Spacetime stereo and its applications

Li Zhang

A dissertation submitted in partial fulfillment of
the requirements for the degree of

Doctor of Philosophy

University of Washington

2005

Program Authorized to Offer Degree: Computer Science and Engineering

University of Washington
Graduate School

This is to certify that I have examined this copy of a doctoral dissertation by

Li Zhang

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Chair of Supervisory Committee:

Steven M. Seitz

Reading Committee:

Steven M. Seitz

Brian Curless

Richard Szeliski

Date:

University of Washington

Abstract

Spacetime stereo and its applications

Li Zhang

Chair of Supervisory Committee:

Professor Steven M. Seitz
Computer Science and Engineering

A long-standing challenge in computer vision is to recover 3D shapes from images, especially when the shapes change over time. Among the numerous existing shape acquisition methods, very few are able to accurately measure time-varying shapes at both high spatial and temporal resolution. Furthermore, some previous methods have strict assumptions on surface color and texture and do not work robustly for complex object shapes. In this dissertation, we address these limitations and present a new approach, *spacetime stereo*, for dynamic shape acquisition.

Spacetime stereo formulates stereo matching as a 3D window warping problem in video volumes. This new formulation incorporates the temporal variations of image pixels as a cue for better shape reconstruction, and is a general framework for different variants of triangulation-based shape reconstruction methods, namely, passive stereo, active stereo, one-shot structured light, and multi-shot structured light. Under this framework, we develop algorithms that accurately reconstruct deforming objects at high spatial and temporal resolution. Spacetime stereo is also effective for a class of natural scenes, such as waving trees and flowing water, which have repetitive textures and chaotic behaviors and are challenging for existing stereo algorithms.

To demonstrate an application of spacetime stereo, we build a dynamic face capture system that consists of off-the-shelf cameras, projectors, and computers. We also propose a new template fitting and tracking method that computes a sequence of 3D face meshes with vertex-to-vertex correspondence by combining shape and optical flow estimation. The resulting system is able to continuously (20Hz) capture high resolution (about 20K points) facial motion simultaneously from multiple viewpoints without the need to paint markers on the face. Finally, we develop data-driven facial animation tools that take advantage of the captured 3D face sequence. These tools enable untrained users to produce realistic facial expressions and animations.

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGMENTS

I would like to express my gratitude to my advisers, Steve Seitz and Brian Curless, who have made this dissertation possible. I sincerely appreciate them for their guidance, inspiration, and encouragement throughout my graduate studies. They have two different research and advising styles but both of them are equally remarkable. I have learned a great deal from them about doing research, presenting results, and interacting with students. I consider myself very fortunate to have both of them as my advisers.

This dissertation is built upon part of the research projects I have done with several collaborators. Noah Snavely has contributed tremendous efforts on our SIGGRAPH 2004 paper; he implemented the face graph animation tool, generated the animation results, and created beautiful slides for our SIGGRAPH presentation. Aaron Hertzmann introduced me to the field of Machine Learning. It is extremely enlightening to discuss with him; we collaborated on our ICCV 2003 paper and his thoughts without any doubt inspired me on our CVPR 2005 paper. Guillaume Dugas-Phocion is a very fun person; our collaboration on single view modeling is highly enjoyable.

I want to thank Lindsay Michimoto, Tina Loucks, Jennifer Wagner, and Cris Mesling for their timely guidance and help on paper works; and Stephen Spencer and CSE support group for their invaluable help on system maintenance. In the course of my graduate study, the GRAIL faculties and students are incredible, making the lab the most pleasant place to do research. In particular, I would like to thank Kiera Henning, Terri Moore, Ethel Evans, Brett Allen, and Daichi Sasaki for allowing us to use their face data. I really appreciate the various helps from Jiwon Kim, Kiera

Henning, Brett Allen, Yung-Yu Chuang, and Wilmot Li during the SIGGRAPH 2004 submission season, and Daichi Sasaki's effort for generating some mesh tracking results in this dissertation.

I also want to thank Richard Szeliski, Tom Duchamp, and Thomas Daniel for being on my committee. In particular, I would like to thank Richard Szeliski for carefully reading my dissertation and giving thoughtful comments.

Finally, I would like to thank my parents, grandparents, uncles, aunts, and cousins for their love and support. I felt very lucky to be within such a warm family. Most of all, I am deeply indebted to Nan Li for her endless enthusiasm and creativity, which make these last several years truly memorable.

DEDICATION

To my grandma who passed away during my graduate study.

Chapter 1

INTRODUCTION

Understanding the way objects change shape is a fundamental problem in many scientific disciplines. For example, modeling how human beings change facial shapes to express their emotion is an interesting problem in psychology, and relating these shape changes to muscle actuation is an important problem in bio-mechanics and plastic surgery. Additional examples include predicting liver deformation after a surgical incision, which is a critical problem in medical robotics, and simulating the way clothing wrinkles and folds in response to body pose changes, which is a fundamental problem in computer graphics. Solutions to these problems will have a wide spectrum of applications, ranging from computer games and visual effects in movies to medical practice. Yet these problems are not fully understood, because the underlying complex, dynamic shape variations are hard to measure and, as a result, difficult to model.

In this dissertation, we address the problem of accurately capturing three dimensional (3D) shapes of dynamic objects at high spatial and temporal resolution from image streams. Specifically, we propose a novel binocular stereo method, *spacetime stereo*, for recovering time-varying 3D shapes. We also combine the spacetime stereo method with existing optical flow methods to estimate high resolution 3D motion. As an application, we demonstrate how the resulting 3D shape and motion data can be used for data-driven expression synthesis and facial animation in computer graphics.

In this chapter, we first give a formal problem statement of the dissertation and list the desired properties of its solution. We then review a variety of previous shape acquisition methods that may be applied for dynamic scenes and summarize their pros and cons. Finally, we give an overview of this dissertation and outline its organization.

1.1 Problem statement

The objective of this dissertation is as follows:

> *Given a dynamic object, compute the object's time-varying surface shape from different view points, using video cameras and video projectors.*

We seek shape recovery solutions with the following desired properties.

- *High spatial and temporal resolution.* We are interested in the dynamic object's detailed shape variation over time. Therefore, we seek to measure its shape at both high spatial *and* high temporal resolution. Specifically, we would like to use high frame rate cameras to record the object and reconstruct the 3D position for every pixel in the recorded videos.

- *Accurate reconstruction.* We seek to reconstruct shape accurately so that the resulting data can be used for shape synthesis and recognition applications, e.g., synthesizing winkles on a human face or expression recognition. For an object comparable in size to a human face, the reconstruction accuracy should be sub-millimeter.

- *Robustness.* The new methods should handle complex shapes, which may have disconnected components and large depth discontinuities. These types of shapes are not well handled in some previous methods. The new methods should also be robust to both texture-less and highly textured surfaces, to both neutral color and saturated color surfaces.

- *Simultaneous multi-view acquisition.* Complex dynamic scenes usually can not be captured completely from a single view point. We seek methods that are able to capture the scene from multiple viewpoints simultaneously.

- *Off-the-shelf equipment.* For the purpose of easy replication, the new method should be implementable with off-the-shelf devices.

In this dissertation, we present a novel method for reconstructing dynamic shapes that possesses all these properties. Previous methods, by contrast, possess only subsets of these properties. In the next section, we briefly review the previous work in dynamic shape acquisition.

1.2 Previous work in dynamic shape acquisition

Over the last century, a vast number of shape acquisition methods have been proposed. Curless [26] provides an excellent categorization of most existing shape acquisition methods. These methods vary significantly in terms of their capabilities and equipment requirements. Some are designed to capture high resolution shapes for static scenes, while others are designed to capture dynamic scenes at low spatial resolution. Very few methods are capable of robustly capturing dynamic scenes at high spatial and temporal resolution. In this section, we focus on the methods that have been demonstrated to capture dynamic shapes, which includes both a subset of methods discussed in [26] and several more recently published results.

1.2.1 A taxonomy of dynamic shape acquisition

From the top level in Figure 1.2, dynamic shape acquisition can be divided into contact and non-contact methods.

The contact methods include Magnetic Motion Capture (Mocap), Shape Tape, and Optical Mocap. These methods attach some set of “sensors” on a moving object,

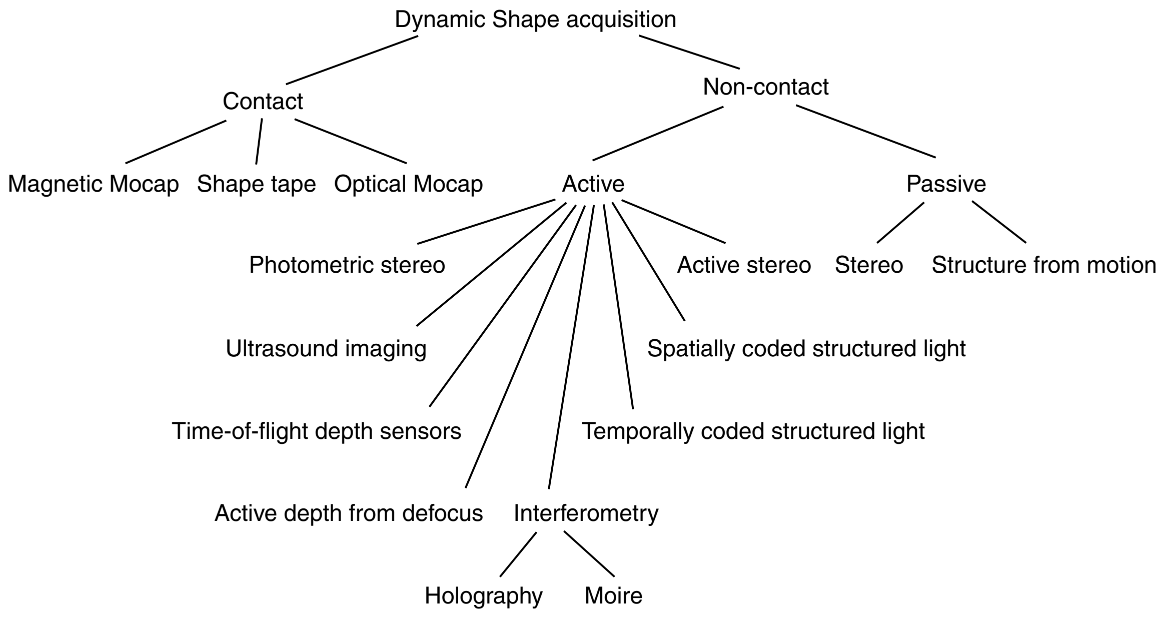


Figure 1.1: A taxonomy of dynamic shape capture

usually a human subject, to measure the object motion. Magnetic Mocap uses a set of coil sensors which can measure their own location and orientation within a source magnetic field [24]. The Shape Tape technique [65] uses a fiber optic sensor to measure 3D bend and twist continuously along its length. Both Magnetic Mocap and Shape Tape are mainly used to measure body poses, not detailed body deformation, e.g., muscle bulging. The Optical Mocap technique requires a few reflective markers placed on a human body and computes 3D positions of the markers by simultaneously videotaping them from multiple viewpoints [68]. Optical Mocap is mainly used to capture human poses as Magnetic Mocap does. Recently, engineers in the movie industry have tried placing more than a hundred small markers on an actor's face to capture his facial motion for the movie "Polar Express". However, this approach still falls short of capturing expressive nuances in facial motion. As camera resolution increases, we can imagine putting smaller and denser sets of markers on a target surface for higher resolution capture. However, doing so is cumbersome and greatly increases the overhead of shape measurement.

Compared to contact methods, non-contact methods do not require placing sensors on the object surface; they reconstruct 3D shapes by recording videos of the scene. These methods can be further divided into active and passive methods, depending on whether the videos are recorded under controlled or natural conditions.

Passive methods

Passive methods reconstruct 3D shape from videos taken under natural lighting, e.g., outdoor video. These methods are typically based on the principle of *triangulation*, as illustrated in Figure 1.2. Based on this principle, a point's 3D position can be reconstructed by intersecting the lines of sight of the corresponding points in two or more images. Two fundamental computer vision problems, stereo and structure from motion, are based on this principle.

Stereo assumes the camera positions and orientations are known and seeks to compute pixel correspondence for 3D reconstruction. Stereo has been studied for decades; Scharstein and Szeliski [85] provide an excellent overview of the state of the art. A known difficulty for stereo is that pixels in texture-less regions cannot be matched accurately. Most stereo algorithms address this difficulty by assuming a smoothness prior model for the underlying shape to regularize the reconstruction. As a result, existing stereo algorithms provide very good approximate shape reconstructions, but not very accurate ones.

Structure from motion assumes that, for a set of image features, their correspondences across different images are given, and seeks to compute camera motion and the 3D positions of these image features. Hartley and Zisserman [46] provide a comprehensive overview of existing methods, which all assume the scene is rigid. More recent work [15, 93] addresses the cases of non-rigid scenes, e.g., humans and animals. A known difficulty in structure from motion is that pixel correspondences can be reliably computed only for the salient feature points. Therefore, structure from motion

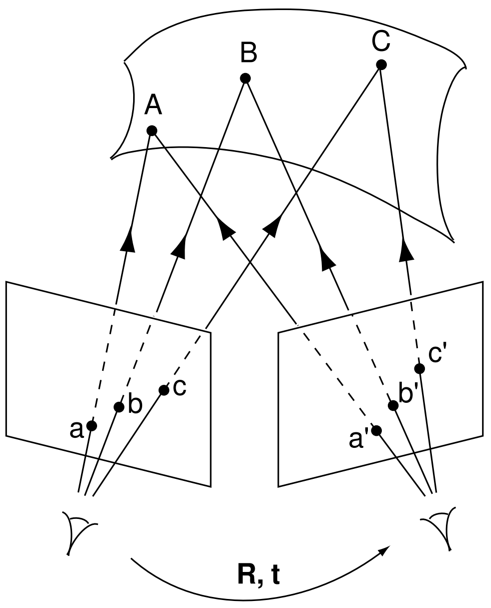


Figure 1.2: The principle of shape from triangulation. The 3D shape of an object can be reconstructed by intersecting corresponding lines of sight from two different viewpoints. The 3D reconstruction problem boils down to two subproblems: computing pixel correspondence and estimating camera motion. Given the camera rotation $\mathbf{R}$ and translation $\mathbf{t}$, stereo seeks to establish pixel correspondence $a \rightarrow a'$, $b \rightarrow b'$, and $c \rightarrow c'$, so that the 3D positions of A, B, and C can be computed. Given the pixel correspondence $a \rightarrow a'$, $b \rightarrow b'$, and $c \rightarrow c'$, structure from motion seeks to compute the camera motion $\mathbf{R}$ and $\mathbf{t}$, and the 3D positions of A, B, and C.

often only recovers sparse shape.

Active methods

Active methods reconstruct 3D shape by sending certain types of energy waves onto a surface and imaging the reflected waves. A large body of methods have been proposed in this area. While many of the methods use lights as energy waves, other types of waves, e.g., ultrasound waves, are also used. These methods differ in the way they control the waves and the way they reconstruct shapes from the reflected signals.

Active stereo Active stereo seeks to alleviate the difficulty caused by low contrast regions in passive stereo by projecting a high contrast texture pattern, e.g., a sinusoidally varying stripe pattern [56] or a random dot pattern [62], onto the surface. This approach introduces synthetic textures over the surface without the overhead of physically touching the surface. However, this approach usually results in reconstructions with limited spatial resolution, because reliable matches are limited to the image pixels with high local contrast in the projected pattern. For example, in some commercial active stereo systems [62], surface positions are only computed for a selective subset of pixels.

Temporally coded structured light Since the main challenge in the triangulation based shape acquisition is establishing pixel correspondence, a large body of research has focused on simplifying the correspondence problem by encoding the correspondence using specially designed lighting patterns. One group of such techniques seeks to uniquely encode the illumination rays using multiple lighting patterns. Specifically, the temporal intensity variation of each ray from the light source is unique. Representative examples of such methods include laser scanners (e.g., [29, 49]) and hierarchical stripe scanners [83]. These methods traditionally require the scene to be completely static over the multiple frames. More recently, with the advance of high speed projection and imaging technologies, researchers [45, 47, 52] have started to apply multi-frame encoding methods to moving scenes, because the scene in a few neighboring frames can be approximated as being static.

Spatially coded structured light Instead of encoding illumination rays over time using multiple frames, another group of methods seeks to design a single lighting pattern such that the intensity variation within a spatial neighborhood is unique. This approach has the advantage that the scene structure can be recovered using only one pattern. Furthermore, this approach can be readily applied to dynamic scenes by repeating the reconstruction for each single frame. Examples of this approach include color stripes [13] and random dot patterns [67]. However, this approach also has a limitation similar to the active stereo methods: only a subset of pixels can be encoded and reconstructed accurately.

Interferometry Interferometry methods operate by projecting a periodic pattern onto a surface. The surface geometry will warp the phase of the periodic pattern. The warped phase can be computed by demodulating the reflected pattern using the original periodic pattern. There are two kinds of interferometry, as illustrated in Figure 1.2. One of them uses the wave nature of light (holography) to represent the sinusoidal pattern and get sub-wavelength accuracy. The other does intensity modulation (moire) and is intended for large scale surfaces. Both require phase unwrapping that does not work with disconnected components and depth discontinuity. Interferometry also assumes uniform surface albedo so that surface geometry is the only source of phase change. In practice, highly textured surfaces violate this assumption. Recently, Fong and Buron [39] combines the phase unwrapping approach and the one-shot color structured light scanning method to address the issues of depth discontinuity and disconnected components. However, the highly textured surfaces still remain to be a challenging case for these methods.

Photometric stereo Photometric stereo works by taking multiple (at least three) images from one fixed viewpoint under varying illumination[97]. It then computes the surface orientation for each pixel based on its shading variation under the different lighting conditions. The surface shape can be generated by integrating over the estimated surface normal filed. Traditionally, photometric stereo has been used only for static scenes. Recently, Wengeret al. [96] have applied photometric stereo to reconstruct the 3D face of a performing actor using a 2160 fps camera with a computer controlled lighting rig. They report that, at such a high frame rate, a moving face can be approximated as being static for every three or four consecutive frames.

Active depth from defocus Depth from defocus [73, 69] recovers the distance of a surface point by estimating its blur in the image. This method works by taking multiple images while varying the distance between the imaging plane and the lens. Then the depth for each pixel can be computed based on the pixel blur in different images. Depth from defocus shares the same weakness with stereo triangulation: it requires the object surface to have high frequency textures. In practice, this requirement is achieved by projecting lighting patterns onto the surface, the same idea used in active stereo. Among many systems based on this approach, Nayar et al. [69] demonstrated the most impressive system, which is able to produce 512x480 depth estimates at 30Hz, with an accuracy of 0.3%. This performance is achieved by taking only two simultaneous scene images and designing an optimal projection pattern taking into account optical transfer functions, image formation, and focus measure operators. Their method, however, assumes the projected light pattern is the dominant texture on the scene surface for optimal performance. Further, it is unclear whether more than one projector-camera system can be used to simultaneously capture the scene from different viewpoints, because the interference among the pattern projections may pose a problem for the optimal depth estimation.

Time-of-flight depth sensors Time-of-flight depth sensors measure the distance to a surface by sending a light pulse to the surface and recording the time delay of the reflected pulse. This principle has been used in radar and sonar for decades, mainly for measuring targets at a large physical scale. Recently, Canesta [50] and 3DV Systems [91] commercialized 3D depth sensors based on this time-of-flight principle that capture at video rates. Their depth accuracy is around a few centimeters, and this accuracy is constant regardless of the object distance from the camera. Although this accuracy may suffice for large scale environments, it is not good enough for small objects on the order of less than one meter in size. To achieve sub-millimeter accuracy with such a working volume, the time-of-flight measurement must be accurate in the femtosecond range [26], and building timing circuity at such a high speed is a significant challenge. Also the time-of-flight systems are sensitive to surface reflectance variations, though some recent work [42] attempts to address this issue.

Ultrasound imaging The time-of-flight principle has also been used in medical imaging where an ultrasound pulse, instead of a light pulse, is sent to the surface and the time delay of the echo is then used to reconstruct the 3D surface. Ultrasound imaging has been used in obstetrics and gynecology for measuring the size of the fetus to determine the due date, in cardiology for seeing the inside of the heart to identify abnormal structures or functions, and in urology for measuring blood flow through the kidney. However, ultrasound waves do not pass through air, limiting its operation range to solid organs that are inside bodies.

1.2.2 A summary of the taxonomy

Among the numerous dynamic shape acquisition methods, there are very few methods that robustly measure shape at both high spatial and temporal resolution. Contact methods capture shapes at high temporal resolution but low spatial resolution. Time-of-flight methods capture shape at both high spatial and temporal resolution, but their results on small objects are not accurate. Interferometry methods can recover dynamic shape under the assumption that the surfaces are textureless. Active defocus sensors can recover dynamic shape at high spatial and temporal resolution. However, they require customized hardware, and multiple scanners may interfere with each other when used simultaneously from multiple viewpoints. Various off-the-shelf structured light methods either capture shape at high spatial resolution or at high temporal resolution, but not both. Recent work on applying existing temporal structured light methods on dynamic scenes with high speed imaging seems a promising approach.

However, our experience indicates that, even at 500 frames per second, inter-frame motion many not be negligible for fast motion, e.g., cloth waving. Therefore, there is a clear need for a new method that can robustly acquire dynamic shapes at high spatiotemporal resolution using off-the-shelf equipments.

1.3 Dissertation overview

This dissertation consists of two main parts. The first part is focused on the solutions to the problem of dynamic shape reconstruction stated in Section 1.1. The second part is focused on how the captured 3D data can be used in synthesizing facial animation. We summarize these two parts as follows.

1.3.1 Dynamic shape reconstruction

The most straightforward approach to recovering time-varying 3D shapes from videos is to recover a shape at each frame independently, so-called one-shot methods. Chapter 2 overviews such methods in detail and proposes a new one-shot method. This new method estimates the 3D shape of an object from a single image taken when the object is illuminated by one color stripe lighting pattern. The method works by matching the projected color transitions with observed edges in the image. It eliminates some restrictions posed by previous methods in this area, e.g., global smoothness assumptions, strict ordering constraints, and singly-connected surface constraints. The resulting approach is suitable for generating high-speed scans of moving objects when projecting a single stripe pattern and recording an image stream.

Our one-shot method, as well as most other one-shot methods, suffer from the limitation that only a subset of pixels along edges can be accurately reconstructed. To resolve this limitation, Chapter 3 considers the problem of recovering dynamic shapes from a pair of stereo sequences. The key observation is that we should match sequence to sequence, instead of frame-by-frame as in one-shot methods, in order to have more accurate 3D estimation for every pixel. Based on this observation, we propose a new stereo framework, *spacetime stereo*, in which the temporal variation of each pixel is used as a cue for pixel correspondence between different image sequences. Under the new stereo formulation, we propose a window warping stereo matching algorithm that is able to reconstruct time-varying shapes accurately (down to sub-millimeter) for each pixel. The resulting approach has all the desired properties listed in Section 1.1.

Although the algorithm proposed in Chapter 3 improves the state-of-art in dynamic shape capture, it has an over-fitting problem. Specifically, it neglects the consistency constraints for depth estimation between neighboring pixels. Chapter 4 proposes a global optimization technique for spacetime stereo matching that overcomes this over-fitting deficiency.

To evaluate the spacetime stereo algorithms, we have built several camera systems to capture 3D shapes. Chapter 5 describes these experimental setups in detail, which include a portable stereo system consisting of a laptop and two Firewire cameras and a multi-camera system for capturing moving human faces from different viewpoints.

1.3.2 Application in facial animation

Spacetime stereo generates surface shape measurement for dynamic objects. When using the measured shapes for 3D animation, it is useful to know the 3D motion as well, i.e., the point correspondence between the shapes. In Chapter 6, we propose a new surface tracking and fitting procedure to compute the 3D surface motion that optimally fits both the 3D shape measurements and 2D optical flow of the surface color in image streams. This method generates a sequence of surface meshes with vertex-to-vertex correspondence and also fills in the missing data in the shape measurement. The end result of Chapter 6 is a dense, marker-less facial motion capture system.

In Chapter 7, we address the problem of how to use the dense 3D motion estimation for expression synthesis and facial animation. Specifically, we have developed an intuitive expression synthesis tool, faceIK, that is able to synthesize a wide range of new expressions by exploiting the correlation in the captured 3D faces. The tool enables posing new expressions by dragging surface points interactively. For example, pulling up on one corner of the mouth causes the entire face to smile. The tool also enables creating new expressions that do not exist in captured faces. For example, asymmetric faces can be created from a set of symmetric faces. We also develop a face graph representation that encodes the dynamics of a face sequence. This graph can be traversed to create a variety of animations via key-framing or texture-synthesis techniques. Our approach enables untrained users to produce expressive facial animations.

Finally, we conclude in Chapter 8 by stating our contributions in this dissertation and suggesting topics for future research. In this dissertation, Chapter 2 and 3 describe joint work with Brian Curless and Steven M. Seitz, first presented in IEEE 3DPVT 2002 [99] and IEEE CVPR 2003 [100], respectively. Chapter 4, 6, and 7 describe joint work with Noah Snavely, Brian Curless, and Steven M. Seitz, first presented in ACM SIGGRAPH 2004 [102]. In the next chapter, we start from discussing the one-shot methods.

Chapter 2

ONE-SHOT SCANNING USING COLOR STRUCTURED LIGHT

Among the various shape reconstruction methods discussed in Chapter 1, triangulation based methods, e.g., stereo and structured light, are probably the most widely used due to their reconstruction accuracy, their scalability to a large variety of working volumes, and their requirements of only off-the-shelf devices. Traditional triangulation methods reconstruct a dynamic scene by taking one or more videos synchronously and then estimating the underlying shape at each time instant independently. As these methods use images taken at one time instant, or sometimes even just a single image, they are often referred as *one-shot* methods. Previous one-shot methods face a few challenges. First, scenes with complex geometry, e.g., disconnected components or large depth discontinuities, often cause errors for the reconstructions. Second, only a subset of the camera pixels, instead of all pixels, are usually reconstructed. In this chapter, we focus on the one-shot approach, and present a new method that addresses the first challenge.[1] In Chapter 3 and 4, we propose methods to address both challenges; these methods are not one-shot methods.

Specifically, in this chapter, we present an accurate triangulation system by projecting a single structured light pattern consisting of high frequency alternation of color stripes. The shape is reconstructed by matching the projected color transitions with observed edges in the image. Due to the finite number of distinct color transitions available in three color channels, some matching ambiguity remains. Furthermore,

[1]This chapter describes joint work with Brian Curless and Steven M. Seitz, first presented in IEEE 3DVPT 2002 [99].

since the surface reflectance modulates the projected lighting pattern, uncertainty also exists for determining the type of color transition for each edge in the image. As a result, the edges themselves may have non-zero likelihood of being associated with several different color transition or even none. We address this multi-hypothesis correspondence problem with a novel multi-pass dynamic programming algorithm. This algorithm reconstructs piecewise-continuous surfaces and overcomes the monotonic surface limitation in previous dynamic programing based methods.

The rest of this chapter is structured as follows. Section 2.1 overviews some of the structured light methods suitable for capturing shapes for dynamic scenes and proposes the architecture of our scanner. In Section 2.2 we formulate the edge correspondence problem as a multi-hypothesis code matching problem and present the multi-pass dynamic programming solution. Next, in Section 2.3, we describe the design of a color-coded projection pattern and its use in reconstructing shape from a single image. In Section 2.4, we describe our implementation and show results. Finally, in Section 2.5, we summarize the work and suggest avenues for future research.

2.1 Related work

Optical triangulation has been an active area of research for decades. The techniques that have been developed range from those that require many images to reconstruct a surface to those that require only a single image. To handle faster motion, many methods use only one pattern. These patterns can be divided into two categories: pseudorandom patterns and periodic patterns.

Pseudorandom patterns seek to encode projected light rays using spatial neighborhoods. For example, Boyer and Kak [13] designed a color stripe pattern in which each window of spatially adjacent stripes has a unique color intensity configuration. The stripe color is chosen from red, green, and blue, and the windows of stripes are separated by black stripes. Following Boyer and Kak's work, Hügli and Maitre [48] use de Bruijn sequences, a type of sequences within which each subsequence of certain length is unique. De Bruijn sequences avoid the need for black stripes as special tags. In both cases, the authors first classify the color of each stripe in the images before matching it to the projected pattern. This two-step procedure is sensitive to errors in stripe identification. Monks and Carter [66] and Chen et al. [22] combine the color identification and stripe matching together as an optimization problem and use dynamic programming to solve both simultaneously. The constraint in these cases is that the order of stripes in the projected pattern is the same as that in the observed image.

Pseudo randomness can also be used for patterns that are not based on stripes. For example, a 2D extension of de Bruijn sequence is called an M-array, which is an $R \times C$ matrix with each $r \times c$ sub-matrices being unique. These sub-matrices can be used as codes to establish the correspondence between a projected pattern and a recorded image. Griffin and Narasimhan [43] and Davies and Nixon [30] used M-arrays to design patterns for structured light. However, these methods are sensitive to surface colors and depth discontinuities, both of which can cause decoding errors for their methods.

Aside from pseudo random patterns, periodic patterns have also been used. Proesmans et al. [78, 77] demonstrate a scanner that projects a grid pattern onto the scene and matches the observed grid to the projected pattern by a global 2D grid optimization algorithm. In this case, the constraint is that the visible portion of the object consist of a single connected component.

Wust and Capson [98] present a three-step phase shift method. They designed a pattern with three sinusoid stripe patterns encoded in the red, green, and blue channels. These three sinusoidal pattern differ 90° in phase. In the observed image, the phase of each pixel can be recovered from its red, green, and blue pixel values. This approach assumes the scene surface has neutral colors and does *not* have large depth discontinuities.

Custom hardware solutions have been developed to solve the problem of shape capture for dynamic scenes. For example, Nayar et al. [69] build a real-time focus range sensor using customized cameras and projector. Although these methods generate good results under certain constraints, the need for customized hardware makes it hard to replicate these methods. We seek to construct range finders from more commonly available and easily assembled components, making our methods more accessible to other researchers.

To this end, we describe a technique that uses a video camera to record a dynamic scene when it is illuminated with a color stripe pattern from a projector.[2] By matching projected color transitions and observed color edges with a novel multi-pass dynamic programming method, we overcome the limitations presented in previous work, the monotonicity constraint in [22], the singly-connected component constraint in [78], and depth discontinuity ambiguity in [98, 47].

2.2 Multi-hypothesis code matching

The basic principle of one-shot triangulation is illustrated in Figure 2.1(a): an illumination pattern is projected onto an object and the reflected light is captured by a camera. The relative distance between a point in the illumination pattern and its position in the captured image is inversely related to depth, allowing the 3D position of the point to be reconstructed, assuming the camera and projector parameters are known.

The primary challenge in one-shot triangulation is obtaining correspondence between points in the projected pattern and pixels in the image. This correspondence problem is mitigated by two observations. First, as shown in Figure 2.1(a), the 2D correspondence problem reduces to determining correspondences between each row of the projected pattern and its corresponding row of the *rectified* camera image [37].

[2]In our implementation, we use a video projector, but it can be replaced by a slide projector.

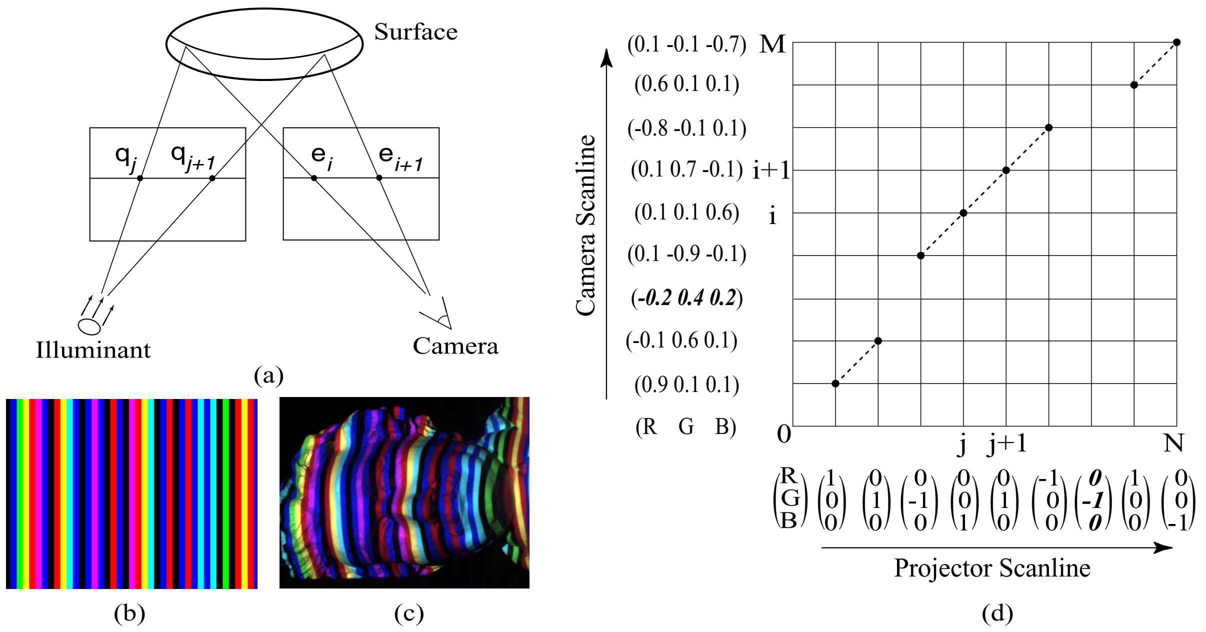


Figure 2.1: Summary of the one-shot method. (a) In optical triangulation, an illumination pattern is projected onto an object and the reflected light is captured by a camera. The 3D point is reconstructed from the relative displacement of a point in the pattern and image. If the image planes are rectified as shown, the displacement is purely horizontal (one-dimensional). (b) An example of the projected stripe pattern and (c) an image captured by the camera. (d) The grid used for multi-hypothesis code matching. The horizontal axis represents the projected color transition sequence and the vertical axis represents the detected edge sequence, both taken for one projector and rectified camera scanline pair. A match represents a path from left to right in the grid. Each vertex (j, i) has a score, measuring the consistency of the correspondence between e_i, the color gradient vectors shown by the vertical axis, and q_j, the color transition vectors shown below the horizontal axis. The score for the entire match is the summation of scores along its path. We use dynamic programming to find the optimal path. In the illustration, the camera edge in bold italics corresponds to a false detection, and the projector edge in bold italics is missed due to, e.g., occlusion.

Second, we can choose whatever pattern we wish to project; therefore, we should choose a pattern that simplifies the correspondence. Laser triangulation scanners take this strategy to the extreme by projecting a narrow beam or plane of light that is easily identified in the image as a point or contour on the surface. Since very little information is available in each image, reconstruction from laser scanners typically requires a very large number of images. Multi-stripe techniques, on the other hand, often use a detailed pattern to obtain as much information about the scene as possible from a small number of images. In this chapter, we treat the problem of obtaining correspondence using multi-stripe techniques, in particular from color stripe patterns of the form shown in Figure 2.1(b) that yield camera images like the one shown in Figure 2.1(c).

While the concepts from this chapter are applicable to a wide range of patterns, we begin by considering specifically the case of patterns consisting of equal-width color stripes to be used for capturing shape with a single image. We can enumerate these stripes by a string of colors $P = (p_0, p_1, \ldots, p_N)$. The information we will use for triangulation is encoded in the transitions between colors. This transition sequence, call it $Q = (q_0, q_1, \ldots, q_{N-1})$, is comprised of elements $q_j = (q_j^r, q_j^g, q_j^b)$ where each color channel takes on a value of -1, 0, or 1 corresponding to a falling, constant, or rising transition, respectively. For example, $(0, -1, 1)$ indicates no change in the red channel, a fall from 1 to 0 in the green channel, and a rise from 0 to 1 in the blue channel. For convenience, we adopt a notation in which, q_j refers to the j-th projector edge, q refers to any (generic) projector edge, and q^c refers to the transition in color channel $c \in \{r, g, b\}$ of projector edge q.

The reflection of the projected pattern from the scene is detected in the image as a sequence of color edges, $E = (e_0, e_1, \ldots, e_{M-1})$ for each sensor scanline, where (using the simplified notation mentioned above) a color edge $e = (e^r, e^g, e^b)$ is described by its 1D intensity gradients in each of the three color channels. Our objective is to solve the correspondence between the transition sequence Q and the edge sequence E.

Correctly identifying the correspondence between projected and imaged stripes, shown in Figure 2.1(b) and (c) respectively, brings out two key difficulties:

- **Mislabeling**: In addition to the projected pattern, the color of image pixels depends on factors such as surface reflectance and shading, viewing direction, color cross-talk between projector spectra and sensor filters, and sensor noise. Consequently, obtaining reliable estimates of color directly from pixel values is not at all straightforward, and misclassifications can result.

- **Occlusions**: Real scenes often have occlusions, shadows, and surface discontinuities. It is therefore not realistic to assume that every projected transition is visible in the image, nor that every observed edge has a corresponding color transition in the projected pattern.

We address the mislabeling problem by introducing a technique that allows for *multiple hypotheses*. Rather than assigning a unique label to every stripe in the image, every label assignment is represented, along with its probability of matching. A final labeling is then obtained by applying a global optimization technique that accounts for occlusions, shadows, and discontinuities.

Specifically, a global match hypothesis ϕ between the projected transition sequence Q and observed edge sequence E is defined as a sequence of integer pairs

$$\phi = \left\{ \begin{pmatrix} j_1 \\ i_1 \end{pmatrix}, \begin{pmatrix} j_2 \\ i_2 \end{pmatrix}, \ldots, \begin{pmatrix} j_\Phi \\ i_\Phi \end{pmatrix} \right\} \tag{2.1}$$

where Φ is the number of integer pairs of ϕ, $j_1 < j_2 < \ldots < j_\Phi$, and $i_1, i_2, \ldots, i_\Phi$ are distinct from each other. Each integer pair $(j_k, i_k)^\mathtt{T}$ indicates that edge e_{i_k} corresponds to the transition q_{j_k}.

The match ϕ is equivalent to a path in a 2D grid, as illustrated in Figure 2.1(d). The horizontal axis represents the projected transition sequence Q and the vertical axis represents the observed edge sequence E. The match ϕ represents a path from left to right in the grid, intersecting each row and each column no more than once.

The quality of a match is computed by assigning each vertex (j, i) a score, $\mathtt{score}(q_j, e_i)$, measuring the consistency of the correspondence between edge e_i and transition q_j.

The specific definition of $\texttt{score}(q_j, e_i)$ is described in Section 2.3. The score of the entire match ϕ is the summation of scores of all the vertices in ϕ, defined as

$$\sigma(\phi) = \sum_{k=1}^{\Phi} \texttt{score}(q_{j_k}, e_{i_k}) \tag{2.2}$$

The optimal match is therefore defined as

$$\phi^* = \arg\max_{\phi}\{\sigma(\phi)\} \tag{2.3}$$

In general, the possible paths represented by the ϕ's may go up and down and have disconnected components due to occlusion, texture edges, etc. Therefore, the space of all possible matches is enormous, of size $O(M^N)$. A common technique for making this optimization problem tractable is to introduce an assumption of depth-ordering, or *monotonicity* [3]

$$i_1 < i_2 < \ldots < i_\Phi. \tag{2.4}$$

With this assumption, Eq. (2.3) may be solved efficiently using dynamic programming [3, 71, 41, 25, 6, 22, 51, 10]. The monotonicity assumption should be used with care, however, since it is violated in the presence of occlusions, and can produce artifacts. In practice, we have observed that violations of the monotonicity assumption result in dropouts, i.e., portions of the scene that are not reconstructed. We show how this problem can be addressed by use of a multi-pass dynamic programming technique. First, we review the basics of dynamic programming in the following subsection.

2.2.1 Correspondence by dynamic programming

We adopt a notation similar to that of Cox et al. [25]. Let $G_{j,i}$ be the sub-grid defined by $[0, j] \times [0, i]$, and $\phi^*_{j,i}$ be the optimal path in $G_{j,i}$. Three possible configurations of $\phi^*_{j,i}$ exist: (1) it consists of vertex (j, i) and the optimal path $\phi^*_{j-1,i-1}$ in $G_{j-1,i-1}$, (2) it is entirely in the sub-grid $G_{j-1,i}$, or (3) it is entirely in $G_{j,i-1}$. In the latter two cases, $\phi^*_{j,i} = \phi^*_{j-1,i}$ or $\phi^*_{j,i} = \phi^*_{j,i-1}$, respectively. Consequently, $\sigma(\phi^*_{j,i})$ may be

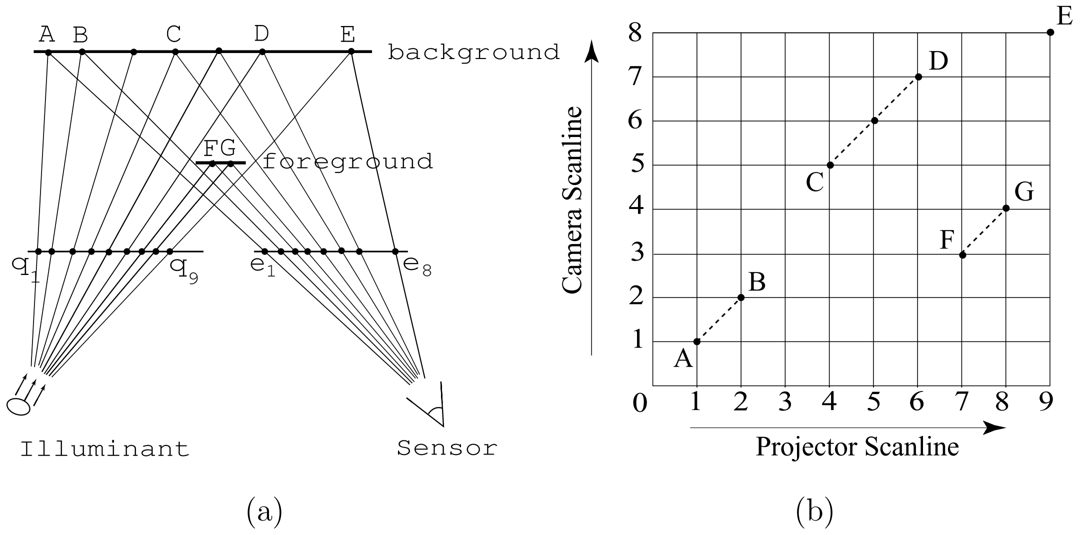


Figure 2.2: Violation of the monotonicity assumption. (a) 9 transitions, $q_1, q_2, \ldots, q_9$, are projected onto a scene comprised of a thin foreground surface against a background surface, and 8 edges, $e_1, e_2, \ldots, e_8$, are detected. Projected edge q_3 is occluded by the foreground element and not detected. (b) The resulting match grid shows that camera edge indices do not increase monotonically with corresponding projector edge indices. A multi-pass DP algorithm can recover the (A,B,C,D,E) path in the first pass, and the (F,G) path in the second pass.

recursively computed as

$$\sigma(\phi^*_{j,i}) = \begin{cases} 0, & \texttt{if j} = 0 \texttt{ or i} = 0; \\ \max\left\{ \begin{array}{l} \sigma(\phi^*_{j-1,i-1}) + \texttt{score}(q_j, e_i), \\ \sigma(\phi^*_{j-1,i}), \\ \sigma(\phi^*_{j,i-1}) \end{array} \right\}, & \texttt{otherwise}. \end{cases} \tag{2.5}$$

The cost of the optimal solution ϕ^* to Eq. (2.3) is given by $\sigma(\phi^*_{N,M})$, and evaluating every $\sigma(\phi^*_{j,i})$ takes $O(MN)$ space and time. $\phi^*_{N,M}$ is computed by tracing back through the cost matrix [25], which takes $O(M + N)$ time. A common technique to reduce the complexity is to restrict the depth range to a user-defined value (e.g., 10% of the maximum possible depth range).

2.2.2 Multi-pass dynamic programming

A fundamental limitation of matching algorithms based on dynamic programming (DP, for short) is the assumption of monotonicity, which is violated in the presence of occlusions. An example of such a violation is shown in Figure 2.2(a). In the figure, a thin foreground element lies in front of a background plane. Due to the occlusion, the order of projected transitions and detected edges is not the same, resulting in a non-monotonic path in the grid, shown in Figure 2.2(b).

The DP algorithm therefore fails to find the optimal path; instead, it will identify the optimal monotonic solution. While this solution could potentially be quite different than the optimal path, in practice we have seen that it corresponds to a monotonic component of the optimal solution. In the case of Figure 2.2(b), DP identifies the sub-path consisting of (A, B, C, D, E). The rest of the optimal solution, sub-path (F, G), is itself monotonic and can be identified by applying DP on the sub-grid obtained by removing columns $(1, 2, 4, 5, 6, 9)$, and rows $(1, 2, 5, 6, 7, 8)$ from the original grid. The same procedure may be repeated until all rows and columns are exhausted. This procedure, which we call `MultiPassDP`, is summarized as follows

```
procedure  MultiPassDP($grid$)
  set $path$ to be empty;
  while($path1 := $DP($grid$) is not empty)
    $path := path \bigcup path1$;
    remove columns and rows in $path1$ from $grid$;
  return $path$;
end  MultiPassDP
```

`MultiPassDP` computes the monotonic components of the optimal path in multiple passes, enabling solution of correspondence problems with occlusions that are not possible with traditional DP. Instead of exhausting the positive monotonic components, $path1$, in the grid, the number of DP passes can also be specified by a user, based on prior knowledge of how many "layers" of structure the scene contains.

2.3 One-shot patterns and scoring functions

The previous section describes the machinery needed to compute an optimal surface given a projected pattern and observed image. In this section, we discuss design decisions in choosing a pattern to project and a scoring function to be used in computing the optimal one-shot surface.

2.3.1 De Bruijn illumination patterns

Choosing a good pattern to project is important for achieving accurate correspondence with optical triangulation techniques, particularly one-shot methods. Assuming the patterns and images are rectified, the pattern may be designed for a single scanline and replicated to produce a 2D vertical pattern. One design choice is whether to project a smooth or a piecewise-constant pattern. For a one-shot method, encoding information in smooth intensity variations is difficult to do, because surface shading can affect these variations substantially. Thus, we choose to encode information on edges and resort to a piecewise-constant illumination pattern.

In addition, a good pattern has the property that correspondences between projected edges and observed edges is easy to determine. As noted earlier, each edge element of Q is comprised of three color edge transition labels that can each take values -1, 0, or 1. This value assignment implies an "edge alphabet" of 27 unique edges; however, since the edge $(0, 0, 0)$ is really no edge at all, 26 edges are actually available to us.

Projecting exactly 26 edges would lead to well-determined correspondence, since each edge is unique, but this would yield very sparse reconstructions. Alternatively, we could lay out the same sequence of edges and repeat them as many times as needed to fill out the desired total number of projected edges. Consider, though, the case of an object that is 26 projected stripes in width. The dynamic programming algorithm will prefer to find a sequence of matches that corresponds to a single connected surface (rather than a set of scattered matches), but due to the repetition in the pattern, a set of equivalent answers will arise. In fact, such a repetitive pattern can result in ambiguity for even larger objects. To minimize such ambiguity, we instead we seek to find a sequence that has a good *windowed uniqueness* property, i.e., a sequence that has the desired number of transitions with a small window size n such that each sub-sequence of n consecutive stripes is unique within the entire sequence.

Our goal now is to devise a color sequence P that yields an edge sequence Q with a good windowed uniqueness property. First, we consider the fact that given a color p_j, the next color p_{j+1} must be different in at least one channel for an edge transition to occur. If we think of these colors as being mapped to base-2 numbers $\{000, 001, \ldots, 111\}$ (i.e., black, blue, ..., white), then a legal edge is produced by performing a bitwise `XOR` (exclusive or) of a given color with a number in the range $\{001, \ldots, 111\}$. For example, the color index corresponding to green, 010, could change to 110 (yellow) by flipping the red channel, i.e., by `XOR`'ing with 100. We could then attain local and global non-periodicity if we could choose a sequence of `XOR` patterns (of which we have 7 to choose from) that are unique when taken in groups of n at a time.

De Bruijn sequences [40] are ideally suited to this problem. In particular, a k-ary de Bruijn sequence of order n is a circular sequence $d_0, d_1, \ldots, d_{k^n-1}$, where each element d_j is taken from a set of k values and for which any sub-sequence of length n appears exactly once. In our case, we can construct up to a 7-ary sequence with each element $d_l \in \{001, \ldots, 111\}$. Then, we can generate a color index sequence $(p_0, p_1, \ldots p_{7^n})$ by choosing an initial color $p_0 \in \{000, \ldots, 111\}$ and following the iteration:

$$p_{j+1} = p_j \texttt{ XOR } d_j \tag{2.6}$$

In practice, we only needed 125 stripes and thus worked with $k = 5, n = 3$. We

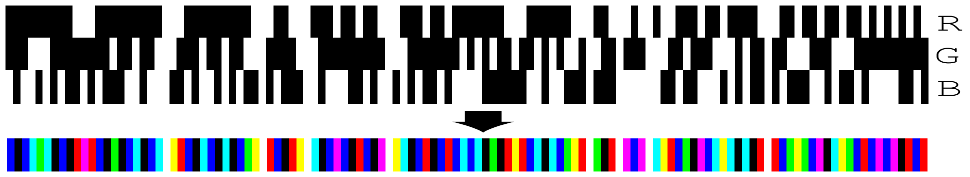


Figure 2.3: Using a de Bruijn sequence, we can generate binary R, G, and B patterns that combine to make a sequence for which each three consecutive color transitions are unique. This example is a complete sequence for $k = 5, n = 3$, which is used to generate the results in this paper.

eliminated 110 and 111 from the de Bruijn sequence, as we found that simultaneous red and green transitions suffered the most from residual crosstalk errors after approximate decoupling (see Section 2.3.2). Figure 2.3 shows a complete color stripe pattern generated in this manner. Note that while de Bruijn sequences are not straightforward to derive, they have been previously tabulated by researchers in combinatorics. We use generators available online [82] to obtain de Bruijn cycles.

2.3.2 Color edge detection

After the illumination pattern from the previous section is projected onto an object, a camera records an image of the reflected light. In an ideal world, the light from each projector color channel reaches only its correspondingly colored sensor pixel (e.g., red light is seen only by red pixels). In practice, however, each projector color channel has some influence on all three sensor channels, a phenomenon known as color crosstalk. This coupling is complicated by the intervention of a surface that may modify the projector spectrum in unknown ways before it is observed at the sensor. One solution would be to assume that the surface is spectrally uniform, so that we need only measure a projector-camera color crosstalk matrix that indicates how much each projector channel influences each camera channel. Caspi et al. [19], however, demonstrate that a less severe restriction can work fairly well. In particular, they relate observed camera color s to projector color p as:

$$\underbrace{\begin{bmatrix} s^r \\ s^g \\ s^b \end{bmatrix}}_{s} = \underbrace{\begin{bmatrix} \chi_{11} & \chi_{12} & \chi_{13} \\ \chi_{21} & \chi_{22} & \chi_{23} \\ \chi_{31} & \chi_{32} & \chi_{33} \end{bmatrix}}_{X} \underbrace{\begin{bmatrix} \rho^r & 0 & 0 \\ 0 & \rho^g & 0 \\ 0 & 0 & \rho^b \end{bmatrix}}_{F} \underbrace{\begin{bmatrix} p^r \\ p^g \\ p^b \end{bmatrix}}_{p} + \underbrace{\begin{bmatrix} o^r \\ o^g \\ o^b \end{bmatrix}}_{o} \tag{2.7}$$

where X is the projector-camera color channel crosstalk matrix, F is the scene albedo matrix at a point on the surface, and o is the ambient light observed by the camera for the same surface point. By pre-multiplying each camera color by X^{-1}, we obtain new camera colors:

$$\tilde{s} = X^{-1}s = \begin{bmatrix} \rho^r p^r + \tilde{o}^r \\ \rho^g p^g + \tilde{o}^g \\ \rho^b p^b + \tilde{o}^b \end{bmatrix} \tag{2.8}$$

where $\tilde{o} = X^{-1}o$. Using this model, crosstalk has largely been factored out so that each color channel can be analyzed independently and correlated more closely to projected colors.

Given a color-corrected camera scanline, we can now localize color edges by looking for local extrema in gradients (1D derivatives) in each of the color channels. In practice, however, this will lead to distinct localizations in each color channel. Instead, we compute a combined gradient function along a scanline that is comprised of the sum of the squares of the gradients in each of the color channels. The edges are then determined to be local maxima of this function, and their strengths are the color gradient values at the localized edges.

2.3.3 Edge-based scoring functions

We now consider the problem of evaluating a match between a projected color transition q and an observed edge e in an image by defining a score function $\texttt{score}(q, e)$. Let $e = (e^r, e^g, e^b)$, where $e^c \in [-1, 1]$ is the 1D intensity gradient of e in channel c, and let transition $q = (q^r, q^g, q^b)$, with $q^c \in \{-1, 0, 1\}$. e and q are consistent only

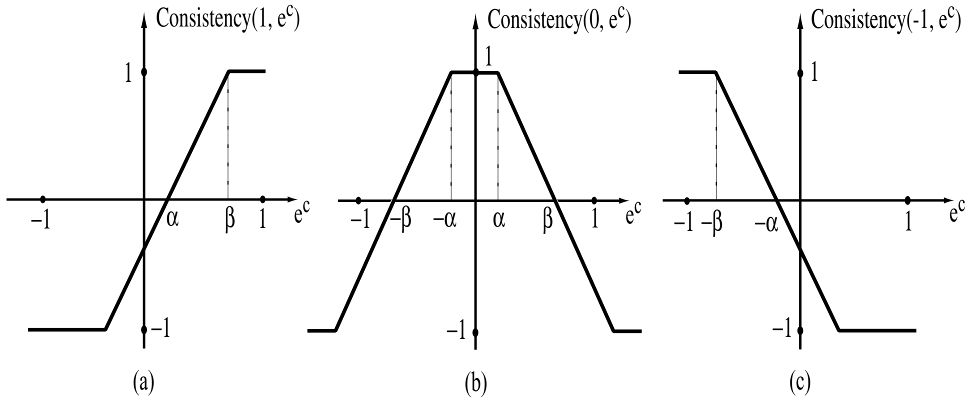


Figure 2.4: Consistency measure $\texttt{consistency}(q^c, e^c)$ between projector transition q^c and e^c. (a) $q^c = 1$, (b) $q^c = 0$, and (c) $q^c = -1$.

if they match in all three channels. Accordingly, we use the following definition of `score`:

$$\texttt{score}(q, e) = \min_{c \in \{r,g,b\}} \{\texttt{consistency}(q^c, e^c)\} \tag{2.9}$$

where $\texttt{consistency}(q^c, e^c) \in [-1, 1]$ measures the agreement between corresponding color channels of q and e. For example, when $q^c = 1$, $\texttt{consistency}(1, e^c)$ should be 1 if e^c is sufficiently large, 0 if $|e^c|$ is sufficient small, and negative if e^c is negative. More formally, $\texttt{consistency}(q^c, e^c)$ is defined by the following equation, Eq. 2.10(a), and illustrated in Figure 2.4(a). For the cases of $q^c = 0$ and -1, $\texttt{consistency}(0, e^c)$ and $\texttt{consistency}(-1, e^c)$ are similarly defined in Eq. 2.10(b,c) and illustrated in Figure 2.4(b,c) respectively.

$$\begin{array}{rll} \texttt{consistency}(1, e^c) = & \texttt{CLAMP}(\frac{e^c-\alpha}{\beta-\alpha}; -1, 1) & (a) \\ \texttt{consistency}(0, e^c) = & \texttt{CLAMP}(1 - \frac{|e^c|-\alpha}{\beta-\alpha}; -1, 1) & (b) \\ \texttt{consistency}(-1, e^c) = & \texttt{consistency}(1, -e^c) & (c) \end{array} \tag{2.10}$$

where

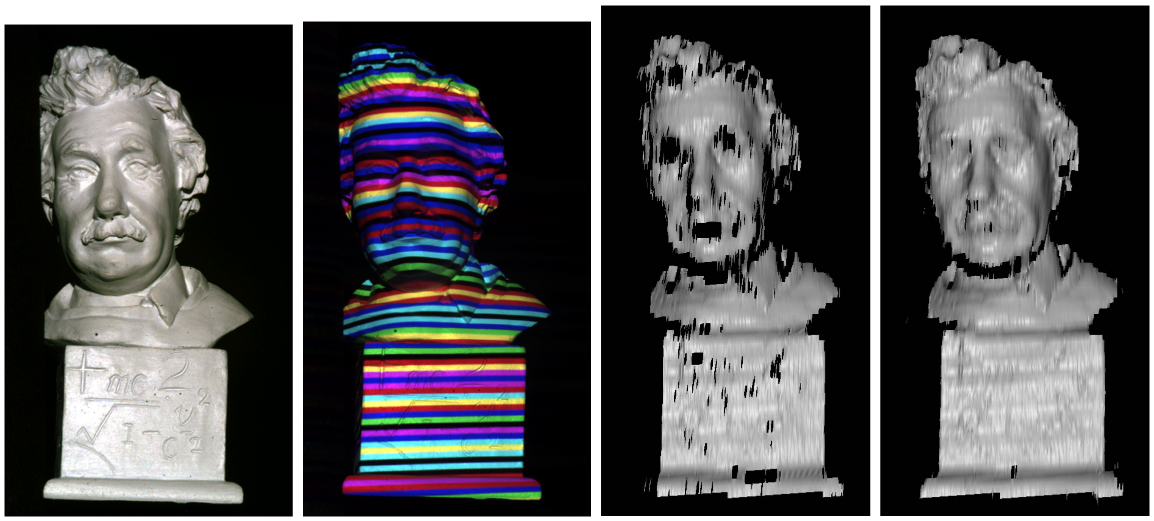
$$\mathtt{CLAMP}(x; x_0, x_1) = \begin{cases} x_0 & \mathtt{if}\ \ x < x_0; \\ x & \mathtt{if}\ \ x_0 < x \leq x_1; \\ x_1 & \mathtt{if}\ \ x_1 < x. \end{cases} \tag{2.11}$$

and $0 \leq \alpha < \beta \leq 1$ are *soft thresholds* that are chosen based on the uncertainty of edge measurement. In particular, gradients in the range of $[\alpha, \beta]$ can be classified with fractional values that reflect their uncertainty, whereas gradients with absolute values that are sufficiently large or small are assigned either -1, 0, or 1. The decision on how to label each edge is deferred to the global optimization stage. In the degenerated case when $\alpha = \beta$, the gradients are classified with *hard thresholds*, as in [13]. The larger the value of $\beta - \alpha$, the more uncertain `consistency`. Less certain consistency measures are useful when there are significant differences in the intensity of projected and reflected patterns, due for instance to noise, shading, or surface texture.

Note that edge pair (q, e) will get matched by DP only if $\mathtt{score}(q, e)$ is positive and it will not get matched if any of its channel consistency measures is negative. As a result, clamping of negative consistency values is not necessary in theory, but in practice avoids possible numerical problems of large negative numbers when α and β and nearly equal.

2.4 Results

We have developed an experimental system for testing our one-shot shape capture methods. The hardware consists of a Kodak DCS520 digital still camera and a Compaq MP1800 digital projector. To simulate resolutions more comparable to a video camera (the sensor type we ultimately plan to use for realtime capture), we downsampled the Kodak images by a factor of 2X in each dimension, yielding 864x576 images. The projector operates at 1024x768 resolution. For geometric calibration, we image a checkerboard textured plane in a variety of poses. For each pose, we also *project* a distinct checkerboard pattern onto the plane and take an additional image. The



(a) (b) (c) (d)

Figure 2.5: Comparison between DP and CFG [13]. (a) Photo of original Einstein bust. (b) Stripe image used for one-shot reconstruction. The bust was photographed on its side, but the image shown here is rotated by 90 degrees for convenient visualization. (c) Shaded rendering of the CFG reconstruction. (d) Shaded rendering of the DP reconstruction. In areas of high curvature, false edges are more prevalent, and result in dropouts in the CFG reconstruction, whereas DP is free to ignore such edges in pursuit of a global optimum.

images taken without the projected pattern are used to estimate the camera intrinsics and plane poses (see Bouguet [12]). For the remaining images, we can compute the 3D coordinates of the the projected pattern features (corresponding to projector rays) and thus calibrate the projector. We employ a linear projective model for both the camera and projector; these projective matrices are used to rectify [38] the projected pattern and the recorded images before the edge detection and code matching procedures are performed. In addition, to improve color channel alignment in the digital camera, we image the checkerboard textured plane an additional time and compute separate 2D homographies for the red and blue channels relative to the green.

The X matrix in Eq. 2.7 is approximated by projecting solid red, green, and blue patterns to a fronto-parallel white board and capturing three images accordingly. The

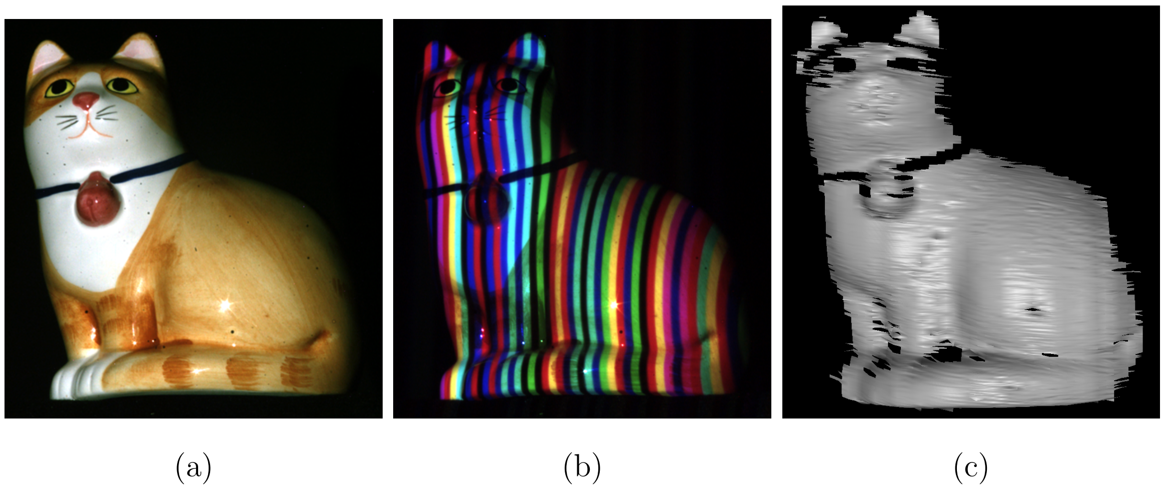


Figure 2.6: Sensitivity to surface reflectance. (a) Photo of original porcelain cat model. (b) Stripe image used for one-shot reconstruction. (c) Shaded rendering of the DP reconstruction. Although this model is not "colorless" the reconstruction behaves reasonably well. Completely black regions, of course, lead to range dropouts.

mean colors of the three captured images constitute the three columns of X.

We must note that, while we have taken some measures to reduce color misalignments and account for color crosstalk, we still observe some residual misalignments and non-linear crosstalk behaviors that have not been accounted for. As a result, some of the rendered reconstructions shown in this section exhibit coherent ridges at color transition boundaries.

We have tested the one-shot capture method on a variety of scenes. In each case, the projected pattern consists of de Bruijn generated sequences with 7 pixel wide stripes (roughly 150 stripes total).

First, we compare the dynamic programming approach to the *Crystal Fitting and Growing* (CFG) algorithm developed by Boyer and Kak [13], for which results are shown in Figure 2.5. CFG first labels each edge transition code by computing the signs of intensity transitions in the three channels, followed by matching the labeled edge sequence by iteratively finding the longest matching sub-sequences. In practice,

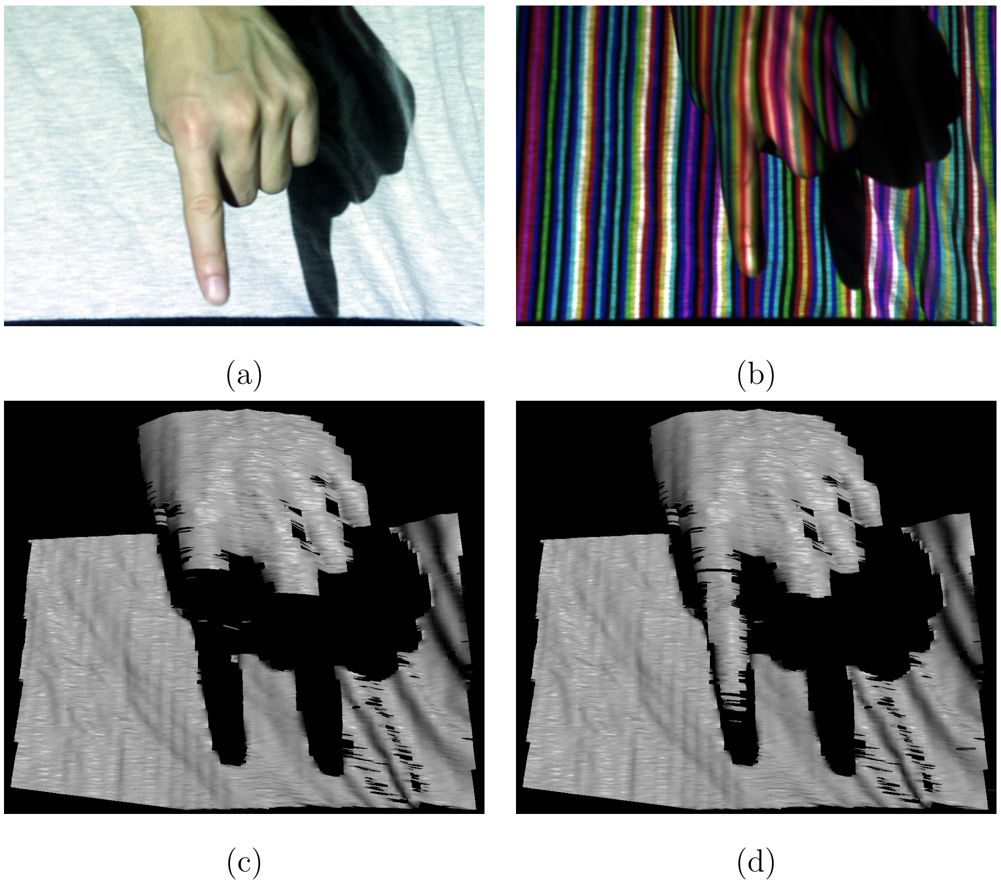


Figure 2.7: Comparison between one-pass DP and two-pass DP. (a) Photo of original scene of a hand and finger in front of a cloth background. (b) Stripe image used for one-shot reconstruction. Note that image (a) is taken at a different time instant than the image (b); therefore the hand poses in these two images are slightly different. (c) Shaded rendering of a one-pass DP reconstruction. (d) Shaded rendering of a two-pass DP reconstruction. The second pass recovers most of the finger that violated monotonicity and was not recoverable in a single pass. The "double-finger" hole in the background corresponds to projector and sensor visibility shadows.

we have found this technique to be fragile in the presence of erroneous edge labels, resulting in outlier points and holes in the reconstruction as shown in the figure. One-pass DP, on the other hand, is able to cope with these mis-labelings by searching for a globally optimal, monotonic solution. The result is fewer holes in the reconstruction.

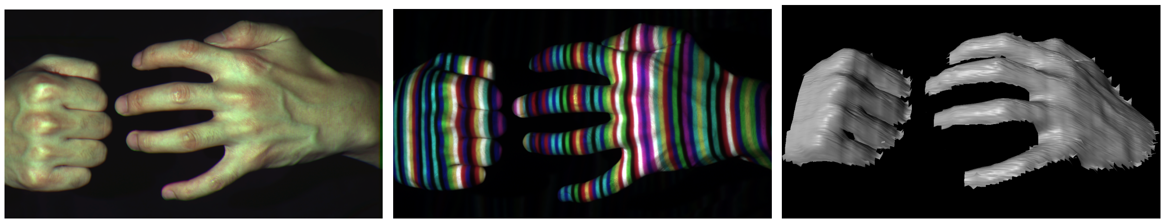


Figure 2.8: In this paper, we show how to reconstruct the shape of a scene, such as the two hands shown on the left, given a single photograph of the scene under color-striped illumination shown at center. A novel dynamic programming method leads to the geometric reconstruction on the right, shown as a shaded rendering from a new viewpoint.

The Einstein bust in the previous example is fairly "white," similar to the color crosstalk calibration target. To test sensitivity to fairly non-white surfaces, we took one-shot images of human hands (Figure 2.8) and of a painted porcelain cat (Figure 2.6). In both cases, the reconstructions are fairly accurate and complete, including, for instance, regions around the cat's red nose and over its orange body. Particularly dark areas result in holes in the reconstructions, since edges in these areas are not detected by the sensor.

Note that one noticeable artifact in Figure 2.8 is the occurrence of false edge extensions at the boundary of the object. This artifact arises, because DP is free to add points onto the boundary while still increasing the score. Thresholding out low intensity gradients minimizes the effect. Further, these points can be downweighted when used, e.g., to reconstruct surfaces [94]

Finally, to demonstrate the multi-pass DP method, we show a simple example that violates the monotonicity constraint: a finger in front a piece of cloth (Figure 2.7). Using a single DP pass, the finger is lost in favor of reconstructing the cloth background. The second pass, however, recovers much of the lost finger. Note that this example, together with Figure 2.8, also demonstrates that our scanning method is applicable to scenes with disconnected components, which can not be reconstructed by methods that rely on traversing edge graphs spatially within a single connected component (as in the case of, e.g., Proesmans et al. [78, 77]).

In the above experiments, each range map takes less than 1 minute to compute offline using a 900 MHz Pentium III PC. The exact time generally depends on the number of edges detected and the depth range of the scene. Reconstructions typically contain tens of thousands of range points, with dense vertical sampling along stripe edges and comparatively sparse sampling horizontally. For a triangulation angle of approximately 17 degrees, and an x-y field of view of about 40cm x 25cm, we have found plane-fit accuracy to have a standard deviation of 0.18mm.

Video demonstration of applying our method repeatedly to a sequence of images for recovering 3D dynamic scenes are available on the website `http://grail.cs.washington.edu/projects/moscan/`.

2.5 Discussion

This chapter presents a general multi-pass dynamic programming algorithm to solve the multiple hypothesis code matching problem in structured light scanning. Dynamic programming has long been used in stereo vision, but has a number of limitations that have made it less desirable than other methods of stereo reconstruction. In the context of color structured light rangefinding, however, we show that it can be quite powerful. One limitation of dynamic programming in this setting is the requirement that the surface be monotonic with respect to the projector and camera. Our multi-pass dynamic programming overcomes this limitation by constructing piecewise-continuous matchings for surfaces that violate monotonicity. A second problem with dynamic programming as applied to traditional stereo correspondence is that incorporating inter-scanline constraints is problematic, resulting in abrupt disparity discontinuities between adjacent scanlines. With the aid of color structured light, however, we have observed that the inter-scanline inconsistency problem is much less severe. This is primarily due to the fact that the structured light pattern is very coherent; i.e., it is identical across scanlines.

This work has potential for improvement in several directions. Although the dynamic programming method works reasonably well in our experiments, it would interesting to investigate its theoretical convergence properties. Also, we wish to explore the possibility of using 2D optimization techniques, such as graph cuts [57], and compare them with dynamic programming in future work. Compared to the list of desired properties in section 1.1, one limitation of this work is that it only estimates shape at edges. To obtain shape reconstructions at the higher resolution, we would like to estimate depth for every pixel. Another limitation is that the shape estimation is based on edge detection. This is problematic, because the projected color transitions may be corrupted by surface texture. Further, the projected color transitions may interfere with each other if several patterns are projected simultaneously from different viewpoints for multi-view shape capture. To overcome these limitations, we present a more general method in the next chapter for accurately recovering per-pixel depth that is resilient to surface texture and overlapping structured light projections.

Chapter 3

SPACETIME STEREO

The appearance of the real world varies over time, due to lighting changes, motion, and changes in shading or texture over time. Traditional one-shot triangulation methods [13, 78] handle these variations by treating each time instant in isolation and computing spatial correspondences between pixels in a single pair of images for a static moment in time. While they enable reconstructing moving scenes, they typically provide limited shape detail and resolution. In this chapter, we instead argue that better results may be obtained by considering how each pixel varies over time and using this variation as a cue for correspondence, an approach we call *spacetime stereo*.[1]

The basic principle of spacetime stereo is straightforward. First, consider conventional stereo algorithms, which generally represent correspondence in terms of disparity between a pixel in one image and the corresponding pixel in another image. The matching function used to compute disparities often compares spatial neighborhoods around candidate pairs of pixels. Spacetime stereo simply adds a temporal dimension to the neighborhoods used in the matching function. The spacetime window can be a rectangular 3D volume of pixels, which is useful for reconstructing scenes in which changes in lighting and appearance, rather than shape changes, dominate. When the object is moving significantly, the disparity must be treated in a time-dependent fashion. In this case, we compute matches based on oriented spacetime windows that allow the matching pixels to shift linearly over time. In either case, the match scores

[1]This chapter describes joint work with Brian Curless and Steven M. Seitz, first presented in IEEE CVPR 2003 [100].

based on spacetime windows are easily incorporated into existing stereo algorithms.

The spacetime stereo approach has several advantages. First, it serves as a simple yet general framework for computing shape when only appearance or lighting changes. These changes may be natural, e.g., imparted by the weathering of materials or the motion of the sun, or they may be artificially induced, as in the case of structured light scanning. We present results specifically for the latter. Second, when shape changes are small and erratic and the appearance has complex semi-repetitive texture, such as one might find when observing a waterfall or a collection of leaves on a tree blowing in the wind, spacetime stereo allows robust computation of average disparities or "mean shapes." Finally, for objects in motion, possibly deforming, the oriented spacetime window matching provides a way to compute accurate disparity maps when standard stereo methods fail. This last case is shown to be particularly effective for structured light scanning of moving scenes.

The rest of this chapter is organized as follows. Section 3.1 discusses prior work both in stereo and in structured light scanning. Section 3.2 presents spacetime metrics used to evaluate candidate correspondences. Section 3.3 describes how to adapt existing stereo algorithms to perform spacetime stereo matching. Section 3.4 presents experimental results. Section 3.5 concludes with a discussion of the strengths and weakness of the approach.

3.1 Previous work

Our work builds upon a large literature on stereo correspondence and structured light scanning. Here we summarize the most relevant aspects of this body of work.

Stereo matching has been studied for decades in the computer vision community. We refer readers to Scharstein and Szeliski's recent survey paper [85] for a good overview and comparison of the current state of the art. Traditional stereo algorithms match two or more images acquired at a single time instant and do not exploit other cues that exist over time. Recently, a number of researchers have proposed integrating stereo matching and motion analysis cues, an approach called *motion stereo*. For example, Mandelbaum *et al.* [64] and Strecha and van Gool [90] recover static scenes with a moving stereo rig; in particular, they acquire multiple frames over time to better reconstruct occluded regions in a single stereo pair. Tao *et al.* [92] represent a scene with piecewise planar patches and assume a constant velocity for each plane to constrain dynamic depth map estimation. Vedula *et al.* [95] present a linear algorithm to compute 3D scene flow based on 2D optical flow and estimate 3D structures from the scene flow. Zhang *et al.* [104] compute 3D scene flow and structure in an integrated manner, in which a 3D affine motion model is fit to each local image region and an adaptive global smoothness regularization is applied to the whole image. They later improve their results by fitting parametric motion to each local image region obtained by color segmentation, so that discontinuities are preserved [105]. Carceroni and Kutulakos [18] present a method to recover piecewise continuous geometry and parametric reflectance under non-rigid motion with known light positions.

Unlike this previous motion stereo work, spacetime stereo does *not* estimate inter-frame motion, but rather uses a linear model to approximate local disparity variation in space and time. The locally linear model is generally valid for interior regions of continuous surfaces so long as the cameras have a sufficiently high frame rate with respect to the 3D motion. A key advantage of our approach is that it does not require brightness constancy over time. Caspi and Irani [20] use a related idea to align image sequences assuming a global affine transformation between sequences. Our approach can be viewed as extending this method to compute a full stereo correspondence instead of a global affine transformation.

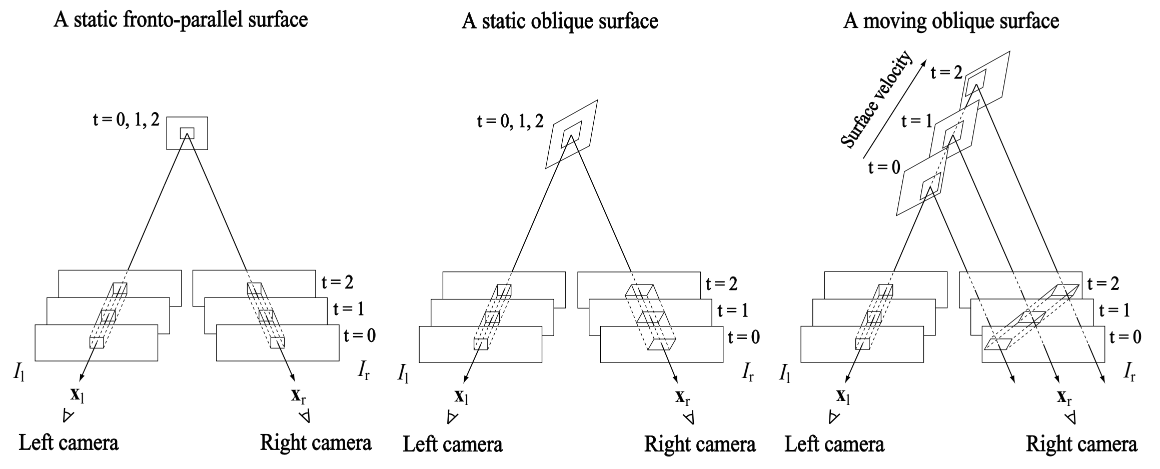
One important application where brightness constancy is especially violated is structured light scanning, where controlled lighting is used to induce appearance variation and used to reconstruct accurate shape models. In section 2.1, we reviewed several representative structured light methods that use a single projection pattern.

Better results can be obtained if multiple patterns and images are used. Projecting many frames of single stripe patterns, Kanade *et al.* [54] and Curless and Levoy [27] used temporal intensity variation (the latter authors called it "spacetime analysis") to resolve correspondence between sweeping laser stripes and sensor pixels. Pulli *et al* [79] adapted spacetime analysis to match a sweeping projector stripe observed by multiple video cameras. Bouguet and Perona [11] applied spacetime analysis to shadow profiles simultaneously cast onto a scene and a calibrated plane as observed by a single video camera. These single-stripe spacetime analysis methods are limited to static scenes and require hundreds of images for reconstructing an object.

In the direction of using fewer images, Sato and Inokuchi [84] describe a set of hierarchical stripe patterns to give range images with $\log N$ images, where N is the number of resolvable stripes. In particular, each camera pixel observes a bit code over time that uniquely determined its correspondence. In [86], Scharstein and Szeliski combine hierarchical stripe patterns and sine wave patterns at different frequencies and phases to obtain accurate depth maps as ground truth for stereo matching benchmarks. They also combine camera-to-camera and camera-to-projector correspondences to obtain more complete shape reconstructions. Both these methods are limited to static scenes. Hall-Holt and Rusinkiewicz [45] describe a method that consists of projected *boundary-coded* stripe patterns that vary over time. By finding nearest stripe patterns over time, a unique code can be determined for any stripe at any time. The constraint in this case is that the object move slowly to avoid erroneous temporal correlations, and only depths at stripe boundaries are measured. Huang et al. [47] implemented the three-step phase shift method [98] sequentially over time using a customized high speed projector. Specifically, they project a sequence of sinusoidal stripe patterns in which each frame is obtained by shifting the previous one by $120°$. A dynamic shape is approximated as being static during every three consecutive frames, and the recovered phase of each pixel can be used to infer the pixel's depth. Note that each of these methods is specially designed for a certain class of patterns and is not applicable under more general illumination changes. The spacetime stereo framework presented here subsumes these previous methods as special cases and extends the ones that only work for static scenes to handle moving scenes.

Concurrently with our work, Davis et al. [31] developed a similar spacetime stereo framework to the one presented here. However, their work is focused on analyzing and presenting results for geometrically static scenes imaged under varying illumination. In this chapter, we describe how to handle both illumination variation and geometrically moving scenes.

3.2 Spacetime stereo metrics



(a) (b) (c)

Figure 3.1: Illustration of spacetime stereo. Two stereo image streams are captured from stationary sensors. The images are shown spatially offset at three different times, for illustration purposes. For a static surface (a,b), the spacetime windows are "straight", aligned along the line of sight. For an oblique surface (b), the spacetime window is horizontally stretched and vertically sheared. For a moving surface (c), the spacetime window is also temporally sheared, i.e., "slanted". The best affine warp of each spacetime window along epipolar lines is computed for stereo correspondence.

In this section, we formulate the spacetime stereo problem, and define the metrics that are used to compute correspondences.

Consider a Lambertian scene observed by two synchronized and pre-calibrated video cameras.[2] Spacetime stereo takes as input the two rectified image streams $I_l(x, y, t)$ and $I_r(x, y, t)$ from these cameras. To recover the time-varying 3D structure of the scene, we wish to estimate the disparity function $d(x, y, t)$ for each pixel (x, y) in the left image at each time t. Many existing stereo algorithms solve for $d(x, y, t)$ at some position and moment (x_0, y_0, t_0) by minimizing the following error function

$$E(d_0) = \sum_{(x,y) \in W_0} e(I_l(x, y, t_0), I_r(x - d_0, y, t_0)) \tag{3.1}$$

where d_0 is shorthand notation for $d(x_0, y_0, t_0)$, W_0 is a spatial neighborhood window around (x_0, y_0), and $e(p, q)$ is a similarity measure between pixels from two cameras. Depending on the specific algorithm, the size of W_0 can vary from being a single pixel to, say, a 10x10 neighborhood, or can be adaptively estimated for each pixel [55]. A common choice for $e(p, q)$ is simply:

$$e(p, q) = (p - q)^2 \tag{3.2}$$

in which case Eq. (3.1) becomes the standard sum of squared differences (SSD). We have obtained better results in practice by defining $e(p, q)$ to compensate for radiometric differences between the cameras:

$$e(p, q) = (s{\cdot}p + o - q)^2 \tag{3.3}$$

where s and o are window dependent scale and offset constants to be estimated. Other forms of $e(p, q)$ are summarized in [4]. Note that Eq. (3.3) is similar enough to a squared difference metric that, after substituting into Eq. (3.1), we still refer to the result as an SSD formulation.

[2] Here we assume the two cameras are offset horizontally. However our formulation can be used for any stereo pair orientation.

We seek to incorporate *temporal appearance variation* to improve stereo matching and generate more accurate and reliable depth maps. In the next two subsections, we'll consider how multiple frames can help to recover static and nearly static shapes, and then extend the idea for moving scenes.

3.2.1 Static scenes

Scenes that are geometrically static may still give rise to images that change over time. For example, the motion of the sun causes shading variations over the course of a day. In a laboratory setting, projected light patterns can create similar but more controlled changes in appearance.

Suppose that the geometry of the scene is static for a period of time $T_0 = [t_0 - \Delta t, t_0 + \Delta t]$. As illustrated in Figure 3.1(a), we can extend the spatial window to a spatiotemporal window and solve for d_0 by minimizing the following sum of SSD (SSSD) cost function:

$$E(d_0) = \sum_{t \in T_0} \sum_{(x,y) \in W_0} e(I_l(x, y, t), I_r(x - d_0, y, t)) \tag{3.4}$$

This error function reduces matching ambiguity in any single frame by simultaneously matching intensities in multiple frames. Another advantage of the spacetime window is that the spatial window can be shrunk and the temporal window can be enlarged to increase matching accuracy. This principle was originally formulated as spacetime analysis in [27, 54] for laser scanning and was applied by several researchers [11, 79] for structured light scanning. However, they only consider either single-frame spatial window or single-pixel temporal window and their specialized formulation is only effective for single stripe illumination. Here we are casting this principle in a general spacetime stereo framework that is valid for any illumination variation.

We should point out that both Eq. (3.1) and Eq. (3.4) treat disparity as being constant within the window W_0, which assumes the corresponding surface is fronto-parallel. For a static but oblique surface, as shown in Figure 3.1(b), a more accurate (first order) local approximation of the disparity function is

$$d(x, y, t) \approx \hat{d}_0(x, y, t) \overset{\text{def}}{=} d_0 + d_{x_0} \cdot (x - x_0) + d_{y_0} \cdot (y - y_0) \tag{3.5}$$

where d_{x_0} and d_{y_0} are the partial derivatives of the disparity function with respect to spatial coordinates x and y at (x_0, y_0, t_0). This local spatial linearization results in the following SSSD cost function to be minimized:

$$E(d_0, d_{x_0}, d_{y_0}) = \sum_{t \in T_0} \sum_{(x,y) \in W_0} e(I_l(x, y, t), I_r(x - \hat{d}_0, y, t)) \tag{3.6}$$

where $\hat{d}_0$ is a shorthand notation for $\hat{d}_0(x, y, t)$, which is defined in Eq. (3.5) in terms of (d_0, d_{x_0}, d_{y_0}) and is estimated for each pixel. Non-zero d_{x_0} and d_{y_0} will cause a horizontal stretch or shrinkage and vertical shear of the spacetime window respectively, as illustrated in Figure 3.1(b). These window warping effects can also be described by the following equation:

$$\begin{bmatrix} x_r - x_0 \\ y_r - y_0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 - d_{x_0} & -d_{y_0} & -d_0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_l - x_0 \\ y_l - y_0 \\ 1 \end{bmatrix} \tag{3.7}$$

where (x_l, y_l) and (x_r, y_r) are pixel coordinates in the left and right images, respectively.

3.2.2 Quasi-static scenes

The simple SSSD method proposed above can also be applied to an interesting class of time-varying scenes. Some natural scenes, like water flow in Figure 3.7, have spatially varying texture and motion, but an overall shape that is roughly constant over time. Although these natural scenes move stochastically, people tend to fuse the image stream into an average shape over time. We refer to this class of natural scenes as *quasi-static*. By applying the SSSD method from the previous section, we can compute a temporally averaged disparity map which corresponds roughly to the "mean shape" of the scene. In graphics applications where a coarse geometry is sufficient, one could, for instance, use the mean shape as static geometry with time-varying color texture mapped over the surface.

3.2.3 Moving scenes

Next, let's consider the case where the object is moving in the time interval $T_0 = [t_0 - \Delta t, t_0 + \Delta t]$, as illustrated in Figure 3.1(c). Because of the object motion, the window in the left video is deformed in the right sequence. The temporal trajectory of window deformation in the right video is determined by the object motion and could be arbitrarily complex. However, if the camera has a high enough frame rate relative to the object motion and there are no changes in visibility, we can locally linearize the temporal disparity variation in much the same way we linearized spatial disparity in Eq. (3.5). Specifically, we take a first order approximation of the disparity variation with respect to both spatial coordinates x and y and temporal coordinate t as

$$d(x,y,t) \approx \tilde{d}_0(x,y,t) \overset{\text{def}}{=} d_0 + d_{x_0} \cdot (x - x_0) + d_{y_0} \cdot (y - y_0) + d_{t_0} \cdot (t - t_0) \tag{3.8}$$

where d_{t_0} is the partial derivative of the disparity function with respect to time at (x_0, y_0, t_0). This local spatial-temporal linearization results in the following SSSD cost function to be minimized:

$$E(d_0, d_{x_0}, d_{y_0}, d_{t_0}) = \sum_{t \in T_0} \sum_{(x,y) \in W_0} e(I_l(x,y,t), I_r(x - \tilde{d}_0, y, t)) \tag{3.9}$$

where $\tilde{d}_0$ is a shorthand notation for $\tilde{d}_0(x,y,t)$, which is defined in Eq. (3.8) in terms of $(d_0, d_{x_0}, d_{y_0}, d_{t_0})$ and is estimated for each pixel at each time. Note that Eq. (3.8) assumes a linear model of disparity within the spacetime window, i.e., $(d_0, d_{x_0}, d_{y_0}, d_{t_0})$ is constant within $W_0 \times T_0$.

We use the term *straight window* to refer to a spacetime window whose position and shape is fixed over time, such as the windows shown in Figure 3.1(a,b). If the position of the window varies over time, we say that the spacetime window is *slanted*, such as the one in the right camera in Figure 3.1(c). Similar to Eq. (3.7), the window warping in this case can be characterized by the follow equation:

$$\begin{bmatrix} x_r - x_0 \\ y_r - y_0 \\ t_r - t_0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 - d_{x_0} & -d_{y_0} & -d_{t_0} & -d_0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_l - x_0 \\ y_l - y_0 \\ t_l - t_0 \\ 1 \end{bmatrix} \tag{3.10}$$

where (x_l, y_l, t_l) and (x_r, y_r, t_r) are pixel coordinates in the left and right videos, respectively.

In its current formulation, spacetime stereo requires matching a straight spacetime window around each pixel (x, y) at each time t in the left image stream to, in general, a slanted spacetime window in the right image stream. It would be straightforward to make the metric more symmetric by matching a slanted window in the left to a slanted window in the right through the minimization of the following objective function:

$$E(d_0, d_{x_0}, d_{y_0}, d_{t_0}) = \sum_{t \in T_0} \sum_{(x,y) \in W_0} e(I_l(x + \frac{\tilde{d}_0}{2}, y, t), I_r(x - \frac{\tilde{d}_0}{2}, y, t)) \tag{3.11}$$

In practice, however, we have found this symmetric formulation results in similar reconstruction results as the asymmetric formulation, Eq. (3.9), does. Therefore, we use Eq. (3.9) as our general objective function throughout this chapter.

3.3 Spacetime stereo matching

A wide variety of existing stereo algorithms are easily adapted for spacetime stereo. Most stereo algorithms have a "data term" $C(x_l, x_r; y, t)$ that describes the similarity of the pixel or region around (x_l, y, t) in the left image and (x_r, y, t) in the right. To use these algorithms for spacetime stereo, one can simply replace the data term with

$$C(x_l, x_r; y, t) = \min_{d_x, d_y, d_t} E(x_l - x_r, d_x, d_y, d_t) \tag{3.12}$$

Eq. (3.12) can be incorporated as the cost function in most existing stereo algorithms, e.g., window-based correlation, dynamic programming, graph cuts, and so forth (see [85] for a description and evaluation of these methods). For our experiments, we used dynamic programming to compute a discrete correspondence followed by Lucas-Kanade registration [63] to obtain floating point disparities. The dynamic programming method has been presented in Section 2.2, and here we focus on extending the Lucas-Kanade method in spacetime.

3.3.1 The Lucas-Kanade method in spacetime

Lucas and Kanade [63] first applied the Gauss-Newton nonlinear least square method to the image registration problem. Here we first briefly summarize the Gauss-Newton method.

The Gauss-Newton method seeks to estimate a parameter vector $\mathbf{p}$ by minimizing a sum of squared residual errors $\{r_i\}$, which are all functions of $\mathbf{p}$, defined as:

$$E(\mathbf{p}) = \sum_i r_i^2(\mathbf{p}) \tag{3.13}$$

It starts with an initial value $\mathbf{p}_0$ and iteratively minimizes E by updating $\mathbf{p}_k = \mathbf{p}_{k-1} + \delta\mathbf{p}_k$. $\delta\mathbf{p}_k$ is computed by first approximating $E(\mathbf{p})$ near $\mathbf{p}_{k-1}$ as a quadratic function:

$$\begin{aligned} E(\mathbf{p}_{k-1} + \delta\mathbf{p}) &= \sum_i r_i^2(\mathbf{p}_{k-1} + \delta\mathbf{p}) \\ &\approx \sum_i \left(r_i(\mathbf{p}_{k-1}) + \frac{\partial r_i}{\partial \mathbf{p}}(\mathbf{p}_{k-1})^{\mathrm{T}} \delta\mathbf{p} \right)^2 \\ &= \sum_i r_i(\mathbf{p}_{k-1})^2 + 2r_i(\mathbf{p}_{k-1}) \frac{\partial r_i}{\partial \mathbf{p}}(\mathbf{p}_{k-1})^{\mathrm{T}} \delta\mathbf{p} + \delta\mathbf{p}^{\mathrm{T}} \frac{\partial r_i}{\partial \mathbf{p}}(\mathbf{p}_{k-1}) \frac{\partial r_i}{\partial \mathbf{p}}(\mathbf{p}_{k-1})^{\mathrm{T}} \delta\mathbf{p} \\ &= c + 2\mathbf{g}^{\mathrm{T}} \delta\mathbf{p} + \delta\mathbf{p}^{\mathrm{T}} \mathbf{H} \delta\mathbf{p} \end{aligned} \tag{3.14}$$

where

$$\begin{aligned} \mathbf{H} &= \sum_i \frac{\partial r_i}{\partial \mathbf{p}}(\mathbf{p}_{k-1}) \frac{\partial r_i}{\partial \mathbf{p}}(\mathbf{p}_{k-1})^{\mathrm{T}} \\ \mathbf{g} &= \sum_i \frac{\partial r_i}{\partial \mathbf{p}}(\mathbf{p}_{k-1}) r_i(\mathbf{p}_{k-1}) \\ c &= \sum_i r_i(\mathbf{p}_{k-1})^2 \end{aligned} \tag{3.15}$$

To minimize Eq. (3.14), the optimal update $\delta \mathbf{p}_k$ is

$$\delta \mathbf{p}_k = \arg\min_{\delta \mathbf{p}} E(\mathbf{p}_{k-1} + \delta \mathbf{p}) = -\mathbf{H}^{-1}\mathbf{g} \tag{3.16}$$

In the context of spacetime stereo, the parameter $\mathbf{p}$ is the disparity and its derivatives in space and time, i.e., $\mathbf{p} = [d_0, d_x, d_y, d_t]^{\mathrm{T}}$. If the standard SSD is used as the error metric, the error term $r_i(\mathbf{p})$ for pixel (x_i, y_i, t_i) within the window centered at (x_0, y_0, t_0) is

$$r_i(\mathbf{p}) = I_l(x_i, y_i, t_i) - I_r(x_i - [d_0, d_x, d_y, d_t] \begin{bmatrix} 1 \\ x_i - x_0 \\ y_i - y_0 \\ t_i - t_0 \end{bmatrix}, y_i, t_i) \tag{3.17}$$

In this case,

$$\frac{\partial r_i}{\partial \mathbf{p}} = \begin{bmatrix} 1 \\ x_i - x_0 \\ y_i - y_0 \\ t_i - t_0 \end{bmatrix} \frac{\partial I_r}{\partial x} \tag{3.18}$$

and $\mathbf{H}$ and $\mathbf{g}$ can be computed by substituting Eq. (3.17) and Eq. (3.18) into Eq. (3.14). The SSD error metric assumes brightness constancy across two views, which is not exactly satisfied in practice, due to radiometric differences of cameras and non-lambertian reflectance of the surface. To compensate for these factors, we can use the gain-bias error metric as

$$r_i(\mathbf{p}) = I_l(x_i, y_i, t_i) - s \cdot I_r(x_i - [d_0, d_x, d_y, d_t] \begin{bmatrix} 1 \\ x_i - x_0 \\ y_i - y_0 \\ t_i - t_0 \end{bmatrix}, y_i, t_i) - o \tag{3.19}$$

In this case,

$$\frac{\partial r_i}{\partial \mathbf{p}} = \begin{bmatrix} 1 \\ x_i - x_0 \\ y_i - y_0 \\ t_i - t_0 \end{bmatrix} s\frac{\partial I_r}{\partial x} - I_r \frac{\partial s}{\partial \mathbf{p}} - \frac{\partial o}{\partial \mathbf{p}} \tag{3.20}$$

and **H** and **g** can be computed by substituting Eq. (3.19) and Eq. (3.20) into Eq. (3.14). However in this case, we need to know s and o and their derivatives with respect to the disparity parameters $\frac{\partial s}{\partial \mathbf{p}}$ and $\frac{\partial o}{\partial \mathbf{p}}$. s and o can be computed in closed form given $\mathbf{p}_{k-1}$:

$$\begin{bmatrix} s \\ o \end{bmatrix} = \frac{1}{N\sum_i I_r^2 - (\sum_i I_r)^2} \begin{bmatrix} N\sum_i I_l I_r - (\sum_i I_l)(\sum_i I_r) \\ \sum_i I_l \sum_i I_r^2 - (\sum_i I_r)(\sum_i I_l I_r) \end{bmatrix} \tag{3.21}$$

where N is the number of pixels in the window.

From Eq. (3.21), we can compute the derivatives of s and o with respect to disparity parameter **p**.

$$\begin{aligned} \begin{bmatrix} \frac{\partial s}{\partial \mathbf{p}} \\ \frac{\partial o}{\partial \mathbf{p}} \end{bmatrix} &= \frac{1}{N\sum_i I_r^2 - (\sum_i I_r)^2} \begin{bmatrix} N\sum_i I_l \frac{\partial I_r}{\partial \mathbf{p}} - (\sum_i I_l)(\sum_i \frac{\partial I_r}{\partial \mathbf{p}}) \\ 2\sum_i I_l \sum_i I_r \frac{\partial I_r}{\partial \mathbf{p}} - (\sum_i \frac{\partial I_r}{\partial \mathbf{p}})(\sum_i I_l I_r) - (\sum_i I_r)(\sum_i I_l \frac{\partial I_r}{\partial \mathbf{p}}) \end{bmatrix} \\ &- 2\frac{N\sum_i I_r \frac{\partial I_r}{\partial I_l} - (\sum_i I_r)(\sum_i \frac{\partial I_r}{\partial I_l})}{(N\sum_i I_r^2 - (\sum_i I_r)^2)^2} \begin{bmatrix} N\sum_i I_l I_r - (\sum_i I_l)(\sum_i I_r) \\ \sum_i I_l \sum_i I_r^2 - (\sum_i I_r)(\sum_i I_l I_r) \end{bmatrix} \end{aligned} \tag{3.22}$$

In summary, we perform the spacetime stereo matching as follows: given $\mathbf{p}_{k-1}$ (initialized from dynamic programming), we compute s, o, $\frac{\partial s}{\partial \mathbf{p}}$, and $\frac{\partial o}{\partial \mathbf{p}}$ using Eq. (3.21) and Eq. (3.22). Then we compute **H** and **g** using Eqs. (3.15, 3.19, 3.20) and update $\mathbf{p}_{k-1}$ and using Eq. (3.16). We iterate the above procedure a few times (5 in our experiments). In the case that the gain and bias are not modeled, we set $s = 1$, $o = 0$, $\frac{\partial s}{\partial \mathbf{p}} = \mathbf{0}$, and $\frac{\partial o}{\partial \mathbf{p}} = \mathbf{0}$. Notice that, in this case, Eq. (3.19) and Eq. (3.20) are the same as Eq. (3.17) and Eq. (3.18), respectively.

In some of the experiments, we assume certain disparity derivatives are zeros.

For example, in the case of static cases, we assume $d_t = 0$ and the spacetime stereo matching can be performed simply by throwing out the fourth row and column in $\mathbf{H}$ and the fourth element in $\mathbf{g}$ and solving for a 3 by 3 linear system. In the case of quasi-static scenes with insufficient texture, we assume $d_x = d_y = d_t = 0$ and throw out the second, third, and fourth rows and columns in $\mathbf{H}$ and the second, third, and fourth elements in $\mathbf{p}$. In this case, Eq. (3.16) degenerates to a scalar equation.

3.4 Results

We performed several experiments to evaluate the performance of spacetime stereo for static, quasi-static, and moving scenes. In each case, we performed camera calibration using Bouguet's calibration toolbox [12].

3.4.1 Static Scenes

To test the performance with static scenes, we applied varying illumination to induce appearance changes. In the first experiment, we employed a stereo pair of synchronized Basler A301f monochrome video cameras and projected light onto a scene with a 60 Hz Compaq MP1800 digital projector. We defer the description of the experimental to Chapter 5. Spacetime stereo places no restriction on the type of projected pattern. For instance, a hierarchical stripe (Gray code) pattern [84], as shown in Figure 3.2(a) should yield reasonable depth maps. In practice, we obtained more accurate results by projecting patterns with high spatial frequencies in each frame. Intuitively, this happens because the coarse patterns, while providing some disambiguation cues for the finer patterns, give little detailed information for disparity matching. Our approach then is to take the Gray code and randomly shuffle the temporal columns so that each is still unique, but each resulting horizontal pattern has high frequency spatial detail (Figure 3.2(b)).

Our pattern sequence is comprised of 8 patterns, each of which has 256 4-pixel-

	stripe intensities															
t=0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
t=1	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0
t=2	0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0
t=3	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0

(a) Gray code

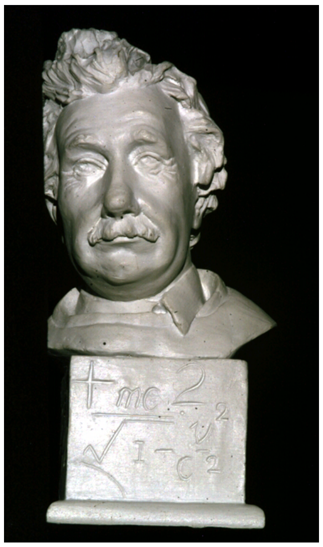
	stripe intensities															
t=0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
t=1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1
t=2	0	1	0	1	1	0	1	0	0	1	0	1	1	0	1	0
t=3	1	0	1	0	1	0	1	0	0	1	0	1	0	1	0	1

(b) Modified Gray code

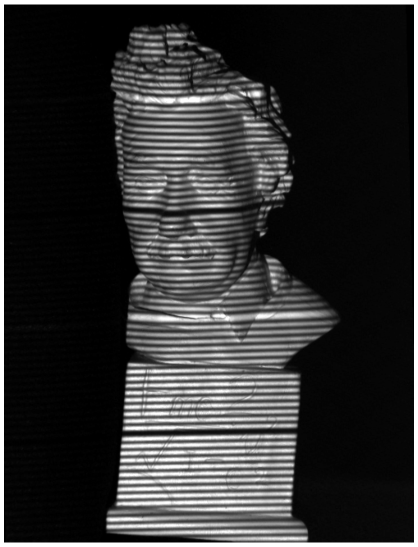
Figure 3.2: Structured light illumination patterns. Each pattern consists of a set of black and white vertical stripes, defined by the sequence of zeros and ones shown above. At each time instant, a different pattern is displayed. (a) A Gray code pattern for 16 stripes. Each column is unique, but some rows contain only low spatial frequencies. (b) After shuffling columns of the Gray code, we obtain a modified pattern that still has unique columns, but also has high spatial frequencies in each row.

wide stripes. Each pattern is low-pass filtered (using Gaussian kernel $\sigma = 1$) to obtain patterns with continuous intensity variation. This gentle low-pass filtering avoids the aliasing problem of imaging the hard stripe boundaries and results in more accurate per-pixel matching.

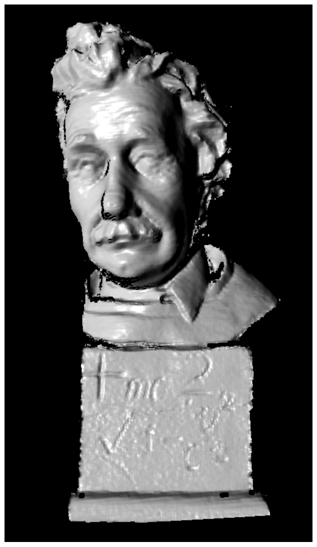
Figure 3.3 shows the results obtained by imaging a small sculpture (a plaster bust of Albert Einstein) and using 10 images taken from each camera matched with a spacetime window of 5x5x10 (5x5 pixels per frame by 10 frames). The shaded rendering reveals details visually comparable to those obtained with a laser range scanner. Eq. (3.3) is used as the pixel similarity measure in the Lucas-Kanade refinement for



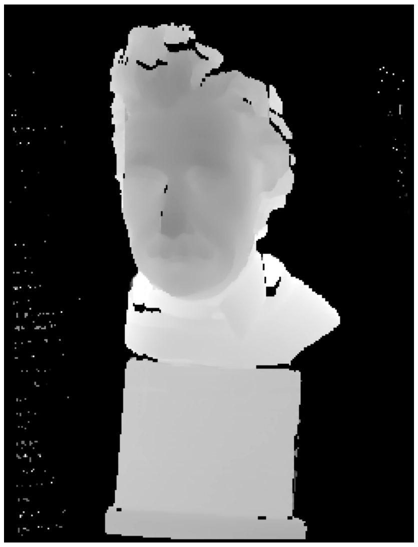
(a)



(b)



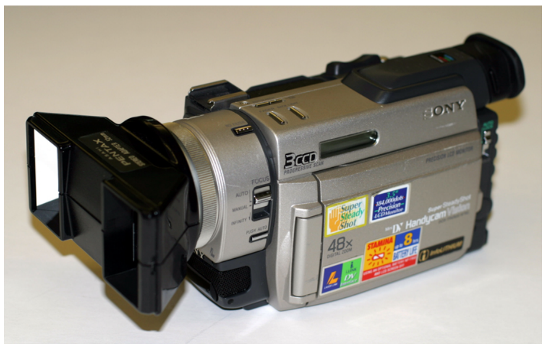
(c)



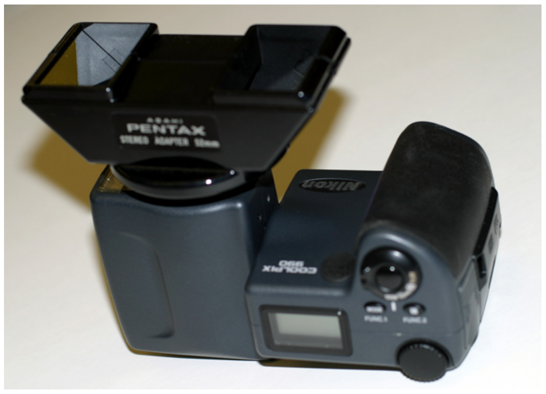
(d)

Figure 3.3: Structured light result with modified Gray code. (a) Einstein bust under natural lighting. (b) One image taken from the set of 10 stereo pair images. The images were taken with the bust laying on its side, but the one shown here has been rotated 90 degrees. (c) Shaded rendering of geometric reconstruction. (d) Reconstructed disparity map.

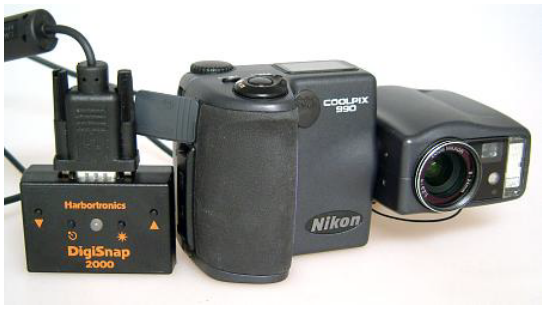
(a)



(b)



(c)

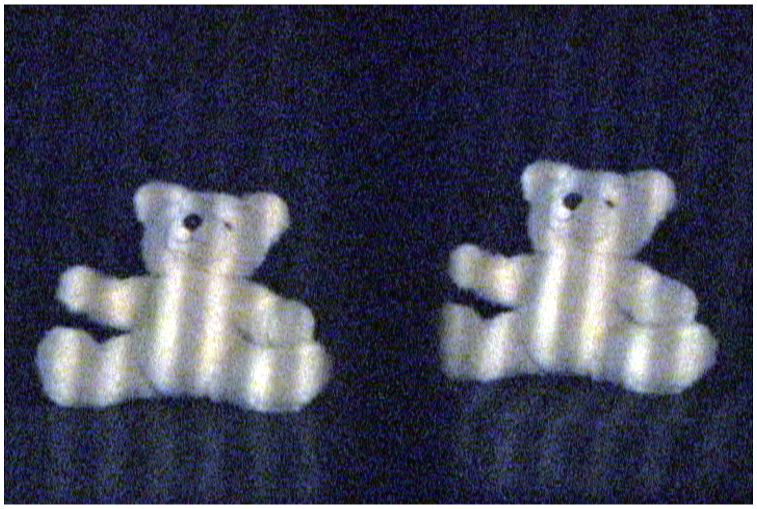


(d)

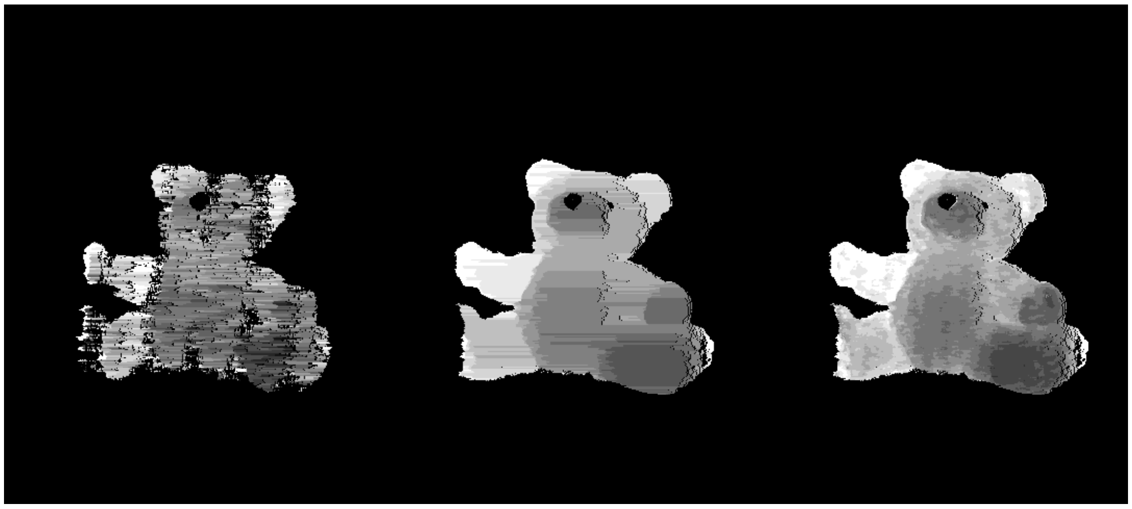
Figure 3.4: Single camera stereo with mirror adapter. (a) The Pentax stereo mirror adapter. (b) Sony TRV-900 camcorder mounted with the stereo adapter. (c) Nikon Coolpix-990 camera mounted with he stereo adatper. (d) Nikon Coolpix-990 with DigiSnap 2000 time lapse controller.

this example (also in the moving hand and deforming face examples later).

Next, we tried a simpler imaging system for static shape capture using much looser structured light. For the stereo pair, we attached a Pentax stereo adapter to a single SONY-TRV900 camcorder, as shown in Figure 3.4(a,b). For illumination, we shined an ordinary desk lamp through a transparency printed with a black and white square wave pattern onto the subject (a teddy bear) and moved the pattern by hand in a free-form fashion. In this case, we captured 125 frames and tried both single frame stereo for one of the image pairs using a 5x1 window and spacetime stereo over all frames



(a)



(b) (c) (d)

Figure 3.5: Loosely structured light using transparency and desk lamp. (a) One out of the 125 stereo pair images. (b) Disparity map for a traditional stereo reconstruction. (c) Disparity map for spacetime stereo after dynamic programming. (d) Same as (c) after Lucas-Kanade refinement.

using a 5x1x125 window. In both cases (also in the waterfall example later), spatial disparity variation is ignored within the windows, i.e., $d_x = d_y = 0$, and Eq. (3.2) is used as the pixel similarity measure. Figure 3.5 shows marked improvement of

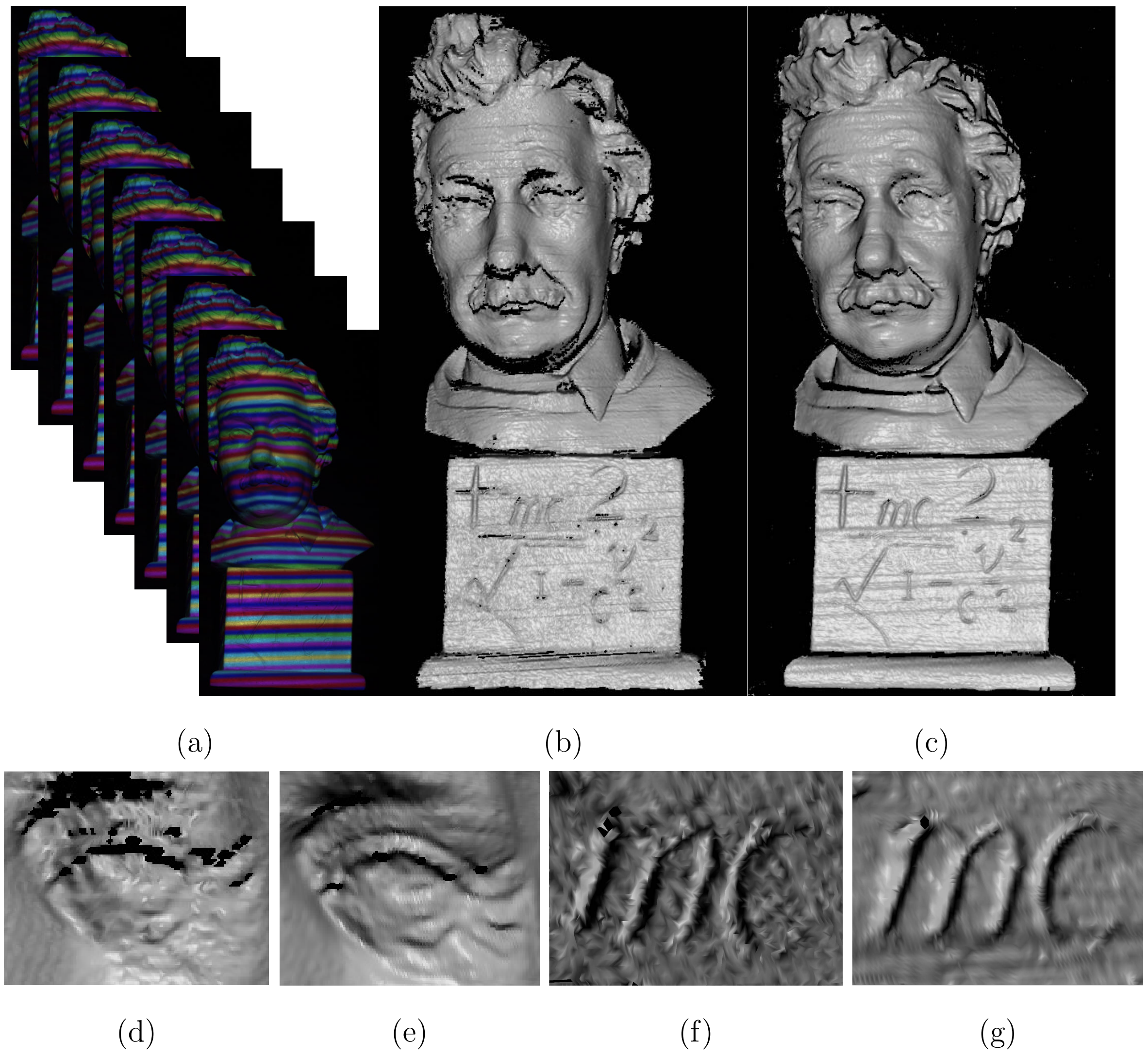


Figure 3.6: Comparison between multiple one-shots and spacetime analysis. (a) Stack of 7 stripe images taken of the Einstein bust for use with spacetime analysis. (b) Shaded rendering of reconstruction produced by combining 7 one-shot results (using shifted one-shot patterns). (c) Shaded rendering of reconstruction produced by spacetime analysis (using the patterns in (a)). (d) and (e) Renderings of the left eye (on right side of the (b) and (c) images) using multiple one-shots and spacetime, respectively. Notice the improved resolution in the wrinkles under spacetime. (f) and (g) Renderings of the letters "mc" on the base of the bust using multiple one-shots and spacetime, respectively. Notice the crisper, less noisy reconstruction under spacetime.

spacetime stereo over regular stereo, in addition to improvement due to the final Lucas-Kanade subpixel refinement step.

Matching between a projector and a camera

In the previous experiments, we sought to establish pixel correspondences between two camera images using the spacetime stereo method. However, spacetime stereo can also be used to compute correspondence between a projector and a camera, as shown in Figure 3.6. In this case, we project a shifted sequence of 7 patterns onto the Einstein bust. For comparison with a non-spacetime method, we first choose the same pattern as in the one-shot method described in Chapter 2, shift the pattern by one pixel 7 times, and independently estimate a range map for each image. We then combine these range maps into a single high resolution range map as shown. For the spacetime method, we blur the same pattern with a Gaussian filter ($\sigma = 1.5$ pixels), shift it by two pixels at a time, and perform the reconstruction using spacetime stereo. Specifically, we directly match the projected pattern and the observed image using the gain-bias matching metric, over a $1 \times 1 \times 7$ spacetime window, to compensate for the intensity differences between them due to surface albedo. As the figure shows, the spacetime method does a substantially better job of resolving fine detail. In particular, the edge detection method used in the one-shot technique is susceptible to the rapid shading changes in high curvature areas, whereas the spacetime technique is much less so. To evaluate the reconstruction accuracy, we select a planar region on the base of the Einstein bust and fit a plane to the region. The deviation of the plane-fitting residuals for the multiple one-shot reconstruction is 0.18mm, while spacetime method significantly reduce this deviation to 0.048mm.

3.4.2 Quasi-static scenes

For a quasi-static scene, we took a sequence of 45 images of a small but fast-moving waterfall using a Nikon Coolpix 900 still camera with the stereo adapter attached,

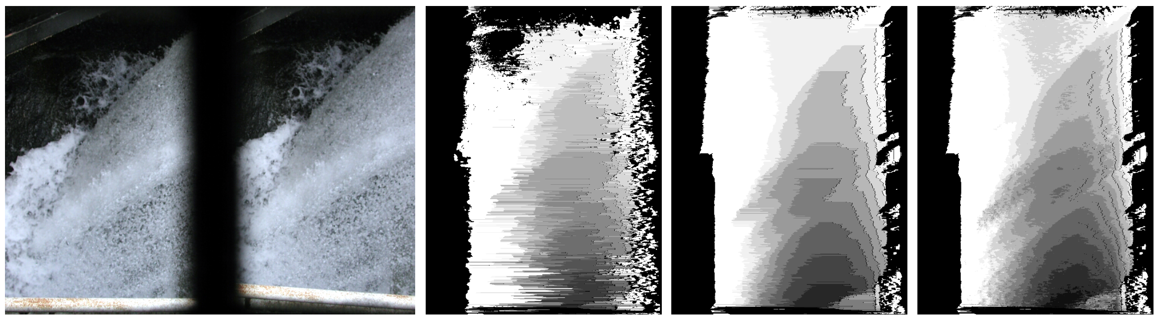


Figure 3.7: Quasi-static rushing water experiment. (a) One out of the 45 stereo pair images. (b) Disparity map for a traditional stereo reconstruction. (c) Disparity map for spacetime stereo after dynamic programming. (d) Same as (c) after Lucas-Kanade refinement.

shown in Figure 3.4(c). The camera is controlled by a time lapse trigger device, Digisnap 2000, as shown in Figure 3.4(d). Figure 3.7 shows a comparison of the results obtained with traditional stereo for one of the image pairs, followed by results obtained with the same spacetime stereo technique used for the teddy bear in Figure 3.5. Note how much more consistent and complete the spacetime result is.

3.4.3 Moving Scenes

For moving scenes, we tried two experiments. In the first, we projected the modified Gray code pattern onto a human hand that is moving fairly steadily away from the stereo rig. In this case, the disparity is changing over time, and the straight spacetime window approach fails to reconstruct a reasonable surface. By estimating the temporal derivative of the disparity using slanted windows, we obtain a much better reconstruction, as shown in Figure 3.8.

We also tried comparing spacetime stereo and one-shot stereo for moving scenes. In this case, a short sequence of stripe patterns is generated randomly and projected repeatedly to a deforming human face. To demonstrate the effect of extruding the

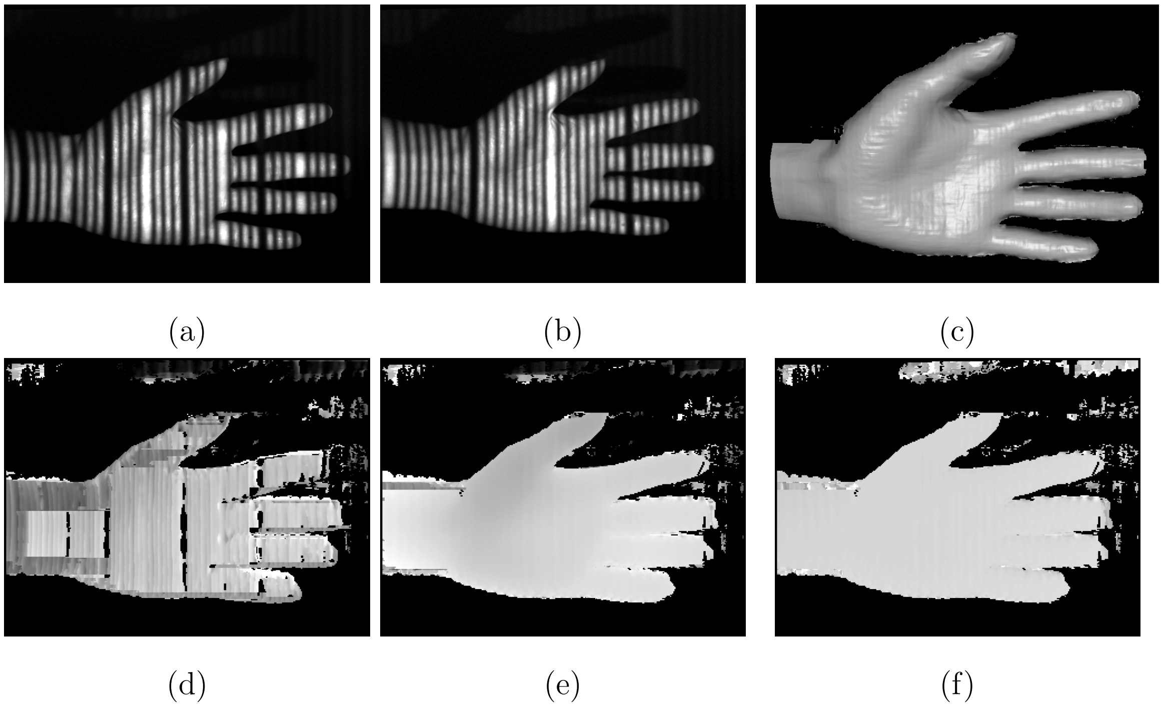


Figure 3.8: Moving hand under structured light. (a), (b) Two images taken from one of the cameras. The hand is moving away from the stereo rig, which is why it is getting smaller. (c) Shaded rendering of the reconstructed model using slanted window spacetime analysis. (d) Disparity map with straight spacetime windows. (e) Disparity map with slanted spacetime windows. (f) Temporal derivative of disparity. Since the hand is translating at roughly a constant speed, the disparity velocity is fairly constant over the hand.

matching window over time, we choose a $9 \times 5 \times 5$ spacetime window and a 15×15 spatial window, which have the same number of pixels, to compare spacetime stereo and frame-by-frame stereo on reconstructing facial expressions. Notice that the spacetime reconstruction generates significantly more accurate results, as shown in Figure 3.9. On our website,

`http://grail.cs.washington.edu/projects/ststereo/`,

we also show this comparison in video format. Notice that spacetime stereo results in temporally much more stable reconstruction than frame-by-frame stereo.

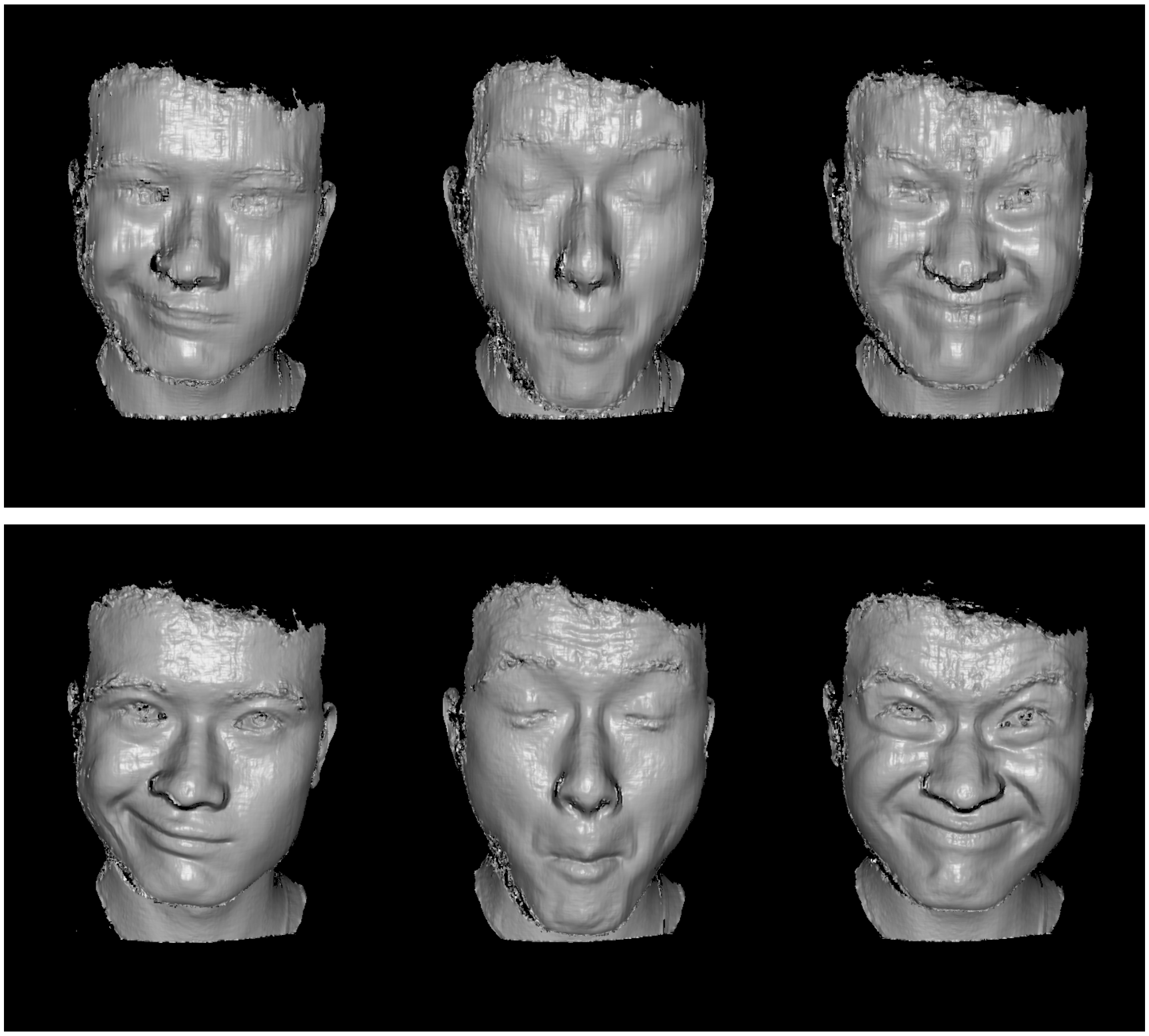


Figure 3.9: Comparison of spacetime and frame-by-frame stereo for a moving face reconstruction. Top row: five face expressions reconstructed using a 15×15 spatial window. Bottom row: the same face expressions reconstructed using a $9 \times 5 \times 5$ spacetime window. Notice that although both the spatial and spacetime windows have the same number of pixels, spacetime stereo results in a more detailed reconstruction.

Finally, we tried imaging moving and deforming textured objects under more natural lighting conditions. In practice, we found that spacetime stereo performed roughly the same as regular stereo. In the next section, we offer an explanation of why we did not observe substantial improvement with spacetime stereo under natural illumination.

3.5 Discussion

In this chapter, we described a simple extension to traditional stereo that incorporates temporal information into the disparity estimation problem. Here we discuss the "operating range" of the technique and suggest ideas for future work.

For geometrically static scenes, the spacetime stereo framework has proved effective for reconstructing shape from changes in appearance. We have demonstrated results for tightly controlled illumination changes, as well as more loosely controlled variations, both in a laboratory setting. The framework subsumes previous structured light triangulation methods as special cases; when implemented as triangulation between two cameras, it has the advantage that the illumination needs not be calibrated. Further, the approach is general enough to handle more natural appearance variations including shadows sweeping across architecture and objects that change texture naturally over time (e.g., metallic patinas, food that browns with age, ants crawling over surfaces, etc.). We conjecture that the approach should also demonstrate some small amount of improvement over per-frame stereo for image streams of geometrically static scenes with no temporal variations, because the influence of noise should be averaged out over time.

For quasi-static scenes, we have also demonstrated improvement over per-frame stereo. Instead of using our spacetime approach, one could compute the disparities per-frame and then average them together, though we would expect a number of outliers and non-matches to complicate this process, and it would actually be slower to execute because of the need to perform stereo matching n times for n frames instead of only once. An area for future work would be to try to model the statistical variation of quasi-static and moving scenes, e.g., to model the stochastic changes in

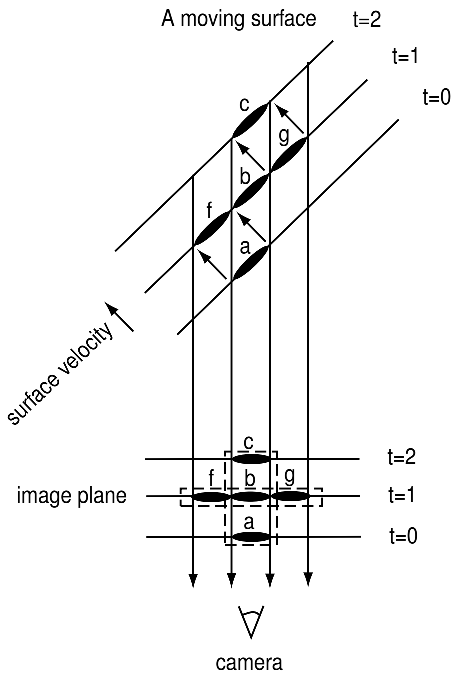


Figure 3.10: Trade-off between spatially narrow windows for spacetime stereo and wide windows for traditional stereo.

disparity for a waterfall.

For dynamic scenes, our most compelling results have been for structured light systems. In this setting, the fact that we do not perform motion stereo analysis (i.e., 2D optical flow in tandem with stereo) is essential, since the brightness constancy constraint does not apply. Our results indicate that dense structured light stereo is possible even as the subject moves.

For dynamic scenes under more natural (smoother) and constant illumination, we have observed less benefit with the spacetime stereo method. Let's consider in particular a scene with constant lighting and with Lambertian reflectance at each surface point. In this case, consider the trade-off between using a wider spatial window and a narrower (but temporally longer) spacetime window, as illustrated in Figure 3.10. The spacetime window $\{a, b, c\}$ over time $t = 0, 1, 2$ contains the same information as the spatial window $\{f, b, g\}$ at time $t = 1$ because the window f at $t = 1$ is the same as window a at $t = 0$ and window c at $t = 2$ is the same as window g at $t = 1$. Therefore, using a spacetime window is not always more powerful than a purely spatial window. An existing frame by frame stereo matching algorithm with a spatial window size $\{f, b, g\}$ is expected to have the same performance as spacetime stereo with spacetime window $\{a, b, c\}$.

The above analysis of the spacetime window tradeoff depends on the direction of surface motion. For example, if the surface moves straight back along the light of sights of the camera, increasing the temporal window size does not include new appearance changes within the spacetime window, although we conjecture before that spacetime stereo in this case may slightly improve the reconstruction by averaging out imaging noise within the spacetime window. An area for future work is to formalize these reasonings and extend the adaptive window method [55] to search for optimal spacetime windows. Finally, another interesting direction would be to adapt global optimization methods like graph cuts [57] to compute depth maps that are piecewise smooth both in space and time.

Chapter 4

GLOBALLY CONSISTENT SPACETIME STEREO

One major artifact of spacetime stereo matching is that it produces quite noticeable ridging artifacts, evident in Figure 4.1(e). Our study indicates that these artifacts are due primarily to the fact that Eq. (3.9) is minimized for each pixel independently, without taking into account constraints between neighboring pixels. Specifically, computing a disparity map with M pixels introduces $4M$ unknowns: M disparities and $3M$ disparity gradients. While this formulation results in a system that is convenient computationally, it is clearly over-parameterized, since the $3M$ disparity gradients are a function of only M disparities. Indeed, the estimated disparity gradients may not agree with the estimated disparities. For example, $d_x(x, y, t)$ may be quite different from its central difference approximation, $\frac{1}{2}(d(x+1, y, t) - d(x-1, y, t))$, because $d_x(x, y, t)$, $d(x+1, y, t)$, $d(x-1, y, t)$ are independently estimated for each pixel. This inconsistency between disparities and disparity gradients results in inaccurate depth maps as shown in Figure 4.1(e,i). In this chapter, we reformulate spacetime stereo as a global optimization problem to overcome this inconsistency deficiency.[1]

4.1 A global optimization formulation for spacetime stereo

The global spacetime stereo method computes the disparity function while taking into account gradient constraints between pixels that are adjacent in space and time. Given image sequences $I_l(x, y, t)$ and $I_r(x, y, t)$, the desired disparity function $d(x, y, t)$

[1]This chapter describes joint work with Noah Snavely, Brian Curless, and Steven M. Seitz, first presented in ACM SIGGRAPH 2004 [102].

minimizes

$$\Gamma(\{d(x,y,t)\}) \triangleq \sum_{x,y,t} E(d, d_x, d_y, d_t) \tag{4.1}$$

subject to the following constraints[2]

$$\begin{aligned} d_x(x,y,t) &= \tfrac{1}{2}(d(x+1,y,t) - d(x-1,y,t)) \\ d_y(x,y,t) &= \tfrac{1}{2}(d(x,y+1,t) - d(x,y-1,t)) \\ d_t(x,y,t) &= \tfrac{1}{2}(d(x,y,t+1) - d(x,y,t-1)) \end{aligned} \tag{4.2}$$

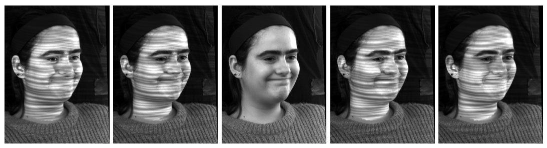
Eq. (4.1) defines a nonlinear least squares problem with linear constraints. We solve this problem using the Gauss-Newton method [70] with a change of variables. To explain our approach, we use $\mathbf{D}$ to denote the concatenation of $d(x,y,t)$ for every (x,y,t) into a column vector. $\mathbf{D}_x$, $\mathbf{D}_y$, and $\mathbf{D}_t$ are defined similarly, by concatenating values of $d_x(x,y,t)$, $d_y(x,y,t)$, and $d_t(x,y,t)$, respectively. Given an initial value of $\mathbf{D}$, $\mathbf{D}_x$, $\mathbf{D}_y$, and $\mathbf{D}_t$, we compute the gradient $\mathbf{q}$ and local Hessian $\mathbf{J}$ of Eq. (4.1) using the Gauss-Newton approximation. Then, the optimal updates $\delta\mathbf{D}$, $\delta\mathbf{D}_x$, $\delta\mathbf{D}_y$, and $\delta\mathbf{D}_t$ are given by

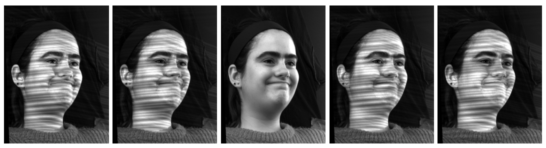
$$\mathbf{JP}\begin{bmatrix} \delta\mathbf{D} \\ \delta\mathbf{D}_x \\ \delta\mathbf{D}_y \\ \delta\mathbf{D}_t \end{bmatrix} = -\mathbf{q} \tag{4.3}$$

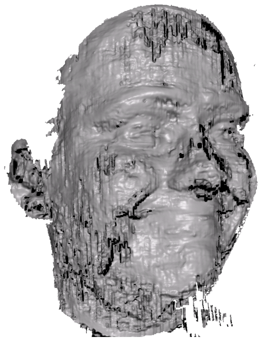
where $\mathbf{J}$ is a block-wise concatenation of the 4 by 4 Hessian matrices $\mathbf{H}$ in Eq. (3.15), $\mathbf{q}$ is a concatenation of the 4 by 1 vector $\mathbf{g}$ in Eq. (3.15),

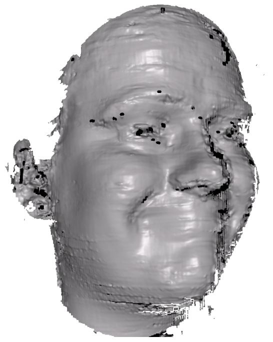
$$\mathbf{J} = \begin{bmatrix} \mathbf{H}_1 & & & \\ & \mathbf{H}_2 & & \\ & & \ddots & \\ & & & \mathbf{H}_M \end{bmatrix} \quad \mathbf{q} = \begin{bmatrix} \mathbf{g}_1 \\ \mathbf{g}_2 \\ \vdots \\ \mathbf{g}_M \end{bmatrix} \tag{4.4}$$

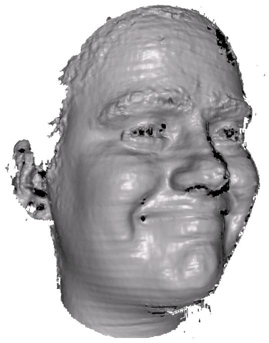
[2]At spacetime volume boundaries, we use forward or backward differences instead of central differences.

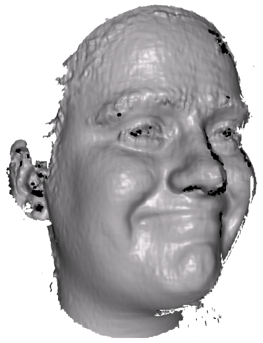

(a)

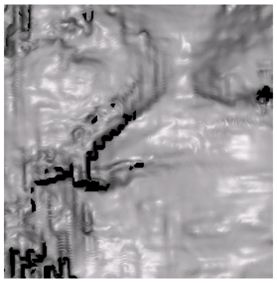

(b)

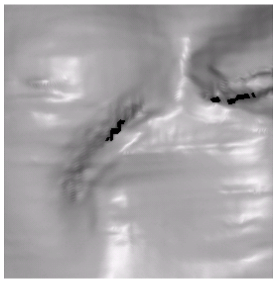

(c)

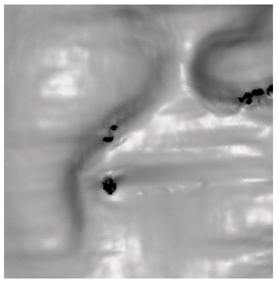

(d)


(e)


(f)


(g)


(h)


(i)

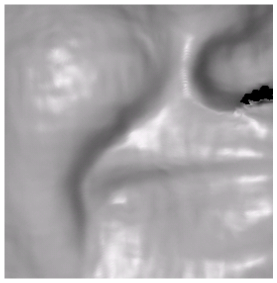

(j)

Figure 4.1: Comparison of four different stereo matching algorithms. (a,b) Five consecutive frames from a pair of stereo videos. The third frames are non-pattern frames. (c) Reconstructed face at the third frame using traditional stereo matching with a [15×15] window. The result is noisy due to the lack of color variation on the face. (d) Reconstructed face at the second frame using stereo matching with a [15×15] window. The result is much better because the projected stripes provide texture. However, certain face details are smoothed out due to the need for a large spatial window. (e) Reconstructed face at the third frame using local spacetime stereo matching with a [9×5×5] window. Even though the third frame has little intensity variation, spacetime stereo recovers more detailed shapes by considering neighboring frames together. However, it also yields noticeable striping artifacts due to the over-parameterization of the depth map. (f) Reconstructed face at the third frame using our new global spacetime stereo matching with a [9×5×5] window. The new method removes most of the striping artifacts while preserving the shape details. (g-j) Closeup comparison of the four algorithms around the nose and the corner of the mouth.

and $\mathbf{P}$ is a permutation matrix that reorders the elements in $\delta\mathbf{D}$, $\delta\mathbf{D}_x$, $\delta\mathbf{D}_y$, and $\delta\mathbf{D}_t$ as follows:

$$\begin{bmatrix} \delta d_1 \\ \delta d_2 \\ \vdots \\ \delta d_M \\ \delta d_{x1} \\ \delta d_{x2} \\ \vdots \\ \delta d_{xM} \\ \delta d_{y_1} \\ \delta d_{y_2} \\ \vdots \\ \delta d_{y_M} \\ \delta d_{t1} \\ \delta d_{t2} \\ \vdots \\ \delta d_{tM} \end{bmatrix} \xrightarrow{\mathbf{P}} \begin{bmatrix} \delta d_1 \\ \delta d_{x1} \\ \delta d_{y_1} \\ \delta d_{t1} \\ \delta d_2 \\ \delta d_{x2} \\ \delta d_{y_2} \\ \delta d_{t2} \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \delta d_M \\ \delta d_{xM} \\ \delta d_{y_M} \\ \delta d_{tM} \end{bmatrix} \tag{4.5}$$

Specifically, $\mathbf{P}$ can be defined as

$$\mathbf{P} = \begin{bmatrix}
1 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\
0 & 0 & \cdots & 0 & 0 & 1 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\
0 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 1 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\
0 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 1 & 0 & \cdots & 0 & 0 \\
0 & 1 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\
0 & 0 & \cdots & 0 & 0 & 0 & 1 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\
0 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 1 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\
0 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 1 & \cdots & 0 & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
0 & 0 & \cdots & 1 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\
0 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 1 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\
0 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 1 & 0 & 0 & 0 & \cdots & 0 & 0 \\
0 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 1 & 0 \\
0 & 0 & \cdots & 0 & 1 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\
0 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & 1 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\
0 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & 1 & 0 & 0 & \cdots & 0 & 0 \\
0 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & 1
\end{bmatrix} \tag{4.6}$$

Since Eqs. (4.2) are linear constraints, we represent them by matrix multiplication:

$$\mathbf{D}_x = \mathbf{F}_x\mathbf{D} \quad \mathbf{D}_y = \mathbf{F}_y\mathbf{D} \quad \mathbf{D}_t = \mathbf{F}_t\mathbf{D} \tag{4.7}$$

where $\mathbf{F}_x$,$\mathbf{F}_y$, and $\mathbf{F}_t$ are sparse matrices encoding the finite difference operations. For example, suppose $d(x, y, t)$ is the i'th component of $\mathbf{D}$, then the only nonzero columns in row i of $\mathbf{F}_x$ are j and j' which correspond to $d(x + 1, y, t)$ and $d(x - 1, y, t)$ and take values of 0.5 and -0.5, respectively. Substituting Eq. (4.7) into Eq. (4.3) and multiplying $\begin{bmatrix} \mathbf{I} \\ \mathbf{F}_x \\ \mathbf{F}_y \\ \mathbf{F}_t \end{bmatrix}^{\mathtt{T}} \mathbf{P}^{\mathtt{T}}$ on both sides, we obtain the optimal update $\delta\mathbf{D}$ by solving

$$\begin{bmatrix} \mathbf{I} \\ \mathbf{F}_x \\ \mathbf{F}_y \\ \mathbf{F}_t \end{bmatrix}^{\mathtt{T}} \mathbf{P}^{\mathtt{T}}\mathbf{J}\mathbf{P} \begin{bmatrix} \mathbf{I} \\ \mathbf{F}_x \\ \mathbf{F}_y \\ \mathbf{F}_t \end{bmatrix} \delta\mathbf{D} = - \begin{bmatrix} \mathbf{I} \\ \mathbf{F}_x \\ \mathbf{F}_y \\ \mathbf{F}_t \end{bmatrix}^{\mathtt{T}} \mathbf{P}^{\mathtt{T}}\mathbf{q} \tag{4.8}$$

where $\mathbf{I}$ is an identity matrix of the same dimension as $\mathbf{D}$. We initialize $\mathbf{D}$ using dynamic programming with the spacetime window metric given in Eq. (3.4),[3] and set $\mathbf{D}_x$, $\mathbf{D}_y$, and $\mathbf{D}_t$ to be zero. Then we iteratively solve Eq. (4.8) and re-compute $\mathbf{J}$ and $\mathbf{q}$, until convergence.

4.1.1 The constraints on boundaries

Eq. (4.2) describes the neighborhood constraints for continuous interior regions. At spacetime volume boundaries and depth discontinuities, the central difference is not well defined; Instead, we use forward or backward differences in these cases. In our implementation, we detect depth discontinuity boundaries after disparity initialization using dynamic programming. Specifically, we define a discontinuity between neighboring pixels if their disparity difference is larger than a certain threshold. In our implementation, the threshold is set to 4. The discontinuity condition is then used to select the finite difference scheme for computing disparity derivatives. For example, we use the forward finite difference for $d_x(x, y, t)$ if the pixel is on the right of a discontinuity. Similarly, we use the backward finite difference for $d_t(x, y, t)$ if the pixel's disparity is discontinuous from its next frame.

[3]When using dynamic programming for initialization, we use a [1×3] image window for frame-by-frame matching and a [1×3×3] window for spacetime matching.

4.1.2 Scalable implementation

Although $\mathbf{D}_x$, $\mathbf{D}_y$, $\mathbf{D}_t$, and $\mathbf{J}$ are very sparse, solving Eq. (4.8) using the conjugate gradient method [76] over the whole video is not practical; a 10-second video of 640×480 resolution at 60Hz comprises nearly 180 million depth variables! To apply global spacetime stereo matching over a long video, we take the follow two strategies.

First, we use the conjugate gradient method [76] to solve Eq. (4.8) iteratively. At each iteration, the conjugate gradient method does not require knowing the product of matrices on the left hand side of $\delta\mathbf{D}$ explicitly; Instead it only needs to evaluate their product with the $\delta\mathbf{D}$. To do so, we only need to save the sub-matrices $\{\mathbf{H}_1, \mathbf{H}_2, \cdots, \mathbf{H}_M\}$ during the iteration, avoiding storing the whole $M \times M$ matrix product.

Second, we divide the video into a 3D $(\mathsf{X}, \mathsf{Y}, \mathsf{T})$ array of blocks that are optimized in sequence. For each block, we load the video data and the associated $\mathbf{H}$ and $\mathbf{g}$ for each pixel into RAM. In practice, we have found the 80×80×90 blocksize works well with our 2G RAM machine. When optimizing a particular block, we treat as boundary conditions the disparity values in its adjacent blocks that have already been optimized. To speed up the procedure, we distribute the computation over multiple machines while ensuring that adjacent blocks are not optimized simultaneously. While many traversal orders are possible, we found that the following simple strategy suffices: We first optimize blocks with odd T values, and distribute blocks with different T values to different CPU's. On each CPU, we traverse the blocks from left to right and top to bottom. We then repeat the same procedure for blocks with even T values. Our prototype implementation takes 2 to 3 minutes to compute a depth map on a 2.8GHz CPU and each depth map contains approximately 120K depth points. Therefore, it takes 20 CPU's about 7.5 hours to finish 3000 frames (a sequence of 50 seconds at 60fps).

4.2 Results

To evaluate our global spacetime stereo method and also demonstrate it on a real application, we built a camera rig to continuously capture dynamic facial performance. Our system takes as input 6 synchronized video streams (4 monochrome and 2 color) running at 60 Hz, and global spacetime stereo is used to recover the time varying 3D facial shapes. The camera rig is shown in Figure 5.2. Three of the cameras capture the left side of the face, and the other three capture the right side. All of the components are off-the-shelf, and their specifics are described in Chapter 5.

To facilitate depth computation, we use two video projectors that project grayscale random stripe patterns onto the face. The projectors send a solid black pattern every three frames, and surface color texture is captured at these frames. Figure 4.1 shows the improvement using global spacetime stereo by comparing it to the spacetime stereo matching method presented in Chapter 3, as well as standard frame-by-frame stereo. On our website,

`http://grail.cs.washington.edu/projects/stfaces/`,

we also provide a video of a pair of depth map sequences reconstructed from both left and right sides of a moving face using global spacetime stereo.

4.3 Discussion

In this chapter, we presented a global spacetime stereo matching algorithm. This method resolves the over-fitting problem of the method presented in Chapter 3, and further improves the result. There are several important topics to explore in future work.

Although global spacetime stereo removes a significant amount of ridging artifacts, some minor ridging artifacts still remain. To quantify the magnitude of these remaining striping artifacts, we reconstruct a planar surface using spacetime stereo and then fit a plane to the reconstruction. The standard deviation of the fitting residual is 0.033mm, which is the approximate range of ridging artifacts and reconstruction noise. Although these artifacts are numerically small, they are visually distracting when the mesh is shaded, because shading a mesh requires computing surface normals and surface normals can vary dramatically even for small surface bumps. As a result, the tiny coherent surface bumps result in a more substantial visual artifact. To further reduce the coherent ridging artifacts, one approach is to choose projection patterns that do not contain coherent structures like stripes. For example, we have tried using random dot patterns for projection and have not observed striping artifacts in the final reconstruction. Some examples are shown in Figure 5.7 and the detailed experiment is described in Section 5.3.2.

Our spacetime stereo matching techniques are based on window warping using a locally linear disparity variation model, which fails near discontinuity boundaries. For example, rapid lip movements during speech result in temporal depth discontinuities in the scanned sequence. It is desirable to improve spacetime stereo algorithms for better performance near spatial and temporal discontinuities. Specifically, the window should be adaptively selected based on local spacetime shape variation. In the extremely case of large shape variation over time due to fast motion, the algorithm should automatically switch to one-shot stereo. Kanade and Okutomi [55] proposed an adaptive window selection theory for one-shot stereo assuming front-parallel disparity model; adaptive window selection for locally linear disparity model in spacetime remains an important topic for future research.

Stereo matching algorithms involve various parameters, such as window sizes in spacetime stereo and regularization weight in Markov Random Field (MRF) stereo. In practice, different data sets require different parameters for optimal performance. Learning parameters automatically is therefore of great importance for these algorithms to be deployed in automated vision systems for real applications. Toward this goal, we address the problem of learning optimal parameters for MRF stereo matching in [101]. How to apply this mechanism to spacetime stereo and other visual reconstruction algorithms is a very interesting problem for future research.

Although the accuracy of spacetime stereo is good enough for several applications, e.g., face modeling and animation, a current limitation is speed. It takes 2-3 minutes to compute a disparity map. Some applications, such as 3D shape recognition of faces, expressions, and poses for natural human computer interfaces, require 3D reconstruction techniques that operate in real time. Techniques that can work both accurately and in real time are not available yet, to our best knowledge. An important future research avenue is to design new techniques and accelerate existing techniques to fulfill both the speed and accuracy requirements for real time 3D reconstruction.

Chapter 5

EXPERIMENTAL STRUCTURED LIGHT STEREO SYSTEMS

In this chapter, we present two stereo systems we built to evaluate the spacetime stereo algorithms presented in this dissertation. First, we describe a binocular stereo system consisting of two video cameras connected to a laptop. Second, we describe a multiple projector-camera system for face capture. Finally, we describe methods of multiplexing shape acquisition and surface texture acquisition.

5.1 Portable two camera stereo system

We first describe the binocular stereo system used to capture the Einstein bust (Figure 3.3), hand (Figure 3.8), and face (Figure 3.9) sequences in Chapter 3. This system uses two Basler A301f Firewire (a.k.a IEEE 1394) video cameras. These cameras output 656×494 monochrome image streams at up to 80 frame per second (fps). Each camera has a Tokina T10Z0814 C-mount lens.

For the purpose of portability, we connect these two cameras to an IBM Thinkpad T40 laptop through two Ratoc CBFW1U PCMCIA Firewire cards,[1] as shown in Figure 5.1. Note that each Firewire card has one Firewire bus and each Firewire bus has a bandwidth of 50MB. The Basler camera's bandwidth requirement is 26MB/s when running at the maximum 80fps frame rate. Therefore the two cameras have to be on two different Firewire cards; attaching two cameras on the same card will saturate

[1]It is reported that Belkin Firewire CardBus Adapter works as well. We found that some Firewire cards, such as the Pyro 1394DV, did not work successfully with Basler cameras.

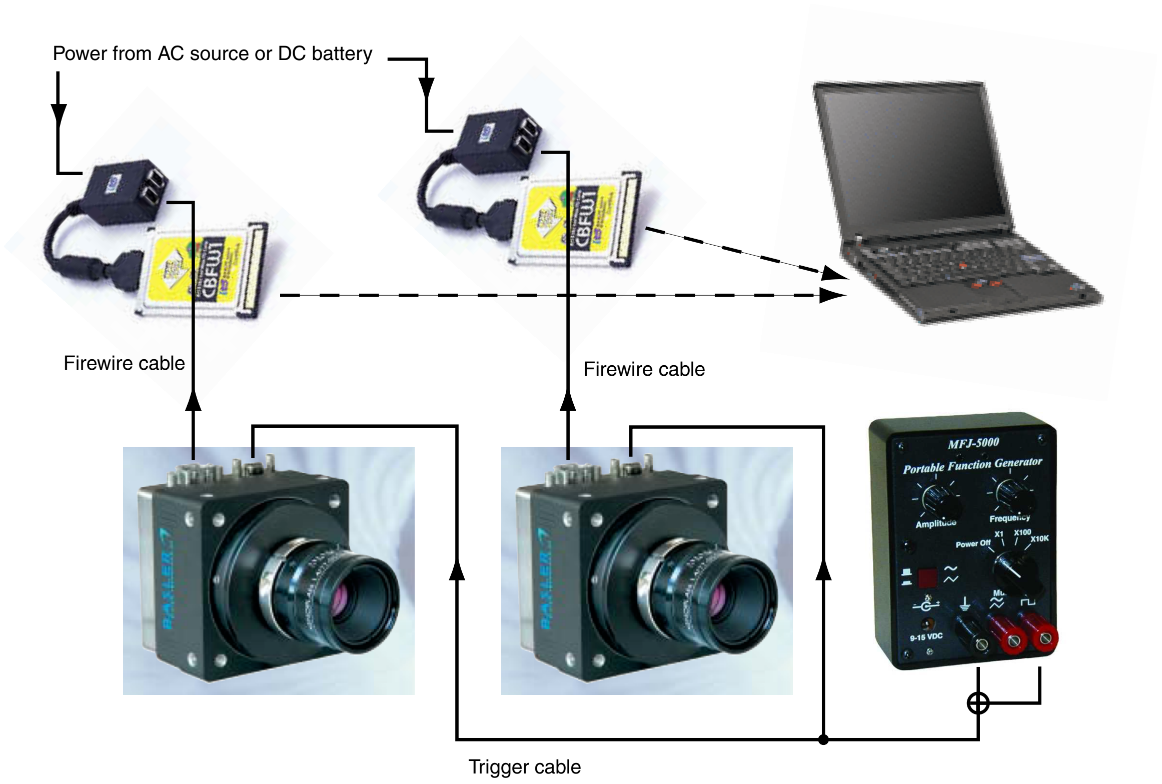


Figure 5.1: The portable stereo system

the Firewire bus. The Ratoc Firewire card requires a separate power source. We used an AC/DC adapter for indoor experiments and a battery, e.g., Bescor NMH-54, for outdoor experiments.

To synchronize the cameras, we trigger the cameras using an external square wave signal. This signal can be from any signal generator. We use an MFJ-5000 portable function generator to create the needed signal.

5.2 A face capture rig

A binocular stereo system can only recover mutually visible portions of an object. To enable more complete shape capture for a face, we built a multi-camera system that enables simultaneously recovering depth maps of left and right sides of the face. We

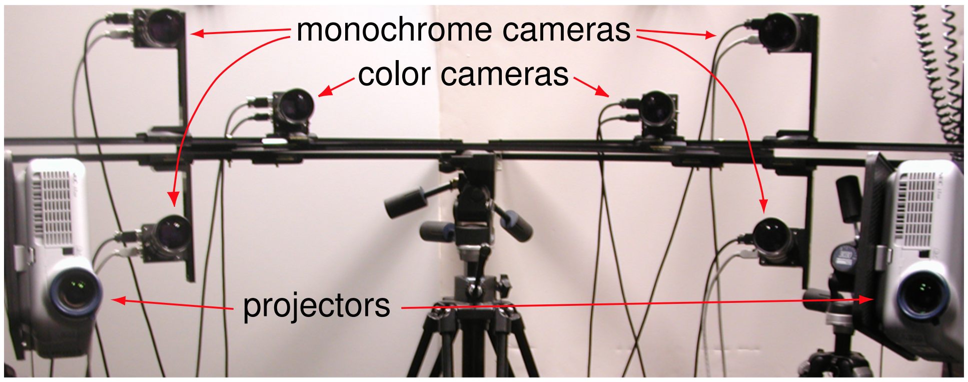


Figure 5.2: Our face capture rig consists of six video cameras and two data projectors. The two monochrome cameras on the left constitute one stereo pair, and the two on the right constitute a second stereo pair. The projectors provide stripe pattern textures for high quality shape estimation. The color cameras record video streams used for optical flow and surface texture.

used this system to record image sequences shown in Chapters 4 and 6. This system employs four Basler A301f cameras to form two stereo pairs. It also uses two NEC LT260K DLP projectors (DLP stands for “Digital Light Processing”) for structured light patterns, as shown in Figure 5.2. One stereo pair and one projector capture shape from left side and the others capture from right side. The cameras are mounted on several baseline bars, ordered from `http://www.stereoscopy.com/jasper/`.

5.2.1 Configuration of the workstations

We use two Dell Precision 650 workstations to receive the image streams. Each workstation has 5 empty PCI slots, as shown in Figure 5.3, which can be used for streaming in image sequences. The configuration of these slots is as follows. Slots #1 and #3 both have a 4601-22 Dual Firewire card and each card has *two* Firewire buses on it. (Note this card is different from the card used for the laptop.) Other slots are left empty. Because each card can be connected to two Basler cameras working at full speed (80fps) and the integrated Firewire port can be connected to one more camera,

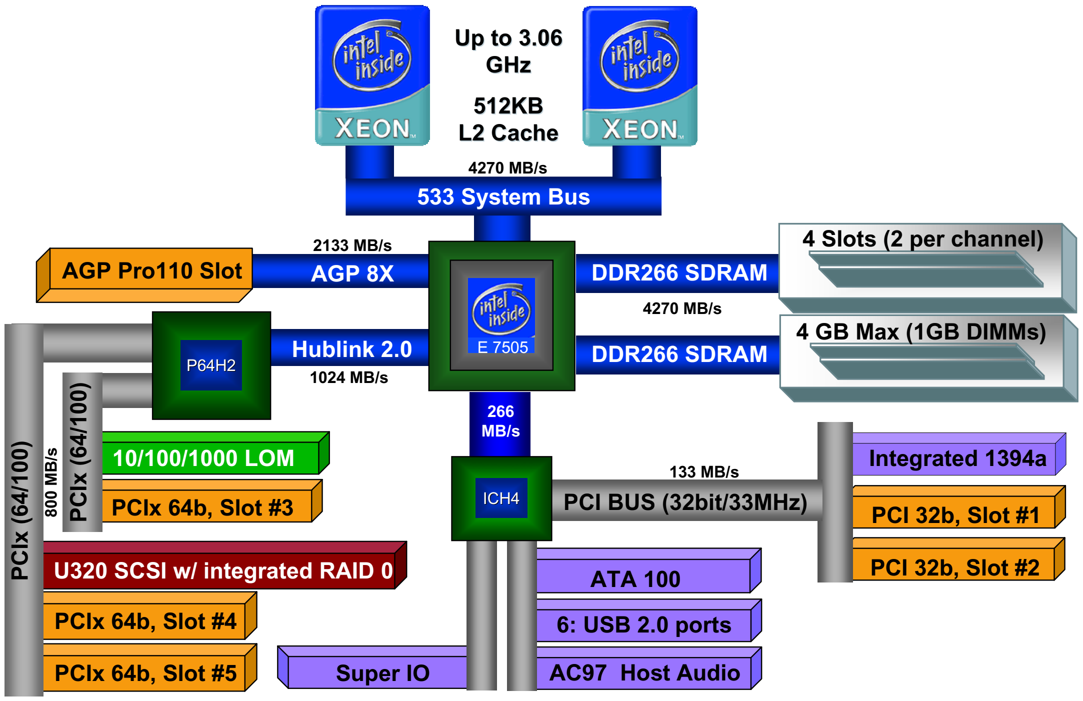


Figure 5.3: Dell Precision 650 Architecture

this configuration can take 5 Basler cameras in total. The integrated U320 SCSI card can then be used to stream image sequences from the Firewire card to SCSI hard drives. In this case, slots #4 and #5 should be left empty to avoid diminishing the bandwidth needed by the integrated SCSI controller.[2] (The system drive is connected to the ATA 100 controller.)

5.3 *Capture surface texture under structured light*

The four-camera systems described so far require projecting patterns on the surface while taking images. These structured light patterns help to improve stereo matching

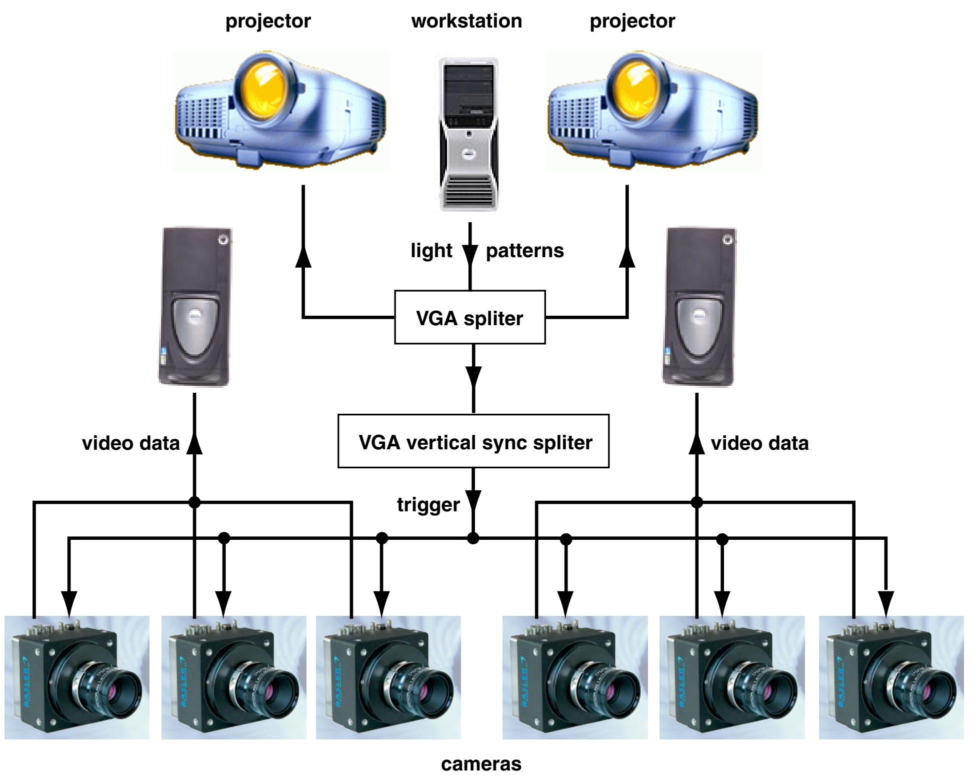
[2]Slots #3, #4, and #5 are all on PCIx buses (64bits/100MHz), which have 8 times the bandwidth of a PCI bus (32bit/33MHz). However, the Firewire cards we used are 32bit/33MHz devices. When such devices appear on a PCIx bus, the PCIx bus will work like an ordinary PCI bus with lower bandwidth.

but prevent us from capturing surface texture. However, capturing surface texture is equally important in many applications, e.g., computing optical flow for estimating 3D motion as described in Chapter 6. In order to capture surface texture while capturing the shapes, we use two additional color cameras and operate the camera rig using the following two strategies.

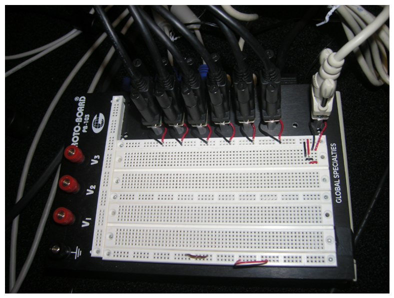
5.3.1 Time multiplexing

The first strategy is "time multiplexing" illumination, i.e., alternating between structured light and ambient light. Specifically, we project a solid black frame for every third frames, during which the surface texture is captured by the 2 color cameras under ambient light. Our ambient light consists of several light bulbs illuminating the face from different angles. We also synchronize the projectors and cameras as follows, shown in Figure 5.4(a). We use the video output from a computer to drive projectors with structured light patterns at 60fps, and use the vertical sync signal in the video output to trigger the cameras. Specifically, we split the video output from the computer, a 15-pin VGA signal, into three video signals using a video splitter. Two of these outputs are used as projector video inputs and the vertical sync (v-sync) signal of the third video signal is used to trigger all six cameras through a signal splitter. The VGA splitter, an Aten VS-138, is an off-the-shelf device, as shown in Figure 5.4(b). The v-sync signal splitter is something we built; it is simply a breadboard on which the input v-sync pin of the VGA connector is connected to all the output trigger pins of the camera IO connectors, as shown in Figure 5.4(c).

We should point out that the above system architecture may not work well for every type of camera and projector. Some cameras' shutters are not very accurately synchronized to their trigger signals. Almost all commercial DLP projectors have internal frame rates of 60Hz. However, some of them synchronize their internal frame rates to the input video and others do not. All of these issues affect whether the cameras and projectors can be synchronized. We tested our projector and camera



(a)



(b) (c)

Figure 5.4: A synchronized projector-camera system. (a) A Dell Precision 330 workstation sends out structured light patterns through its VGA output. This VGA signal is sent two NEC LT 260K projectors through a video splitter. In addition, the vertical sync of this VGA signal is used to trigger four black and white Basler A301f's and two color Basler A310fc's through a sync signal splitter. The video data from the cameras are streamed into two Dell Precision 650 workstations in real time. (b) VGA splitter: Aten VS-138. (c) Sync splitter: a breadboard on which the v-sync pin of the VGA connector is connected to all the trigger pins of the camera I/O connectors.

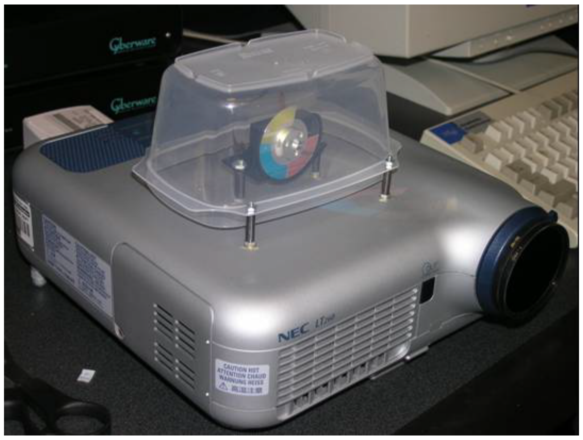


Figure 5.5: We build a near infrared projector by moving the projector color wheel outside and attaching a UV block filter and an IR pass filter in front of the projector lens.

models through many experiments, and verified that they work well.

The advantage of time multiplexing is that it is fairly straightforward to implement. The disadvantage is that it requires the synchronization between projector and camera, limiting the cameras to operate at 60fps. Also, blank frames do not have stripes, which may lower the shape reconstruction quality. Finally, the color data is only acquired at one third the rate of shape data.

5.3.2 Spectrum multiplexing

Another way to capture surface texture is to project infrared (IR) structured light for shape acquisition and use color cameras to capture surface texture in the visible portion of the spectrum. Doing so avoids shining bright light into the subject's eyes. Furthermore, it enables capturing both shape and color at 60 fps. However, IR projectors are not commonly available. Here we describe a prototype system we developed to explore this idea.

We modified an off-the-shelf projector for IR projection. To allow IR light to flow through the optical path of the NEC LT 260K DLP projector, we removed the color

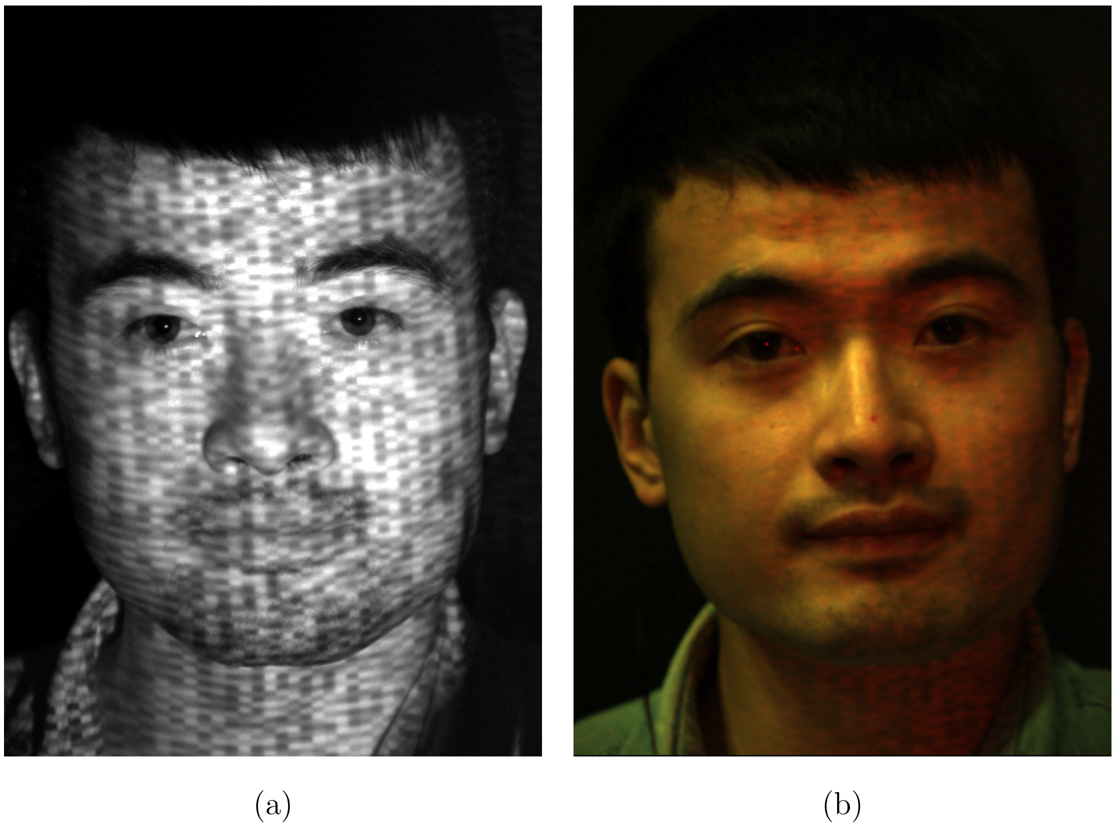


Figure 5.6: Images taken under infrared structured light projection. (a) An image from a black white camera with an IR filter attached. (b) An image from a color camera. While most of the pattern is removed, notice that there is still a faint pattern in the red channel in the color image. This is because the IR filter on the projector, B+W 092, lets a small amount of red light through.

wheel filter from the path. The resulting light has much broader spectral content, including IR and ultraviolet (UV).[3] Then we attach a B+W 415 UV-block filter and a B+W 092 IR-pass filter to the projector lens, so that the projector only passes IR and not potentially hazardous UV radiation. Figure 5.5 shows our modified projector. When using IR projection, we also attach the same IR pass filters to the black and white cameras.

[3]In practice, we can not simply throw away the color wheel. If the color wheel is not detected running, the projector electronics will shut down. Therefore, we mount the color wheel outside the projector body and let it spin.

Figure 5.6(a,b) shows images seen by a monochrome camera and a color camera, respectively, when the IR projector is projecting a random dot pattern. Specifically, the pattern is generated by randomly assigning each cell of a rectangular grid with one of the gray levels $\{0, 0.2, 0, 4, 0.6, 0.8, 1.0\}$. Using two monochrome cameras with IR-pass filters attached, we captured stereo sequences under infrared structured light projection and applied spacetime stereo on these sequences. Figure 5.7 shows 4 examples from a reconstructed face sequence. These results are noisier than those obtained using visible structured light, because our cameras are not very sensitive to infrared light and the infrared images are much noisier than the images recorded under visible structured light projection.

In Figure 5.6(b), we can still see a faint pattern in the red channel in the color image. This is because the B+W 092 IR filter lets a small amount of red light through. We did try using a more opaque IR filter (Hoya R72), but found that our monochrome camera recorded much darker images; it is not very IR sensitive. Therefore, we use the time-multiplex approach for the image acquisition in Chapter 4 and 6. IR sensitive cameras are becoming more available because they also have applications in video surveillance. We believe spectrum multiplexing is a very promising alternative to allow simultaneous capture of both shape and texture with imperceptible structured light.

Instead of spectrum multiplexing, Raskar et al. [81] implement an imperceptible structured light system by consecutively projecting a pattern and its complement. The human visual system then temporally averages out the flickering light patterns and perceives the projector as a white light source with constant brightness. However, with this approach, the projector may still be uncomfortably bright for human eyes. In addition, due to the surface motion, the surface texture can not be easily obtained by averaging out neighboring frames in the color image sequences.

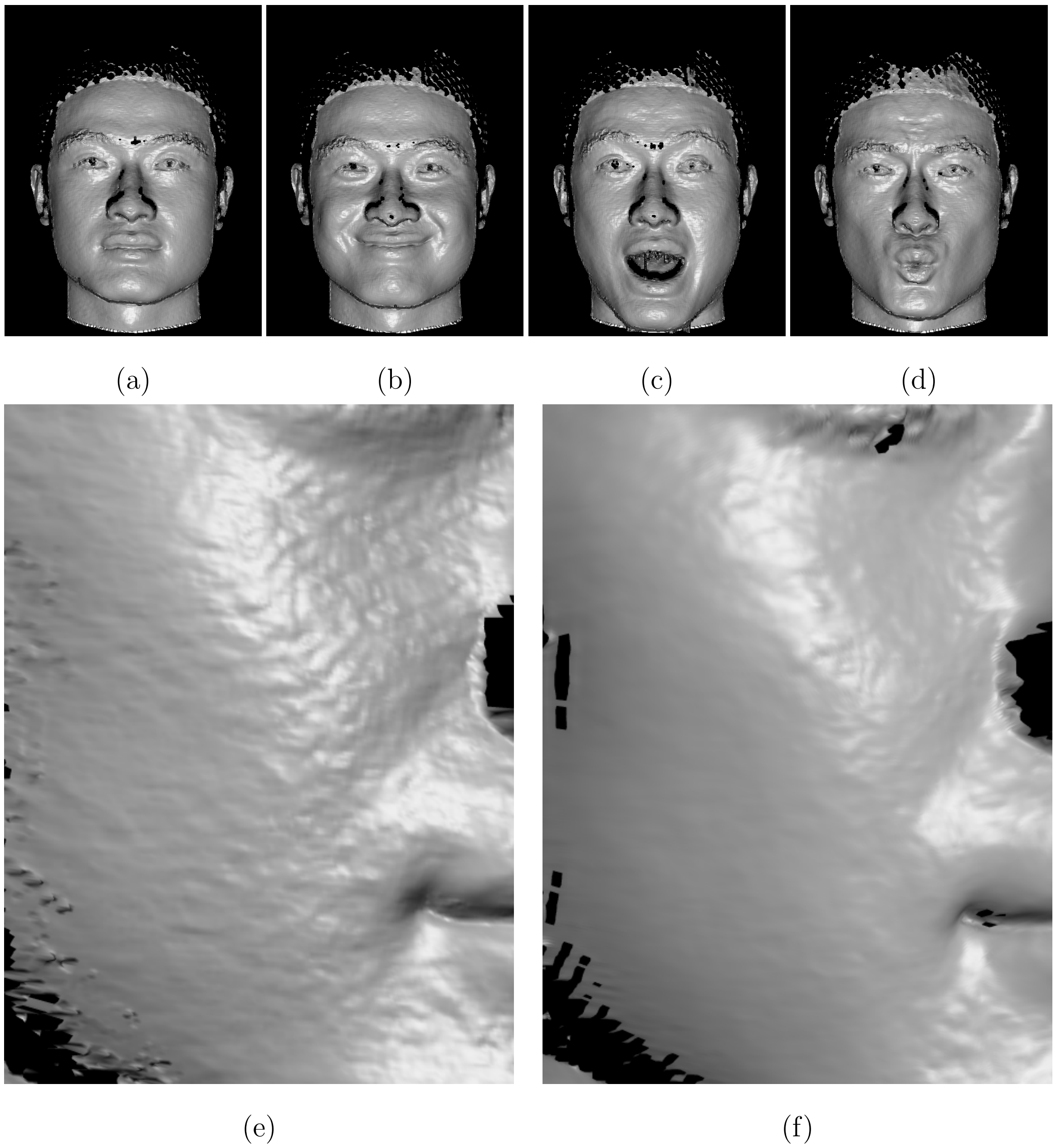


Figure 5.7: (a-d) Four examples of face reconstruction using infrared projection and imaging. (e) A closeup view of the cheek region in (a). (f) The cheek region of a face reconstruction using visible structured light projection and imaging. Notice that (e) is noisier than (f) because our cameras are not very sensitive to infrared light and record infrared images that are much noisier than images taken under visible light.

Chapter 6

DENSE FACE MOTION CAPTURE

In previous chapters, we presented spacetime stereo methods and the design of a camera rig to capture both shape and texture for a dynamic surface. The captured shapes are represented as depth maps. These depth maps have missing data due to occlusions. Also they lack correspondence information that specifies how a point on the 3D surface moves over time. These limitations make it difficult to re-pose or re-animate the captured faces. In this chapter, we present a novel method for computing a single time-varying mesh that closely approximates the depth map sequences while optimizing the vertex motion to be consistent with optical flow computed between color frames.[1] In the next section, we first review previous work in creating dynamic face models and then present our approach.

6.1 Previous work on 3D face modeling

Face modeling is an active research topic in computer graphics and computer vision. Here we review the most related work on 3D face tracking. 3D face tracking techniques compute the facial motion by deforming a 3D face model to fit a sequence of images [35, 75, 5, 32, 8] or 3D marker positions [44]. Blanz and Vetter [9] construct particularly high quality models, represented as linear subspaces of laser-scanned head shapes. Although subspace models are flexible, they fail to reconstruct shapes that are outside the subspace. In order to handle expression variation, Blanz and Vetter [8] laser-scanned faces under different expressions, a time-consuming process that

[1]This chapter describes joint work with Noah Snavely, Brian Curless, and Steven M. Seitz, first presented in ACM SIGGRAPH 2004 [102].

requires the subject to hold each expression for tens of seconds. A problem with existing face tracking methods in general is that the templates have relatively few degrees of freedom, making it difficult to capture fine-scale dimples and folds which vary from one individual to another and are important characteristic features. We instead work with a generic high resolution template with thousands of degrees of freedom to capture such fine-grain features. This approach is related to the work of Allen et al. [1] for fitting templates to human body data, except that they rely on markers to provide partial correspondence for each range scan, whereas we derive correspondence information almost entirely from images.

An interesting alternative to traditional template-based tracking is to compute the deformable template and the motion directly from the image sequence. Torresani et al. [93] and Brand [15] recover non-rigid structure from a single video assuming the shape lies within a linear subspace. Although these methods are promising and work from regular video streams, they produce relatively low-resolution results, compared to, e.g., structured light stereo.

Compared to previous methods, our method seeks to combine both 3D shape measurement and 2D optical flow for an optimal 3D motion estimation. Our method starts by fitting a template mesh to the pair of depth maps captured in the first non-pattern frame, initialized with a small amount of user guidance. The method then tracks the template mesh through other non-pattern frames in the sequence automatically and without the need for putting markers on the subject's face. In the next section, we first present template fitting at the first frame. Then we present template tracking through the sequence in Section 6.3. We show results and discuss the pros and cons of our methods in Sections 6.4 and 6.5, respectively.

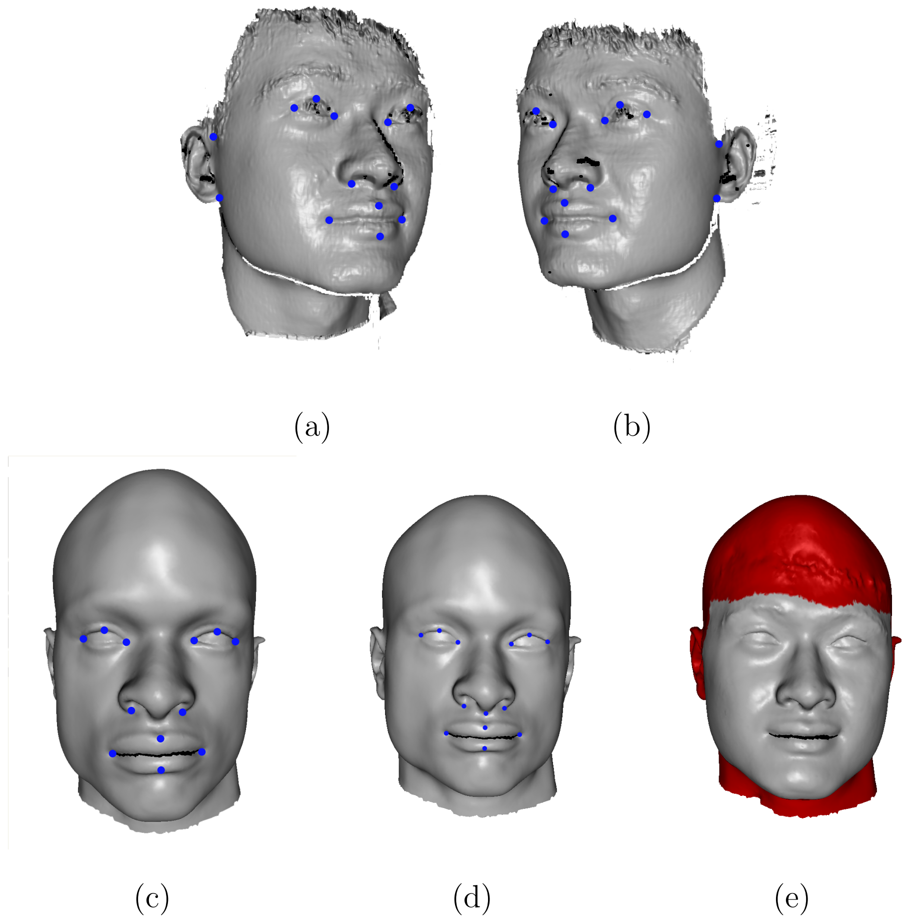
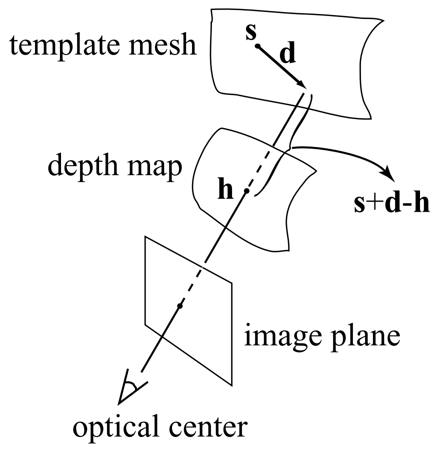


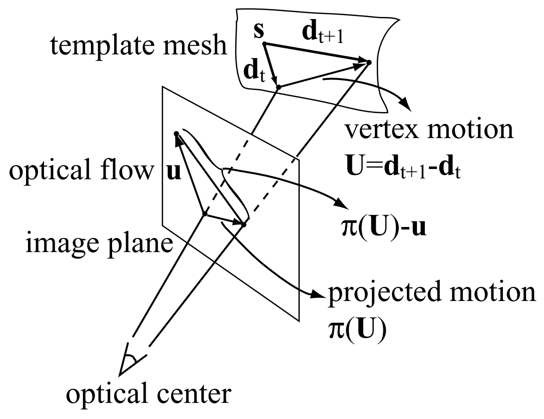
Figure 6.1: Illustration of the template fitting process. (a,b) Depth maps from two viewpoints at the first frame. (c) A face template. A few corresponding shape feature positions are manually identified on both the face template and the first two depth maps. (d) The template after initial global warp using the feature correspondence. (e) Initial mesh after fitting the warped template to the first two depth maps. The initial mesh is colored red for regions with unreliable depth or optical flow estimation.

6.2 Template fitting

Let $M = (\mathsf{V}, \mathsf{E})$ be an N-vertex triangle mesh representing a template face, with vertex set $\mathsf{V} = \{\mathbf{s}_n\}_{n=1}^{N}$ and edge set $\mathsf{E} = \{(n_1, n_2) \mid \mathbf{s}_{n_1} \texttt{ and } \mathbf{s}_{n_2} \texttt{ are connected}\}$. Let $h_j(x, y)$, $j \in \{1, 2\}$, be the two depth maps at frame 1, as shown in Figure 6.1(a,b).



(a)



(b)

Figure 6.2: Illustration of the error metric of a vertex used in template fitting. (a) $\mathbf{s}$ is a vertex on the template mesh and $\mathbf{d}$ is its displacement vector. Let $\mathbf{h}$ be the intersection of the depth map, shown as a surface, and the line from optical center to $\mathbf{s}+\mathbf{d}$. $\mathbf{s}+\mathbf{d}-\mathbf{h}$ is the difference vector between $\mathbf{s}+\mathbf{d}$ and the depth map. (b) $\mathbf{d}_{t+1}$ and $\mathbf{d}_t$ are the displacements for a vertex $\mathbf{s}$ on the template mesh at frame t and $t+1$, respectively. $\mathbf{U}=\mathbf{d}_{t+1}-\mathbf{d}_t$ is the vertex motion from frame t to $t+1$. The projection of $\mathbf{U}$ in the image plane, $\pi(\mathbf{U})$, should be the same as the optical flow $\mathbf{u}$. $\|\pi(\mathbf{U})-\mathbf{u}\|$ is used as a metric for consistency between vertex motion and optical flow.

Given the relative pose between these depth maps,[2] we wish to solve for a displacement $\mathbf{d}_n$ for each vertex such that the displaced mesh M_1, with vertex set $\{\mathbf{s}_n+\mathbf{d}_n\}_{n=1}^{N}$, optimally fits the depth maps. Our fitting metric has two terms: a depth matching term, E_s, and a regularization term E_r.

The depth matching term E_s measures the difference between the depths of vertices of M_1 as seen from each camera's viewpoint and the corresponding values recorded in each depth map. Specifically,

$$E_s(\{\mathbf{d}_n\}) = \sum_{j=1}^{2}\sum_{n=1}^{N} w_{n,j}\rho([\mathbf{s}_n+\mathbf{d}_n-\mathbf{h}_{n,j}]_{\mathbf{z}_j}, \sigma_s) \tag{6.1}$$

where N is the number of mesh vertices, $\mathbf{h}_{n,j}\in\mathbf{R}^3$ is the intersection of the depth

[2]We obtain the relative pose between depth maps using the rigid registration tool provided in Scanalyze [80].

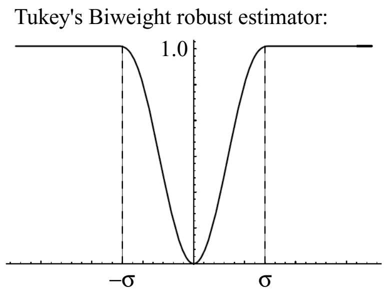


Figure 6.3: A plot of Tukey's Biweight robust estimator.

map $h_j(x,y)$ and the line from the j'th camera's optical center to $\mathbf{s}_n + \mathbf{d}_n$, as shown in Figure 6.2(a); $[\cdot]_{\mathbf{z}_j}$ is the z component of a 3-d point in the j'th camera's coordinate system; $\rho(\cdot, \sigma_s)$ is Tukey's biweight robust estimator, shown in Figure 6.3; and $w_{n,j}$ is a weight factor governing the influence of the j'th depth map on the template mesh. In experiments, we set $\sigma_s = 20$, which rejects potential depth map correspondences that are over 20mm away from the template mesh. For $w_{n,j}$, we use the product of the depth map confidence, computed as in [28], and the dot product of the normals[3] at $\mathbf{h}_{n,j}$ and $\mathbf{s}_n + \mathbf{d}_n$; we set $w_{n,j}$ to 0 if the dot product is negative. Note that, in practice, we do not need to intersect a line of sight with a surface to compute each $\mathbf{h}_{n,j}$. Instead, we project each displaced template point into the depth map $h_j(x,y)$ and perform bilinear interpolation of depth map values to measure depth differences.

In general, the shape matching objective function, Eq. (6.1), is under-constrained. For instance, template surface points after being displaced could bunch together in regions while still matching the depth maps closely. Furthermore, the depth maps do not completely cover the face, and so the template can deform without penalty where there is no data. Thus, we add a regularization term E_r that penalizes large displace-

[3]We compute the normal of a vertex as the area-weighted average of the normals of its neighboring triangles.

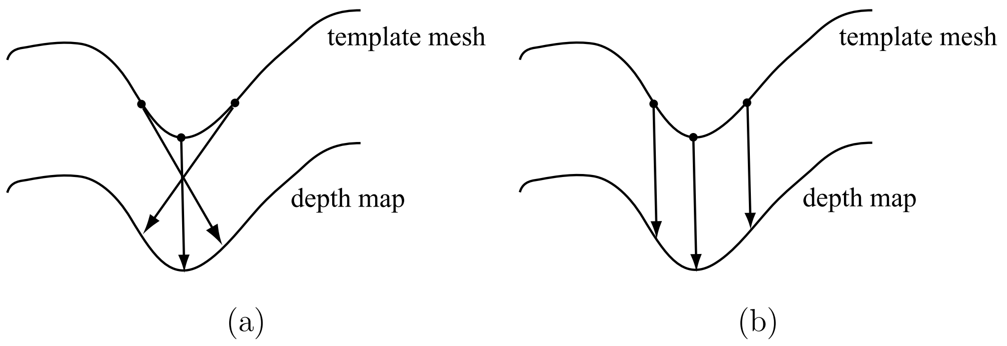


Figure 6.4: The need of regularization in template fitting. (a) Without additional constraints, template surface points could bunch together while still matching the depth map closely. (b) A smooth vertex displacement field can be obtained by adding a regularization that penalizes large displacement differences between neighboring vertices on the template mesh.

ment differences between neighboring vertices on the template mesh. Specifically,

$$E_r(\{\mathbf{d}_n\}) = \sum_{(n_1,n_2)\in \mathsf{E}} \|\mathbf{d}_{n_1} - \mathbf{d}_{n_2}\|^2 / \|\mathbf{s}_{n_1} - \mathbf{s}_{n_2}\|^2 \tag{6.2}$$

where the denominator $\|\mathbf{s}_{n_1} - \mathbf{s}_{n_2}\|^2$ helps to penalizes neighboring displacement differences according to the neighboring vertex distance. Notice that preferring a smooth displacement field is different from preferring a smooth displaced template mesh; the former preserves the shape details in the template mesh while the latter often smooths out the details [1].

To fit the template mesh M to the depth maps at frame 1, we minimize a weighted sum of Eqs. (6.1) and (6.2)

$$\Phi = E_s + \alpha E_r \tag{6.3}$$

with $\alpha = 2.0$ in our experiments.

We minimize Eq. (6.3) using the Gauss-Newton method. We initialize the optimization by manually aligning the template with the depth maps. Specifically, we select several corresponding feature positions on both the template mesh and the depth maps (Figure 6.1(a,b,c)). Next, from these feature correspondences, we solve for an over-constrained global affine transformation to deform the mesh. To determine an affine transformation, at least 4 correspondences are needed; in practice, we specify about 15 feature correspondences. Finally, we interpolate the residual displacements at the feature positions over the whole surface using a normalized radial basis function [17] similar to the method of Pighin et al. [74]. After the initial warp, the selected feature correspondences are no longer used for template fitting. As shown in Figure 6.1(d), the initial alignment does not have to be precise in order to lead to an accurate final fitting result, as illustrated in Figure 6.1(e).

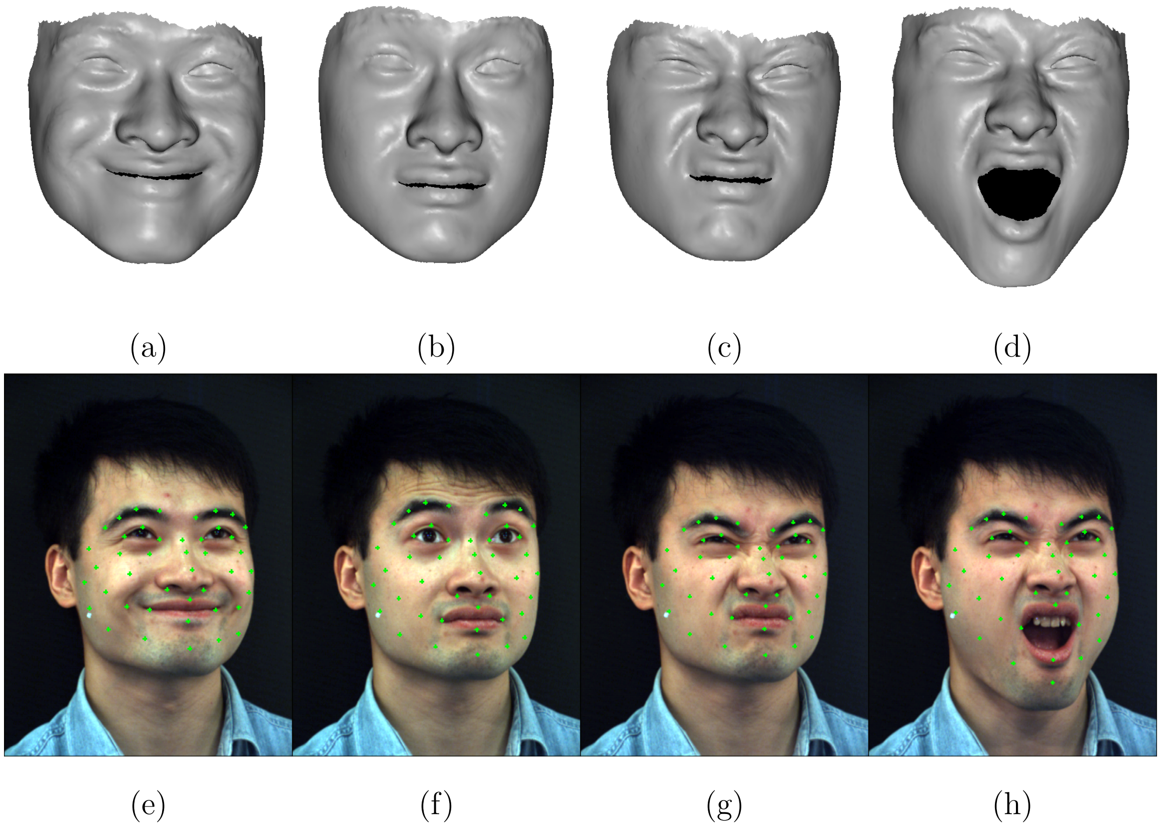
6.3 Template tracking

Given the mesh M_1 at the first frame, we would now like to deform it smoothly through the rest of the sequence such that the shape matches the depth maps and the vertex motions match the optical flow computed for the non-pattern frames of the color image streams. Specifically, we seek to compute $\{\mathbf{d}_{n,t}\}$, which gives the time-varying shape $\{\mathbf{s}_{n,t} = \mathbf{s}_n + \mathbf{d}_{n,t}\}$. Let $I_k(x, y, t)$, $k \in \{1, 2\}$, be color image sequences from two viewpoints with pattern frames removed. We first compute optical flow $\mathbf{u}_k(x, y, t)$ for each sequence using Black and Anandan's method [7]. The flow $\mathbf{u}_k(x, y, t)$ represents the motion from frame t to $t+1$ in the k'th image plane. We measure the consistency of the optical flow and the vertex inter-frame motion $\mathbf{U}_{n,t} = \mathbf{d}_{n,t+1} - \mathbf{d}_{n,t}$, called *scene flow* in [95], by the following metric:

$$E_m(\{\mathbf{d}_{n,t+1}\}) = \sum_{k=1}^{2} \sum_{n=1}^{N} \rho(\|\pi_k(\mathbf{U}_{n,t}) - \mathbf{u}_{n,t,k}\|, \sigma_m) \tag{6.4}$$

where $\pi_k(\mathbf{U}_{n,t})$ is the image projection of $\mathbf{U}_{n,t}$ in the k'th image plane and $\mathbf{u}_{n,t,k}$ is the value of optical flow $\mathbf{u}_k(x, y, t)$ at the corresponding location, $\pi_k(\mathbf{s}_n + \mathbf{d}_{n,t})$, shown in Figure 6.2(c); $\rho(\cdot, \sigma_m)$ is the same Tukey's biweight robust estimator as in Eq. (6.1), with $\sigma_m = 20$ pixels.

Starting from M_1, we recursively compute M_{t+1} given M_t by optimizing a weighted



(a) (b) (c) (d)

(e) (f) (g) (h)

Figure 6.5: Illustration of the template tracking result. (a,b,c,d): selected meshes after tracking the initial mesh through the whole sequence, using both depth maps and optical flows. The process is marker-less and automatic. (e,f,g,h): the projection of a set of vertices from the selected meshes on the image plane, shown as green dots, to verify that the vertex motion is consistent with visual motion. Note that the subject had no markers on his face during capture; the green dots are overlaid on the original images purely for visualization purposes.

sum of Eq. (6.3) and Eq. (6.4) where $\mathbf{d}_n$ is replaced with $\mathbf{d}_{n,t}$:

$$\Psi = E_s + \alpha E_r + \beta E_m \tag{6.5}$$

with $\alpha = 2.0$ and $\beta = 0.5$ in our experiments.

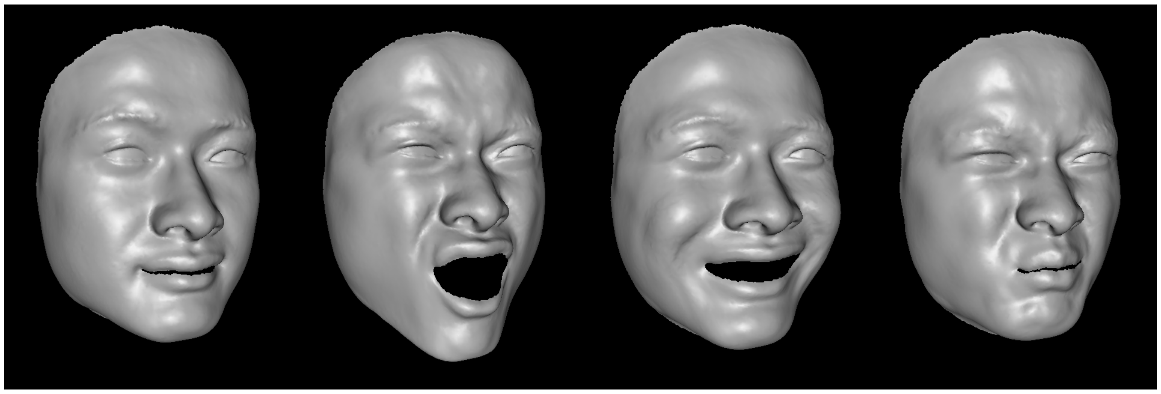


Figure 6.6: Four examples of the template tracking results for another sequence. In this sequence, the subject wears a hair wrapper and his forehead region is correctly recovered.

6.4 Results

We tried our mesh fitting and tracking methods on images sequences for several subjects and show results for three subjects here. In Figure 6.5(a-j), sample results of mesh tracking for the first subject are shown in gray shaded rendering. The video on our website `grail.cs.washington.edu/projects/stfaces` shows the full sequence both as gray-shaded and color-textured rendering. The sequence has 332 face meshes and each mesh has 14883 vertices. Template tracking takes less than 1 minute per frame.

Notice that in Figure 6.5, the forehead data is missing. This is because the hair hanging over the forehead and the forehead itself form two layers of motion which confuse the optical flow algorithm. Therefore, the forehead motion is not correctly recovered and we manually specify a mask and cut it out. In Figure 6.3, we record sequences of the same subject when he wears a hair wrapper. In this case, the full facial shape and motion are recovered. This sequence has 384 mesh models, and each mesh has about 23728 vertices.

In Figure 6.7, we show 4 examples of template tracking results for the second

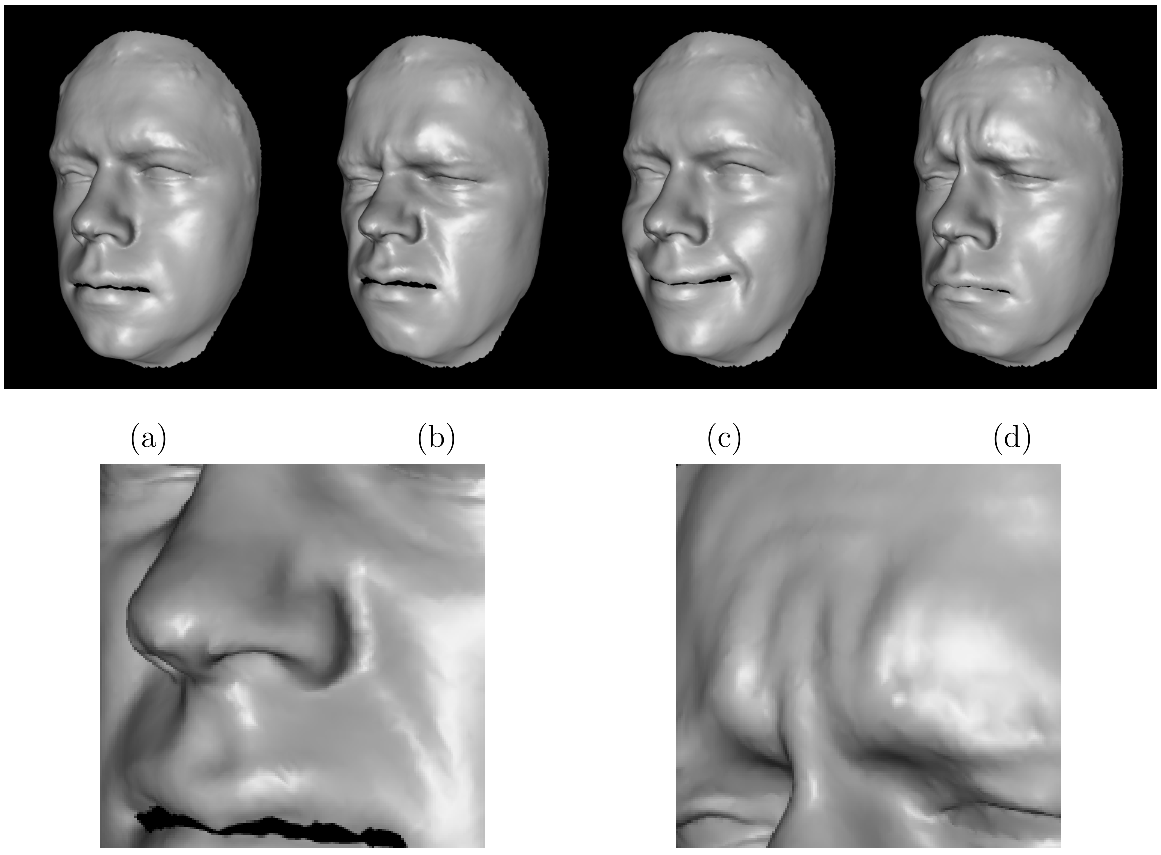
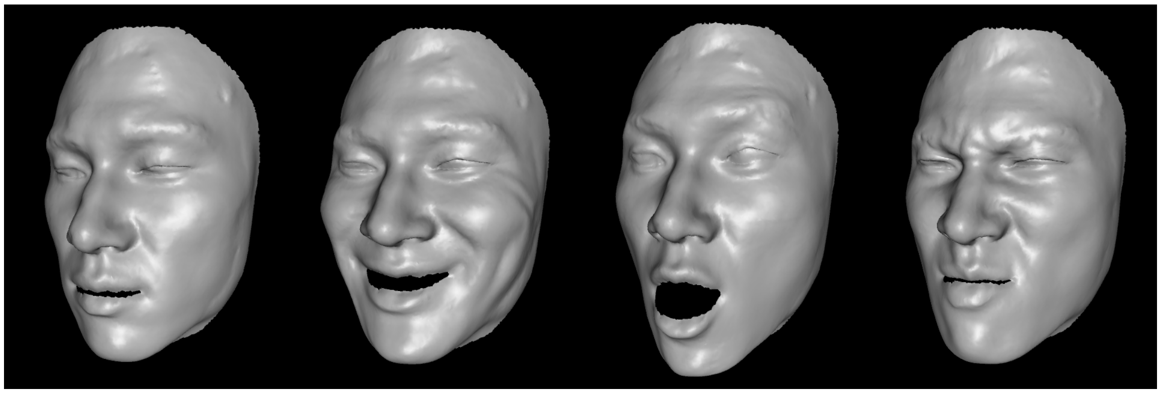
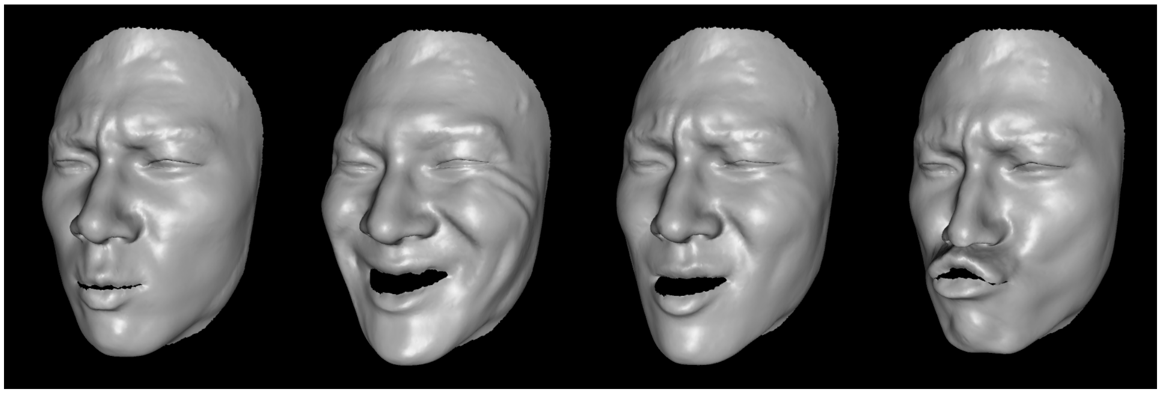


Figure 6.7: (a-d) Four examples of the template tracking results for the second subject. (e) A closeup view of the skin deformation near the nose in the "disgusted" expression shown in (b). (f) A closeup view of the wrinkles on the "frowning" forehead of (d).

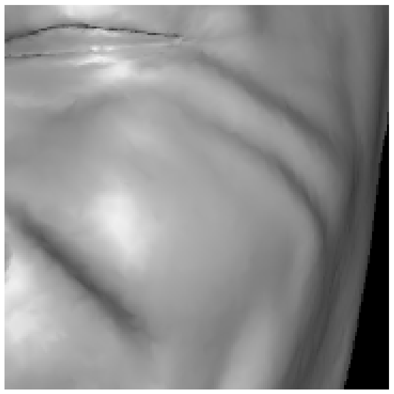
subject. This sequence consists of 580 meshes and each mesh has 23728 vertices. In Figure 6.8, we show 8 examples of the template tracking results for the third subject. This sequence consists of 339 meshes and each mesh has 23728 vertices. Notice that, in Figure 6.7(e,f) and Figure 6.8(i,j), we show closeup views of some of the face meshes, demonstrating the ability to capture fine features such as the wrinkles on a frowning forehead and near a squinting eye. These subtle shape details are extremely important for conveying realistic expressions, because our visual system is well tuned



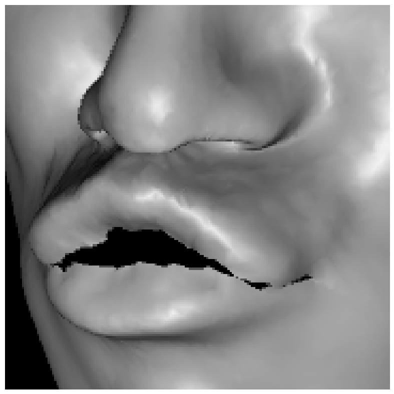
(a) (b) (c) (d)



(e) (f) (g) (h)



(i)



(j)

Figure 6.8: (a-h) Eight examples of the template tracking results for the third subject. (i) A closeup view of the wrinkles near the left eye in (f). (j) A closeup view of the pouting lips in (h).

to recognize human faces. The uniqueness of our system is its capability to capture not only these shape details, but also how these shape details change over time. In the end, our system is an automatic, dense, and marker-less facial motion capture system.

6.5 Discussion

In this chapter, we described a new template fitting and tracking procedure that brings the surfaces into correspondence across the entire sequence without the use of markers. This procedure enables us to use the captured 3D data for expression synthesis and facial animation as presented in the next chapter.

There are many important open research topics to explore for shape fitting and tracking. First, we only fit the template to one-third of the depth maps (at non-pattern frames). A natural extension of our current work is to interpolate color along optical flow to obtain face models at 60Hz. Second, our template face mesh, which is cut from a whole body human model, has a resolution of about one-eighth of the depth maps. In the future, it would be interesting to subdivide the template mesh until its vertex count is comparable to the depth map resolution before applying the fitting procedure. Although fitting a higher resolution mesh may result in more detailed face models, it could also lead to fitting measurement noise. Therefore, it would also be interesting to devise robust regularization that removes noises while preserving features.

Notice that the definition of regularization term E_r in Eq. (6.2) is translation-invariant but not rotation-invariant, meaning that E_r is zero if $\{\mathbf{d}_n\}$ is constant, i.e., a translation motion field, but non-zero if $\{\mathbf{d}_n\}$ is a rotation field. Therefore, it will not work well for turning heads. In our experiments, we let subjects fix their head poses and only change expressions. To handle face motion with large head rotations, the affine-invariant regularization energy used in [1] could be a better alternative.

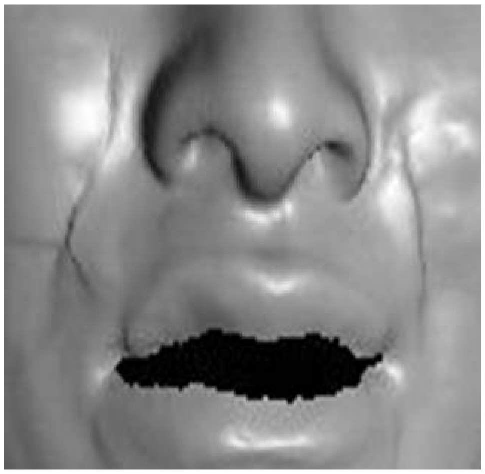


Figure 6.9: Bunching and folding artifacts in tracking results due to optical flow errors.

Our fitting and tracking techniques depend on optical flow estimation which can be unreliable for textureless regions. Although the regularization of displacements helps to produce smooth and overall pretty detailed meshes, we did observe some folding and bunching of vertices in the textureless regions, as shown in Figure 6.9, especially when the mouth moves fast. In the future, we hope to incorporate physical skin models and temporal coherence to avoid these artifacts. If subjects wish, we could also paint markers on their faces to simplify the tracking problem.

Our fitting and tracking methods are semi-automatic, requiring manual initialization for the first shape in each sequence. It would very useful to fully automate this procedure to efficiently register measured shapes, especially for large data sets with many different individuals.

Chapter 7

DATA-DRIVEN 3D FACIAL ANIMATION

Creating face models that look and move realistically is an important problem in computer graphics. It is also one of the most difficult, as even the most minute changes in facial expression can reveal complex moods and emotions. In previous chapters, we presented new techniques to continuously capture a moving human face at high resolution. In this chapter, we study the problem of using the captured data for expression synthesis and facial animation.[1]

Our objective is to create new facial poses and motion that accurately reflects the shape and time-varying behavior of a real person's face based on the captured face models. In particular, we seek real-time, intuitive controls to edit expressions and create animations. For instance, dragging the corner of the mouth up should result in a realistic expression, such as a smiling face. Rather than programming these controls manually, we wish to extract them from correlations present in the input video. Furthermore, we wish to use these controls to generate desired animations which preserve the captured dynamics of a real face. (By "dynamics," we mean the time-varying behavior, not the physics *per se.*)

To achieve this goal, we propose a data-driven, inverse kinematics technique we call *faceIK* to interactively manipulate captured models to create new expressions. FaceIK blends the models in a way that is automatically adaptive to the number of user-specified controls. FaceIK can be used to create key frames for desired animation. We also present a representation called *face graph* which encodes the dynamics of the

[1]This chapter describes joint work with Noah Snavely, Brian Curless, and Steven M. Seitz, first presented in ACM SIGGRAPH 2004 [102].

captured face sequence. The face graph can be traversed to generate in-between frames given keyframes created by FaceIK. While our animation results do not match the artistry of what an expert animator can produce, our approach enables untrained users to produce realistic facial animations.

7.1 Previous work in 3D face editing and animation

Modeling and synthesizing faces is an active research field in computer graphics and computer vision. Here we review two topics most related to this chapter: constraint-based face editing, and data-driven facial animation.

Direct 3D face editing Following Parke's pioneering work [72] on blendable face models, most face editing systems are based on specifying blending weights to linearly combine a set of template faces. These weights can be computed indirectly from user-specified constraints [75, 53] or fit directly to images [9]. All these methods blend faces based on predefined feature regions, e.g., eyes and mouths, to create more varieties of facial expressions from a limited set of example faces. In particular, the editing interface in [75] allows users to paint blending weights for face mesh vertices.

Our *faceIK* tool, as a general expression editing interface, is similar to the one developed by Joshi et al. [53]. However, they segment a face into a region hierarchy *a priori*, which decouples the natural correlation between different parts of the face. Zhang et al. [103] address this problem with a hierarchical PCA technique in which user edits may propagate between regions. Our faceIK method instead maintains the correlation across the whole face and only decouples it — automatically and adaptively — as the user introduces more constraints.

Data-driven 3D face animation A focus of our work is to use captured models of human face dynamics to drive animatable face models. Several others have explored performance-based methods for animating faces, using either video of an actor [8, 21], or speech [16, 14, 36] to drive the animation. These techniques can be considered *data-driven* in that they are based on a sequence of example faces.

Still others have explored data-driven animation techniques in the domains of human figure motion [61, 2, 58, 60] and video sprites [88]. We adapt ideas from these other domains to devise 3D face animation tools.

7.2 FaceIK

In this section we describe a real-time technique for editing a face to produce new expressions. The key property of the technique is that it exploits correlations in a set of input meshes to propagate user edits to other parts of the face. So, for instance, pulling up on one corner of the mouth causes the entire face to smile. This problem is analogous to the *inverse kinematics* (*IK*) problem in human figure animation in which the goal is to compute the pose of a figure that satisfies one or more user-specified constraints. We therefore call it *faceIK*.

Our approach is based on the idea of representing faces as a linear combination of basis shapes. While linear combinations have been widely used in face modeling [23, 9, 74], the problem of generating reasonable faces using only one constraint (e.g., the corner of the mouth), or just a few constraints, is more difficult, because the problem is severely underconstrained. One solution is to compute the coefficients which maximize their likelihood with respect to the data, using principle component analysis (PCA) [9, 1]. The maximum likelihood criterion works well for modeling face variations under similar expressions and human body variations under similar poses. However, applying PCA to facial expressions does not produce good results unless the face is segmented *a priori* into separate regions, e.g., eyes, nose, and mouth [9, 53, 103]. Unfortunately, segmenting faces into regions decouples the natural correlation between different parts of a face. Further, the appropriate segmentation is not obvious until run-time when the user decides what expressions to create. For example, to create an asymmetric expression, the left and right sides of the face must be decoupled. As discussed in Section 7.2.2, under- or over-segmenting the face can result in undesirable editing behavior. We instead describe a method that avoids these problems by adaptively segmenting the face into soft regions based on user edits. These regions are independently modeled using the captured face sequence, and then they are blended into a single expression. We could model these regions using PCA; however, because they are formed by user edits, we would then have to compute principal components, a costly operation, at run-time for each region. To address this problem, we introduce a fast method, *Proximity-Based Weighting* (PBW), to model the regions. We start by describing how to use PBW to model the entire face as a single region, and then extend it to handle multiple, adaptively defined regions.

7.2.1 Proximity-based weighting

Suppose we are given as input F meshes, each with N vertices. We use $\mathbf{s}_{n,f}$ to denote the n'th vertex in mesh f. Let $\{\mathbf{p}_l\}_{l=1}^{L}$ be user-specified 3D constraints, requiring that vertex l should be at position $\mathbf{p}_l$; we call these constraints *control points.*[2] We seek a set of blend coefficients c_f such that for every l,

$$\sum_{f=1}^{F} c_f \mathbf{s}_{l,f} = \mathbf{p}_l \qquad \text{and} \qquad \sum_{f=1}^{F} c_f = 1 \tag{7.1}$$

The second equation poses an affine constraint on the blend coefficients. This constraint makes the blend coefficients invariant to global translation. Specifically, if we move all the meshes and control points by a same displacement, the same blend coefficients will still satisfy Eq. (7.1) for the displaced meshes under displaced constraints.

Because the number of constraints L is generally far fewer than the number of meshes F, we advocate weighting example faces based on *proximity to the desired expression*, i.e., nearby faces are weighted more heavily, a scheme we call proximity

[2]We assume the L constraints are for the first L mesh vertices, to simplify notation without loss of generality.

based weighting. Specifically, we penalize meshes whose corresponding vertices are far from the control points by minimizing

$$g(\mathbf{c}) = \sum_{f=1}^{F} \phi(\|\bar{\mathbf{s}}_f - \bar{\mathbf{p}}\|)c_f^2 \tag{7.2}$$

where $\mathbf{c} = [c_1\ c_2\ \ldots\ c_F]^{\mathsf{T}}$, $\bar{\mathbf{s}}_f = [\mathbf{s}_{1,f}^{\mathsf{T}}\ \mathbf{s}_{2,f}^{\mathsf{T}}\ \ldots\ \mathbf{s}_{L,f}^{\mathsf{T}}\ 1]^{\mathsf{T}}$, $\bar{\mathbf{p}} = [\mathbf{p}_1^{\mathsf{T}}\ \mathbf{p}_2^{\mathsf{T}}\ \ldots\ \mathbf{p}_L^{\mathsf{T}}\ 1]^{\mathsf{T}}$, $\|\bar{\mathbf{s}}_f - \bar{\mathbf{p}}\| = \sqrt{\sum_{l=1}^{L} |\mathbf{s}_{l,f} - \mathbf{p}_l|^2}$, and $\phi(\cdot)$ is a monotonically increasing function. In our experiments, we found that a simple function $\phi(r) = 1 + r$ worked well. Minimizing Eq. (7.2) subject to Eq. (7.1) encourages small weights for faraway meshes, and can be solved in closed-form as:

$$c_f = \frac{1}{\phi_f}\bar{\mathbf{s}}_f^{\mathsf{T}}\mathbf{a} \tag{7.3}$$

where $\phi_f = \phi(\|\bar{\mathbf{s}}_f - \bar{\mathbf{p}}\|)$ and $\mathbf{a} = (\sum_{f=1}^{F} \frac{1}{\phi_f}\bar{\mathbf{s}}_f\bar{\mathbf{s}}_f^{\mathsf{T}})^{-\mathbf{1}}\bar{\mathbf{p}}$. The derivation of this solution is given in Appendix A.

Screen-space constraints

Rather than requiring that constraints be specified in 3D, it is often more natural to specify where the *projection* of a mesh vertex should move to. Given a set of user-specified 2D constraints $\{\mathbf{q}_l\}_{l=1}^{L}$, Eq. (7.2) is modified as follows

$$g(\mathbf{c}) = \sum_{f=1}^{F} \phi(\|\pi(\bar{\mathbf{s}}_f) - \bar{\mathbf{q}}\|)c_f^2 \tag{7.4}$$

such that

$$\pi(\sum_{f=1}^{F} c_f\mathbf{s}_{l,f}) = \mathbf{q}_l \qquad \text{and} \qquad \sum_{f=1}^{F} c_f = 1 \tag{7.5}$$

where $\pi(\cdot)$ is the image projection operator, $\pi(\bar{\mathbf{s}}_f) \overset{\text{def}}{=} [\pi(\mathbf{s}_{1,f})^{\mathsf{T}}\ \pi(\mathbf{s}_{2,f})^{\mathsf{T}}\ \ldots\ \pi(\mathbf{s}_{L,f})^{\mathsf{T}}\ 1]^{\mathsf{T}}$, and $\bar{\mathbf{q}} = [\mathbf{q}_1^{\mathsf{T}}\ \mathbf{q}_2^{\mathsf{T}}\ \ldots\ \mathbf{q}_L^{\mathsf{T}}\ 1]^{\mathsf{T}}$. Since π is in general nonlinear, we approximate Eq. (7.5) by

$$\sum_{f=1}^{F} c_f\pi(\mathbf{s}_{l,f}) = \mathbf{q}_l \qquad \text{and} \qquad \sum_{f=1}^{F} c_f = 1 \tag{7.6}$$

This approximation works well in our experience, and minimizing Eq. (7.4) subject to Eq. (7.6) yields the closed-form solution:

$$c_f = \frac{1}{\phi_f}\pi(\bar{\mathbf{s}}_f)^{\mathtt{T}}\mathbf{a} \tag{7.7}$$

where $\phi_f = \phi(\|\pi(\bar{\mathbf{s}}_f) - \bar{\mathbf{q}}\|)$ and $\mathbf{a} = (\sum_{f=1}^{F} \frac{1}{\phi_f}\pi(\bar{\mathbf{s}}_f)\pi(\bar{\mathbf{s}}_f)^{\mathtt{T}})^{-\mathbf{1}}\bar{\mathbf{q}}$. The derivation of this solution is the same as that for Eq. (7.3) with $\bar{\mathbf{s}}_f$ and $\bar{\mathbf{p}}$ replaced by $\pi(\bar{\mathbf{s}}_f)$ and $\bar{\mathbf{q}}$.

7.2.2 Local influence maps

The faceIK method presented so far assumes that the entire face is created by linear interpolation of input meshes with nearby meshes given larger weights. However, this assumption is too limiting, since, for example, an asymmetric smile cannot be generated from a data set that contains only symmetric smiles. We therefore propose a method to blend different face regions by defining an *influence map* for each control point. Specifically, we give each control point greater influence on nearby mesh points, and then blend the influences over the entire mesh to allow for a broader range of expressions that do not exist in the input data.

Accordingly, for each of the L control points, we compute a set of blending coefficients $\mathbf{c}_l$ by minimizing Eq. (7.2) or Eq. (7.4); This process is done independently for each control point. These L sets of coefficients will result in have L meshes; each mesh satisfies one of the L constraints but not necessarily the others. The resulting L meshes are then blended together, using normalized radial basis functions [17] to define spatially-varying weights. Specifically, we set the blending coefficient for vertex $\mathbf{s}_n$ as follows

$$\mathbf{c}(\mathbf{s}_n) = \sum_{l=1}^{L} B(\mathbf{s}_n, \mathbf{s}_l)\hat{\mathbf{c}}_l \tag{7.8}$$

where $B(\mathbf{s}_n, \mathbf{s}_l) = \frac{\exp(-\|\mathbf{s}_n - \mathbf{s}_l\|^2/r_l^2)}{\sum_{l'=1}^{L}\exp(-\|\mathbf{s}_n - \mathbf{s}_{l'}\|^2/r_{l'}^2)}$ with $r_l = \min_{l' \neq l} \|\mathbf{s}_l - \mathbf{s}_{l'}\|$ and $\hat{\mathbf{c}}_l$ is a F-dimensional vector defined at each control point l. In Appendix B, we describe how to compute

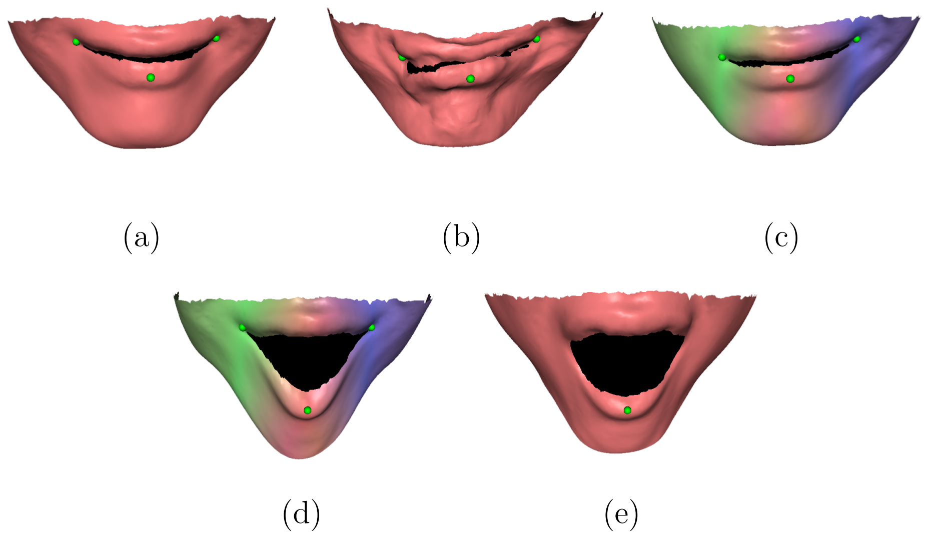
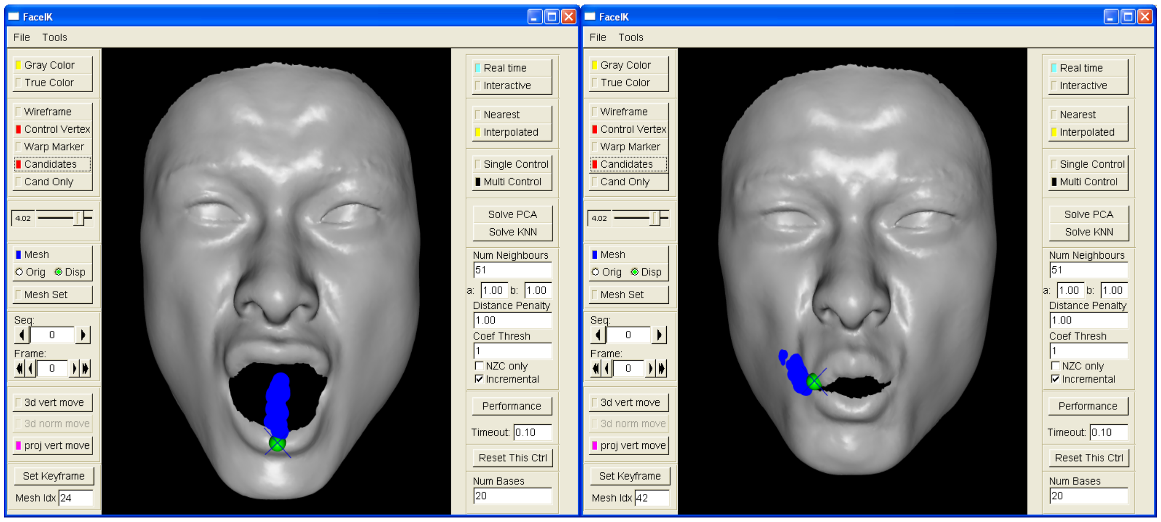


Figure 7.1: Advantages of adaptive face segmentation with faceIK. Many face editing techniques pre-segment a face into regions (e.g., mouth, nose, eyes) and model each region separately with PCA. (a) Three symmetric control points are used to create a symmetric smile by applying PCA on the mouth region [1] to compute the maximum likelihood (ML) shape. (b) When the control points become asymmetric, ML behaves poorly, since all input mouth shapes are roughly symmetric. (c) For the same control point positions, faceIK creates an asymmetric smile by dividing the mouth into three soft regions (indicated by color variations) and blending the influence of each control point. Each control point influences its region using PBW in real-time. By contrast, using PCA would require computing principal components, a costly operation, at run-time for each new region. (d) With the same control vertices as in (c), if the point on the lower lip is moved by itself, the mouth opens unnaturally, because the two control points on the mouth corners decouple their correlation to the lower lip. (e) With only one control point on the lower lip, the mouth opens more naturally. These comparisons indicate that it is more desirable to adaptively segment a face into regions based on user edits, rather than *a priori*.

$\hat{\mathbf{c}}_l$ and prove that the components of $\mathbf{c}(\mathbf{s}_n)$ sum to 1 given that $\sum_{l=1}^{L} B(\mathbf{s}_n, \mathbf{s}_l) = 1$. For each vertex $\mathbf{s}_n$, we use $\mathbf{c}(\mathbf{s}_n)$ to blend corresponding vertices in the mesh data set.

Figure 7.1 shows the advantage of using local influence maps to adaptively segment the face based on user-interaction, rather than specifying regions *a priori*. The main observation is that the optimal set of regions depends on the desired edit; for in-



(a): 4.02; Seq: 0; Frame: 0; Mesh Idx 24; Num Neighbours 51; a: 1.00 b: 1.00; Distance Penalty 1.00; Coef Thresh 1; Timeout 0.10; Num Bases 20

(b): 4.02; Seq: 0; Frame: 0; Mesh Idx 42; Num Neighbours 51; a: 1.00 b: 1.00; Distance Penalty 1.00; Coef Thresh 1; Timeout 0.10; Num Bases 20

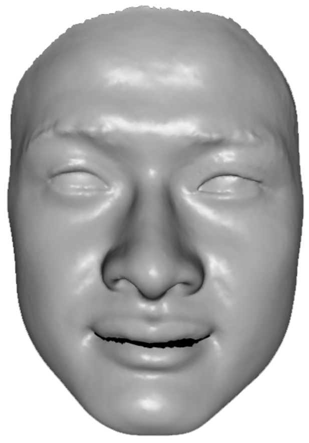
(a) (b)

Figure 7.2: FaceIK user interface. A user can click any point on the face to define a control point constraint, shown as the green ball. The positions of the control point in the input mesh sequence are then shown as a cloud of blue points. The user can move the control point within or slightly outside this point cloud to synthesize a new face that satisfies the constraint. For example, (a) when dragging the lower lip down, the user opens the mouth widely; (b) when dragging the left corner of the mouth to the right, the user pouts the lips.

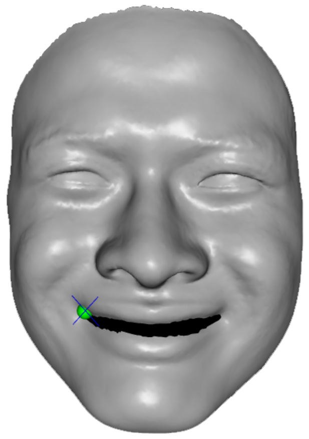
stance, generating an asymmetric smile from a set of roughly symmetric faces requires decoupling the left and right sides of the mouth. However, an edit that opens the mouth is more easily obtained *without* this decoupling. Our PBW scheme supports the adaptive segmentation in real-time.

7.2.3 FaceIK user interface

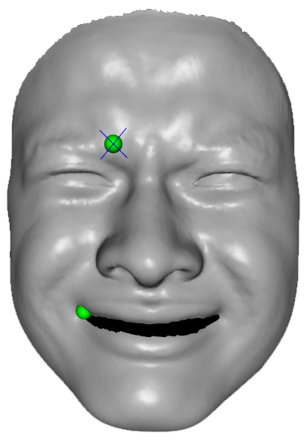
We implemented the proximity-based weighting and local influence map blending in real-time, providing an interactive tool for expression synthesis by direct-manipulation. Figure 7.2 shows the interface of this tool. Figure 7.3 shows a sequence of edits that leads from a neutral face to a complex expression. A screen capture of a real-time



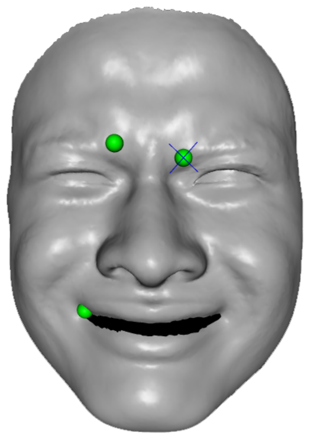
(a)



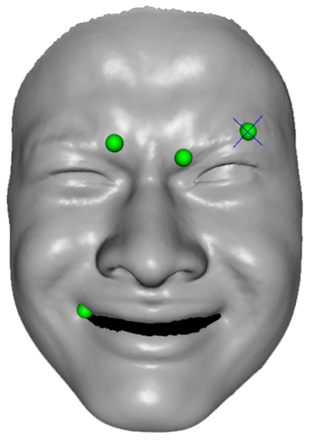
(b)



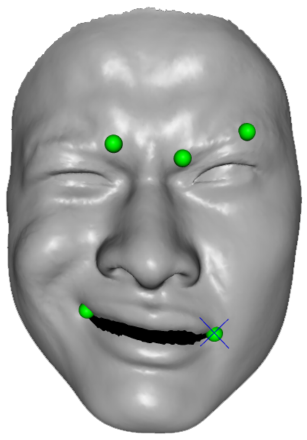
(c)



(d)



(e)



(f)

Figure 7.3: A faceIK editing session. From (a) to (f), we show the creation of a complex expression by adding and moving control points one at a time, starting from neutral.

FaceIK demo is available on our website at

`http://grail.cs.washington.edu/projects/stfaces/capture2_fast2.avi`.

7.3 Data-driven animation

Producing realistic animations of the human face is extremely challenging, as subtle differences in dynamics and timing can have a major perceptual effect. In this section, we exploit the facial dynamics captured in our reconstructed 3D face sequences to create tools for face animation. In particular we describe two such tools, one for producing random infinite face sequences, and another for data-driven interpolation of user-specified key frames.

Before introducing these tools, we first describe our model of face dynamics using a graph representation. Our approach adapts related graph techniques used in video textures [87] and character animation [58, 2, 60] to the domain of face animation.

7.3.1 Face graphs

Let $M_1, \ldots, M_F$ be a sequence of face meshes. We represent face dynamics using a fully connected graph with F nodes corresponding to the faces; we call this the *face graph*. The weight of the edge between nodes i and j specifies the cost of a transition from M_i to M_j in an animation sequence. This cost should respect the dynamics present in the input sequence, balanced with a desire for continuity. Given two frames M_i and M_j in the input sequence, we define the weight w of edge (i, j) as

$$w(i,j) = \texttt{dist}(M_{i+1}, M_j) + \lambda \texttt{dist}(M_i, M_j) \tag{7.9}$$

where `dist` is a distance metric between meshes. (We use the L_2 norm, summed over all the vertices.) The first term prefers following the input sequence, while the second term penalizes large jumps. λ is a weight that balances the two terms. In our experiments, we fixed $\lambda = 2$ for all texture synthesis examples and manually selected λ for each segment in the final keyframe animation to achieve the best visual results.

7.3.2 Random walks through face space

Video textures [87] generates non-repeating image sequences of arbitrary length. The same technique can be used to generate random, continuously-varying face animations. To do so we simply perform a random walk through the face graph. As in the work by Schödl et al. [87], we define the probability of a transition from mesh M_i to mesh M_j to be $P_{ij} = \frac{1}{Z}e^{-w(i,j)/\sigma}$, where Z is a normalization factor so that the sum of P_{ij} over all j is 1. The parameter σ is used to define the frequency of transitions between different parts of the input sequence; lower values create animations that closely follow the input sequence, whereas higher values promote more random behavior. Using P_{ij} as the transition matrix, we can generate random walks on the face graph, and here is an example shown in the online video

`http://grail.cs.washington.edu/projects/stfaces/videotexture.avi`.
The random walks provide a way to visualize the space of captured faces.

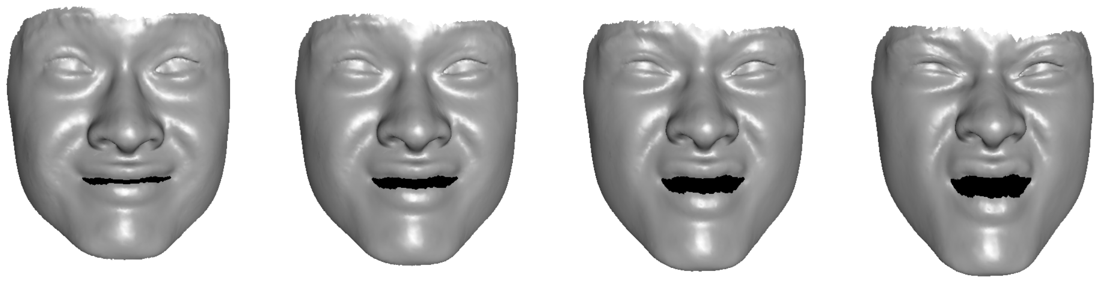
Animation with regions

A limitation of the method described so far is that the frames composing the animation are constrained to lie within the set of input meshes. We therefore generalize this approach by defining regions on the face, animating the regions separately using the above method, and then blending the results into a single animation.

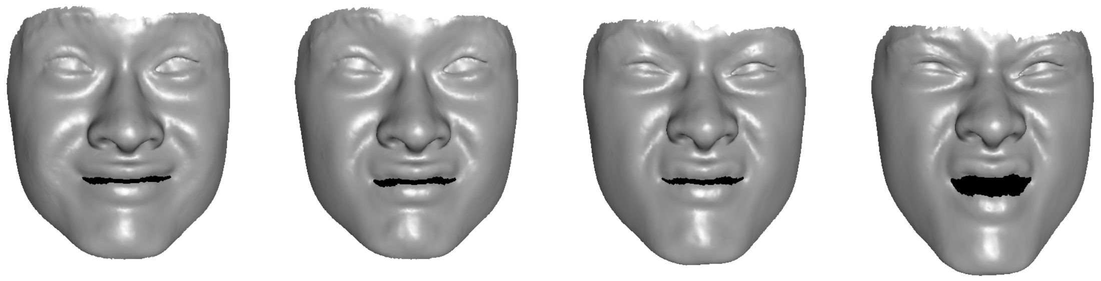
Specifically, we independently generate multiple random walk sequences. For each sequence, we associate a user specified control point; each of these control points defines a influence map as we described in the Section 7.2.2. Then we use these influence maps to blend the multiple random walks into a single random walk. The online video,

`http://grail.cs.washington.edu/projects/stfaces/videotexture.avi`,
also shows an animation generated from this method using two control points. While a majority of the resulting sequence looks natural and compelling, it also contains

some unnatural frames and transitions, due to the fact that different parts of the face are animated independently. The region-based random walks provide a way to visualize the space of faces that can be synthesized given the configuration of control points.



Linear interpolation



Data-driven interpolation

Figure 7.4: Illustration of linear interpolation (top row) vs. data-driven interpolation (bottom row), with the first and last columns as key frames. Linear interpolation makes the mouth and the eyes move synchronously, which looks less realistic when played as an animation (also shown in the online video `http://grail.cs.washington.edu/projects/stfaces/keyframe.avi`). Data-driven interpolation, instead, first purses the mouth, then squints the eyes, and finally opens the mouth. The sequential nature of the data-driven interpolation for this example arose naturally because that is the way the real subject behaved.

7.3.3 Data-driven keyframe animation

While the random walk technique produces animation very easily, it does not provide a mechanism for user control. However, the same concepts may be used to support traditional keyframe animations, in which in-between frames are automatically generated from user-specified constraint frames. The in-between frames are generated using a data-driven interpolation method, which seeks to follow minimum-cost paths through the graph [58, 2, 60, 88].

Suppose that an animator has a sequence of meshes available, and wants to animate a transition between two expressions that appear in the sequence. In the simplest case, the expressions comprise the endpoints of a subsequence of the input sequence. More generally, the interpolation must blend two or more noncontiguous subsequences.

To find a path between two keyframes M_i and M_j, we construct the graph defined above, then search for the shortest path connecting M_i and M_j using Dijkstra's algorithm [59]. The result is a sequence of meshes. However, the length of this sequence may not be the desired number of frames that need to be filled up between the two keyframes. We therefore treat this sequence of faces as a sequence of high-dimensional sample points along a curve in the face space. We then fit a cubic-spline for the samples and resample the spline uniformly to create the described number faces between the two keyframes.

Keyframe animation with regions

The keyframe interpolation method is extended to work with multiple regions using the same technique as described for random walks. In particular, a separate graph is defined for each region. Optimal paths on each graph are computed independently, and the resulting animations are blended together using the influence functions to produce a composite key-frame animation. Figure 7.4 shows an example keyframe animation, comparing our data-driven interpolation to traditional linear interpolation. We have also created a forty three second animation (shown in the online video `http://grail.cs.washington.edu/projects/stfaces/keyframe.avi`) using our data-driven technique. The animation uses nineteen keyframes, and each keyframe has three control points.

7.4 Discussion

In this chapter, we demonstrated a data-driven, interactive method for face editing that draws on the large set of fitted templates and allows specification of new expressions by dragging surface points directly. We also described new tools that model the dynamics in the input sequence to enable new animations, created via key-framing or video texture synthesis techniques. There are many important topics to explore in both face editing and animation for future work directions.

Our faceIK method generates natural results for control points that are relatively near the input faces, but can produce bizarre (and even disturbing) results for larger extrapolations, as shown in Figure 7.5. Although this behavior is not surprising, the user must explore the space of faces by trial and error. More useful would be if the tool could constrain the edits to the range of *reasonable* faces, by learning a set of good controls. Another limitation of faceIK is that, although it allows the user to segment the face adaptively, within an animation the segmentation cannot be changed. Therefore, before creating an animation, the user must specify enough control points so that any desired keyframe can be created. This limitation did not pose a problem when we created the animation in the accompanying video; three control points (on both eyebrows and the lower lip) were enough to create the expressions we wanted. However, it would be desirable to allow the number and positions of control points to vary across keyframes.

An alternative to computing blending coefficients using our PBW method is to

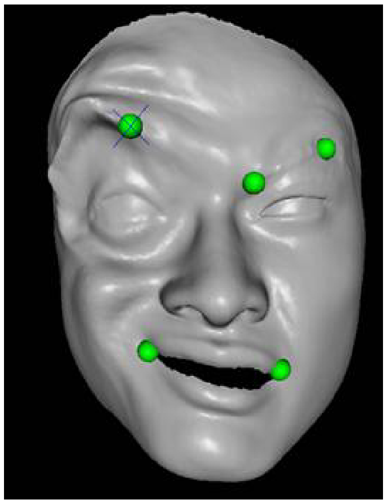


Figure 7.5: An example of weird expression due to large extrapolation.

formulate face editing as a scattered data interpolation problem from the configuration space of control vertices to the configuration space of all other vertices. The difference between PBW and this alternative approach is that the interpolation coefficients of the latter may not make the control points satisfy the user-specified constraints. Since we could avoid applying the interpolation coefficients on the control vertices by simply setting their values to be the user specified constraints, this alternative method is still worth further study.

The animation tools presented in this paper are quite simple and could be extended and improved in several ways. One limitation is that we assume that the key frames are blends of the input frames. Although different regions can be controlled separately, the approach does not provide good support for extrapolated faces, since such faces may not be part of a motion sequence (i.e., there is no natural successor frame). Another limitation is that our data-driven interpolation technique requires a rich face sequence as input in order to produce natural-looking output transitions. If our technique fails to find a desired transition in the face sequence, it may choose to use linear interpolation or an unnatural backwards transition instead. In addition, eye blinks may occur at inopportune moments and gaze direction is fixed by the input sequence; more careful modeling of the eyes, as well as insertion of teeth, would improve the quality of resulting animations. Finally, more intuitive control of timing would also help produce more realistic keyframe animations. All of these problems are important areas for future work.

While we have focused on animating a single face, it would be interesting to explore variations in dynamic behaviors among different human faces, similar in spirit to what has been done for static shape variations [9, 1].

Chapter 8

CONCLUSION

In this dissertation, we have presented new techniques that take raw image streams recorded with off-the-shelf hardware and produce high-resolution, time-varying surface models. We have also developed tools that use these models for 3D facial expression synthesis and animation applications. In this section, we review our technical contributions and suggest areas for future work.

8.1 Contributions

This dissertation has five major contributions. First, we have proposed an improved structured light method for generating high-speed scans of moving objects. The method works by projecting a single, static stripe pattern of alternating colors while simultaneously recording an image stream. The 3D shape is recovered by matching the projected color transitions with observed edges in the image. We solve the correspondence problem using a novel, multi-pass dynamic programming algorithm that eliminates global smoothness assumptions, strict ordering constraints, and singly-connected surface limitations present in previous formulations.

Second, we have introduced an approach called spacetime stereo, in which the temporal variation of each pixel is used as a cue for correspondence. Specifically, instead of using a spatial window to define matching cost as in traditional frame-by-frame stereo, spacetime stereo adds a temporal dimension to the window. The spacetime window can be a rectangular 3D volume if the local shape variations in space and time are negligible. For significant shape variations, we have proposed a locally linear model to approximate the variations and formulated spacetime stereo as a spacetime window warping problem. Starting with our local warping algorithm, we have demonstrated that spacetime stereo improves spatial resolution and temporal stability for shape capture of moving objects. However, the local method does not take into account intrinsic constraints between windows of neighboring pixels, resulting in an over-fitting problem. We have therefore developed a global optimization technique that takes into account all such constraints and further improves spacetime stereo reconstruction.

Spacetime stereo works effectively for dynamic scenes with smoothly changing geometry and rapidly changing texture. First, it serves as a general framework for structured light scanning, in which laser scanning, hierarchical striping, and active stereo are subsumed as special cases. Within this framework, the spacetime window warping algorithm is one of very few existing methods that can robustly reconstruct objects that are moving and deforming over time. Beyond the controlled structured light scanning application, spacetime stereo is also effective for a class of natural scenes, such as waving trees and flowing water, which have repetitive textures and chaotic behaviors and are challenging for existing stereo algorithms.

Third, we have presented a template fitting and tracking method to integrate shape estimation and optical flow estimation from multiple viewpoints. This method computes a sequence of surface meshes with vertex to vertex correspondence, such that each surface mesh approximates the input shapes and each vertex moves in agreement with the optical flow.

Fourth, based on our proposed algorithms, we have developed a research prototype system that consists of off-the-shelf cameras, projectors, and computers for dynamic 3D face capture. Our system advances the state of the art of facial motion capture solutions: it is able to continuously capture (20fps) high resolution (about 20K vertices) facial motion of a dynamic face simultaneously from multiple viewpoints without the need to paint markers on the face.

Finally, we have developed data-driven facial animation tools that take advantage of the spatio-temporally dense data. Specifically, we have presented an expression synthesis tool, faceIK, that exploits the correlation in the large set of captured 3D faces and allows for posing new expressions by dragging surface points directly. Based on user edits, the tool also decouples the correlation between different regions on the face — adaptively and automatically — at runtime. This is achieved by computing spatial varying weights to blend input faces, thereby enabling the creation of new expressions that do not exist in the database of captured facial expressions for that individual. For example, asymmetric faces can be created from a set of symmetric faces. We have also developed a representation called *face graph* that encodes the dynamics of a face mesh sequence. This graph can be traversed to create a variety of animations via key-framing or video texture techniques. We demonstrated that interpolating keyframes via the face graph preserves the relative timing of the motion over different regions of the face in the input sequence, and thereby generates more realistic animation than classic linear interpolation does. Our approach makes it simple for untrained users to produce expressive facial animations.

8.2 Future work

At the end of each previous chapter, we discussed limitations of our methods and suggested relatively immediate research topics for future work. In this section, we propose a few more ambitious problems that are based on measuring and modeling 3D dynamic shapes.

The dynamics of rigid bodies, solids, and fluids are relatively well understood; however, realistic modeling of living biological forms, such as human faces and organs, remains an important unsolved problem in bioengineering research. Today cameras can be deployed to capture visual information almost everywhere, even inside human bodies. Based on the reconstructed 3D models, many interesting research topics can be explored in dynamic shape modeling.

Learning controls for facial animation The low-level facial animation tools developed in this dissertation, e.g., dragging points on faces and creating key frames, are only a first step toward facial modeling. A more ambitious problem is to learn computational models that describe human emotions. To address this problem, one approach is to first capture facial motions and voices of human subjects under various moods. Then we can consider the following open questions: How does emotion affect our expression? How does it affect our voice tone? Answering these questions will be very useful in graphics for modeling human behavior, as it helps to create high-level controls for animation. For example, can we edit an existing facial animation by tweaking a "knob" to make the face look happier across the entire sequence? It may also be useful as a computational framework for psychologists [33] to quantify their description of human behaviors and for expression and emotion recognition [34] in computer vision and human computer interaction.

Learning dynamic skin models When we move, our bones interact with the skin through body tissues in a complex way. For example, when we open our hands, we see tendons contracting on the back of the hand; when we wave, a muscle quivers on the forearm. How to model and simulate the relationship between bones and skin is still an open problem in biomechanics. In graphics, existing methods for describing this relationship, a.k.a., *skinning*, either treat it as a static mapping or simulate it assuming body tissues are elastic solids. Consequently, existing skinning methods do not effectively reproduce realistic shape details and secondary motion of human bodies. An important research challenge is to learn a *dynamic* statistical model that directly relates captured dynamic body shapes to skeleton motions obtained from motion capture data. This approach may avoid complex physical simulation and yet still realistically reproduce both shape and secondary motion.

Learning organ mechanics Today even human organs can be observed by cameras, for example, in minimally invasive surgeries. There, surgeons operate on an organ by looking at videos taken inside the body through laparoscopes. An organ has extremely complex (anisotropic, inhomogeneous) mechanical properties, and these properties change as the organ changes its shape. Therefore, it is very hard to predict the state and the property of the organ once the surgeon bends or cuts it. An important research challenge is to reconstruct organ shapes from laparoscopic imagery and register these shapes to the organ model acquired before surgery, e.g., using M.R.I. With the aid of force sensors, another interesting problem is how to learn the mechanical properties of the organs from the reconstructed shape sequence [89]. Solving these problems will aid in surgical planning during an operation and surgical simulation for training. Understanding human tissue properties can also be used to simulate bodily injuries, e.g., from an automobile accident or in a battle field, and to design better seats and beds to maximize comfort and minimize injury risk.

Comparative study in populations Orthogonal to the study of statistical and physical properties of human faces, bodies, and organs, an important research challenge is to model these property variations across people. The study of human body measurement is the main goal of anthropometry, in which only static shapes are considered. Today, techniques allow us to measure not only static but also dynamic shapes, not only from outside but also from inside the human body. Studying the distribution of these new measurements over a population is of great scientific importance. Continuing on Allen et al.'s work on modeling static shape variations [1], an interesting research topic is to apply machine learning methods to study how much static attributes, like body size, weight, age, and sex, affect the aforementioned statistical and dynamical properties of different people. Understanding these correlations will enrich the applications of dynamic human shape modeling.

BIBLIOGRAPHY

[1] B. Allen, B. Curless, and Z. Popovic. The space of human body shapes: reconstruction and parameterization from range scans. In *SIGGRAPH Conference Proceedings*, pages 587–594, 2003.

[2] O. Arikan and D. A. Forsyth. Synthesizing constrained motions from examples. In *SIGGRAPH Conference Proceedings*, pages 483–490, 2002.

[3] H. H. Baker and T. O. Binford. Depth from edges and intensity based stereo. In *Int. Joint Conf. on Artificial Intelligence*, pages 631–636, 1981.

[4] Simon Baker and Iain Matthews. Lucas-kanade 20 years on: A unifying framework. *International Journal of Computer Vision*, 56(3):221 – 255, March 2004.

[5] Sumit Basu, Nuria Oliver, and Alex Pentland. 3d lip shapes from video: A combined physical-statistical model. *Speech Communication*, 26(1):131–148, 1998.

[6] P. N. Belhumeur. A bayesian approach to binocular stereopsis. *Int. J. on Computer Vision*, 19(3):237–262, 1996.

[7] M. J. Black and P. Anandan. Robust dense optical flow. In *Proc. Int. Conf. on Computer Vision*, pages 231–236, 1993.

[8] V. Blanz, C. Basso, T. Poggio, and T. Vetter. Reanimating faces in images and video. In *Proceedings of EUROGRAPHICS*, pages 641–650, 2003.

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

[10] A. F. Bobick and S. S. Intille. Large occlusion stereo. *Int. J. on Computer Vision*, 33(3):181–200, 1999.

[11] J.-Y. Bouguet and P. Perona. 3D photography on your desk. In *Proc. Int. Conf. on Computer Vision*, pages 43–50, 1998.

[12] Jean-Yves Bouguet. *Camera Calibration Toolbox for Matlab*. http://www.vision.caltech.edu/bouguetj/calib_doc/index.html, 2001.

[13] K. L. Boyer and A. C. Kak. Color-encoded structured light for rapid active ranging. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 9(1):14–28, 1987.

[14] M. Brand. Voice puppetry. In *SIGGRAPH Conference Proceedings*, pages 21–28, 1999.

[15] M. Brand. Morphable 3D models from video. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 456–463, 2001.

[16] C. Bregler, M. Covell, and M. Slaney. Video rewrite: Visual speech synthesis from video. In *SIGGRAPH Conference Proceedings*, pages 353–360, 1997.

[17] D. S. Broomhead and D. Lowe. Multivariable functional interpolation and adptive networks. *Complex Systems*, 2:321–355, 1988.

[18] R. L. Carceroni and K. N. Kutulakos. Scene capture by surfel sampling: from multi-view streams to non-rigid 3D motion, shape and reflectance. In *Proc. Int. Conf. on Computer Vision*, pages 60–67, 2001.

[19] D. Caspi, N. Kiryati, and J. Shamir. Range imaging with adaptive color structured light. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 20(5):470–480, 1998.

[20] Y. Caspi and M. Irani. A step towards sequence to sequence alignment. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 682–689, 2000.

[21] J. Chai, X. Jin, and J. Hodgins. Vision-based control of 3d facial animation. In *Proceedings of Eurographics/SIGGRAPH Symposium on Computer Animation*, pages 193–206, 2003.

[22] C.S. Chen, Y.P. Hung, C.C. Chiang, and J.L. Wu. Range data acquisition using color strucred lighting and stereo vision. *Image and Vision Computing*, 15:445–456, 1997.

[23] T. F. Cootes, C. J. Taylor, D. H. Cooper, and J. Graham. Active shape models—their training and application. *Computer Vision and Image Understanding*, 61(1):38–59, 1995.

[24] Ascension Technology Corporation. *MotionStar*. http://www.ascension-tech.com/products/motionstar.php.

[25] I. J. Cox, S. L. Hingorani, S. B. Rao, and B. M. Maggs. A maximum likelihood stereo algorithm. *Computer Vision and Image Understanding*, 63(3):542–567, 1996.

[26] B. Curless. *New methods for surface reconstruction from range images.* PhD thesis, Stanford University, 1997.

[27] B. Curless and M. Levoy. Better optical triangulation through spacetime analysis. In *Proc. Int. Conf. on Computer Vision*, pages 987–994, June 1995.

[28] B. Curless and M. Levoy. A volumetric method for building complex models from range images. In *SIGGRAPH Conference Proceedings*, pages 303–312, 1996.

[29] Cyberware Inc., http://www.cyberware.com/. *Model 15 scanner*, 2001.

[30] C. J. Davies and M. S. Nixon. A hough transformation for detecting the location and orientation of three-dimensional surfaces via color encoded spots. *IEEE Trans. on Systems, Man, and Cybernetics*, 28(1B):90–95, 1998.

[31] J. Davis, R. Ramamoorthi, and S. Rusinkiewicz. Spacetime stereo: A unifying framework for depth from triangulation. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 359–366, 2003.

[32] D. DeCarlo and D. Metaxas. Adjusting shape parameters using model-based optical flow residuals. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 24(6):814–823, 2002.

[33] P. Ekman, W. V. Friesen, and J. C. Hager. *THE FACIAL ACTION CODING SYSTEM.* Weidenfeld and Nicolson, 2002.

[34] I. Essa and A. Pentland. Coding, analysis, interpretation and recognition of facial expressions. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 19(7), 1997.

[35] Irfan Essa, Sumit Basu, Trevor Darrell, and Alex Pentland. Modeling, tracking and interactive animation of faces and heads using input from video. In *Proceedings of the Computer Animation*, pages 68–79. IEEE Computer Society, 1996.

[36] T. Ezzat, G. Geiger, and T. Poggio. Trainable videorealistic speech animation. In *SIGGRAPH Conference Proceedings*, pages 388–398, 2002.

[37] O. Faugera. *Three-Dimensional Computer Vision*. MIT Press, 1993.

[38] O. Faugeras. *Three-Dimensional Computer Vision*. MIT Press, 1993.

[39] P. Fong and F. Buron. Sensing moving and deforming objects with commercial off the shelf hardware. In *IEEE International Workshop on Projector-Camera Systems (PROCAMS)*, 2005.

[40] H. Fredricksen. The lexicographically least debruijn cycle. *Journal of Combinatorial Theory*, 9(1):509–510, 1970.

[41] D. Geiger, B. Ladendorf, and A. Yuille. Occlusions and binocular stereo. *Int. J. on Computer Vision*, 14(3):211–226, 1995.

[42] H. Gonzales-Banos and J. Davis. A method for computing depth under ambient illumination using multi-shuttered light. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 234–241, 2004.

[43] P. Griffin and L. Narasimhan. Generation of uniquley encoded light patterns for range data acquisition. *Pattern Recognition*, 25(6):609–616, 1992.

[44] B. Guenter, C. Grimm, D. Wood, H. Malvar, and F. Pighin. Making faces. In *SIGGRAPH Conference Proceedings*, pages 55–66, 1998.

[45] O. Hall-Holt and S. Rusinkiewicz. Stripe boundary codes for real-time structured-light range scanning of moving objects. In *Proc. Int. Conf. on Computer Vision*, pages 359–366, 2001.

[46] R. I. Hartley and A. Zisserman. *Multiple View Geometry*. Cambridge University Press, 2000.

[47] P. S. Huang, C. P. Zhang, and F. P. Chiang. High speed 3-d shape measurement based on digital fringe projection. *Optical Engineering*, 42(1):163–168, 2003.

[48] H. Hügli and G. Maïtre. Generation and use of color pseudo random sequences for coding structured light in active ranging. In *Proceedings of Industrial Inspection*, volume 1010, pages 75–82, 1989.

[49] Impact Studio, http://www.laserscan3d.com/. *3D Laser Scanning Services*, 2001.

[50] Canesta Inc. *Electronic Perception Development Kit*. http://www.canesta.com.

[51] H. Ishikawa and D. Geiger. Occlusions, discontinuities, and epipolar lines in stereo. In *Eur. Conf. on Computer Vision*, pages 232–248, 1998.

[52] A. Jones, A. Gardner, M. Bolas, I. McDowall, and P. Debevec. Performance geometry capture for spatially varying relighting. In *ACM SIGGRAPH Conference Sketch*, page to appear, August 2005.

[53] P. Joshi, W. C. Tien, M. Desbrun, and F. Pighin. Learning controls for blend shape based realistic facial animation. In *Proceedings of Eurographics/SIGGRAPH Symposium on Computer Animation*, pages 187–192, 2003.

[54] T. Kanade, A. Gruss, and L. Carley. A very fast vlsi rangefinder. In *Proc. Int. Conf. on Robotics and Automation*, volume 39, pages 1322–1329, April 1991.

[55] T. Kanade and M. Okutomi. A stereo matching algorithm with an adaptive window: Theory and experiment. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 16(9):920–932, 1994.

[56] S. B. Kang, J. A. Webb, C. L. Zitnick, and T. Kanade. A multibaseline stereo system with active illumination and real-time image acquisition. In *Proc. Int. Conf. on Computer Vision*, pages 88–93, June 1995.

[57] V. Kolmogorov and R. Zabih. Multi-camera scene reconstruction via graph cuts. In *Proc. Eur. Conf. on Computer Vision*, pages 82–96, 2002.

[58] L. Kovar, M. Gleicher, and F. Pighin. Motion graphs. In *SIGGRAPH Conference Proceedings*, pages 473–482, 2002.

[59] D. C. Kozen. *The Design and Analysis of Algorithms.* Springer, 1992.

[60] J. Lee, J. Chai, P. S. S. Reitsma, J. K. Hodgins, and N. S. Pollard. Interactive control of avatars animated with human motion data. In *SIGGRAPH Conference Proceedings*, pages 491–500, 2002.

[61] Y. Li, T. Wang, and H.-Y. Shum. Motion texture: A two-level statistical model for character motion synthesis. In *SIGGRAPH Conference Proceedings*, pages 465–472, 2002.

[62] 3Q Technologies Ltd. *QloneratorPRO.* http://www.3q.com.

[63] B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Int. Joint Conf. on Artificial Intelligence*, pages 674–679, 1981.

[64] R. Mandelbaum, G. Salgian, and H. Sawhney. Correlation based estimatin of ego-motion and structure from motion and stereo. In *Proc. Int. Conf. on Computer Vision*, pages 544–550, 1999.

[65] Measurand. *Shape Tape.* http://www.measurand.com/products/ShapeTape.html.

[66] T. Monks and J. Carer. Improved stripe matching for color encoded structured light. In *Internationl Conference on Computer Analysis of Images and Patterns*, pages 476–485, 1993.

[67] H. Morita, K. Yajima, and S. Sakata. Reconstruction of surfaces of 3d objects by m-array pattern projection method. In *Proc. Int. Conf. on Computer Vision*, pages 468–473, 1988.

[68] Meta Motion. *Motion Captor.* http://www.metamotion.com/captor/motion-captor.htm.

[69] Shree K. Nayar, Masahiro Watanabe, and Minori Noguchi. Real-time focus range sensor. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(12):1186–1198, 1996.

[70] J. Nocedal and S. J. Wright. *Numerical Optimization.* Springer, 1999.

[71] Y. Ohta and T. Kanada. Stereo by intra- and inter-scanline searching using dynamic programming. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 7(2):139–154, 1985.

[72] Frederick I. Parke. Computer generated animation of faces. In *Proceedings of the ACM annual conference*, pages 451–457. ACM Press, 1972.

[73] A. Pentland. A new sense for depth of field. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 9(4):523–531, 1987.

[74] F. Pighin, J. Hecker, D. Lischinski, D. H. Salesin, and R. Szeliski. Synthesizing realistic facial expressions from photographs. In *SIGGRAPH Conference Proceedings*, pages 75–84, 1998.

[75] F. Pighin, D. H. Salesin, and R. Szeliski. Resynthesizing facial animation through 3D model-based tracking. In *Proc. Int. Conf. on Computer Vision*, pages 143–150, 1999.

[76] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recipes in C : The Art of Scientific Computing.* Cambridge University Press, 2nd edition, 1993.

[77] M. Proesmans, L. Van Gool, and A. Oosterlinck. Active acquisition of 3d shape for moving objects. In *Int. Conf. on Image Processing*, pages 647–650, 1996.

[78] M. Proesmans, L. Van Gool, and A. Oosterlinck. One-shot active 3d shape acquization. In *Int. Conf. on Pattern Recognition*, pages 336–340, 1996.

[79] K. Pulli, H. Abi-Rached, T. Duchamp, L. Shapiro, and W. Stuetzle. Acquisition and visualization of colored 3D objects. In *Proc. Int. Conf. on Pattern Recognition*, pages 11–15, 1998.

[80] K. Pulli and M. Ginzton. *Scanalyze.* http://graphics.stanford.edu/software/scanalyze/, 2002.

[81] Ramesh Raskar, Greg Welch, Matt Cutts, Adam Lake, Lev Stesin, and Henry Fuchs. The office of the future: A unified approach to image-based modeling and spatially immersive displays. In *SIGGRAPH Conference Proceedings*, pages 179–188, 1998.

[82] Frank Ruskey. *The Combinatorial Object Server.* http://www.theory.csc.uvic.ca/˜cos/, 2000.

[83] K. Sato and S. Inokuchi. Three-dimensional surface measurement by space encoding range imaging. *Journal of Robotic System*, 2:27–39, 1985.

[84] K. Sato and S. Inokuchi. Range-imaging system utilizing nematic liquid crystal mask. In *Proc. Int. Conf. on Computer Vision*, pages 657–661, 1987.

[85] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *Int. J. on Computer Vision*, 47(1):7–42, 2002.

[86] D. Scharstein and R. Szeliski. High-accuracy stereo depth maps using structured light. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, volume 1, pages 195–202, 2003.

[87] A. Schödl, S. Szeliski, D. H. Salesin, and I. Essa. Video textures. In *SIGGRAPH Conference Proceedings*, pages 489–498, 2000.

[88] Arno Schödl and Irfan A. Essa. Controlled animation of video sprites. In *Proceedings of Eurographics/SIGGRAPH Symposium on Computer Animation*, pages 121–127. ACM Press, 2002.

[89] M. Sermesant, C. Forest, X. Pennec, H. Delingette, and N. Ayache. Deformable biomechanical models: Application to 4d cardiac image analysis. *Medical Image Analysis*, 7(4):475–488, 2003.

[90] C. Strecha and L. V. Gool. Motion - stereo integration for depth estimation. In *Proc. Eur. Conf. on Computer Vision*, pages 170–185, 2002.

[91] 3DV Systems. *ZCam*. http://www.3dvsystems.com.

[92] H. Tao, H. S. Sawhney, and R. Kumar. Dynamic depth recovery from multiple synchronized video streams. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 118–124, 2001.

[93] L. Torresani, D. B. Yang, E. J. Alexander, and C. Bregler. Tracking and modeling non-rigid objects with rank constraints. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 493–500, 2001.

[94] G. Turk and M. Levoy. Zippered polygonal meshes from range images. In *SIGGRAPH*, pages 311–318, 1994.

[95] S. Vedula, S. Baker, P. Rander, R. Collins, and T. Kanade. Three-dimensional scene flow. In *Proc. Int. Conf. on Computer Vision*, pages 722–729, 1999.

[96] A. Wenger, A. Gardner, C. Tchou, J. Unger, T. Hawkins, and P. Debevec. Postproduction relighting and reflectance transformation with time-multiplexed illumination. In *SIGGRAPH Conference Proceedings*, page to appear, August 2005.

[97] R. J. Woodham. Photometric method for determining surface orientation from multiple iamges. *Optical Engineering*, 19(1):139–144, 1980.

[98] C. Wust and D. W. Capson. Surface profile measurement using color fringe projection. *Macine Vision and Applications*, 4:193–203, 1991.

[99] L. Zhang, B. Curless, and S. M. Seitz. Rapid shape acquisition using color structured light and multi-pass dynamic programming. In *Proc. Int. Symp. on 3D Data Processing, Visualization, and Transmission*, 2002.

[100] L. Zhang, B. Curless, and S. M. Seitz. Spacetime stereo: Shape recovery for dynamic scenes. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 367–374, 2003.

[101] Li Zhang and Steven M. Seitz. Parameter estimation for mrf stereo. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, page to appear, June 2005.

[102] Li Zhang, Noah Snavely, Brian Curless, and Steven M. Seitz. Spacetime faces: High-resolution capture for modeling and animation. In *ACM Annual Conference on Computer Graphics*, pages 548–558, August 2004.

[103] Q. Zhang, Z. Liu, B. Guo, and H. Shum. Geometry-driven photorealistic facial expression synthesis. In *Proceedings of Eurographics/SIGGRAPH Symposium on Computer Animation*, pages 177–186, 2003.

[104] Y. Zhang and C. Kambhamettu. Integrated 3D scene flow and structure recovery from multiview image sequences. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 674–681, 2000.

[105] Y. Zhang and C. Kambhamettu. On 3D scene flow and structure estimation. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 592–606, 2001.

Appendix A

THE CLOSED-FORM SOLUTION TO PBW

In this appendix, we show that Eq. (7.3) minimizes Eq. (7.2) subject to Eq. (7.1). We first rewrite this constrained minimization problem using matrices as follows,

$$\begin{aligned} &\texttt{Mininize} \quad g(\mathbf{c}) = \mathbf{c}^{\mathsf{T}}\mathbf{Q}\mathbf{c} \\ &\texttt{such that} \quad \mathbf{B}\mathbf{c} = \bar{\mathbf{p}} \end{aligned} \tag{A.1}$$

where

$$\mathbf{Q} = \begin{bmatrix} \phi_1 & 0 & \cdots & 0 \\ 0 & \phi_2 & \cdots & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & \cdots & \phi_F \end{bmatrix} \qquad \mathbf{B} = \begin{bmatrix} \mathbf{s}_{1,1} & \mathbf{s}_{1,2} & \cdots & \mathbf{s}_{1,F} \\ \mathbf{s}_{2,1} & \mathbf{s}_{2,2} & \cdots & \mathbf{s}_{2,F} \\ \vdots & \vdots & \vdots & \vdots \\ \mathbf{s}_{L,1} & \mathbf{s}_{L,2} & \cdots & \mathbf{s}_{L,F} \\ 1 & 1 & \cdots & 1 \end{bmatrix} \tag{A.2}$$

Using Lagrange multipliers, we know the minimizer of Eq. (A.1) should satisfy the following equation,

$$\mathbf{Q}\mathbf{c} = \mathbf{B}^{\mathsf{T}}\mathbf{a} \tag{A.3}$$

where **a** is the Lagrange multiplier. From Eq. (A.3), we obtain the following relation between **c** and **a**

$$\mathbf{c} = \mathbf{Q}^{-1}\mathbf{B}^{\mathsf{T}}\mathbf{a} = \begin{bmatrix} \frac{1}{\phi_1}\bar{\mathbf{s}}_1^{\mathsf{T}}\mathbf{a} \\ \frac{1}{\phi_2}\bar{\mathbf{s}}_2^{\mathsf{T}}\mathbf{a} \\ \vdots \\ \frac{1}{\phi_F}\bar{\mathbf{s}}_F^{\mathsf{T}}\mathbf{a} \end{bmatrix} \tag{A.4}$$

Substituting Eq. (A.4) into the constraint equation in Eq. (A.1), we have

$$\mathbf{B}\mathbf{c} = \mathbf{B}\mathbf{Q}^{-1}\mathbf{B}^{\mathsf{T}}\mathbf{a} = \bar{\mathbf{p}} \tag{A.5}$$

Solving Eq. (A.5), we have

$$\mathbf{a} = (\mathbf{B}\mathbf{Q}^{-1}\mathbf{B}^{\mathsf{T}})^{-1}\bar{\mathbf{p}} = (\sum_{f=1}^{F} \frac{1}{\phi_f} \bar{\mathbf{s}}_f \bar{\mathbf{s}}_f^{\mathsf{T}})^{-\mathbf{1}}\bar{\mathbf{p}} \tag{A.6}$$

Eq. (A.6) and Eq. (A.4) together show that Eq. (7.3) indeed minimizes Eq. (7.2) subject to Eq. (7.1).

Appendix B

THE NORMALIZED PROPERTY OF SOFT REGION WEIGHTS

In this appendix, we prove that the blending coefficients for vertex $\mathbf{s}$, $\mathbf{c}(\mathbf{s})$, sum up to 1 given that $\sum_{l=1}^{L} B(\mathbf{s}, \mathbf{s}_l) = 1$. Our proof is based on two facts. To state the facts succinctly, we first introduce two concepts. A vector is called *normalized* if its components sum to 1. A matrix is called *normalized* if all of its rows are normalized.

FACT1. If an invertible square matrix $\mathbf{A} = [a_{i,j}]$ is normalized, then $\mathbf{B} = \mathbf{A}^{-1}$ is also normalized.

PROOF. Let $\mathbf{B} = [b_{i,j}]$. $\mathbf{B} = \mathbf{A}^{-1} \Rightarrow \forall i, \forall j, \sum_k b_{i,k} a_{k,j} = \delta_{i,j} \Rightarrow \forall i, 1 = \sum_j \delta_{i,j} = \sum_j \sum_k b_{i,k} a_{k,j} = \sum_k \sum_j b_{i,k} a_{k,j} = \sum_k b_{i,k} \sum_j a_{k,j} = \sum_k b_{i,k}$.

FACT2. If an m by n matrix $\mathbf{A} = [a_{i,j}]$ is normalized and an m dimensional vector $\mathbf{b} = [b_i]$ is normalized, then the n dimensional vector $b^{\mathsf{T}}\mathbf{A}$ is also normalized.

PROOF. $b^{\mathsf{T}}\mathbf{A} = [\sum_i b_i a_{i,j}] \Rightarrow \sum_j \sum_i b_i a_{i,j} = \sum_i \sum_j b_i a_{i,j} = \sum_i b_i \sum_j a_{i,j} = \sum_i b_i = 1$.

Let $\mathbf{C} = [\mathbf{c}_1\ \mathbf{c}_2\ \ldots\ \mathbf{c}_L]^{\mathsf{T}}$ and $\hat{\mathbf{C}} = [\hat{\mathbf{c}}_1\ \hat{\mathbf{c}}_2\ \ldots\ \hat{\mathbf{c}}_L]^{\mathsf{T}}$. We know from the construction of the RBF that $\mathbf{C} = [B(\mathbf{s}_{l'}, \mathbf{s}_l)]\,\hat{\mathbf{C}}$. Because both $\mathbf{C}$ and $[B(\mathbf{s}_{l'}, \mathbf{s}_l)]$ are normalized, the matrix $\hat{\mathbf{C}} = [B(\mathbf{s}_{l'}, \mathbf{s}_l)]^{-1}\mathbf{C}$ is also normalized, according to FACT1 and FACT2. Again, from the definition of RBF, we know that $\mathbf{c}(\mathbf{s})^{\mathsf{T}} = [B(\mathbf{s}, \mathbf{p}_l)]^{\mathsf{T}}\hat{\mathbf{C}}$. Because both vector $[B(\mathbf{s}, \mathbf{p}_l)]$ and matrix $\hat{\mathbf{C}}$ are normalized, $\mathbf{c}(\mathbf{s})$, the blending coefficient for vertex $\mathbf{s}$, is also normalized.

VITA

Li Zhang obtained his B.E. in Automation, with a minor in Radio Technology and Information System, at Tsinghua University in Beijing, P.R. China, in 1998. He then began his PhD study in the Robotics Institute at Carnegie Mellon University. After two years, he transferred, with his advisor Steve Seitz, to the Department of Computer Science and Engineering at University of Washington, where he was awarded a Master and a PhD degree in 2001 and 2005, respectively. His PhD thesis focuses on 3D shape recovery from videos and its applications in animation.