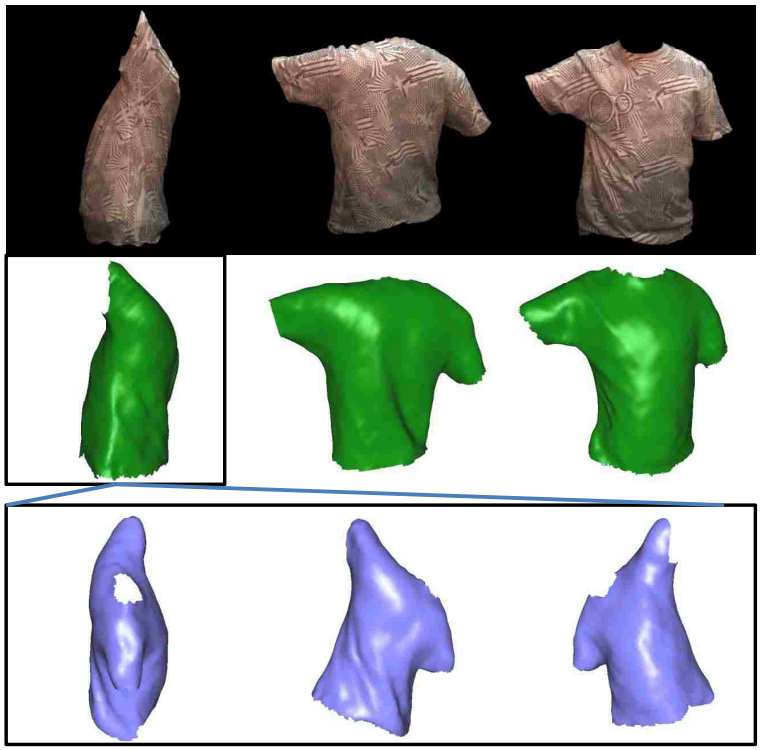

Figure 4.10: $1^{st}$ and $2^{nd}$ rows, from left to right: Frame 5, 7, 14 out of total 14 frames of a deforming shirt. $3^{rd}$ row shows 3 different views of the 3D model of frame 5.

## Chapter 5 2D-3D Video Conversion

The above chapter presents the possibility with the existing depth cameras. However, if we only have a regular camcorder instead of a depth camera, which means we are given a video sequence without any depth information. Can we still explore 3D with the input video? The answer is yes. In this chapter, I explore the possibilities to turn it into stereoscopic video pairs, enabling 3D perception and lifelike viewer experiences.

In particular, I present a semi-automatic system that converts conventional videos into stereoscopic videos by combining motion analysis with user interaction, aiming to transfer as much as possible labeling work from the user to the computer. In addition to the widely-used structure from motion (SFM) techniques, I develop two new methods that analyze the optical flow to provide additional qualitative depth constraints. They remove the camera movement restriction imposed by SFM so that general motions can be used in scene depth estimation C the central problem in mono-to-stereo conversion. With these algorithms, the user's labeling task is significantly simplified. I further developed a quadratic programming approach to incorporate both quantitative depth and qualitative depth (such as these from user scribbling) to recover dense depth maps for all frames, from which stereoscopic view can be synthesized. In addition to visual results, I present user study results showing that my approach is more intuitive and less labor intensive, while producing 3D effect comparable to that from current state-of-the-art interactive algorithms.

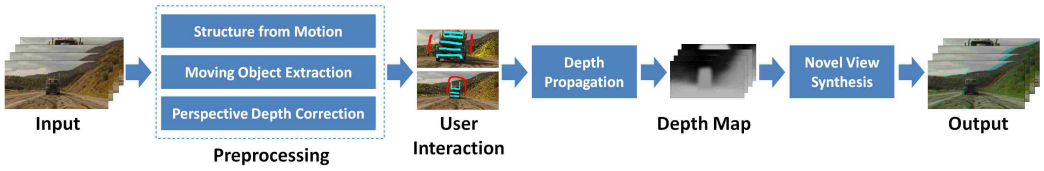


Figure 5.1: The pipeline of my system.

## 5.1 Automatic Pre-Processing

Figure 5.1 shows the pipelines of my system. In the pre-processing step, the input image sequence is first passed through three individual automatic modules: *structure-from-motion* (SFM), *moving object segmentation* (MOS), and *perspective depth correction* (PDC). The SFM algorithm is applied to the input image sequence with dominant rigidly moving objects to recover a sparse set of 3D points. The MOS module is used to automatically segment the foreground, it is particularly effective in a *follow* shot in which the foreground is relatively static and the background is rapidly changing. Finally, the PDC module inspects the size change of an object's image to estimate relative depth changes between frames. After automatic processing, the users are presented with images showing area with known depth (from SFM and MOS). If there are still undefined regions, the users need to label them in some key frames by simple scribbling. The user's input as well as all the automatically calculated depth cues will be integrated in a quadratic programming framework to generate dense depth maps for all frames.

Finally the novel view is generated via shifting every pixel horizontally by a certain amount base on the depth maps, simulating the perspective from the other eye. Since in my cases, the baseline between the synthesized view and the input view is

small, I use a simple technique to deal with the gaps in the synthesized view due to disocclusion. I simply fill the uncolored region with neighboring pixels of larger depth values. I found this method worked well in practice. View synthesis is not the emphasis here, therefore I will not discuss it further.

### 5.1.1 Structure from Motion and Optical Flow Estimation

The structure from motion algorithm is conventionally employed to recover the 3D positions of the feature points and the positions of the camera in a sequence of static scene. After the positions of the camera track are computed, a dense depth map for each frame can be obtained by multiple view stereo [13].

My system does not make any assumption on the input image sequences. The scene may be captured by a camera with fixed viewpoint or may contain non-rigid motion and hence, the SFM algorithm does not always work. However, in most of the moving camera shots, SFM can automatically track sparse features on the static background, leaving only the moving foreground objects to be labeled by the users. Both spacial and temporal depth variations for these feature points can be accurately computed in my approach. By contrast, Guttmann et al.'s approach [2] requires users to careful label the depths at different potions of the background, as well as how the depth changes across different frames, which can be both cumbersome and imprecise for scenes with complex background.

The depth information extract by SFM are incorporated into the final solution through equality constraint. Basically, for each feature point $\mathbf{p}$ in frame $t$ with assigned depth $D^t_{SFM}(\mathbf{p})$, I add the following constraint to the final linear equation:

$$d^t(\mathbf{p}) = D^t_{SFM}(\mathbf{p}) \tag{5.1}$$

where $d^t(\mathbf{p})$ is the depth value of pixel $\mathbf{p}$ at frame $t$.

I also estimate the optical flow between adjacent frames. Similar to [276], the optical flow is estimated by solving an optimization problem defined on region-tree by dynamic programming. The region-trees of over-segmented regions are constructed from the input images. The optical flow will be used in both perspective depth correction and moving object extraction.

### 5.1.2 Moving Object Extraction

In follow shots, where the camera tracks the moving foreground objects, it is often than not that the tracked (foreground) objects are visually consistent in the video sequence, whereas the appearance of the backdrops varies. Based on this observation, we can achieve the automatic segmentation by counting the statistical coherence of appearance variation at each pixel, followed by making a per-pixel decision whether it is foreground or not.

Given a sequence of video frames $\{I_1, I_2, \cdots I_m\}$, my goal is to segment the foreground object $\mathcal{O} = \bigcap_{i=1}^{m} I_i$. However, segmenting all frames simultaneously needs to solve a large number of unknowns. Instead, I focus on segmenting one frame at each time, but with additional information propagated from other frames. I first establish dense correspondence maps between any pair of frames $I_i$ and $I_j$ by traversing through a connected path from $I_i$ to $I_j$ related by optical flows.

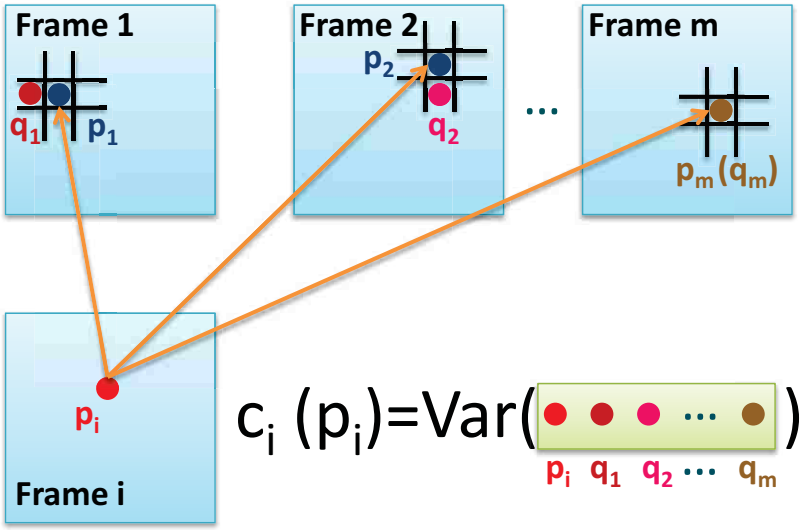


Figure 5.2: An illustration to compute the color variance $c_i(\mathbf{p}_i)$ at pixel $\mathbf{p}_i$ in frame $I_i$. Suppose pixel $\mathbf{p}_i$ finds its correspondences $\mathbf{p}_1$ in frame $I_1$, $\mathbf{p}_2$ in frame $I_2$, and up to $\mathbf{p}_m$ in frame $I_m$ as shown by the orange arrows, and $\mathbf{p}_1$'s neighboring pixel $\mathbf{q}_1$ has the smallest color difference with $\mathbf{p}_i$ and so on and so forth; then the cost $c_i(\mathbf{p}_i)$ is the variance of the color values from $\mathbf{p}_i, \mathbf{q}_1, \mathbf{q}_2 \cdots \mathbf{q}_m$.

In the next step, I will compute a color variance map $\mathcal{C}_i$ for frame $I_i$, where each element $c_i(\mathbf{p}_i)$ corresponds to the color variance of the series consisting of pixel $\mathbf{p}_i$ in frame $I_i$ and its correspondent pixels in other frames. Suppose we find $\mathbf{p}_i$'s correspondences $\mathbf{p}_1$ in frame $I_1$, $\mathbf{p}_2$ in frame $I_2$, and up to $\mathbf{p}_m$ in frame $I_m$; we can simply compute $c_i(\mathbf{p}_i)$ as the color variance of the series:

$$c_i(\mathbf{p}_i) = \mathrm{Var}\left(\theta_i(\mathbf{p}_i), \theta_1(\mathbf{p}_1), \theta_2(\mathbf{p}_2), \cdots \theta_m(\mathbf{p}_m)\right), \tag{5.2}$$

where $\theta_j(\mathbf{p}_j)_{1 \le j \le m}$ is the color of the pixel $\mathbf{p}_j$ in frame $I_j$. However, to be more robust to image noise and alignment offsets, I use the approximate matching strategy to compute $\mathcal{C}_i$, that is instead of selecting $\mathbf{p}_i$'s exact matched pixel $\mathbf{p}_j$ in frame $I_j$, I search the $\mathbf{p}_j$'s neighboring pixel $\mathbf{q}_j$, where color $\theta_j(\mathbf{q}_j)$ is closest to the reference color $\theta_i(\mathbf{p}_i)$, and efficiently compute the color variance in a greedy manner as:

$$\mathbf{q}_j = \underset{\mathbf{q}_j \in N(\mathbf{p}_j)}{\arg\min} \left|\theta_j(\mathbf{q}_j) - \theta_i(\mathbf{p}_i)\right|, \quad j \neq i, \tag{5.3}$$

$$c_i(\mathbf{p}_i) = \mathrm{Var}\left(\theta_i(\mathbf{p}_i), \theta_1(\mathbf{q}_1), \theta_2(\mathbf{q}_2), \cdots \theta_m(\mathbf{q}_m)\right), \tag{5.4}$$

where $N(\mathbf{p}_j)$ represents a search window centered at pixel $\mathbf{p}_j$ in frame $I_j$ with the size of $w$, and in the experiments, I typically set $w = 7 \times 7$. This procedure is illustrated in Figure 5.2.

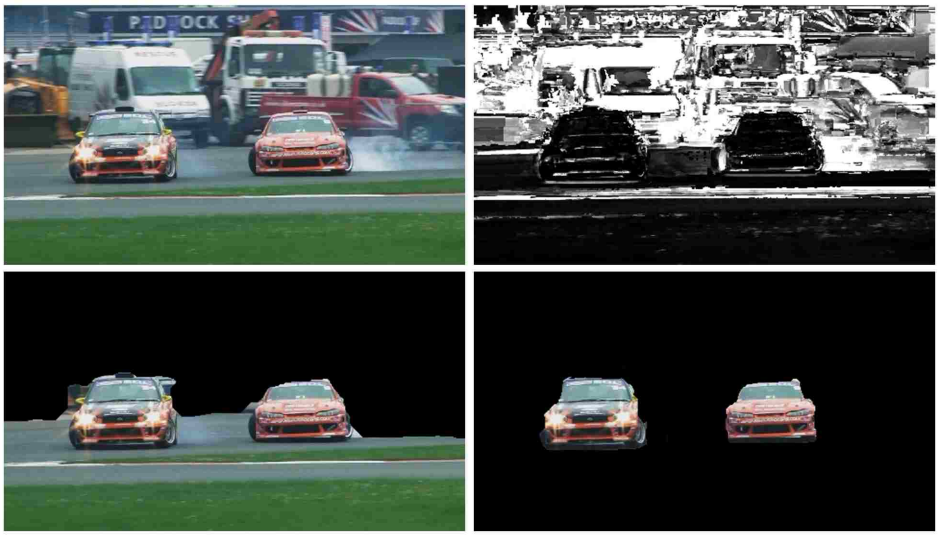

Figure 5.3: An intermediate result of the moving object extraction. Given one frame $I_i$ (upper left image), we can first compute its variance map $\mathcal{C}_i$ (upper right image). Based on $\mathcal{C}_i$, we can mask out the high-variance (bright in the variance map) parts using graph-cut algorithm to get the initial segmentation (lower left image). Furthermore, a user can provide only a few color seeds, e.g., the green color in the grass and gray in the ground, to remove the unexpected extracted parts and get the final segmentation result (lower right image).

Finally, I formulate the variance map $\mathcal{C}_i$ as an additional data term $V$ in the binary labeling Markov Random Field (MRF) energy minimization framework for image segmentation. $V$ is defined in equation 5.5:

$$V(\mathbf{p}) = \begin{cases} \exp(-c_i(\mathbf{p})/\alpha_f) & \text{if } \mathbf{p} \in \text{Background}, \\ 1 - \exp(-c_i(\mathbf{p})/\alpha_b) & \text{if } \mathbf{p} \in \text{Foreground}, \end{cases} \tag{5.5}$$

where $\alpha_f = 7 \times v_f$, $\alpha_b = 0.5 \times v_b$; $v_f$ is set to the 0.1 percentile in $\mathcal{C}_i$ and $v_b$ is set to the 99th percentile in $\mathcal{C}_i$. The explanation of equation 5.5 is that if the color variance $c_i(\mathbf{p})$ is small, which implies that the pixel $\mathbf{p}$ in frame $I_i$ finds visually consistent matches in the rest of frames, the pixel should be considered as foreground, i.e., the cost of assigning it to background is high. On the contrary, a large color variance $c_i(\mathbf{p})$ indicates pixel $\mathbf{p}$ should be labeled as background. Combining the other widely used visual cues, such as color and edge information [238, 239], I adopt the graph-cut approach to solve the minimization problem using the min-cut/max-flow algorithm [238] and eventually extract the moving foreground objects from the video sequence. Figure 5.3 shows the segmentation result for one frame in a follow shot.

When the foreground and background are separated, they will be treated as two different depth layers. However, not the entire background region is marked as further than the foreground, but only the areas that is above and to the both sides of the foreground object, which I call defined background. This is under the observation that most follow shots are following objects on the ground, so that the ground below the object is actually closer and should not be regarded as behind the object. Let $\mathbf{p}$ and $\mathbf{q}$ be arbitrary pixels that lie on the foreground and defined background respectively. The following constraints are plugged into the final formulation:

$$d(\mathbf{p}) - d(\mathbf{q}) \geq \Delta d \tag{5.6}$$

where $\Delta d$ is a predefined threshold that denotes the minimum depth difference between two layers.

### 5.1.3 Perspective Depth Correction

Another useful vision cue is that, under perspective projection with fixed focal length, a rigid object gets bigger when it moves closer to the camera and appears smaller when it moves away. As shown in figure 5.4, the size change of the object shows up on the estimated optical flow as local flow expansion or shrinking. Here I use this cue to automatically infer depth changes of moving objects. It is noteworthy that this observation is based on the assumption of local rigidity, i.e., the local structure of the objects in the scene undergoes rigid motion in short time intervals even though deformable objects are allowed in the video.

The optical flow of an object does not show up as pure expansion or shrinking unless it is at the center of the image and is moving along the optical axis of the camera. In general cases, the optical flow of the object is a combination of the motion along the depth direction and the one parallel to the camera plane. To extract the portion of the flow caused by depth change, here I first apply Helmholtz-Hodge decomposition to decompose the 2D optical flow into a divergence-free vector field and a divergence field. The divergence-free field is ignored and the divergence field undergos the following test to see if it is caused by the depth change of a rigid object.

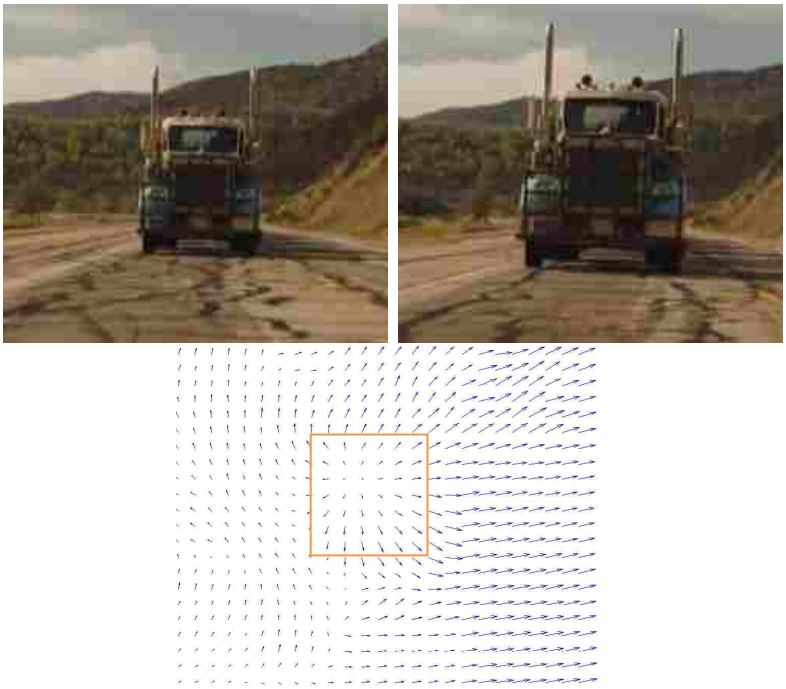

Figure 5.4: The illustration of the expanding optical flow.

The test is performed based on a local structure consisting of a pixel and its 4 spatial neighbors, which I refer as a unit structure. The 5 pixels in the unit structure are traced into the next frame by the pre-computed optical flow.

Figure 5.5 shows four of the possible shapes the unit structure could take in the next frame under the rigidity assumption. 1) If the unit structure is scaled evenly in all the directions, it suggests that the corresponding 3D points of the unit structure pixels are at the same depth and move toward (or away from) the camera. 2) If the unit structure is evenly scaled and rotated, the motion of the corresponding 3D structure is exactly the same as in the first case except that there is also in-image-plane rotation. 3) If the unit structure is unevenly scaled, it could result from the depth variation among the structure's points, causing different points moving at different observed velocities. 4) If the unit structure is skewed, it may caused by the

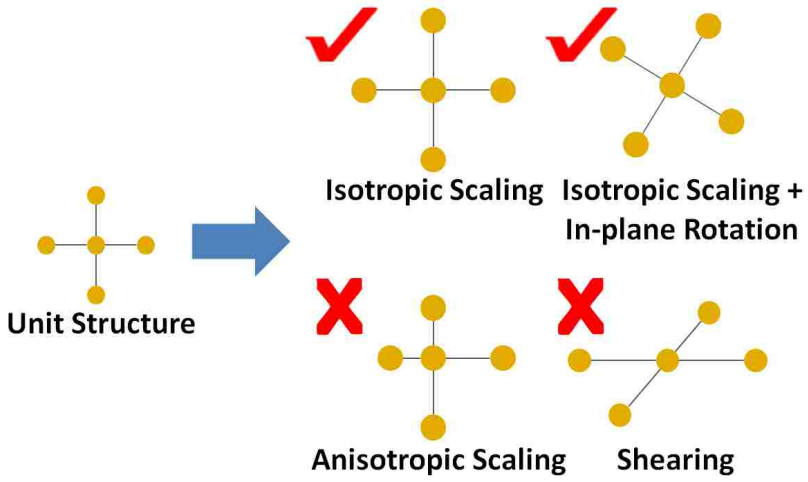


Figure 5.5: The unit structure in frame t could be transformed into the 4 shapes in frame t+1 illustrated on the right hand side. I only make use of the first two cases to infer the depth change.

off-image-plane rotation. Other shapes of the unit structure are possible, which may caused by either the combination of the above rigid motions or non-rigid motion. Here, I only look for the occurrences of the first two cases, and use them to perform depth correction.

In both cases that we consider, the unit structure is evenly scaling along all direction and the scaling factor can be calculated using the average length of the 4 edges. Figure 5.6 explains the relation between scaling factor and depth change using a single line segment. As line segment L moves closer to the camera, its projection size on the image plane changes from $l_1$ to $l_2$. Assuming that the unknown line segment's depth changes from $d_1$ to $d_2$, we have:

$$\left.\begin{aligned} l_1 &= \frac{f}{d_1}L \\ l_2 &= \frac{f}{d_2}L \end{aligned}\right\} \Longrightarrow s = \frac{l_2}{l_1} = \frac{d_1}{d_2} \tag{5.7}$$

where $f$ is the focal length, $s$ is the scaling factor of the line segment's image.

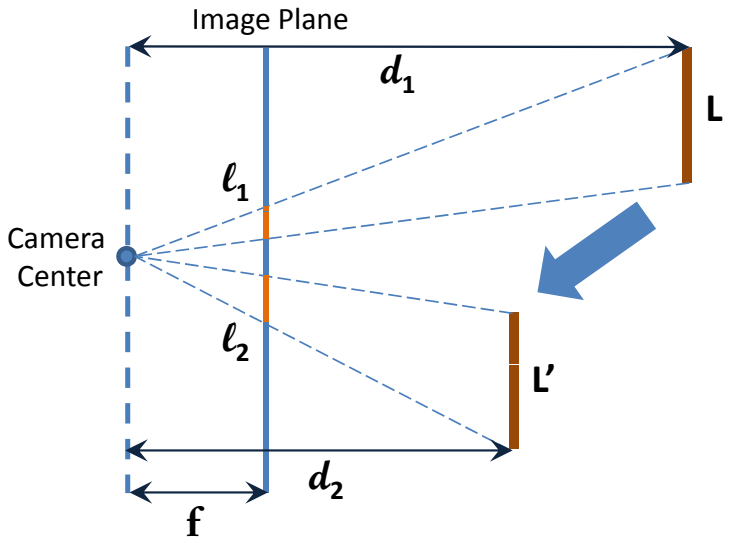


Figure 5.6: The length of the image of a line segment changes when it moves closer to the camera. The ratio of the image length $\frac{l_1}{l_2}$ is inverse proportional the ratio of the depth $\frac{d_1}{d_2}$.

Equation 5.7 tells us that the length of the line segment's projection is inversely proportional to the depth of the line segment's position. Therefore, by calculating the scaling factor of the line segment, we find out its relative depth change without knowing the absolute depth value.

The same property holds for the unit structure. Let $d^t(\mathbf{p})$ be the depth of the center pixel $\mathbf{p}$ of the unit structure in frame $t$, $d^{t+1}(\mathbf{p}')$ be the depth of the corresponding pixel $\mathbf{p}'$ in the next frame, and $s$ be the scaling factor of the unit structure, the following perspective depth correction constraint is included in the final quadratic programming formulation:

$$d^t(\mathbf{p}) - s \cdot d^{t+1}(\mathbf{p}') = 0 \tag{5.8}$$

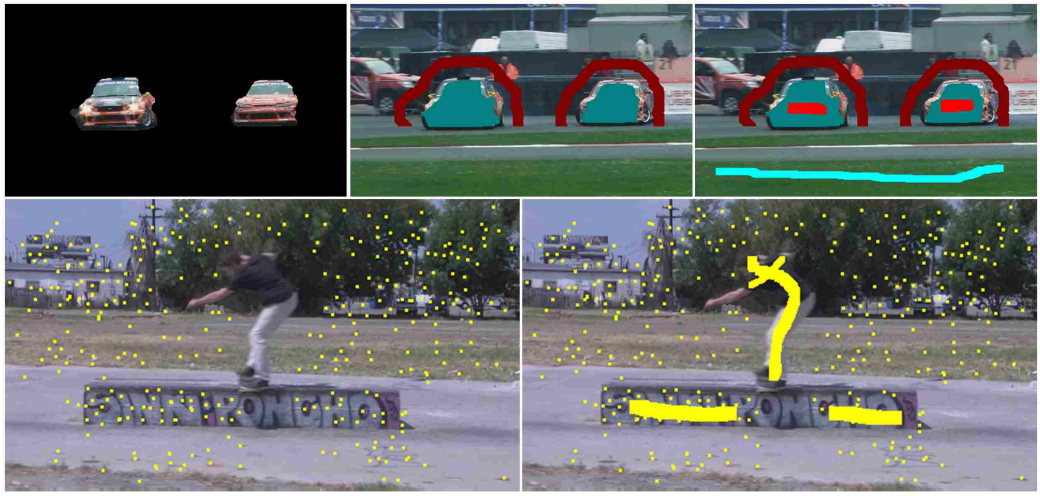

Figure 5.7: The segmented results (row 1, left image) is input in the user interface and treated as predefined depth difference constraints (row 1, middle image). The users need to use depth difference brush to indicate the depth variation in the rest part of the image (row 1, right image). Generally, cyan regions are closer than red regions and dark color scribblings are from segmentation while light color scribblings are from user interaction. The reconstructed 3D points are marked as yellow (row 2, left image) in the user interface, and the users need to assign the computed depth value to undefined regions (row 2, right image).

## 5.2 User Interaction

All those automatically computed output will be presented in the user interface and be marked as defined regions. The reconstructed 3D feature points will be highlighted in yellow on the input image frames (figure 5.7, row 2, left image). The segmented foreground/background objects will be marked as at distinct depth (figure 5.7, row 1, middle image). The effect of the perspective depth correction will not show up in the user interface, but directly input into the final quadratic programming formulation. With the assistance of defined regions, the users need to do some labeling work that points out the depth in the undefined regions.

I introduce two types of brushes: the depth difference brush and depth equivalence

brush. The former one is mainly intended to distinguish the surfaces that lie on different depth (figure 5.7, row 1, right image). And the latter one is employed to assign the recovered depth value to undefined regions (figure 5.7, row 2, right image). Both types of user inputs are are incorporated into the final solution through equality constraints. Let $\mathbf{p}_n$ and $\mathbf{p}_f$ be two pixels covered by the depth different brush, where $\mathbf{p}_n$ is inside the front area and $\mathbf{p}_f$ is inside the back area. Also assume that pixels $\mathbf{p}_i$ and $\mathbf{p}_j$ are two arbitrary pixels covered by the depth equivalence brush. The following equations will be added to the linear equations to constraint the depth of these pixels based on the user's input:

$$\begin{array}{rcl} d(\mathbf{p}_n) - d(\mathbf{p}_f) & \geq & \Delta d \\ d(\mathbf{p}_i) - d(\mathbf{p}_j) & = & 0 \end{array} \tag{5.9}$$

### 5.3 Depth Propagation

To avoid solving an extremely large problem, we down-sample the input image sequences by a factor of 4 in both dimension, and up-sample with the joint bilateral technique [277].

The propagation is based on the smoothness assumption that spatial and temporal neighboring pixels should have the same depth value if they share the same color. For each pixel $\mathbf{p}$ in the image sequence, we have

$$d(\mathbf{p}) - \sum_{\mathbf{q} \in N_{\mathbf{p}}} w_{\mathbf{pq}} d(\mathbf{q}) = 0 \tag{5.10}$$

where $d(\mathbf{p})$ is the depth value of pixel $\mathbf{p}$ and $w_{\mathbf{pq}}$ is the normalized weight between

pixel $\mathbf{p}$ and $\mathbf{q}$, which is inversely proportional to the color difference of these two pixels. $N_{\mathbf{p}}$ is a set consisting of $\mathbf{p}$'s 8 spatial neighbors in the current frame and 1 temporal neighbor in the next frame, which is located using optical flow. The equation 5.10 for each pixel is stacked as a large sparse linear equation $Ax = 0$. where $A$ is the weight matrix and x is the depth vector defined on every pixel.

The equality constraints from the SFM (equation 5.7), perspective depth correction (equation 5.8) and user's depth equivalence brush (equation 5.9) are stacked as a linear equation $Cx = d$.

And the inequality constraints from the moving object extraction (equation 5.6) and user's depth different brush (equation 5.9) are stacked as a linear inequality equation $Ex \geq f$.

Therefore, the propagation is formulated as the following optimization problem:

$$\min_x \|Ax\|_2^2 \qquad s.t. \quad Cx = d, Ex \geq f \tag{5.11}$$

It is equivalent to solve

$$\min_x (x^T A^T A x) \quad s.t. \quad Cx = d, Ex \geq f \tag{5.12}$$

This is a standard quadratic programming formulation. The solution can be found using the MOSEK optimization toolbox, which can deal with sparse large-scale problems.

Figure 5.8: The results obtained by equal brush labeling along with the assistance from SFM. The $1^{st}$ row shows the user labeling on the first and last frames. The $2^{nd}$ row shows some frames from the generated stereo video. The $3^{rd}$ row are corresponding depth maps.

## 5.4 Results

I tested my system on a variety of video shots under different camera motions. I first demonstrate that the output from the three automatic preprocessing algorithms (SFM, moving object extraction and perspective depth correction) do help the users in the labeling step.

In the case of figure 5.8, the background scene is reconstructed by SFM, and the only undefined object/area is the moving person. By observing that the person is standing upon the rail, the user can simply assign the depth of the rail to the person by applying equal brush on the first and last frames. The depth value is propagated to in-between frames by temporal smoothness constraints.

Figure 5.11 shows the results of moving object extraction. This is a typical follow shot in which the subjects are followed by a panning camera. In this case, SFM fails due to the lack of parallax. However, the moving object extraction algorithm can

Figure 5.9: The effectiveness of perspective depth correction algorithm. The two images on the $1^{st}$ row are the first and last frame from a video. The left image on the $2^{nd}$ row is the depth map of the first frame and the middle and right images are depth maps of the last frame without and with the perspective depth correction respectively.

easily segment out the cars from the constantly changing background audiences and foreground lawns. The results are input to the user interface as the depth difference constraints. In particular, we regard the segmented object to be in front of the area above and to the both sides of it, under observation that most follow shots are following objects on the ground. The remaining work to the users is differentiating the depth between the foreground lawn and the cars (figure 5.11, row 3).

Figure 5.9 shows the effectiveness of perspective depth correction. If the algorithm is not applied on the input video, the depth value will be propagated from the first frame to the last frame under the smoothness constraints defined on spatial and temporal neighbors. Thus, the same depth value on the truck in the first frame is passed onto the last frame, which is obviously invalid. Whereas, the perspective depth correction algorithm captures the depth change and produces correct depth value in the last frame.

My system mainly relies on motion to produce additional information that facilitates the user interaction. However, if the three automatic subsystems cannot do anything in the first place, the two brushes: depth difference brush and depth equality brush can still produce similar 3D effect as does the Guttmann's method. For example in figure 5.10, the scene, consisting of nearly static subjects, is captured by a stationary camera. Guttmann's method directly assigns different depth to different layers of the scene. My method can accomplish the same effect by applying the depth difference brush a couple of times, in order to point out pairwise layers that lie on different depth.

All the shown stereo footage in this section as well as in previous sections can be found in the accompanied video. Please wear a pair of red/cyan 3D glasses to watch them.

## 5.5 User Study

To further compare the my system with the one proposed by Guttmann et al. [2], two user studies are conducted. In the first study, the users are asked to convert videos to 3D using both systems. The qualities of the 3D videos generated by these users are evaluated in the second user study. Based on the design goal of my system, I hypothesize the following:

**H1**: Participants using my system will be able to complete the tasks quicker than using the Guttmann's method.

**H2**: Participants will find it easier to label the relative depths of different objects than labeling the disparity values.

Figure 5.10: The scene of stationary camera and static objects will fail the three algorithms in the preprocessing. However, the two labeling brushes: depth difference brush and depth equivalence brush can still produce similar visual 3D effect as does Guttmann's method. Guttmann's method (left column) assigns different depth (color coded) to different layers of the scene. My method (right column) achieves the same effect by applying the depth difference brush twice. The dark red&cyan scribbling pair indicate the depth difference of the old lady and the young man. And the light red&cyan pair point out the difference between the young man and the background. The results from both methods are visually comparable.

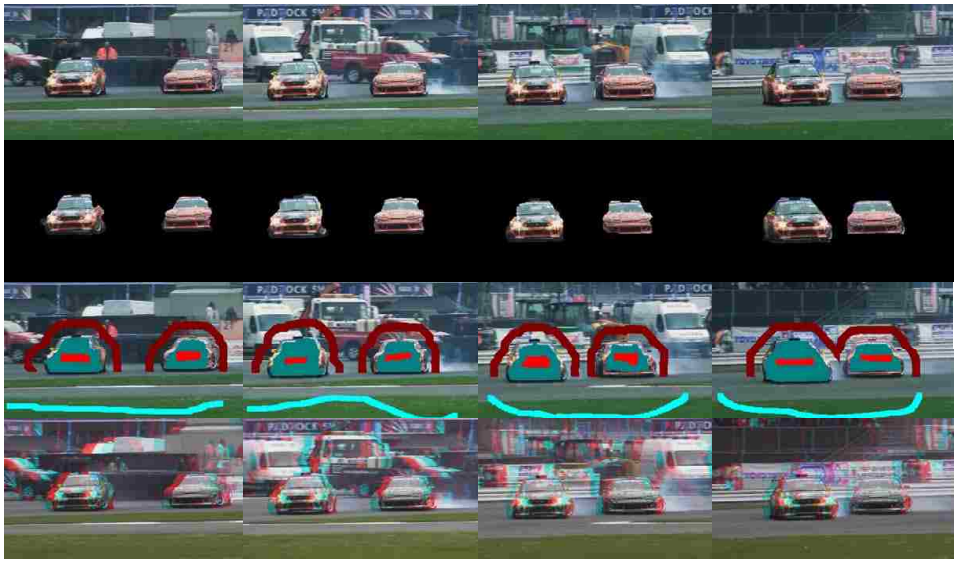

Figure 5.11: The moving object extraction results and its integration into the user interface. The $1^{st}$ and $2^{nd}$ rows show the original images and segmented result. the $3^{rd}$ row shows the look of the user interaction. The dark red&cyan pairs are automatically generated from the segmentation results and the light red&cyan pairs are marked by user. The $4^{th}$ row shows the stereo images from the video.

**H3**: When given a choice of which system to use for future label tasks, participants will prefer my system.

**H4**: The improvement of my system over Guttmann's method is more noticeable when the input videos contain more motion.

### 5.5.1 System Usability Study

This user study is conducted in a controlled setting using a $2 \times 4$ (system $\times$ labeling task) between-subjects design. Each participant performs each of the four labeling tasks only once, two of which using my system and the other two using Guttmann's method. To further alleviate potential learning effects, a Graeco-Latin square was used to vary the order of exposure to the interface and the order of the task assignment. Prior to performing any of the tasks, participants were given a chance to play with both systems.

For each task, measurements of time to task completion and subjective user opinion were made. Pre-study questionnaires were administered to determine prior experience with computer and image/video editing. Post-study questionnaires measures perceived usefulness and ease-of-use, along with an indication of preference.

#### Participant Demographics

Twenty individuals were recruited from the CS graduate student population at two universities to participate in this study. Their response to the pre-study questionnaires shows that all of them are familiar with image/video editing, but do not have prior knowledge about 3D video conversion.

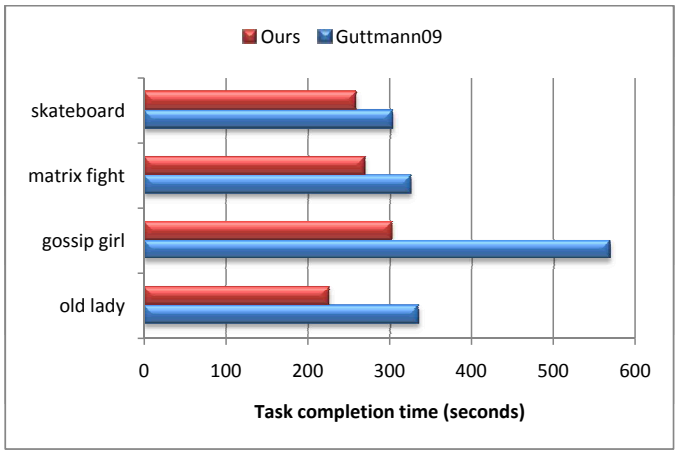


Figure 5.12: The average time that the users took under different systems.

### Time to Task Completion

The average time required to complete the four labeling tasks with the two system are illustrated in Figure 5.12. It shows that, for all four datesets, the users took less time when using my system to label the scene for 3D conversion. ANOVA tests preformed on these measurements confirm that the timing differences are statistical significant ($\mathbf{F(1, 72) = 8.83, p < 0.01}$). This finding confirms the first hypothesis (H1).

### Subjective Reactions

In the post-study questionnaires, participants were asked to indicate their degree of agreement to statements related to the usability of the two systems (using a five-point Likert scale). As shown in Table 5.1, users generally find it difficult to label the depth directly, which is required by Guttmann et al.'s method. Majority users find labeling the relative depth easy and almost all users find my interface makes the 3D conversion task easier (H2). In addition, when asked "which UI would you prefer for future labeling tasks?", all participants selected my UI (H3).

Table 5.1: Users' response to the post-study questionnaires, where $UI_A$ refers to Guttmann et al.'s method and $UI_B$ refers to my approach

| Statements | Strongly disagree | disagree | neutral | agree | strongly agree |
|---|---|---|---|---|---|
| It's easy to label the depth values directly using $UI_A$ | 3 | 10 | 3 | 4 | 0 |
| It's easy to label the depth ordering using $UI_B$ | 1 | 0 | 0 | 8 | 11 |
| The labeling task using $UI_B$ is easier than using $UI_A$ | 1 | 0 | 1 | 6 | 12 |

### 5.5.2 Result Quality Study

To remove the bias that a user may have toward a given system, the result quality evaluation is performed in a separate user study. In this step, all 3D videos generated in the previous study are shuffled and shown to the users. The users are asked to assign a score to each 3D video, without knowing who converted the video and which system was used for the conversion. The score ranges from 0 to 4, with '0' being the poorest and '4' being the best.

The average scores for the 3D videos generated for each dataset using each system are illustrated in Figure 5.13. It shows that, for "skateboard", "matrix fight", and "gossip girl" datesets, the users find the results generated by my system to be much better than the ones generated by Guttmann's method. ANOVA tests confirm that the differences are statistical significant for all three datasets ($\mathbf{F(1, 254) = 114, p < 0.01}$ for "skateboard"; $\mathbf{F(1, 254) = 211, p < 0.01}$ for "fight"; and $\mathbf{F(1, 254) = 8.74, p < 0.01}$ for "gossip girl"). the "old lady" dataset, the average score for the video generated using the Guttmann's method is slightly higher, but the difference is statistically

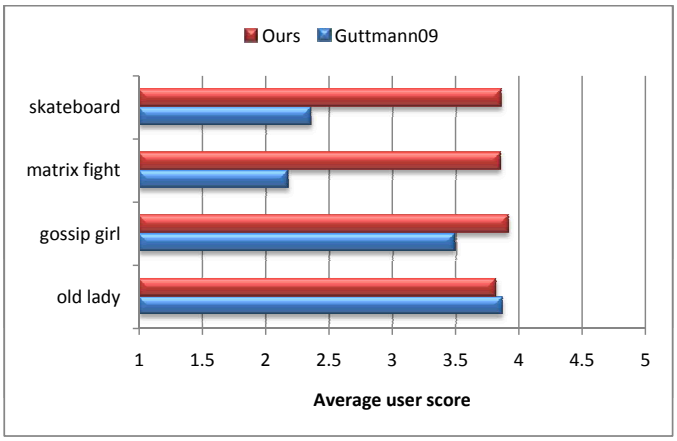


Figure 5.13: The average score that the users gave to videos generated by different systems.

insignificant in $(\mathbf{F(1, 254) = 0.155, p > 0.5})$. This suggests that, when the scene contains little motion, the result qualities of both systems rely solely on the users' input. However, when the scene contains complex motion, the depth labels provided by the users become imprecise and the advantages of automatic pre-processing processes start to emerge (H4).

## 5.6 Conclusion

This chapter presented a hybrid framework to semi-automatically convert conventional videos to stereoscopic videos. My system tries to make full use of the available motion information so that less user interaction is required. Compared to the previous methods [2], the major novelty of my framework lies in the utilization of motion prior analysis, such as automatic moving object extraction and perspective depth correction, to deal with arbitrary camera/object movement. In addition, I also developed a new user interface that requires users to specify relative depth orders with the help of pre-computed 3D visual cues, instead of labeling the depth value directly.

Finally, the depth information automatically extracted by computer algorithms and the interactively labeled by users are propagated to the whole video sequence through solving a quadratic programming problem. As demonstrated in the results and confirmed by the user study, my system is more user-friendly, as well as produces better quality of stereo videos for scenes with complex motions.

Looking into the future, I am planning to automatically cut a video into multiple shots and try to utilize coherence among non-adjacent shots for the same scene. Moreover, I am interested in automatically categorizing the video sequences so that the computer can decide which visual cues can be utilized. How to explore other visual cues for automating the video conversion process is also worth investigating.

## Chapter 6 Conclusion and Future Work

I explore the possibilities of accomplishing traditional computer vision tasks with a single camera. I explored how to recover 3D information from a single view, in order to enable the 3D re-experiencing of recorded scenes. My work consists of three parts:

- Light fall-off stereo depth camera
- 3D Modeling of dynamic and deformable objects
- 2D-3D video conversion

For the 3D modeling, the range data can be either obtained by a stereo camera pair or by a general depth sensor. I actually designed and developed a depth camera based on a novel technique called light falloff stereo (LFS). LFS depth camera produces color+depth images sequences and achieves 30 fps, which is necessary for capturing dynamic scenes. The complete 3D model is reconstructed by aligning surfaces captured at different times. The alignment involves mesh deformation guided by feature correspondences.

For video conversion, I developed a semi-automatic system that converts conventional videos into stereoscopic videos by combining motion analysis with user interaction, aiming to transfer as much as possible labeling work from the user to the computer. I further developed a quadratic programming approach to incorporate both quantitative depth and qualitative depth (such as those from user scribbling),

in order to recover dense depth maps for all frames, from which a stereoscopic view can be synthesized.

## 6.1 Innovations

The general contribution of the presented research is simplifying the data acquisition process for those computer vision tasks by using only a single camera. Modeling deformable and dynamic objects is usually achieved by using an array of synchronized cameras that can see different sides of an object simultaneously. Single view modeling saves people from the tedious synchronization and calibration processes, while still producing plausible results. On the other hand, the view synthesis framework can turn a traditional 2D movie into a 3D movie with limited user interaction. The technical contributions are listed as follows:

**The theory of Light Fall-off Stereo** I developed a novel way to estimate depth information from scenes beyond Lambertian reflectance model. I also developed a global optimization-based method that uses multiple light variations to further improve the accuracy and robustness. The effectiveness of LFS is demonstrated with a variety of real-world scenes exhibiting complex reflectance and geometries.

**The Design of Real-time LFS Camera** I developed a novel depth range system that can generate a VGA (640x480) resolution depth map at 30Hz. In order to toggle between two LED lights in a fast and accurate way, I designed a dedicated circuit to control the state of the LED lights and receive synchronization signals

with the camera. I also immigrated the whole computation of the depth map to GPU, achieving real-time performance. In terms of quantitative accuracy, my system compares favorably to other commercial 3D range sensors, particularly in textured areas. In addition, my system is made of commodity off-the-shelf components, offering an inexpensive solution to real-time, high-resolution, video-rate range sensing.

**The Alignment of Deformable Surfaces** The alignment between surfaces of deformable and dynamic objects captured at different times need not only the rigid transformation [16], but also the laplacian surface deformation [17] controlled by feature points. Laplacian surface deformation ensures that the surface shape is unchanged as much as possible, while at the same time, the control points continue to coincide with their destination positions. However, previous mesh deformation techniques only address the issue of deforming one mesh at a time, which means that we can only sequentially stitch different pieces of surface to reconstruct a complete 3D model. In most cases, sequential alignment may lead to misalignment between the first and the last surface, due to accumulative errors. I modified the traditional mesh deformation in order to align all pieces of surfaces globally and simultaneously, so that a complete model can be obtained. This research work is already published in ICCV 2009 [18].

**Motion Analysis for 2D-3D Video Conversion** I developed two novel techniques that automatically estimate the 3D information from video sequences. Unlike SFM that requires non-axis camera movement (e.g., dolly, crane), my techniques can

work with arbitrary camera/object movement, such as camera pan or zoom, which are frequently used in both everyday video and professional shots.

**Intuitive User Interface** I provided a user-friendly interface that requires users to label depth relationship rather than depth value on the images. My UI design benefits from the already defined 3D cues by the pre-processing of movement, providing users with a more intuitive and less labor intensive UI environment. In the case that none of the 3D cues can be inferred in the pre-processing step, my labeling can still simulate the direct depth labeling with the same amount of manual work.

**Quadratic Programming Formulation for Depth Propagation** I formulate the sparse to dense depth propagation as a quadratic programming problem, that could elegantly integrate both relative (such as layer orders) and absolute (such as that from SFM) depth constraints.

## 6.2 Future Work

As I stated before, the ultimate goal of my dissertation work is to make tomorrow's 3D building as easy as today's photo shooting. My work is two steps towards this goal, even if many more efforts are needed to really reach it. Of course, my attempts here are not the unique ways to this goal. I believe there are various possible directions that are worth exploration. Here are some visions for the future.

**More Reliable Depth Camera** Existing depth cameras are mainly designed for indoor environments. Because most of them rely on active light to detect scene depth,

it fails at capturing the complex lighting conditions in the outdoor environment. For instance, my LFS depth sensor cannot work in outdoor environments because it cannot deal with the ambient light.

Another reason that current depth cameras are not suitable for outdoor capture is that they have fixed focal length. In other words, they have a fixed working distance. For example, some depth cameras are designed to capture human size objects. So it is impossible to use them to capture smaller or bigger objects, such as a miniature car model or a true building. It is very interesting to mak depth cameras work like regular zoom lens cameras, so that they can capture objects within a certain scale range.

The final thing that prevents depth cameras from becoming popular is that the output of the depth cameras is not standardized. Some depth cameras output depth maps and color images, while some of them output text files that record the depth information. I believe there will be more and more demands in the future to create output from the depth cameras that is unified and displayable, just as the images output by a regular camera.

In order to make my proposed 4D reconstruction method more practical, a more common and reliable depth camera is a must.

**Textured Models** Currently, the reconstructed 3D models are not textured. Although a color image is aligned with each captured piecewise surface, it is not easy to stitch together the texture in the images. This is due to the misalignment between the surfaces. A small amount of misalignment between surfaces will cause the loss of

details of the surface shape. Since I am using volumetric method to fuse the aligned surfaces, the final extracted surface will be visually acceptable even if it is distorted by a small amount of misalignment.

However, it is not the case for texture. The misalignment between two surfaces will also cause the misalignment of the texture on the common part between these two surfaces. And the misalignment of the texture will blur out the final result. Compared to the other parts, the blurring of the texture on the shared parts is quite obvious; therefore I have decided not to present the texture on the reconstructed model.

In order the re-explore the captured scenes in interactive 3D, texture has to be added back onto the reconstructed models. Maybe another option is to add the texture at the image level, and then put it onto the complete 3D model, like dressing it up with cloth.

**More Sophisticated Global Method** Currently the global alignment is formulated as a big linear equation. Although it looks simple and intuitive, it is not easy to solve numerically when the equation becomes too big. For example, if we are dealing with 20 surfaces, each of which has about 5000 vertices, the final linear system is about 100,000 x 100,000. In some cases, I will have to deal with many more surfaces and vertices. When the matrix gets bigger, the condition number is likely to get bigger. Plus, it is not always easy to directly solve linear equations at this scale, even using the most sophisticated linear algebra library. This big of a matrix will also cause memory issues, even though we are dealing with a sparse matrix. An iterative

approach might be a solution to this problem, but most of time this cannot guarantee a quick convergence.

Therefore, it will be worth trying a more sophisticated global method, in order to avoid solving big linear equations. Though I am dealing with deformable surfaces in this project, the core idea of distributing the alignment errors evenly across all surfaces is the same as that for rigid surfaces. So, it is interesting to explore those global methods introduced in section 2.2.3 of chapter 2.

This could also be tried for the the depth propagation in the 2D-3D conversion formulation. Because the quadratic programming is turned into linear equations, ultimately, we are still facing the problem of solving big linear equations.

**Automatic Detection of Different Shots** In the 2D-3D video conversion project, the success of my automatic processing modules depends on the camera/object motion. For example, with a dolly shot, SFM will work very well; on the other hand, in a pan or follow-shot in sport video, the automatic segmentation method is effective. How to automatically detect these shot/motion types and apply the appropriate motion analysis tool is an interesting topic in its own right. For the scope of this thesis, I let the user decide which tool to use.

Looking to the future, it would be very useful to automatically cut a video into multiple shots and try to utilize coherence among non-adjacent shots for the same scene. Moreover, I am also interested in automatically categorizing the video sequences so that the computer can decide which visual cues can be utilized. How to explore other visual cues for automating the video conversion process is also worth

investigating.

**Quantitative Evaluations** In this thesis, both the 4D reconstruction and the 2D-3D video conversion only require the results to be visually reasonable, not physically correct. Because both of them are ill-posed problems, and they are based on heuristics. However, it may be quite interesting to quantitatively evaluate my results by comparing it to ground truth.

For the 4D reconstruction project, we could use multiple surrounding cameras to capture the ground truth, and then use the output of one of the cameras to build 3D models. For the 2D-3D video conversion project, we could use a stereo camcorder, such as the Fujifilm FinePix Real 3D camera, to capture a 3D video clip. Then we could input the clip from one view into the system to see the results.

## Bibliography


[1] J. J. Koenderink, A. J. van Doorn, A. M. Kappers, and J. T. Todd. Ambiguity and the 'mental eye' in pictorial relief. In *Perception*, 2001.

[2] Moshe Guttmann, Lior Wolf, and Daniel Cohen-Or. Semi-automatic stereo extraction from video footage. In *ICCV*, 2009.

[3] M. Pollefeys, R. Koch, and L. Van Gool. Self-Calibration and Metric Reconstruction in spite of Varying and Unknown Internal Camera Parameters. In *ICCV*, pages 90–95, 1998.

[4] Michael Goesele, Noah Snavely, Brian Curless, Hugues Hoppe, and Steven M. Seitz. Multi-view stereo for community photo collections. In *Proceedings of ICCV*, 2007.

[5] T. Kanade, P. Rander, S. Vedula, and H. Saito. Virtualized reality: Digitizing a 3d time-varying event as is and in real time. In Yuichi Ohta and Hideyuki Tamura, editors, *Mixed Reality, Merging Real and Virtual Worlds*, pages 41–57. Springer-Verlag, 1999.

[6] C. Lawrence Zitnick, Sing Bing Kang, Matthew Uyttendaele, Simon Winder, and Richard Szeliski. High-quality video view interpolation using a layered representation. *ACM Transactions on Graphics,*, 23(3):600–608, 2004.

[7] C. Tomasi and T. Kanade. Shape and Motion from Image Streams under Orthography: A Factorization Approach. *IJCV*, 9(2):137–154, 1992.

[8] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2000.

[9] Christoph Bregler, Aaron Hertzmann, and Henning Biermann. Recovering non-rigid 3d shape from image streams. In *Proceedings of CVPR*, pages 690–696, 2000.

[10] Lorenzo Torresani, Aaron Hertzmann, and Chris Bregler. Learning non-rigid 3d shape from 2d motion. In *In proceedings of NIPS*, 2003.

[11] Mario Botsch and Olga Sorkine. On linear variational surface deformation methods. In *Transaction on visualization and Computer Graphics*. IEEE, 2008.

[12] Edilson de Aguiar, Carsten Stoll, Christian Theobalt, Naveed Ahmed, Hans-Peter Seidel, and Sebastian Thrun. Performance capture from sparse multi-view video. In *Siggraph*. ACM, 2008.

[13] Guofeng Zhang, Jiaya Jia, Tien-Tsin Wong, and Hujun Bao. Consistent depth maps recovery from a video sequence. *Transactions on Pattern Analysis and Machine Intelligence*, 31(6), 2009.

[14] Sotiris Diplaris, Nikos Grammalidis, Dimitris Tzovaras, and Michael G. Strintzis. Generation of stereoscopic image sequences using structure and rigid motion estimation by extended kalman filters. *ICME*, 2, 2002.

[15] Konstantinos Moustakas, Dimitrios Tzovaras, and Michael G. Strintzis. Stereoscopic video generation based on efficient layered structure and motion estimation from a monoscopic image sequence. *Transactions on Circuits and Systems for Video Technology*, 15(8), 2005.

[16] B. K. Horn. Colosed-form solution of absolute orientation using unit quaternions. In *Journal of the Optical Society of America*, 1987.

[17] O. Sorkine, D. Cohen-Or, Y. Lipman, M. Alexa, C. Rossl, and H.-P. Seidel. Laplacian surface editing. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, 2004.

[18] Miao Liao, Qing Zhang, Huamin Wang, Ruigang Yang, and Minglun Gong. Modeling deformable objects from a single depth camera. In *International Conference on Computer Vision*, 2009.

[19] Paul E. Debevec, Camillo J. Taylor, and Jitendra Malik. Modeling and rendering architecture fromphotographs: A hybrid geometry- and image-based approach. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, pages 11–20, 1996.

[20] M. Pollefeys and L. Van Gool. A Stratified Approach to Self-calibration. In *Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 407–412. IEEE Computer Society Press, 1997.

[21] David Liebowitz and Andrew Zisserman. Combining Scene and Auto-Calibration Constraints. In *Proceedings of International Conference on Computer Vision (ICCV)*, pages 293–300, 1999.

[22] D. Marr and T. Poggio. Cooperative Computation of Stereo Disparity. *Science*, 194:283–287, 1976.

[23] S.T. Barnard and M.A. Fischler. Computational Stereo. *Computer Surveys*, 14(4):553–572, 1982.

[24] U. Dhond and J. Aggrawal. Structure from stereo: a review. *IEEE Transactions on Systems, Man, and Cybernetics*, 19(6):1489–1510, 1989.

[25] D. Scharstein and R. Szeliski. A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms. *IJCV*, 47(1):7–42, May 2002.

[26] D. Marr. *Vision*. W. H. Freeman and Company, 1982.

[27] N. Ayache and C. Hansen. Rectification of Images for Binocular and Trinocular Stereovision. In *Proceedings of International Conference on Pattern Recognition*, pages 11–16, 1988.

[28] D. Papadimitriou and T. Dennis. Epipolar line estimation and rectification for stereo image pairs. *IEEE Transactions on Image Processing*, 5(4):672–676, 1996.

[29] C. Loop and Z. Zhang. Computing Rectifying Homographies for Stereo Vision. In *Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 125–131, 1999.

[30] M. Pollefeys, R. Koch, and L. Van Gool. A Simple and Efficient Rectification Method for General Motion. In *Proceedings of International Conference on Computer Vision (ICCV)*, pages 496–501, Corfu, Greece, 1999.

[31] M. J. Hannah. *Computer Matching of Areas in Stereo Images*. PhD thesis, Stanford University, 1974.

[32] P. Anandan. A Computational Framework and an Algorithm for the Measurement of Visual Motion. *International Journal of Computer Vision (IJCV)*, 2(3):283–310, 1989.

[33] L. Matthies, R. Szeliski, and T. Kanade. Kalman Filter-based Algorithms for Estimating Depth from Image Sequences. *International Journal of Computer Vision (IJCV)*, 3:209–236, 1989.

[34] T. Kanade. Development of a Video-rate Stereo Machine. In *DARPA Image Understanding Workshop*, page 549C557, Monterey, CA, 1994. Morgan Kaufmann Publishers.

[35] T. W. Ryan, R. T. Gray, and B. R. Hunt. Prediction of Correlation Errors in Stereo-pair Images. *Optical Engineering*, 19(3):312–322, 1980.

[36] R. C. Bolles, H. H. Baker, and M. J. Hannah. The JISCT Stereo Evaluation. In *DARPA Image Understanding Workshop*, pages 263–274, 1993.

[37] P. Seitz. Using Local Orientation Information as Image Primitive for Robust Object Recognition. *SPIE Visual Communications and Image Processing IV*, 1199:1630C1639, 1989.

[38] D. Scharstein. Matching Images by Comparing Their Gradient Fields. In *Proceedings of International Conference on Pattern Recognition (ICPR)*, volume 1, pages 572–575, 1994.

[39] R. Zabih and J. Woodfill. Non-parametric Local Transforms for Computing Visual Correspondence. In *Proceedings of European Conference on Computer Vision (ECCV)*, page 151C158, 1994.

[40] M. J. Black and P. Anandan. A Framework for the Robust Estimation of Optical Flow. In *Proceedings of International Conference on Computer Vision (ICCV)*, pages 231–236, 1993.

[41] M. J. Black and A. Rangarajan. On the Unification of Line Processes, Outlier Rejection, and Robust Statistics with Applications in Early Vision. *International Journal of Computer Vision (IJCV)*, 19(1):57–91, 1996.

[42] D. Scharstein and R. Szeliski. Stereo Matching with Nonlinear Diffusion. *International Journal of Computer Vision (IJCV)*, 28(2):155–174, 1998.

[43] S. Birchfield and C. Tomasi. A Pixel Dissimilarity Measure That Is Insensitive to Image Sampling. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 20(4):401–406, 1998.

[44] R. D. Arnold. Automated Stereo Perception. Technical Report AIM-351, Artificial Intelligence Laboratory, Stanford University, 1983.

[45] A. F. Bobick and S. S. Intille. Large occlusion stereo. *International Journal of Computer Vision (IJCV)*, 33(3):181–200, 1999.

[46] M. Okutomi and T. Kanade. A Locally Adaptive Window for Signal Matching. *International Journal of Computer Vision (IJCV)*, 7(2):143–162, 1992.

[47] T. Kanade and M. Okutomi. A Stereo Matching Algorithm with an Adaptive Window: Theory and Experiment. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(9):920 – 932, September 1994.

[48] S. B. Kang, R. Szeliski, and J. Chai. Handling Occlusions in Dense Multi-view Stereo. In *Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR)*, 2001.

[49] O.Veksler. Stereo Matching by Compact Windows via Minimum Ratio Cycle. In *Proceedings of International Conference on Computer Vision (ICCV)*, pages 540–547, 2001.

[50] Y. Boykov, O. Veksler, and R. Zabih. A Variable Window Approach to Early Vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(12), December 1998.

[51] K. Prazdny. Detection of Binocular Disparities. *Biological Cybernetics*, 52(2):93–99, 1985.

[52] W. E. L. Grimson. Computational experiments with a feature based stereo algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 7(1):17–34, 1985.

[53] R. Szeliski and G. Hinton. Solving Random-dot Stereograms Using the Heat Equation. In *Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 284–288, 1985.

[54] J. Shah. A Nonlinear Diffusion Model for Discontinuous Disparity and Half-occlusion in Stereo. In *Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 34–40, 1993.

[55] Tomaso Poggio, Vincent Torre, and Christof Koch. Computational Vision and Regularization Theory. *Nature*, 317(314-319), 1985.

[56] A. Blake and A. Zisserman. *Visual Reconstruction*. MIT Press, 1987.

[57] S. Geman and D. Geman. Stochastic Relaxation, Gibbs Distribution, and the Bayesian Restoration of Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 6(6):721–741, 1984.

[58] J. Marroquin, S. Mitter, and T. Poggio. Probabilistic Solution of Ill-posed Problems in Computational Vision. *Journal of the American Statistical Association*, 82(397):76–89, 1987.

[59] S. T. Barnard. Stochastic Stereo Matching over Scale. *International Journal of Computer Vision (IJCV)*, 3(1):17–32, 1989.

[60] P. B. Chou and C. M. Brown. The Theory and Practice of Bayesian Image Labeling. *International Journal of Computer Vision (IJCV)*, 4(3):185–210, 1990.

[61] D. Geiger and F. Girosi. Parallel and Deterministic Algorithms for MRF's: Surface Reconstruction. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 6(6):721–741, 1984.

[62] S. Roy and I. J. Cox. A Maximum-flow Formulation of the N-camera Stereo Correspondence Problem. In *Proceedings of International Conference on Computer Vision (ICCV)*, pages 492–499, 1998.

[63] H. Ishikawa and D. Geiger. Occlusions, Discontinuities, and Epipolar Lines in Stereo. In *Proceedings of European Conference on Computer Vision (ECCV)*, pages 232–248, 1998.

[64] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 23(11):1222–1239, 2001.

[65] O. Veksler. *Efficient Graph-based Energy Minimization Methods in Computer Vision*. PhD thesis, Cornell University, 1999.

[66] V. Kolmogorov and R. Zabih. Computing Visual Correspondence with Occlusions Using Graph Cuts. In *Proceedings of International Conference on Computer Vision (ICCV)*, pages 508–515, 2001.

[67] C. Buehler, S. J. Gortler, M. Cohen, and L. McMillan. Minimal Surfaces for Stereo Vision. In *Proceedings of European Conference on Computer Vision (ECCV)*, pages 885–899, 2002.

[68] Vladimir Kolmogorov and Ramin Zabih. What Energy Functions Can Be Minimized via Graph Cuts? In *Proceedings of European Conference on Computer Vision (ECCV)*, pages 65–81, 2002.

[69] P. N. Belhumeur. A Bayesian Approach to Binocular Stereopsis. *International Journal of Computer Vision (IJCV)*, 19(3):237–260, 1996.

[70] P. N. Belhumeur and D. Mumford. A Bayesian Treatment of the Stereo Correspondence Problem Using Half-occluded Regions. In *Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 506–512, 1992.

[71] D. Geiger, B. Ladendorf, and A. Yuille. Occlusions and Binocular Stereo. In *Proceedings of European Conference on Computer Vision (ECCV)*, pages 425–433, 1992.

[72] I. J. Cox, S. L. Hingorani, S. B. Rao, and B. M. Maggs. A Maximum Likelihood Stereo Algorithm. *Computer Vision and Image Understanding (CVIU)*, 63(3):542–567, 1996.

[73] Bruce D. Lucas and Takeo Kanade. An Iterative Image Registration Technique with an Application to Stereo Vision. In *Proceedings of International Joint Conference on Artificial Intelligence*, pages 674–679, 1981.

[74] D. Laidlaw, W. Trumbore, and John F. Hughes. Constructive solid geometry for polyhedral objects. *ACM Computer Graphics (SIGGRAPH)*, 20(4):161–170, 1986.

[75] T. Kanade and M. Okutomi. A Stereo Matching Algorithm with an Adaptive Window: Theory and Experiment. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 16(9):920–932, 1994.

[76] J. Mulligan, V. Isler, and K. Daniilidis. Trinocular Stereo: A New Algorithm and its Evaluation. *International Journal of Computer Vision (IJCV), Special Issue on Stereo and Multi-baseline Vision*, 47:51–61, 2002.

[77] D. Scharstein. View Synthesis Using Stereo Vision. *Lecture Notes in Computer Science (LNCS)*, 1583, 1999.

[78] Ronen Basri. On the uniqueness of correspondence under orthographic and perspective projections. In *Proceeding of Image Understanding Workshop*, pages 875–884, 1992.

[79] M. Okutomi and T. Kanade. A Multiple-baseline Stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 15(4):353–363, 1993.

[80] R. Collins. A Space-Sweep Approach to True Multi-Image Matching. In *Proceedings of Conference on Computer Vision and Pattern Recognition*, pages 358–363, June 1996.

[81] Andrew P. Witkin. Recovering surface shape and orientation from texture. *Artificial Intelligence*, 17(1-3):17 – 45, 1981.

[82] A. Blake and C. Marinos. Shape from texture: Estimation, isotropy and moments. 45(3):323–380, October 1990.

[83] D.A. Forsyth. Shape from texture without boundaries. In *In Proc. ECCV*, pages 225–239, 2002.

[84] Jonas Garding. Shape from texture for smooth curved surfaces in perspective projection. *Journal of Mathematical Imaging and Vision*, 2:630–638, 1992.

[85] Jitendra Malik and Ruth Rosenholtz. Computing local surface orientation and shape from texture for curved surfaces, 1997.

[86] Maureen Clerc and Stphane Mallat. Shape from texture through deformations. In *In Proc. 7th Int. Conf. on Computer Vision*, pages 405–410, 1999.

[87] Vilayanur S. Ramachandran. Perceiving shape from shading, 1988.

[88] Harry G. Barrow and J. M. Tenenbaum. *Retrospective on "interpreting line drawings as three-dimensional surfaces"*, pages 71–80. MIT Press, Cambridge, MA, USA, 1994.

[89] E Mingolla and J T Todd. Perception of solid shape from shading. *Biol. Cybern.*, 53:137–152, January 1986.

[90] Berthold K. P. Horn, Richard S. Szeliski, and Alan L. Yuille. Impossible shaded images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(2):15–2, 1993.

[91] Katsushi Ikeuchi and Berthold K. P. Horn. Numerical shape from shading and occluding boundaries, 1981.

[92] B.K.P. Horn and M.J. Brooks. Shape and source from shading. In *International Joint Conference on Artificial Intelligence*, pages 932–936, 1985.

[93] Robert T. Frankot, Rama Chellappa, and Senior Member. A method for enforcing integrability in shape from shading algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10:439–451, 1988.

[94] B.K.P. Horn. Height and gradient from shading. *IJCV*, 5(1):37–76, 1990.

[95] Richard Szeliski. Fast shape from shading. *Computer Vision Graphics and Image Processing: Image Underst.*, 53(2):129–153, 1991.

[96] Omar E. Vaga and Yee-Hong Yang. Shading logic: A heuristic approach to recover shape from shading. *IEEE Trans. Pattern Anal. Mach. Intell.*

[97] Q. Zheng and R. Chellappa. Estimation of illuminant direction, albedo, and shape from shading. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 13(680-702):1222–1239, 1991.

[98] B. K.P. Horn. Shape from shading: A method for obtaining the shape of a smooth opaque object from one view. Technical report, Cambridge, MA, USA, 1970.

[99] Elisabeth Rouy and Agnès Tourin. A viscosity solutions approach to shape-from-shading. *SIAM J. Numer. Anal.*, 29:867–884, June 1992.

[100] J. Oliensis. Shape from shading as a partially well-constrained problem. *CVGIP: Image Underst.*, 54:163–183, July 1991.

[101] J. Oliensis and P. Dupuis. A global algorithm for shape from shading. In *International Conference on Computer Vision*, 1993.

[102] John Oliensis and Paul Dupuis. Shape recovery. chapter Direct method for reconstructing shape from shading, pages 17–28. Jones and Bartlett Publishers, Inc., , USA, 1992.

[103] M. Bichsel and A. P. Pentland. A simple algorithm for shape from shading. pages 459–465, 1992.

[104] R. Kimmel and A. M. Bruckstein. Shape from shading via level sets. In *Israel Institute of Technology, CIS Report 9209*, 1992.

[105] Alex P. Pentland. Shape from shading. chapter Local shading analysis, pages 443–487. MIT Press, Cambridge, MA, USA, 1989.

[106] Chia-Hoang Lee and Azriel Rosenfeld. Shape from shading. chapter Improved methods of estimating shape from shading using the light source coordinate system, pages 323–347. MIT Press, Cambridge, MA, USA, 1989.

[107] A. Pentland. Shape information from shading: A theory about human perception. In *Computer Vision., Second International Conference on*, pages 404 –413, dec 1988.

[108] Ping sing Tsai and Mubarak Shah. Shape from shading using linear approximation. *Image and Vision Computing*, 12:487–498.

[109] Emmanuel Prados and Stefano Soatto. Fast marching method for generic shape from shading. In *In VLSM05*, pages 320–331, 2005.

[110] E. Krotkov. Focusing. *IJCV*, 1(3):223–237, 1987.

[111] S. K. Nayar and Y. Nakagawa. Shape from focus. *IEEE Trans. Pattern Anal. Mach. Intell.*, 16:824–831, August 1994.

[112] P. Favaro. Shape from focus and defocus: Convexity, quasiconvexity and defocus-invariant textures, 2007.

[113] M. Subbarao and G. Surya. Depth from defocus: A spatial domain approach. *IJCV*, 13(3):271–294, 1994.

[114] John Ens and Peter Lawrence. An investigation of methods for determining depth from focus. *IEEE Trans. Pattern Anal. Mach. Intell.*, 15:97–108, February 1993.

[115] A.P. Pentland. A new sense for depth of field. *PAMI*, 9(4):523–531, 1987.

[116] Aggelos Konstantinos Katsaggelos. *Digital Image Restoration*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1991.

[117] A. N. Rajagopalan and Subhasis Chaudhuri. Simultaneous depth recovery and image restoration from defocused images. In *CVPR'99*, pages 1348–1353, 1999.

[118] L. M. Bregman. The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming. *USSR Computational Mathematics and Mathematical Physics*, 7(3):200 – 217, 1967.

[119] Yuri Boykov and Vladimir Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26:1124–1137, September 2004.

[120] Paul Beardsley, Paul Beardsley, Phil Torr, Phil Torr, Andrew Zisserman, and Andrew Zisserman. 3d model acquisition from extended image sequences. pages 683–695. Springer-Verlag, 1996.

[121] P. A. Beardsley, A. Zisserman, and D. W. Murray. Sequential updating of projective and affine structure from motion. *Int. J. Comput. Vision*, 23:235–259, June 1997.

[122] Richard I. Hartley. Estimation of relative camera positions for uncalibrated cameras. In *Proceedings of the Second European Conference on Computer Vision*, ECCV '92, pages 579–587, London, UK, 1992. Springer-Verlag.

[123] M. Pollefeys, R. Koch, M. Vergauwen, and L. Van Gool. Hand-held acquisition of 3d models with a video camera. In *IEEE PROCEEDINGS OF 2 ND . INT. CONF. ON 3D DIGITAL IMAGING AND MODELING (3DIM99)*, pages 14–23. Society Press, 1999.

[124] Olivier Faugeras, Luc Robert, Stphane Laveau, Gabriella Csurka, Cyril Zeller, Cyrille Gauclin, and Imed Zoghlami. 3-d reconstruction of urban scenes from image sequences, 1997.

[125] Andrew W. Fitzgibbon and Andrew Zisserman. Automatic camera recovery for closed or open image sequences. In *Proceedings of the 5th European Conference on Computer Vision-Volume I - Volume I*, ECCV '98, pages 311–326, London, UK, 1998. Springer-Verlag.

[126] Daphna Weinshall and Carlo Tomasi. Linear and incremental acquisition of invariant shape models from image sequences. *IEEE Trans. Pattern Anal. Mach. Intell.*, 17:512–517, May 1995.

[127] Conrad J. Poelman and Takeo Kanade. A paraperspective factorization method for shape and motion recovery. *IEEE Trans. Pattern Anal. Mach. Intell.*, 19:206–218, March 1997.

[128] Peter Sturm and Bill Triggs. A factorization based algorithm for multi-image projective structure and motion, 1996.

[129] Anders Heyden. Projective structure and motion from image sequences using subspace methods. In *IN SCIA97*, pages 963–968, 1997.

[130] F. Schaffalitzky, A. Zisserman, R. I. Hartley, and P. H. S. Torr. A six point solution for structure and motion, 2000.

[131] David Jacobs. Linear fitting with missing data: Applications to structure-from-motion and to characterizing intensity images. In *Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition (CVPR '97)*, CVPR '97, pages 206–, Washington, DC, USA, 1997. IEEE Computer Society.

[132] V. Blanz and T. Vetter. A morphable model for the synthesis of 3d faces. In *Proceedings of SIGGRAPH*, pages 187–194, 1999.

[133] Lorenzo Torresani, Aaron Hertzmann, and Christoph Bregler. Non-rigid structure-from-motion: Estimating shape and motion with hierarchical priors. *PAMI*, 2008.

[134] G. Slabaugh, B. Culbertson, T. Malzbender, and R. Schafer. A Survey of Methods for Volumetric Scene Reconstruction from Photographs. 1, Center for Signal and Image Processing, Georgia Institute of Technology, 2001.

[135] C. R. Dyer. Volumetric scene reconstruction from multiple views. In L. S. Davis, editor, *Foundations of Image Understanding*, pages 469–489. Kluwer, 2001.

[136] A. Laurentini. The Visual Hull Concept for Silhouette Based Image Understanding. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(2):150–162, February 1994.

[137] F. P. Preparata and M. I. Shamos. *Computational Geometry: an Introduction*. Springer-Verlag, 1985.

[138] A. Laurentini. How Far 3D shapes Can be Understood from 2D Silhouettes? *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(2):188–195, 1995.

[139] A. Laurentini. How Many 2D Silhouettes Does It Take to Reconstruct a 3D Object? *Computer Vision and Image Understanding*, 67(1):81–87, 1997.

[140] N. Ahuja and J. Veenstra. Generating Octrees from Object Silhouettes in Orthographic Views. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(2):137–149, 1989.

[141] W. N. Martin and J. K. Aggarwal. Volumetric Description of Objects from Multiple Views. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 5(2):150–158, 1983.

[142] W. Matusik, C. Buehler, R. Raskar, S. Gortler, and L. McMillan. Image-Based Visual Hulls. In *Proceedings of SIGGRAPH 2000*, pages 369–374, New Orleans, August 2000.

[143] M. Potmesil. Generating Octree Models of 3D Objects from their Silhouettes in a Sequence of Images. *Computer Vision, Graphics and Image Processing*, 40:1–20, 1987.

[144] W. Niem. Error analysis for silhouette-based 3D shape estimation from multiple views. In *Proc. Int. Workshop on Synthetic-Natural Hybrid Coding and Three-Dimensional Imaging*, 1997.

[145] S. K. Srivastava and N. Ahuja. Octree Generation from Object Silhouettes in Perspective Views. *Computer Vision, Graphics and Image Processing*, 49(1):68–74, 1990.

[146] R. Szeliski. Rapid Octree Construction from Image Sequences. *Computer Vision, Graphics and Image Processing*, 58(1):23–32, 1993.

[147] S. Moezzi, D.Y. Kuramura, and R. Jain. Reality Modeling and Visualization from Multiple Video Sequences. *IEEE Computer Graphics and Applications*, 16(6):58–63, 1996.

[148] G.K.M. Cheung, T. Kanade, J.-Y. Bouguet, and M. Holler. A real time system for robust 3d voxel reconstruction of human motions. In *In Proceedings of IEEE Conf. Computer Vision and Pattern Recognition*, page 714720, 2000.

[149] D. Snow, P. Viola, and R. Zabih. Exact Voxel Occupancy with Graph Cuts. In *Proceedings of Conference on Computer Vision and Pattern Recognition*, pages 345–352, 2000.

[150] C.H. Chien and J.K. Aggarwal. Volume surface octrees for the presentation of 3d objects. *Computer Vision, Graphics and Image Processing*, 36:100–113, 1986.

[151] S. M. Seitz and C. R. Dyer. Photorealistic Scene Reconstruction by Voxel Coloring. *International Journal of Computer Vision (IJCV),*, 35(2):151–173, 1999.

[152] Adrian Broadhurst and Roberto Cipolla. A Statistical Consistency Check for the Space Carving Algorithm. In *Proceedings of 11th British Machine Vision Conference*, pages 282–291, 2000.

[153] K. Kutulakos and S. M. Seitz. A Theory of Shape by Space Carving. *International Journal of Computer Vision (IJCV),*, 38(3):199–218, 2000.

[154] Vikram Chhabra. Reconstructing specular objects with image based rendering using color caching. Master's thesis, Worcester Polytechnic Institute, 2001.

[155] Ruigang Yang. *View-Dependent Pixel Coloring – A Physically-Based Approach for 2D View Synthesis*. PhD thesis, Depart of Computer Science, University of North Carolina at Chapel Hill, 2003.

[156] B. Culbertson, T. Malzbender, and G. Slabaugh. *Generalized Voxel Coloring*, volume 1883 of *Lecture Notes in Computer Science*, pages 100–115. Springer-Verlag, 1999.

[157] K. N. Kutulakos. Approximate N-view Stereo. In *Proceedings of European Conference on Computer Vision (ECCV)*, pages 67–83, 2000.

[158] G. Slabaugh, B. Culbertson, T. Malzbender, and R. Schafer. Improved Voxel Coloring via Volumetric Optimization. Technical Report 3, Center for Signal and Image Processing, Georgia Institute of Technology, 2000.

[159] Vladimir Kolmogorov and Ramin Zabih. Multi-camera Scene Reconstruction via Graph Cuts. In *Proceedings of European Conference on Computer Vision (ECCV)*, pages 82–96, 2002.

[160] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *Proceedings of SIGGRAPH*, 1996.

[161] M. Goesele, B. Curless, and S. Seitz. Multi-view stereo revisited. In *Proceedings of CVPR*, 2006.

[162] O D Faugeras and M Hebert. The representation, recognition, and locating of 3-d objects. *Int. J. Rob. Res.*, 5:27–52, September 1986.

[163] F. Stein and G. Medioni. Structural indexing: efficient 3-d object recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 14(2):125 –145, feb 1992.

[164] Andrew Edie Johnson and Martial Hebert. Surface registration by matching oriented points. pages 121–128, 1997.

[165] C. Dorai, J. Weng, and A.K. Jain. Optimal registration of object views using range data. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 19(10):1131 –1138, oct 1997.

[166] Yang Chen and Gérard Medioni. Object modeling by registration of multiple range images. *Image Vision Comput.*, 10:145–155, April 1992.

[167] Paul J. Besl and Neil D. McKay. A method for registration of 3-d shapes. *IEEE Trans. Pattern Anal. Mach. Intell.*, 14:239–256, February 1992.

[168] Greg Turk and Marc Levoy. Zippered polygon meshes from range images. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, SIGGRAPH '94, pages 311–318, New York, NY, USA, 1994. ACM.

[169] T. Masuda, K. Sakaue, and N. Yokoya. Registration and integration of multiple range images for 3-d model construction. In *Proceedings of the 1996 International Conference on Pattern Recognition (ICPR '96) Volume I - Volume 7270*, ICPR '96, pages 879–, Washington, DC, USA, 1996. IEEE Computer Society.

[170] S. Weik. Registration of 3-d partial surface models using luminance and depth information. In *In Proc. Int. Conf. on Recent Advances in 3-D Digital Imaging and Modeling*, pages 93–100, 1997.

[171] Guy Godin, Marc Rioux, and Réjean Baribeau. Three-dimensional registration using range and intensity information. In Sabry F. El-Hakim, editor, *Proceedings of the SPIE: Videometrics III*, volume 2350, pages 279–290, Boston, Massachusetts, USA, November 1994. SPIE.

[172] Jerome H. Friedman, Jon Louis Bentley, and Raphael Ari Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Trans. Math. Softw.*, 3:209–226, September 1977.

[173] Michael Erdmann, Eric Grimson Mit, David A. Simon, and David A. Simon. Fast and accurate shape-based registration. Technical report, 1996.

[174] Grard Blais, Martin D. Levine, and Martin D. Levine. Registering multiview range data to create 3d computer objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17:820–824, 1993.

[175] P. J. Neugebauer. Geometrical cloning of 3d objects via simultaneous registration of multiple range images. In *Proceedings of the 1997 International Conference on Shape Modeling and Applications (SMA '97)*, pages 130–, Washington, DC, USA, 1997. IEEE Computer Society.

[176] Raouf Benjemaa and Francis Schmitt. Fast global registration of 3d sampled surfaces using a multi-z-buffer technique. In *Image and Vision Computing*, pages 113–120, 1997.

[177] Chitra Dorai, Gang Wang, Anil K. Jain, and Carolyn Mercer. Registration and integration of multiple object views for 3d model construction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20:83–89, 1998.

[178] Kari Pulli. Surface reconstruction and display from range and color data, 1997.

[179] Kari Pulli. Multiview registration for large data sets. In *SECOND INTERNATIONAL CONFERENCE ON 3D DIGITAL IMAGING AND MODELING*, pages 160–168, 1999.

[180] Szymon Rusinkiewicz and Marc Levoy. Efficient variants of the icp algorithm. In *INTERNATIONAL CONFERENCE ON 3-D DIGITAL IMAGING AND MODELING*, 2001.

[181] O. Hall-Holt and S. Rusinkiewicz. Stripe boundary codes for real-time structured-light range scanning of moving objects. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 2, pages 359 –366 vol.2, 2001.

[182] J. Maver and R. Bajcsy. Occlusions as a guide for planning the next view. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 15(5):417 –433, may 1993.

[183] T. Masuda and N. Yokoya. A robust method for registration and segmentation of multiple range images. In *CAD-Based Vision Workshop, 1994., Proceedings of the 1994 Second*, pages 106 –113, feb 1994.

[184] Xavier Pennec. Multiple registration and mean rigid shapes - application to the 3d case. In *In 16th Leeds Annual Statistical Workshop*, pages 178–185, 1996.

[185] R. Bergevin, M. Soucy, H. Gagnon, and D. Laurendeau. Towards a general multi-view registration technique. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 18(5):540 –547, may 1996.

[186] A. J. Stoddart and A. Hilton. Registration of multiple point sets. In *Proc. 13 th Int. Conf. on Pattern Recognition*, pages 40–44, 1996.

[187] Peter J. Neugebauer. Reconstruction of real-world objects via simultaneous registration and robust combination of multiple range images. *International Journal of Shape Modeling*, 3, 1997.

[188] David W. Eggert, Andrew W. Fitzgibbon, and Robert B. Fisher. Simultaneous registration of multiple range views for use in reverse engineering of cad models, 1996.

[189] John Williams and Mohammed Bennamoun. A multiple view 3d registration algorithm with statistical error modeling, 2000.

[190] Harpreet S. Sawhney, Steve Hsu, and R. Kumar. Robust video mosaicing through topology inference and local to global alignment. In *In Proc. European Conference on Computer Vision*, pages 103–119, 1998.

[191] Heung-Yeung Shum and Richard Szeliski. Systems and experiment paper: Construction of panoramic image mosaics with global and local alignment. *Int. J. Comput. Vision*, 36:101–130, February 2000.

[192] Raouf Benjemaa and Francis Schmitt. A solution for the registration of multiple 3d point sets using unit quaternions. In *European Conference on Computer Vision*, pages 34–50, 1998.

[193] Berthold K. P. Horn. Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America A*, 4(4):629–642, 1987.

[194] F. Lu and E. Milios. Globally consistent range scan alignment for environment mapping. *Autonomous Robots*, 4:333–349, 1997.

[195] Gregory C. Sharp, Sang W. Lee, and David K. Wehe. Toward multiview registration in frame space. In *In IEEE International Conference on Robotics and Automation*, 2001.

[196] G.C. Sharp, S.W. Lee, and D.K. Wehe. Multiview registration of 3d scenes by minimizing error between coordinate frames. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(8):1037 –1050, aug. 2004.

[197] A. Sharf, M. Alexa, and D. Cohen-Or. Context-based surface completion. *ACM Transactions on Graphics*, 23(2):878–887, 2004.

[198] Tao Ju. Robust repair of polygonal models. *ACM Transactions on Graphics*, 23(3):888–895, 2004.

[199] S. Park, X. Guo, H. Shin, and H. Qin. Shape and appearance repair for incomplete point surfaces. In *Proceedings of ICCV*, pages 1260–1267, 2005.

[200] Derek Hoiem, Alexei A. Efros, and Martial Hebert. Geometric context from a single image. In *International Conference on Computer Vision*. IEEE, 2005.

[201] Ashutosh Saxena, Min Sun, and Andrew Y. Ng. Learning 3-d scene structure from a single still image. In *International Conference on Computer Vision*, 2007.

[202] Ashutosh Saxena, Sung H. Chung, and Andrew Y. Ng. Learning depth from single monocular images. In *NIPS*, 2006.

[203] Guofeng Zhang, Zilong Dong, Jiaya Jia, Liang Wan, Tien-Tsin Wong, and Hujun Bao. Refilming with depth-inferred videos. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 15(5):828–840, 2009.

[204] Carlo Tomasi. Shape and motion from image streams under orthography: a factorization method. *International Journal of Computer Vision*, 9:137–154, 1992.

[205] S. Knorr and T. Sikora. An image-based rendering (ibr) approach for realistic stereo view synthesis of tv broadcast based on structure from motion. In *ICIP*, 2007.

[206] E. Rotem, K. Wolowelsky, and D. Pelz. Automatic video to stereoscopic video conversion. In *SPIE*, 2005.

[207] Guofeng Zhang, Wei Hua, Xueying Qin, Tien-Tsin Wong, and Hujun Bao. Stereoscopic video synthesis from a monocular video. *Transactions on Visualization and Computer Graphics*, 13(4), 2007.

[208] Steve Katz. *Film Directing Shot by Shot: Visualizing from Concept to Screen*. Michael Wiese, 1991.

[209] P. Harman. Home based 3d entertainment: An overview. In *ICIP*, pages Vol I: 1–4, 2000.

[210] Phil Harman, Julien Flack, Simon Fox, and Mark Dowley. Rapid 2d to 3d conversion. In *in Stereoscopic Displays and Virtual Reality Systems IX, Andrew*, pages 78–86, 2002.

[211] Anat Levin Dani, Dani Lischinski, and Yair Weiss. Colorization using optimization. *ACM Transactions on Graphics*, 23:689–694, 2004.

[212] Daniel Skora, David Sedlacek, Sun Jinchao, John Dingliana, and Steven Collins. Adding depth to cartoons using sparse depth (in)equalities. *Eurographics*, 2010.

[213] J. J. Koenderink and Van Doorn Aj. Local structure of movement parallax of the plane. *Journal of The Optical Society of America*, 66, 1976.

[214] Muralidhara Subbarao. Bounds on time-to-collision and rotational component from first-order derivatives of image flow. *Graphical Models /graphical Models and Image Processing /computer Vision, Graphics, and Image Processing*, 50:329–341, 1990.

[215] R.C. Nelson. Using flow field divergence for obstacle avoidance: Towards qualitative vision. In *Computer Vision., Second International Conference on*, pages 188 –196, dec 1988.

[216] Roberto Cipolla and Andrew Blake. Surface orientation and time to contact from image divergence and deformation. In *Proceedings of the Second European Conference on Computer Vision*, ECCV '92, pages 187–202, London, UK, 1992. Springer-Verlag.

[217] F. Girosi, A. Verri, and V. Torre. Constraints for the computation of optical flow. In *Visual Motion, 1989.,Proceedings. Workshop on*, pages 116 –124, mar 1989.

[218] S. Negahdaripour and S. Lee. Motion recovery from image sequences using first-order optical flow information. In *Visual Motion, 1991., Proceedings of the IEEE Workshop on*, pages 132 –139, oct 1991.

[219] H. H. Nagel. Direct estimation of optical flow and its derivatives. In *Artificial and Biological Vision Systems*, pages 193–224, 1992.

[220] P. Werkhoven and J. J. Koenderink. Extraction of motion parallax structure in the visual system i. *Biological Cybernetics*, 63:185–191, 1990.

[221] Joseph K. Kearney, William B. Thompson, and Daniel L. Boley. Optical flow estimation: An error analysis of gradient-based methods with local optimization. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, PAMI-9(2):229 –244, march 1987.

[222] M. Campani and A. Verri. Computing optical flow from an overconstrained system of linear algebraic equations. In *Computer Vision, 1990. Proceedings, Third International Conference on*, pages 22 –26, dec 1990.

[223] M. Otte and H.-H. Nagel. Optical flow estimation: advances and comparisons. In *Proceedings of the third European conference on Computer vision (vol. 1)*, ECCV '94, pages 51–60, Secaucus, NJ, USA, 1994. Springer-Verlag New York, Inc.

[224] H.-H. Nagel. On the estimation of optical flow: relations between different approaches and some new results. *Artif. Intell.*, 33:298–324, November 1987.

[225] Berthold K. P. Horn and Brian G. Schunck. Determining optical flow. *ARTIFICIAL INTELLIGENCE*, 17:185–203, 1981.

[226] M. V. Srinivasan. Generalized gradient schemes for the measurement of two-dimensional image motion. *Biol. Cybern.*, 63:421–431, September 1990.

[227] H.-J. Chen, Y. Shirai, and M. Asada. Obtaining optical flow with multi-orientation filters. In *Computer Vision and Pattern Recognition, 1993. Proceedings CVPR '93., 1993 IEEE Computer Society Conference on*, pages 736 –737, jun 1993.

[228] Joseph Weber and Jitendra Malik. Robust computation of optic flow in a multiscale differential framework. *International Journal of Computer Vision*, pages 67–81, 1995.

[229] B. G. Schunk. The motion constraint equation for optical flow. In *International Conference on Pattern Recognition*, 1984.

[230] J. Aisbett. Optical flow with an intensity-weighted smoothing. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 11(5):512 –522, may 1989.

[231] S. Negahdaripour and C.-H. Yu. A generalized brightness change model for computing optical flow. In *Computer Vision, 1993. Proceedings., Fourth International Conference on*, pages 2 –11, may 1993.

[232] Alessandro Verri, Ro Verri, and Tomaso Poggio. Motion field and optical flow: Qualitative properties. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11:490–498, 1989.

[233] J. L. Barron, D. J. Fleet, and S. S. Beauchemin. Performance of optical flow techniques. *INTERNATIONAL JOURNAL OF COMPUTER VISION*, 12:43–77, 1994.

[234] J. Gao, Y. Hu, J. Liu, and R. Yang. Unsupervised learning of high-order structural semantics from images. *ICCV*, 2009.

[235] J. Yuan, Y. Wu, and M. Yang. From frequent itemsets to semantically meaningful visual patterns. *SIGKDD*, 2007.

[236] C. Rother, V. Kolmogorov, and A. Blake. Grabcut: Interactive foreground extraction using iterated graph cuts. *SIGGRAPH*, 2004.

[237] V. Lempitsky, P. Kohli, C. Rother, and T. Sharp. Image segmentation with a bounding box prior. *ICCV*, 2009.

[238] Y. Boykov and M.P. Jolly. Interactive graph cuts for optimal boundary and region segmentation of objects in ND images. *IJCV*, 2001.

[239] Y. Li, J. Sun, C.K. Tang, and H.Y. Shum. Lazy snapping. *SIGGRAPH*, 2004.

[240] J. Cui, Q. Yang, F. Wen, Q. Wu, C. Zhang, L. Van Gool, and X. Tang. Transductive object cutout. *CVPR*, 2008.

[241] C. Liu, J. Yuen, and A. Torralba. Nonparametric scene parsing: Label transfer via dense scene alignment. *CVPR*, 2009.

[242] C. Liu, J. Yuen, A. Torralba, J. Sivic, and W. Freeman. SIFT flow: dense correspondence across different scenes. *ECCV*, 2008.

[243] C. Rother, V. Kolmogorov, T. Minka, and A. Blake. Cosegmentation of image pairs by histogram matching-incorporating a global constraint into MRFs. *CVPR*, 2004.

[244] D.S. Hochbaum and V. Singh. An efficient algorithm for Co-segmentation. *ICCV*, 2009.

[245] Sara Vicente, Vladimir Kolmogorov, and Carsten Rother. Cosegmentation revisited: models and optimization. *ECCV*, 2010.

[246] Dhruv Batra, Adarsh Kowdle, Devi Parikh, Jiebo Luo, and Tsuhan Chen. icoseg: Interactive co-segmentation with intelligent scribble guidance. *CVPR*, 2010.

[247] Armand Joulin, Francis Bach, and Jean Ponce. Discriminative clustering for image co-segmentation. *CVPR*, 2010.

[248] L. Cao and L. Fei-Fei. Spatially coherent latent topic model for concurrent object segmentation and classification. *ICCV*, 2007.

[249] J. Winn and N. Jojic. Locus: Learning object classes with unsupervised segmentation. *ICCV*, 2005.

[250] C. Gu, J. Lim, P. Arbeláez, and J. Malik. Recognition using regions. *CVPR*, 2009.

[251] S. Todorovic and N. Ahuja. Extracting subimages of an unknown category from a set of images. *CVPR*, 2006.

[252] Bryan C. Russell, William T. Freeman, Alexei A. Efros, Josef Sivic, and Andrew Zisserman. Using multiple segmentations to discover objects and their extent in image collections. *CVPR*, 2006.

[253] Yong Jae Lee and Kristen Grauman. Collect-cut: Segmentation with top-down cues discovered in multi-object images. *CVPR*, 2010.

[254] B. Alexe, T. Deselaers, and V. Ferrari. Classcut for unsupervised class segmentation. *ECCV*, 2010.

[255] X. Bai, J. Wang, D. Simons, and G. Sapiro. Video SnapCut: robust video object cutout using localized classifiers. *SIGGRAPH*, 2009.

[256] Yaser Sheikh and Mubarak Shah. Bayesian modeling of dynamic scenes for object detection. *PAMI*, 2005.

[257] Y. Sheikh, O. Javed, and T. Kanade. Background Subtraction for Freely Moving Cameras. *ICCV*, 2009.

[258] Yuri Pekelny and Craig Gotsman. Articulated object reconstruction and markerless motion capture from depth video. In *Eurographics*, 2008.

[259] B.K.P. Horn. *Robot Vision*. MIT Press, 1986.

[260] T. Mitsunaga and S. K. Nayar. Radiometric Self Calibration. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 380–387, 1999.

[261] P. E. Debevec and J. Malik. Recovering High Dynamic Range Radiance Maps from Photographs. *Proceedings of ACM Siggraph*, pages 369–378, 1997.

[262] W.H. Press, B.P. Flannery, S.A. Teukolsky, and W.T. Vetterling. *Numerical Recipes in C*. Cambridge Univ. Press, New York, 1988.

[263] J. Davis and H. Gonzalez-Banos. Enhanced shape recovery with shuttered pulses of light. *Proceedings of IEEE International Workshop on Projector-Camera Systems*, 2003.

[264] L. Zhang, N. Snavely, B. Curless, and S. M. Seitz. spacetime faces: High resolution capture for modeling and animation. In *Transaction on Graphics*. ACM, 2004.

[265] P. Fong and F. Buron. Sensing deforming and moving objects with commercial off the shelf hardware. In *Computer Vision and Pattern Recognition*. IEEE, 2005.

[266] Edgar Acuna and Caroline Rodriguez. A meta analysis study of outlier detection methods in classification. *Technical paper, Department of Mathematics, University of Puerto Rico at Mayaguez*, 2004.

[267] Boris Iglewicz and David Hoaglin. How to detect and handle outliers. *The ASQC Basic References in Quality Control: Statistical Techniques, Edward F. Mykytka, Ph.D., Editor.*, 16, 1993.

[268] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. In *Comm. of the ACM*, 1981.

[269] Berthold K.P. Horn. Colosed-form solution of absolute orientation using unit quaternions. In *Journal of the Optical Society of America*, 1987.

[270] G. A. Korn and T. M. Korn. Mathematical handbook for scientists and engineers. *Mathematics of Computation*, 15, 1961.

[271] John Stuelpnagel. On the parametrization of the three-dimensional rotation group. *Siam Review*, 6, 1964.

[272] Carlo Tomasi and Takeo Kanade. Shape and motion from image streams: a factorization method. In *Technical Report CMU-CS-91-105 Carnegie Mellon University*, 1991.

[273] Desbrun M., Meyer M., Schroder P., and Barra. H. Implicit Fairing of Irregular Mesher Using Diffusion and Curvature Flow. In *In Proceedings of Siggraph*, pages 317–324, 1999.

[274] Olga Sorkine, Daniel Cohen-Or, Yaron Lipman, Marc Alexa, Christian Rossl, and Han-Peter Seidel. Laplacian surface editing. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, 2004.

[275] Stanley Osher and Ronald Fedkiw. *Level Set Methods and Dynamic Implicit Surfaces*. Springer-Verlag, 2002.

[276] C. Lei and Y. H. Yang. Optical flow estimation on coarse-to-fine region-trees using discrete optimization. In *ICCV*, 2009.

[277] Johannes Kopf, Michael Cohen, Dani Lischinski, and Matt Uyttendaele. Joint bilateral upsampling. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2007)*, 26(3):to appear, 2007.

## Vita

Miao Liao was born in Huayang, Chengdu, Sichuan, China, on October 4, 1982, the son of Yongcheng Liao and Xianjun Zhao. After completing his degree at Shuangliu Middle School, Shuangliu, Chengdu, Sichuan, in 2001, he entered the Tsinghua University at Beijing, China, receiving the degree of Bachelor of Engineer in May, 2005. He entered The Graduate School in the Department of Computer Science at the University of Kentucky at Lexington, Kentucky in August, 2005. During his stay at the University, he has published 15 conference and journal papers in the field of computer vision and computer graphics. he served as the president of the Chinese Student and Scholar Association from 2007-2008. He is now a senior researcher in Sharp Labs of America at Camas, Washington.