Silhouettes for Calibration and Reconstruction from Multiple Views

Sudipta N. Sinha

A dissertation submitted to the faculty of the University of North Carolina at Chapel Hill in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Computer Science.

Chapel Hill
2009

Approved by:

Marc Pollefeys, Advisor
Leonard McMillan, Reader
Jean Ponce, Reader
Henry Fuchs, Reader
Guido Gerig, Reader

ABSTRACT

SUDIPTA N. SINHA: Silhouettes for Calibration and Reconstruction from
Multiple Views.
(Under the direction of Marc Pollefeys)

In this thesis, we study how silhouettes extracted from images and video can help with two fundamental problems of 3D computer vision - namely multi-view camera calibration and 3D surface reconstruction from multiple images.

First, we present an automatic method for calibrating a network of cameras that works by analyzing only the motion of silhouettes in the multiple video streams. This is particularly useful for automatic reconstruction of a dynamic event using a camera network in a situation where pre-calibration of the cameras is impractical or even impossible. Our key contribution is a novel RANSAC-based algorithm that simultaneously computes the epipolar geometry and synchronization of a pair of cameras, only from the motion of silhouettes in video. The approach proceeds by first independently computing the epipolar geometry and synchronization for pairs of cameras in the network. In the next stage, the calibration and synchronization for the complete network is recovered.

The fundamental matrices from the first stage are used to determine a projective reconstruction, which is then upgraded to a metric reconstruction using self-calibration. Finally, a visual-hull algorithm is used to reconstruct the shape of the dynamic object from its silhouettes in video. For unsynchronized video streams with sub-frame temporal offsets, we interpolate silhouettes between successive frames to get more accurate visual hulls.

In the second part of the thesis, we address some short-comings of existing volumetric multi-view stereo approaches. First we propose a novel formulation for multi-view stereo that

allows for robust and accurate fusion of the silhouette and stereo cues. We show that it is possible to enforce exact silhouette constraints within the graph-cut optimization step in the volumetric multi-view stereo algorithm. This guarantees that the reconstructed surface will be exactly consistent with the original silhouettes. Contrary to previous work on silhouette and stereo fusion, the silhouette consistency is guaranteed by construction through hard constraints in the graph-cut problem – the silhouette consistency terms are not part of the energy minimization problem which aims to find a surface with maximal photo-consistency.

Finally, we have also developed an adaptive graph construction approach for graph-cut based multi-view stereo to address the inherent high memory and computational overhead of the basic algorithm. The approach does not need any initialization and is not restricted to a specific surface topology which is a limitation with existing methods that use a base surface for initialization. Using this method, we have been able to efficiently reconstruct accurate and detailed 3D models of objects from high-resolution images for a number of different datasets.

This thesis is dedicated to my parents.

ACKNOWLEDGMENTS

I would like to first thank Marc Pollefeys for having me as his student and providing me the opportunity to pursue the research that led to this dissertation. It has been a privilege to have him as my advisor. His enthusiasm and love for his research area motivated me to pursue research as a career and have fun. He has been extremely helpful, provided me with great insight on various problems and immense encouragement through my stint at graduate school.

I would also like to thank Henry Fuchs, Leonard Mcmillan, Guido Gerig and Jean Ponce for being on my committee and for their suggestions, and advice on how to improve this dissertation. I am grateful to the other faculty members at UNC Chapel Hill for their guidance at various points in time and teaching me the nuts and bolts of doing research. I am also grateful to Herman Towles, who during my first year at UNC as a graduate student was instrumental in getting me excited about doing research in the area of computer vision.

I would like to thank several close collaborators and co-authors, which include Professors Marc Pollefeys, Leonard Mcmillan, Philippos Mordohai and Jan-Michael Frahm, UNC students Li Guan and Seon Joo Kim. For other projects I completed while in graduate school [120, 126], my collaborators were Yakup Genc at Siemens Corporate Research, Drew Steedly and Richard Szeliski at Microsoft Research and Maneesh Agrawala.

I would like to thank several researchers in the computer vision community who have generously shared code and data sets with me and helped me immensely in carrying out this research including Yuri Boykov and Vladimir Kolmogorov, for their graph cut energy minimization software, Jean Sebastien Franco and Edmund Boyer for the Exact Polyhedral Visual Hull (EPVH) software, Carlos Hernandez, Francis Schmitt, Yasutaka Furukawa, Jean Ponce,

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
3D modeling from images and video

1.1 Introduction

Recovering 3D models of the real world from images and video is an important research area in computer vision with applications in computer graphics, virtual reality and robotics. Manually modeling photo-realistic scenes for such applications is tedious and requires a lot of effort. The goal in computer vision is to generate such models automatically by processing visual imagery from the real world captured by cameras in the form of images and video or by other specialized sensors and recovering the 3D shape and structure of the scene. In recent years the explosion of digital photography, rapid improvements in cameras and growth in visual surveillance systems coupled with the advances in computer graphics and increasing demand for 3D content in various visual applications has created a growing demand for practical vision-based 3D modeling systems that can reliably capture models from the real world in various different scenarios.

The first area we explore in this dissertation is reconstructing in 3D a dynamic scene that was captured on multiple video streams. The goal is to digitize in 3D a time-varying event involving either rigid or non rigid moving objects such as human beings, for e.g. a dance performance that was recorded by video cameras from multiple viewpoints. Researchers in computer vision have been interested in solving this problem for over a decade now. In 1997, Kanade et. al. [69] coined the term *virtualized reality* and demonstrated the technology by reconstructing real life scenes involving humans using a large cluster of cameras in an indoor environment (various camera setups that were used is shown in Figure 1.1(a–c)). Since then

(a)   (b)

(c)

(d)

(e)
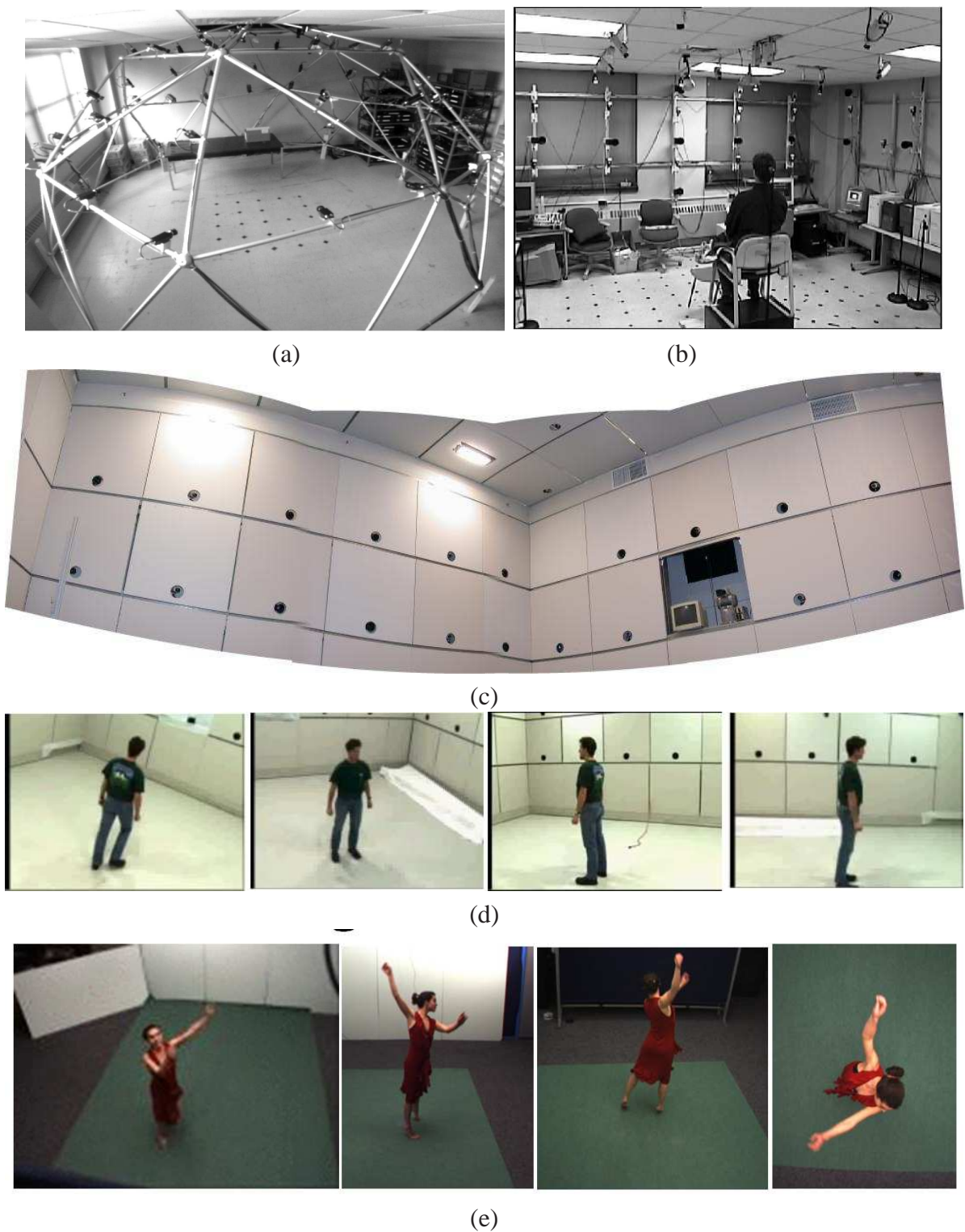
Figure 1.1: Various indoor camera networks at CMU Robotics Labs [69, 70] - (a) 3D dome consisting of 51 cameras, (b,c) 3D room: cameras setup to capture an event from multiple views. (d) 4 out of a 8 view sequence captured at the CMU 3D room (c). (e) Another multiple view sequence of a dancer (courtesy PERCEPTION Group, INRIA Rhone-Alpes).

.

various systems have been developed which can digitize human subjects performing various actions [20, 22, 24, 41, 42, 96, 114] – these systems use an indoor room-sized camera setup that typically consists of 8–15 synchronized cameras recording at 15–30fps. We will refer to such an arbitrary configuration of cameras as a *camera network* . With the popularity of digital cameras and growth of video surveillance systems, nowadays both indoor and outdoor *camera networks* are becoming commonplace. However modeling dynamic scenes outdoors is a much more challenging problem and many assumptions that often hold true in a controlled indoor setup now need to be relaxed. One of the fundamental requirements for reconstructing 3D events observed by camera networks irrespective of the reconstruction method itself is accurate geometric calibration and synchronization of all the cameras in the network.

Currently in all multi-camera systems [20, 22, 24, 41, 42, 96, 114] calibration and synchronization must be done during an offline calibration phase before the actual video capture is done. Someone must be physically present in the scene with a specialized calibration object. This process makes the process of camera deployment and acquisition fairly tedious. Multiple calibration sessions are often required as there is no easy way to maintain the calibration over a longer duration. In this thesis we explore possibilities of making this part of the whole process more flexible by developing methods that recover all the necessary information from the recorded video streams – thus eliminating the need for an explicit offline calibration phase before the video capture. Figure 1.1 shows some examples of camera networks recording a dynamic scene involving a human subject from multiple viewpoints. 3D digitizing such time varying events would enable the user viewing the event to be completely immersed in a virtual world allowing him to observe the event from any arbitrary viewpoint – this is called *viewpoint-free 3D video* and has promising applications in 3D tele-immersion, and in digitizing rare cultural performances, important sports events and generating content for 3D video based realistic training and demonstrations for surgery, medicine or other technical fields.

We also look into the problem of capturing a high quality 3D model of a static object such

Figure 1.2: A 3D model of a statue reconstructed from 36 images. The object was placed on a turn-table and imaged under fairly controlled conditions.

as a statue from ordinary images taken from multiple viewpoints – an example is shown in Figure 1.2. This is also a problem that has been extensively studied in the computer vision community and over the last decade, remarkable progress has been made in terms of quality and accuracy of the recovered 3D models and robustness of the reconstruction approach. More sophisticated and specialized technologies such as laser range scanning have also been succesfully used to scan objects or 3D scenes [87]. However, those require expensive hardware, active lighting in the scene, and are limited to low capture rates and finite depth ranges. In some cases, both structured and unstructured active lighting have been used with ordinary cameras for 3D shape acquisition [160], but performing accurate and robust 3D reconstruction from ordinary images is still a major research goal in the computer vision community because it will make the technology more practical and easily applicable.

In this dissertation, we improve upon the state of the art techniques for the two scenarios described above, namely - (a) modeling dynamic events in 3D from multiple archived video streams and (b) acquisition of high quality 3d models of static objects in the real world.

Our primary goal is to increase the overall flexibility of camera network calibration and synchronization for 3D event reconstruction. We have developed a method for calibrating and

4

synchronizing a network of cameras observing an event from multiple viewpoints. Through this method, we aim to ease the deployment of cameras for reconstructing dynamic scenes as all the necessary information in our method is recovered by analyzing only the silhouettes of moving objects in video. The silhouettes can then be used to recover the 3D shape using a well know *shape-from-silhouette* technique.

For reconstructing accurate 3D models with detailed geometry from images, multi-view stereo approaches have shown promising results in recent years. Although quite sophisticated stereo algorithms have been proposed, the underlying *dense correspondence problem* is highly ill-posed. Therefore all robust method must employ some type of regularization to enforce local surface smoothness. On the other hand, silhouettes of objects observed in multiple views provides a strong constraint on its shape and forms the basis of *shape-from-silhouette* methods. However, silhouette based methods cannot recover any concavities on the surface of the object. Combining stereo and silhouette cues can improve the accuracy of the 3D modeling process. The methods to integrate the two cues that have been proposed [45, 57, 66] do not combine the complementary information provided by silhouettes and stereo in the best possible way. In this thesis, we will investigate this direction further. We will present a new graph-cut based multi-view stereo formulation in which silhouette constraints can be exactly enforced. While the benefit of the graph-cut approach for the 3D reconstruction problem has been reported [13, 85, 148], we show that it is possible to also incorporate the powerful silhouette constraint within the graph-cut framework. The overall goal of this approach is to guarantee robust and accurate fusion of multi-view stereo with silhouette cues.

Finally, we also address the high memory and computational overhead of the volumetric graph-cut based multi-view stereo approach. We propose a formulation that involves an adaptive graph construction. This makes it possible to achieve the fine voxel resolution that is required for reconstructing surfaces with high geometric detail without overshooting the memory bottleneck during the graph-cut optimization. The adaptive construction also avoids

evaluating the photo-consistency measure densely in regions which are unlikely to contain any surface elements. This reduces the overall computation time by an order of magnitude.

1.2   Background

This section provides some background into the state of the art methods used for camera calibration and multi-view 3D reconstruction, and some of the limitations and weaknesses that we plan to address in this dissertation.

1.2.1   Camera Calibration
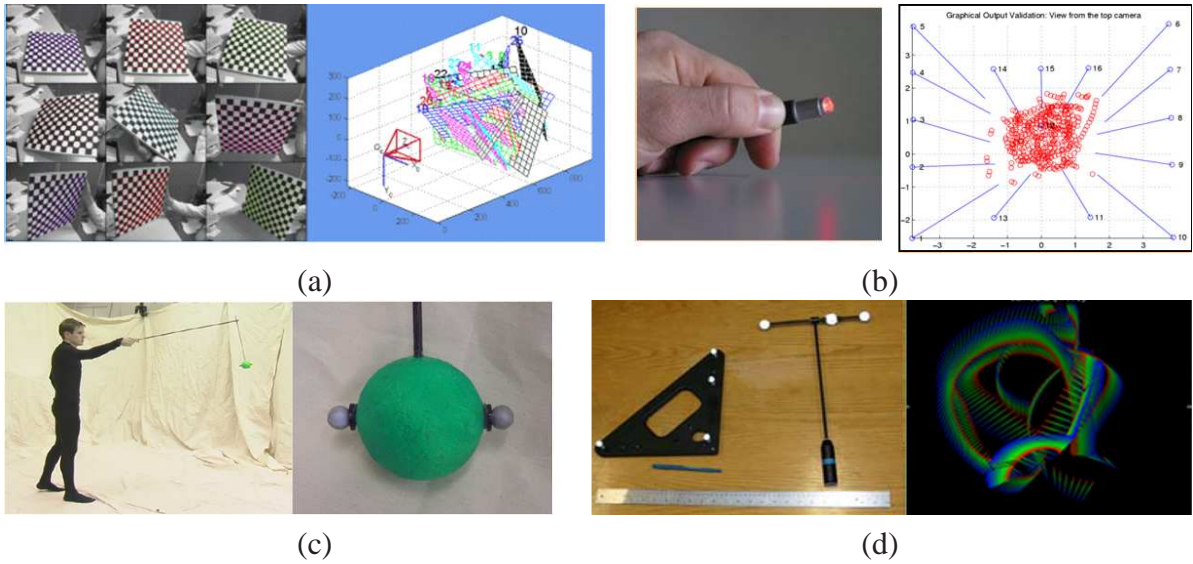
Figure 1.3: (a) Camera Calibration using a planar checkerboard [162]. (b) Calibration using a single LED [137]. (c) Synchronization using motion capture markers [114]. (d) VICON motion capture system and its own calibration wand.

In traditional camera calibration, images of a calibration target (an object whose geometry is known) are first acquired. Correspondences between 3D points on the target and their imaged pixels are then recovered (the target is built in a way to make this step easy). After this, the *camera-resectioning* problem is solved. This involves estimating the intrinsic and extrinsic parameters of the camera (see Appendix A-1.1 for the specific camera model used) by

minimizing the reprojection error of the 3D points on the calibration object. The Tsai camera calibration technique was popular in the past, but required a nonplanar calibration object with known 3D coordinates [143]. The resulting calibration object often had two or three orthogonal planes. This was difficult to construct and made the overall process of camera deployment and calibration quite cumbersome. Zhang et. al.[162] proposed a more flexible planar calibration grid based method in which either the planar grid or the camera can be freely moved. The calibration object is easily created by pasting a checkerboard pattern on a planar board that can be waved around in the scene. An implementation of this calibration method is provided by [10], and has become very popular among computer vision researchers. Figure 1.3(a) shows the result of this calibration procedure in the camera coordinate system. Another similar plane-based camera calibration technique was also proposed around this time [135].

While this method produces fairly accurate calibration in realistic scenarios, obtaining the calibration data for large multi-camera systems can still be quite tedious. Often, with cameras placed all around a scene, the checkerboard can only be seen by a small group of cameras at one time. Hence, only a subset of cameras in the network can be calibrated in one session. By ensuring that these subsets overlap, it is possible to merge the results from multiple calibration sessions and obtain the calibration of the full camera network. However, this requires extra work and makes the overall procedure quite error prone. There is a new method for multi-camera calibration [137] – one that uses a single-point calibration object in the form of a bright LED that is waved around the scene. The advantage it provides is that the LED can be simultaneously observed in all images irrespective of their configuration. In a fairly dark room, detecting it in the images and establishing correspondence are easy and reliable. By moving the LED around, one can calibrate a larger volume than would be possible with a checkerboard, and the arbitrary path taken by the LED during the calibration session can be thought of as a flexible virtual calibration object (see Figure 1.3(b)).

Motion capture systems such as VICON [146] provide their own calibration devices con-

taining retro-reflective sensors (see Figure 1.3(d) or other special markers to establish 2D–3D correspondence. Although the basic idea is similar to LED–based calibration, calibrating large spaces or those in broad daylight is still considered to be a challenge as the LED or the markers become harder to detect when the cameras observe a larger or a brighter scene. Such motion capture sensors are often also used for multi-camera synchronization (see Figure 1.3(c)). Often in controlled scenes, a hardware trigger is used to synchronize all the video cameras together to ensure higher accuracy. A simple alternative is to use a clap sound to manually synchronize the videos, but this can be error prone for videos containing fast-moving subjects or in outdoor, noisy environments.

Although all these traditional methods can produce accurate results, they require physical access to the observed space and involves an offline precalibration stage that precludes reconfiguration of cameras during operation (at least, without an additional calibration session). This is often impractical and costly for surveillance applications and can be impossible for remote camera networks or sensors deployed in hazardous environments.

On the other hand, significant progress has been made in the last decade, in automatic feature detection and feature matching across images. Also, robust structure from motion methods, that allow the recovery of 3D structure from uncalibrated image sequences have been developed. Such structure from motion algorithms were first developed in the context of video [106], but were also extended to handle large unstructured image collections [19, 128]. Although in theory, such techniques could be used for multi-camera calibration, they require more overlap between camera pairs than is often available in camera networks. This is why automatic point correspondence based methods often do not work for camera network calibration and manual calibration is required.

1.2.2 Multi-view 3D Reconstruction

The techniques for 3D reconstruction from images (often called *3D photography*) involves using cameras (and optionally illumination) to acquire the shape and appearance of real objects. Flexible techniques for acquiring the 3D shape and appearance of a variety of real objects under different imaging conditions have been studied. All these methods can be divided into two categories – *active* and *passive* methods. While *active* methods employ some form of artifical lighting in the scene, *passive* methods recover all the information only from the images without making any prior assumptions on illumination information.

Active vision methods such as laser range scanning [87], active stereo with structured or unstructured projected light [160], active depth from defocus [49] typically require expensive hardware or need to deploy specialized equipment. Although they have been shown to produce good results in controlled scenes, there is less flexibility in using them in the real world. This is why there is still a strong interest in the computer vision community to develop robust algorithms for accurate 3D reconstruction using *passive* methods.

In *passive* methods, a wide variety of visual cues can be exploited – these are used to classify the methods into the following categories (not an exhaustive list) – stereo, shape from shading (photometric stereo), shape from silhouettes, shape from focus (and defocus), shape from texture, etc. It has been shown that human vision and perceptual systems rely on these different cues to perceive 3D shape, and this has motivated these various *shape-from-X* approaches. Amongst these, the stereo and silhouette cues dominate and these can be applied in a wide variety of scenes involving various types of objects. In this thesis, we will limit our discussion to 3D reconstruction methods that use only the stereo and silhouette cues.

A single image of an unknown scene does not provide enough information to reconstruct a 3D scene as the associated depth information is lost during image formation. Please refer to Appendix A-1.1 for the mathematics of image formation and the commonly used camera models. Although researchers have been studying the problem of inferring the 3D scene struc-

ture from a single image [29, 62, 115], these methods typically require additional information, require the scene to be structured in a certain way, and produce 3D estimates which are coarse shape and fairly inaccurate.

By extracting dense pixel to pixel correspondences between multiple calibrated images of the same scene, it becomes possible to recover the 3D structure using geometric techniques such as multi-view triangulation. This is the basic idea behind many binocular stereo (two-view) or multi-baseline stereo algorithms. However the dense correspondence problem (or the 3D reconstruction problem in general) is a highly ill-posed inverse problem; the presence of image noise, specularities or other complex illumination effects, presence of texture-less regions on the surface and finally occlusions makes the general 3D reconstruction problem quite challenging. The estimation problem has multiple solution and must employ some form of *regularization* that biases the solution towards smoother shapes. This often boils down to solving a computationally expensive optimization problem involving many unknown variables.

Various mathematical formulations for the optimization problem exist in the literature [38, 13, 14] and are based on different criteria. All the methods can be broadly classified into *local* and *global* methods. While global methods usually ensure higher accuracy and utilize better forms of regularization and shape priors, their computational complexity is usually much higher than the local methods.

Existing multi-view stereo methods can also be classified on the basis of scene representation, photo-consistency measure, shape priors used and the various ways of dealing with the visibility problem (see [117] for a detailed survey of existing methods). We will review some of the relevant stereo and *shape-from-silhouette* methods in Chapters 4 and 5 respectively. Many successful global methods often adopt a variational approach to the problem that aims to recover a weighted minimal surface inside a given bounding volume. One popular strategy is to cast the reconstruction problem into the level-set framework, and represent the surface as

the zero level set of a time-evolving implicit function. The energy functional is solved by using a partial differential equation which drives the evolution of the level set function. The final surface recovered is often quite smooth, but this strongly depends on the initialization because the underlying method can only compute a local minima of the energy functional. Another class of discrete combinatorial *graph-cut* methods have recently been used to minimize similar energy functionals with better guarantees on finding global minimas under some special cases. It was shown that graph-cuts could efficiently compute the weighted minimal surface by solving a mincut problem on a suitable flow graph (see Appendix B-3 for the preliminaries) for which polynomial time algorithms are well known. However, the graph-cut approach for 3D reconstruction has some other weaknesses. These will be analyzed in Chapter 5, and we will explore new methods in this dissertation to address them.

1.3   Our Contributions

The primary contributions discussed in this disseration are two-fold. These are summarized below. See Appendix B-3.1 for the list of relevant publications.

Camera Network Calibration and Synchronization

As surveillance camera networks or video cameras become common, it will be possible to record live video of a dynamic scene involving moving objects, often human subjects from multiple viewpoints. First, we show that it is possible to automatically recover the calibration and synchronization of such a network of cameras using only the input videos. Next, we show how to obtain the 3D reconstruction of the dynamic event as well. The proposed technique can recover all the necessary information by analyzing the silhouettes of moving objects in multiple video streams. This allows us to model dynamic scenes or events in 3D from archived video, and it precludes the need for an offline calibration phase or physical access into the scene for collecting explicit calibration data. This increases the overall ease for

11

camera deployment and acquisition for digitizing 3D events. We demonstrate the benefit of our approach by remotely calibrating several visual hull datasets that were acquired by other researchers in their own labs. Differents parts of this research have been published in the following papers – [125, 122, 123, 121].

The specific contributions are:

- We propose a novel algorithm for recovering the epipolar geometry of a camera pair and the synchronization that recovers the necessary information by analyzing the silhouettes of moving objects in video.

- We use RANSAC [9] in a new way – to explore a low dimensional parameter space within the epipolar geometry estimation problem, rather than using it only for robust estimation which is common in the computer vision literature.

- We show how to incrementally construct a fully calibrated camera network starting from pairwise epipolar geometry estimates. The technique can be applied to other camera network calibration scenarios.

- Finally, the proposed algorithms have been combined within an end-to-end system for calibrating and synchronizing a network of cameras from archived video sequences of a dynamic scene or event. The recovered calibration and synchronization makes it possible to subsequently reconstruct the dynamic scene in 3D.

Improvements in Multi-view Stereo

In the second part of the thesis, we improve upon different aspects of existing multi-view stereo techniques. First, we propose a novel formulation for multi-view stereo that allows for robust and accurate fusion of the silhouette and stereo cues. We show that it is possible to enforce exact silhouette constraints within the graph-cut optimization step in the volumetric multi-view stereo algorithm. This guarantees that the reconstructed surface will be exactly

consistent with the original silhouettes. A preliminary version of this approach was published in [124]. Finally, we have also developed an adaptive graph construction approach for graph-cut based multi-view stereo to address the inherent high memory and computational overhead of the basic algorithm. Using this method, we have been able to efficiently reconstruct accurate and detailed 3D models of objects from high-resolution images for a number of different datasets. This method was published in [108].

The specific contributions are:

- We propose a graph-cut formulation for volumetric multi-view stereo that strictly enforces silhouette constraints. This implies that in addition to being photo-consistent, the final reconstructed surface computed by the graph-cut step will be fully consistent with the original silhouettes. Contrary to previous work on silhouette and stereo fusion, the silhouette consistency in our method is guaranteed by construction (through hard constraints) in the graph-cut problem – the silhouette consistency terms are not part of the energy minimization problem which aims to find a surface with maximal photo-consistency.

- We also propose an alternate formulation that addresses the high memory and computational requirements of the basic volumetric graph-cut stereo algorithm. Our proposed method recovers surfaces with geometric detail by performing a graph-cut on the dual of an adaptive tetrahedral mesh (a CW-complex) created by photo-consistency driven recursive mesh subdivision. The approach does not need any initialization and is not restricted to a specific surface topology, which is a limitation with existing methods that use a base surface for initialization.

1.4 Thesis Outline

This thesis is organized into two parts. Part I deals with the problem of camera network calibration, synchronization and reconstruction of dynamic scenes from multiple video streams. Part II deals with acquisition of 3D models from multiple calibrated images of static objects. In Chapter 2, we discuss a novel method for recovering the epipolar geometry from silhouettes in video. The method is extended to simulateneously recover the epipolar geometry and the temporal offset. In Chapter 3, we describe how the full camera network calibration can be recovered from pairwise epipolar geometries. In Chapter 4, we review exact visual hulls, *shape-from-silhouette* algorithms and present dynamic scene reconstruction results on various datasets. In Chapter 5, we survey the start-of-the-art multi-view stereo methods and those that combine stereo and silhouette cues. Chapter 6 contains our novel graph-cut based multi-view stereo formulation that makes it possible to exactly enforce silhouette constraints. In Chapter 7, we present the efficient graph-cut method that involves an adaptive graph construction. Finally, we present our conclusions in Chapter 8, and point out directions for future work. Miscellaneous topics and relevant background material is reviewed in the Appendices that follow.

Part I

Camera Network Calibration and Synchronization for Modeling Dynamic Scenes

CHAPTER 2
Epipolar Geometry from Silhouettes

2.1 Introduction

The epipolar geometry captures the projective geometry between two views of the same scene, and it depends only on the camera intrinsics and the relative pose of the two cameras. Techniques for computing the epipolar geometry from multiple 2D point correspondences in the two images have been well studied. Estimation of epipolar geometry and other multi-view geometric relations from 2D point correspondences forms the basis for many structure from motion pipelines. However, when dealing with camera networks observing an event, such methods cannot be used to reliably recover the epipolar geometry in several scenarios. This is because these methods depend on the presence of sufficient interest point matches (2D point correspondences) in the two views. Live video of events recorded by a camera network often lacks such point correspondences due to extremely wide baselines of camera pairs, and a low overlap of the static background observed in the two views (an example is shown in Figure 2.1). However, since these cameras are setup to observe dynamic events, the silhouettes of moving foreground objects is often a prominent feature.

This chapter studies how silhouettes of such moving objects observed in two video streams can be used to recover the epipolar geometry of the corresponding camera pair. In fact, the method that we develop can be used to simultaneously recover both the epipolar geometry and the synchronization of the two cameras, as we show later.
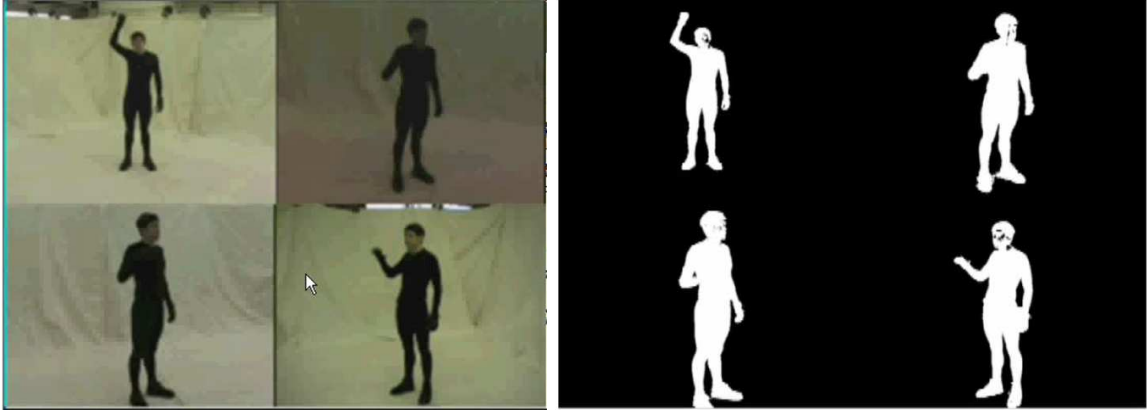
Figure 2.1: A moving person was observed and recorded from four different viewpoints. Here four corresponding frames are shown along with the silhouettes that were automatically extracted. Note that the silhouettes are noisy and the sequences are not synchronized.

2.2   Background

The recovery of camera pose from silhouettes was studied by [68, 100, 147, 152], and recently there has been some renewed interest in the problem [11, 47, 58]. However, most of these techniques can be applied only in specific settings and have requirements that render them impractical for general camera networks observing an unknown dynamic scene. These include that the observed object be static [68, 47], the use of a specific camera configuration (at least partially circular) [58, 152], the use of an orthographic projection model [47, 147], and a good initialization [11, 156].

In our method, we take advantage of the fact that a camera network observing a dynamic object records many different silhouettes, yielding a large number of epipolar constraints that need to be satisfied by every camera pair. At the core of our approach is a robust RANSAC-based algorithm [9], that computes the epipolar geometry by analyzing the silhouettes of a moving object in a video. In every RANSAC iteration, the epipole positions in the two images are randomly guessed and a hypothesis for the epipolar geometry is formed and efficiently verified using all the silhouettes available from the video. Random sampling is used for exploring the 4D space of possible epipole positions as well as for dealing with outliers in

the silhouette data. This algorithm is based on the constraint arising from the correspondence of frontier points and epipolar tangents for silhouettes in two views. This constraint was also used in [47, 100, 111, 152] but for specific camera motion, or camera models, or in a situation where a good initialization was available.

Frontier points were used by [111] to refine an existing estimate of the epipolar geometry. They were used by [100, 152] for the specific case of circular motion, where turntable sequences of static objects were calibrated. Furukawa et. al. [47] directly searched for frontier points on silhouettes in each viewpoint. To recover the epipolar geometry, they require an orthographic camera model. Their method also requires high quality silhouettes and works better with small camera baselines. Many common shapes can have very few frontier point correspondences in two views and the epipolar geometry estimate could be unstable or impossible to compute using this method.

Hernandez et. al. [58] generalized the idea of epipolar tangencies to the concept of *silhouette coherence*, which numerically measures how well a solid 3D shape corresponds to a given set of its silhouettes in multiple views. They performed camera calibration from silhouettes by solving an optimization problem where *silhouette coherence* is maximized. However, they only dealt with circular turntable sequences, which have fewer unknown parameters, so their optimization technique does not generalize to an arbitrary camera network. Boyer et. al. [11] also proposed a criterion back-projected silhouette cones must satisfy such that the true object is enclosed within all of the cones. They use it for refining the calibration parameters of a camera network, but this requires good initial estimates of the camera parameters.

In this chapter, we first study the recovery of epipolar geometry in the case where the two cameras are synchronized. We then show how to extend the algorithm to simultaneously recover the epipolar geometry as well as the temporal offset between a pair of unsynchronized cameras recording at the same frame rate. As our calibration approach relies on silhouettes, it requires a robust background segmentation approach (see Appendix B-1 for the approach used

for silhouette extraction). Moreover, our RANSAC [9]-based algorithm is robust to errors in silhouette extraction. For stable estimates of the epipolar geometry, it is important to sample the 3D space densely, which requires sufficient motion of the foreground object covering a large part of the 3D volume observed by the cameras. This is not a problem, as our method can efficiently handle long video sequences with thousands of silhouettes.

2.2.1 Geometry of Silhouette Formation

When a scene containing an object (or objects) is observed by a pinhole camera (mathematical model described in Appendix A-1.1), it produces a silhouette $\mathcal{S}$ on the image plane. This is shown in Figure 2.2(a). The boundary curve of the silhouette is called the *apparent contour*. It divides the image plane into two regions - the interior of the silhouette and its exterior. The generalized cone in $\mathbb{R}^3$ formed by the set of all rays passing through the interior of the silhouette and the camera center C is called the *viewing cone* or *silhouette cone*. The viewing cone is tangent to the object's surface along a continuous curve called the *contour generator* or *rim*. Note that for a non-convex solid shape, the rim curve can occlude itself giving rise to a *T-junction* on the silhouette. As shown in Figure 2.2(b), in general points that belong to the apparent contour in two different views and lie on matching epipolar lines such as m and m' are not corresponding points.

The only true point correspondences on the apparent contour in two views occur at special locations called *frontier points*. In Figure 2.2(c), one pair of frontier points is denoted by x_1 and x_2, respectively. Note that the viewing rays that correspond to a matching pair of frontier points such as x_1 and x_2 must intersect at a true surface point in the tangent plane of the surface. The contour generators or rims must also intersect at such a surface point. This point, along with the camera baseline defines an epipolar plane that must be tangent to the surface, giving rise to corresponding epipolar lines such as l_1 and l_2, which are tangent to the silhouettes at the frontier points. Frontier point correspondence does not extend to more than
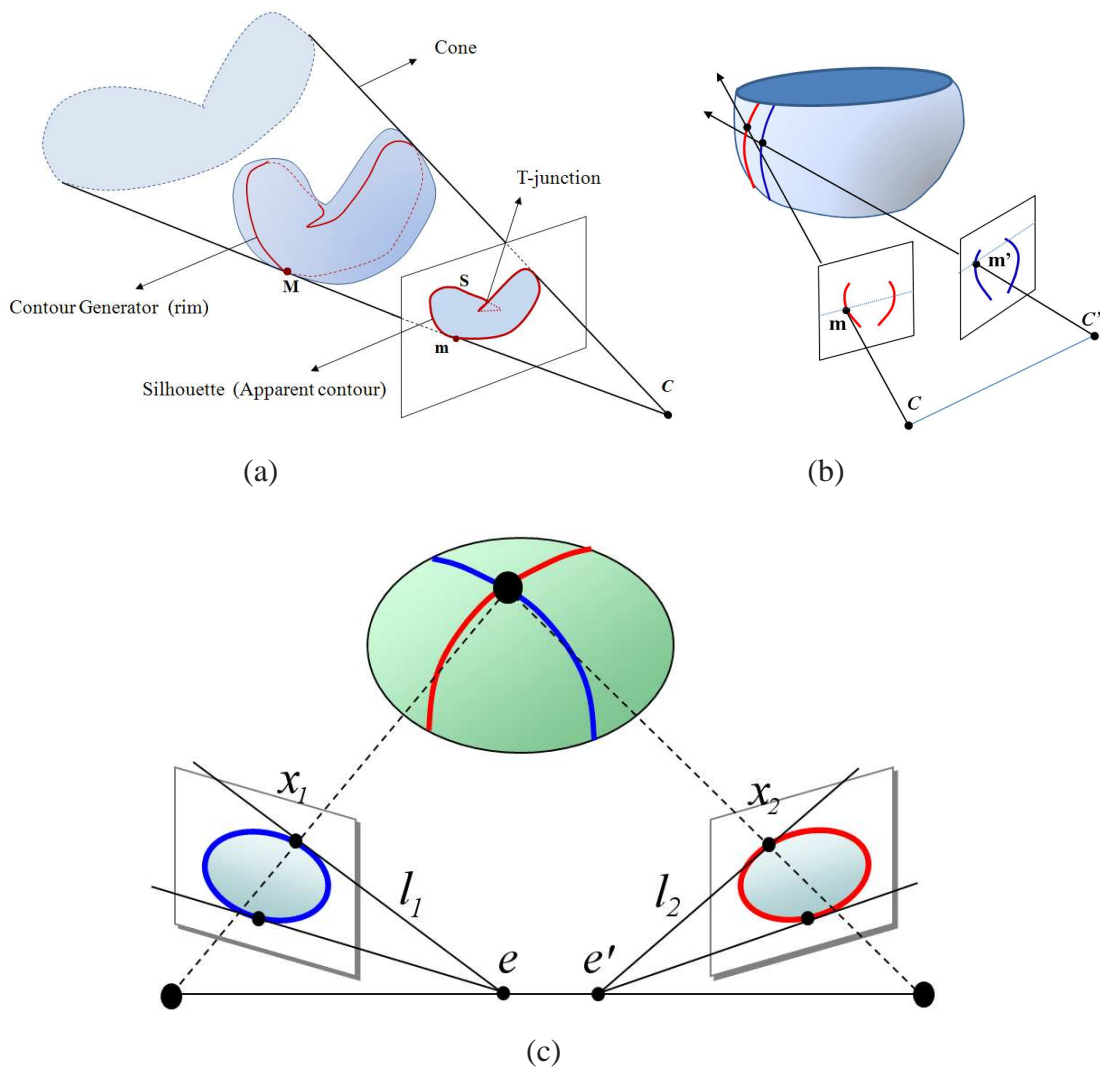
Figure 2.2: (a) Apparent contour and contour generator (rim). (b) In general, any two contour points on matching epipolar lines do not correspond. (c) Frontier points and epipolar tangents for two views.

two views in general. In a three-view case, the frontier points in the first two views do not correspond to those in the last two views.

A convex shape, fully visible in two views, has exactly two pairs of frontier points. For a nonconvex shape such as a human figure, there can be several potential frontier points, but many of them could be occluded. If the location of the epipole in the image plane is known, matching frontier points can be detected by drawing tangents to the silhouettes. However,

20
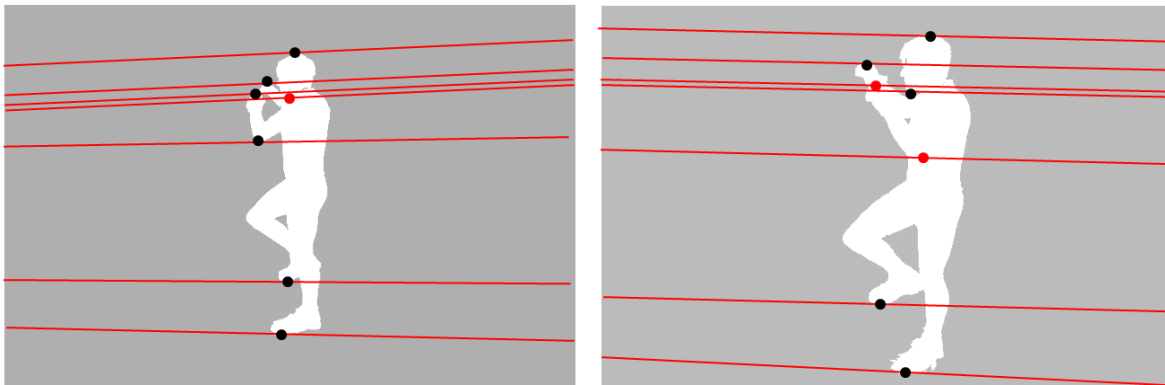
Figure 2.3: If the location of the epipoles in the two images were known, then some corresponding frontier points could be extracted. Note that some of the frontier points corresponding to the elbows, chin of the person are occluded. But the extremal frontier points corresponding to the head or toe will never be self-occluded.

when the epipole location is unknown, it is very difficult to reliably detect the frontier points. In [152] Wong et. al searched for outer-most epipolar tangents for circular motion. In their case, the existence of fixed entities in the images, such as the horizon and the image of the rotation axis, simplified the search for epipoles. We too use only the *extremal* frontier points and outermost epipolar tangents because, for fully visible silhouettes, these are never occluded. An important point to note is that the extremal frontier points must lie on the convex hull of the silhouette in the two views.

2.3 Epipolar Geometry from Dynamic Silhouettes

Given nontrivial silhouettes of a human (see Figure 2.3), if we can detect matching frontier points, we can use the 7-point algorithm to estimate the epipolar geometry by computing the fundamental matrix. However, it is difficult to directly find matching frontier points without knowing the epipoles. The method proposed by Furukawa et. al. [47] assumes an orthographic camera model and small baselines between camera pairs. Their approach requires accurate silhouettes and wouldn't work unless there are at least four unoccluded frontier point matches in general position.

Given an approximate solution, it is possible to refine the epipolar geometry using an optimization approach [5, 111]. However, since we use only silhouettes, an initial solution is not available to us. Therefore, we need to explore the full space of possible solutions. While a fundamental matrix has 7 degrees of freedom (dofs), our method only randomly samples in a 4D space because once the position of the epipoles are fixed, potential frontier point matches can be determined, and from them the remaining degrees of freedom of the epipolar geometry can be computed via an epipolar line homography. Here we propose a RANSAC-based approach that in a single step, allows us to efficiently explore this 4D space as well as robustly deal with inaccurate silhouettes in the sequence.

2.3.1 Silhouette Representation
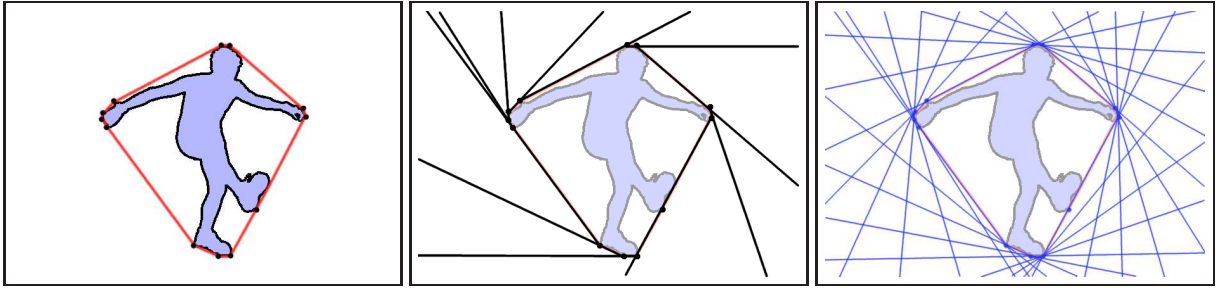
Figure 2.4: (a) The Convex Hull of the silhouette in a video frame. (b) The Tangent Table representation (c) The space of all tangents to the convex hull parameterized by θ.

For every frame in each sequence, a binary foreground mask of the object is computed using background segmentation techniques (see Appendix B-1 for details). Instead of explicitly storing the complete silhouette $\mathcal{S}$, we compute and store only the convex hull $\mathcal{H}_{\mathcal{S}}$ and its dual representation, as shown in Figure 2.4 for every video frame. This is a very compact representation as it allows us to efficiently compute outer tangents to silhouettes in long sequences containing potentially thousands of different silhouettes. The convex hull $\mathcal{H}_{\mathcal{S}}$ is represented by an ordered list of k 2D points in the image ($v_1 \ldots v_k$ in counter-clockwise order (CCW)). The 2D lines tangent to $\mathcal{H}_{\mathcal{S}}$ are parameterized by the angle $\theta = 0 \ldots 2\pi$ (in radians) that the

line subtends with respect to the horizontal direction in the image. For each vertex v_k, an angular interval $[\theta_k^1, \theta_k^2]$ is computed. This set represents all lines that are tangent to $\mathcal{H}_\mathcal{S}$ at the vertex v_k. These tangent lines are directed, that is they are consistently oriented with respect to the convex hull. Thus, for a direction θ, there is always a unique directed tangent $\mathbf{l}_\theta$, which keeps tangency computations in our algorithm quite simple.

Computing the Convex Hull

The convex hull $\mathcal{H}_\mathcal{S}$ for a silhouette $\mathcal{S}$, can be computed in linear time using Melkman's on-line convex hull algorithm [99]. This algorithm requires a simple path traversing all the n input points, which in the general case takes $O(n \log n)$ time to compute. In our case, a top-down scan of the bounding box of $\mathcal{S}$ in the image generates two simple paths in linear time – a left boundary $\mathcal{L}_\mathcal{S}$ and a right boundary $\mathcal{R}_S$. Then in a single pass, Melkman's algorithm is used to compute the left hull in CCW order from $\mathcal{L}_\mathcal{S}$ and the right hull in CW order from $\mathcal{R}_\mathcal{S}$. A union of the left and right hulls produces the convex hull in a consistent CCW order.

When the silhouettes are clipped at the frame boundaries, instead of ignoring the frame completely, we try to use the partial silhouettes in our approach. This is because often partially clipped silhouettes contain 1 or 2 valid extremal frontier points in the parts of the silhouette which are unclipped. We store the convex hull as a single ordered list instead of multiple connected segments. We introduce new vertices where the silhouettes are clipped and use a special bit to indicate the fact that some segments lie outside the image boundary. Also in the scenario where multiple foreground objects have been detected in the image, we compute the unique convex hull of the whole ensemble.

The computational time for the tangent table is linear in the size of the convex hull polygon. Computing the outer tangents to the silhouettes from any external point in the worst case takes $O(\log k)$ time. However, by exploiting high temporal coherence in video, on an average this can be done much faster. In all our experiments, the convex hulls of the silhou-

ettes typically had 20-40 vertices. A single video frame therefore contributes ≤ 500 bytes of storage and the information from thousands of silhouettes in video can easily fits into memory allowing us to efficiently handle long sequences. By handling only outer epipolar tangents, we keep tangency computations efficient and also do not need to worry about occlusions.

2.3.2 Main Idea

In Appendix A-1.2 we discuss the parameterization of $\mathbf{F}_{ij}$ in terms of $\mathbf{e}_{ij}$, $\mathbf{e}_{ji}$, the position of the epipoles in the images and $\mathbf{H}_{ij}$, the epipolar line homography. The basic idea in our approach is the following – to generate a hypothesis for the epipolar geometry, we randomly guess the position of $\mathbf{e}_{ij}$ and $\mathbf{e}_{ji}$ in the two views. This fixes four dofs of the unknown epipolar geometry and the remaining three dofs can be determined by estimating $\mathbf{H}_{ij}$ for the chosen epipole pair. To compute the homography, we need to obtain three pairs of corresponding epipolar lines (epipolar tangents in our case) in the two views. Every homography $\mathbf{H}_{ij}$ satisfying the system of equations $[\mathbf{l}_j^k]_\times \, \mathbf{H}_{ij} \, \mathbf{l}_i^k \ = \ 0$ where $k \ = \ 1 \ldots 3$ is a valid solution. Note that these equations are linear in $\mathbf{H}_{ij}$ and allows it to be estimated efficiently. In a RANSAC [9] like fashion, we then evaluate this hypothesis using all the silhouettes present in the sequences.

(a)   (b)
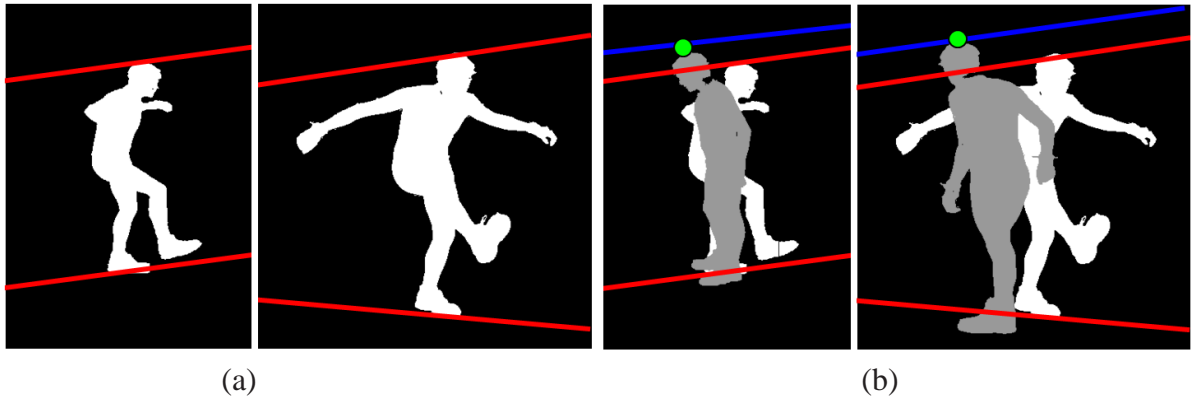
Figure 2.5: (a) Having randomly picked two corresponding frames, two random directions are sampled in each image. The intersection of the corresponding epipolar tangents generates the epipole hypothesis. (b) Outermost epipolar tangents to the new silhouette, computed in another pair of randomly selected corresponding frames. The three pairs of lines can be used to estimate the epipolar line homography.

2.3.3 Hypothesis Generation

At every RANSAC iteration, we randomly choose a pair of corresponding frames from the two sequences. In each of the two frames, we randomly sample two directions each and obtain outer tangents to the silhouettes with these directions. The first direction θ_1 is sampled from the uniform distribution $\mathbf{U}(0, 2\pi)$, while the second direction θ_2 is chosen as $\theta_2 = \theta_1 - x$, where x is drawn from the normal distribution $\mathbf{N}(\pi, \frac{\pi}{2})$. For each of these directions, the convex hull of the silhouette contains a unique directed tangent. The two tangent lines in the first view are denoted by l_i^1 and l_i^2 while those in the second view are denoted by l_j^1 and l_j^2 respectively (these are shown in red in Figure 2.5(a)) [1]. The intersections of the tangent pairs produces the hypothesized epipoles $\mathbf{e}_{ij}$ and $\mathbf{e}_{ji}$ in the two views.

An alternative approach for generating the epipoles involves sampling both epipole directions randomly on a sphere [84], which in the uncalibrated case is equivalent to random sampling on some ellipsoid and yields comparable results to our method. We next randomly select another pair of frames and compute outer tangents from the epipoles $\mathbf{e}_{ij}$ and $\mathbf{e}_{ji}$ to the silhouettes (actually their convex hulls) in both views. If there are two pairs of outer tangents, we randomly select one. This third pair of lines is denoted by l_i^3 and l_j^3, respectively. (these are shown in blue in Figure 2.5(b)).

Now, $\mathbf{H}_{ij}$, the epipolar line homography, is computed from the three corresponding lines [2] $\{l_i^k \leftrightarrow l_j^k\}$ where $k = 1 \ldots 3$. The quantities $(\mathbf{e}_{ij}, \mathbf{e}_{ji}, \mathbf{H}_{ij})$ form the model hypothesis for every iteration of our algorithm.

2.3.4 Model Verification

Each randomly generated model for the epipolar geometry is evaluated using all the data available. This is done by computing outer tangents from the hypothesized epipoles to the whole

---

[1] If silhouettes are clipped in this frame, the second pair of tangents could be chosen from another frame.
[2] There are two ways to pair $\{l_i^1, l_i^2\}$ with $\{l_j^1, l_j^2\}$ and we generate a hypothesis for each of the two cases.
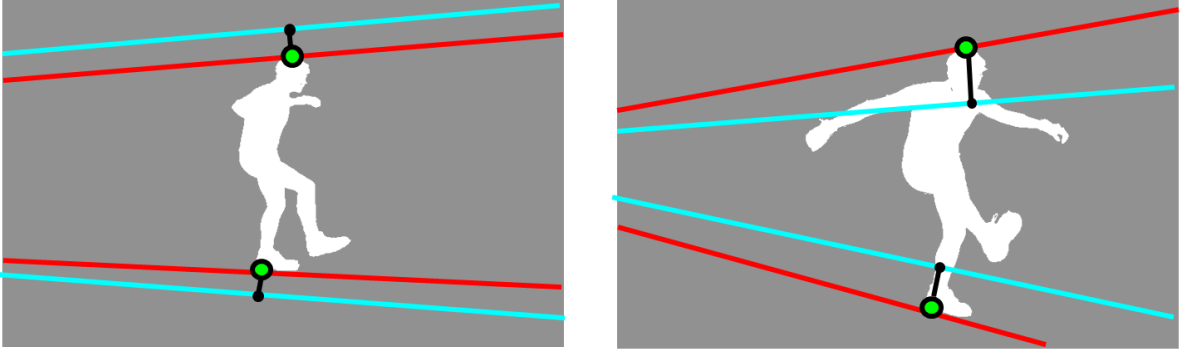
Figure 2.6: The model for epipolar geometry is used to compute the epipolar transfer error for all the silhouettes in video. When the model is completely inaccurate, an early rejection scheme [93] is used.

sequence of silhouettes in each of the two views. For unclipped silhouettes, we obtain two tangents per frame, whereas for clipped silhouettes, there may be one or even zero tangents. Every epipolar tangent in the first view is transferred through $\mathbf{H}_{ij}$ to the second view (see Figure 2.6) and the reprojected epipolar transfer error e is computed based on the shortest distance from the original point of tangency to the transferred line:

$$e = d(\mathbf{x}_i, \mathbf{l}_i^t) + d(\mathbf{x}_j, \mathbf{l}_j^t) \tag{2.1}$$

where $d(\mathbf{x}, \mathbf{l})$ represents the shortest distance from a 2D point $\mathbf{x}$ to a 2D line $\mathbf{l}$ and $\mathbf{x}_i$ and $\mathbf{x}_j$ represent the point of tangencies in the two images which when transferred to the other view, gives rise to epipolar lines $\mathbf{l}_j^t$ and $\mathbf{l}_i^t$ respectively.

Figure 2.7 shows the distribution of e for a typical pair of sequences. We use an outlier threshold denoted by $\tau_{\mathbf{o}}$ to classify a certain hypothesis as a good or bad. The method for automatically computing $\tau_{\mathbf{o}}$ is described in the next section and varies with image size and the amount of motion in the sequence. $\tau_{\mathbf{o}}$ typically to be somewhere in the range of 4–12 pixels (the value 12 pixels was chosen when the image size was 2000×1000 approximately). The $\mathbf{K}^{th}$–quantile denoted by $e_{\mathbf{K}}$ is computed (in all our experiments, K = 0.75, or 75%). If $e_{\mathbf{K}} \leq \tau_{\mathbf{o}}$, then the epipolar geometry model is considered a promising candidate and it is

recorded.

Note that the approach will only work as long as we can deal with slightly wrong guesses for the two epipoles and as long as we sample within the convergence region or attraction basin of the correct epipoles. The size of the attraction basin tends to vary but this can be estimated from how frequently promising candidates are found during the thousands of trials of the algorithm. This is analyzed later.

The RANSAC-based algorithm looks for $\mathbf{n_S}$ promising candidates after which these candidates are ranked based on the inlier count and the best ones amongst these are further refined using non-linear optimization. A stricter threshold $\tau_{\mathbf{in}}$ which is set to 1 pixel in all our experiments, is used to determine the subset of inliers. Frontier point pairs often remain stationary in video and give rise to duplicates, therefore these need to be removed. This is done using a binning approach while computing the error distribution.
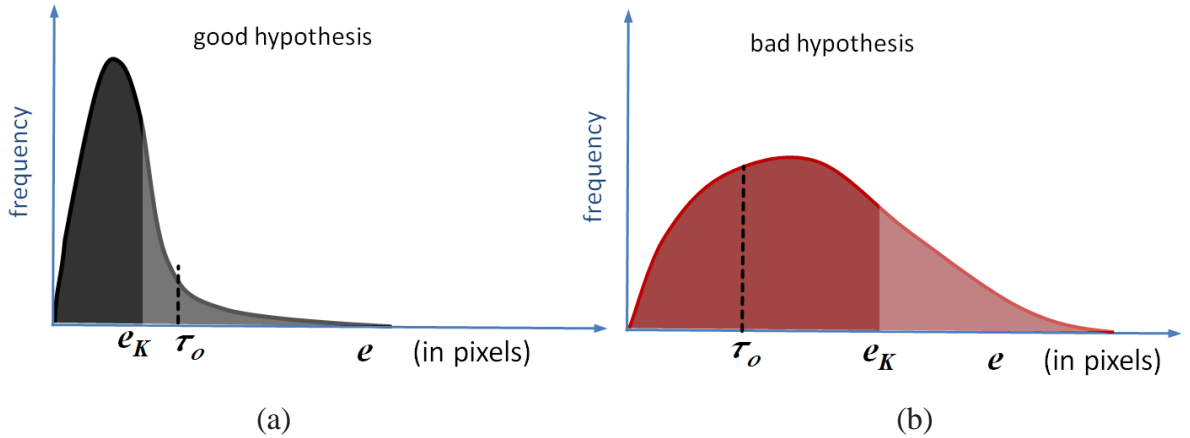
Figure 2.7: (a) Error distribution for a good hypothesis. Note that the $\mathbf{K}^{\text{th}}$–quantile e_K is much smaller than $\tau_{\mathbf{o}}$. (b) Error distribution for a bad hypothesis. Note that it is much more spread out and the $\mathbf{K}^{\text{th}}$–quantile e_K is greater than $\tau_{\mathbf{o}}$.

While evaluating a hypothesis, we count how many tangents exceed the outlier threshold $\tau_{\mathbf{o}}$ and reject a hypothesis when a partial outlier count indicates that the total expected outlier count is likely to be exceeded (i.e. with high probability). This allows us to abort early whenever the model hypothesis is completely inaccurate (a similar optimization was

suggested by [93]), avoiding the redundancy of computing outer tangents from epipoles to all the silhouettes for most random hypotheses which are completely wrong.

The best 25% of the promising candidates are then refined using iterative nonlinear minimization (Levenberg Marquardt method) followed by guided matching. The cost function minimized here, is the symmetric epipolar distance measure in both images. During guided matching, frontier points are recomputed from scratch using the current epipole estimates. In each iteration, the number of inliers steadily increases and the approach terminates when the inlier count stabilizes. Note that as long as the convergence region is large, an inaccurate candidate solution can lead to the true solution as the frontier points are recomputed as the solution gets more and more refined.

In practice, many of the promising candidates from the RANSAC step, when iteratively refined, produce the same solution for the epipolar geometry. Therefore, we stop when three promising candidates converge to the same solution. Otherwise we refine all the candidates. The refined solution with the highest inlier count is reported as the final one. Note that comparing the Frobenius norm of the difference of two normalized fundamental matrices is not a suitable measure, and Zhang et.al. [161] proposes a meaningful measure for deciding whether two fundamental matrices are similar enough. We use this measure to compare two estimates of the epipolar geometry, and decide if they are close enough.

Our algorithm uses a few parameters that tend to vary with datasets: the total number of RANSAC iterations N, the number of promising candidates n_S, and the outlier threshold τ_o. We automatically determine these parameters from the data, making the epipolar geometry estimation completely automatic and convenient to use.

2.3.5 Automatic Parameter Tuning

The number of RANSAC iterations N depends on the number of promising candidates to search for denoted by n_S. Thus N is chosen as $\min(n, N_I)$, where n is the number of it-

erations required to find $\mathbf{n_S}$ candidates and $\mathbf{N_I}$ is a large number set to 1 million in all our experiments. The choice of $\mathbf{n_S}$ is determined by the outlier threshold $\tau_{\mathbf{o}}$. A tighter (i.e. lower) outlier threshold can be used to select very promising candidates but such occurrences are rare. If the threshold is set higher, promising candidates are obtained more frequently but at the cost of finding a few ambiguous ones as well. When this happens, a larger set of promising candidates must be analyzed. Thus, $\mathbf{n_S}$ is set to $\max(2\tau_{\mathbf{o}}, 10)$. The critical parameter $\tau_{\mathbf{o}}$ tends

(a)   (b)
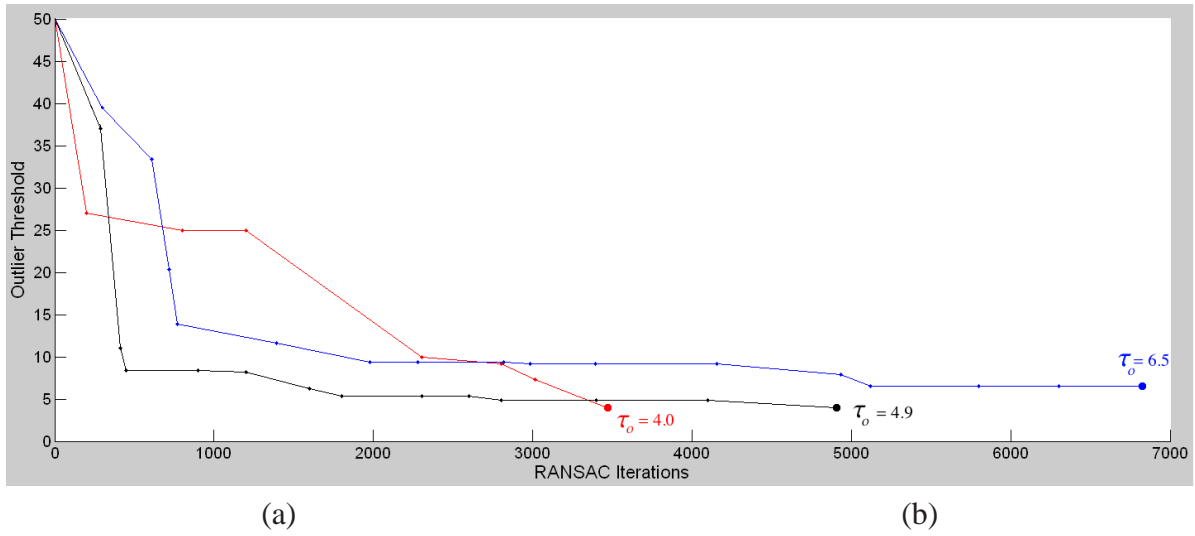
Figure 2.8: For three different test runs from various datasets, the value of $\tau_{\mathbf{o}}$ is plotted against the number of RANSAC iterations required.

to vary between datasets. It depends on the image size, amount of motion in the sequence. Therefore we automatically compute it from the data during some preliminary RANSAC iterations. During this preliminary stage, the hypothesis and verification iterations proceed exactly as described in the previous sections, but these are only used to compute $\tau_{\mathbf{o}}$, and the promising candidates found at this stage are not used later.

We start with a large value of $\tau_{\mathbf{o}}$ (= 50 pixels in our implementation) and iteratively lower it based on the error distribution as shown in Figure 2.7. We compare $\mathbf{e_K}$, the $\mathbf{K}^{\text{th}}$–quantile (K = 75) with the current value of $\tau_{\mathbf{o}}$.

If $\mathbf{e_K} < \tau_{\mathbf{o}}$, we simply reset $\tau_{\mathbf{o}}$ to the smaller value $\mathbf{e_K}$. If $\tau_{\mathbf{o}} \leq \mathbf{e_K} \leq (\tau_{\mathbf{o}} + 1)$, then we increment a counter $\mathbf{C}_{\tau_{\mathbf{o}}}$.

29

If $e_K > (\tau_o + 1)$, then the value of τ_o is not changed. We reset $\mathbf{C}_{\tau_o}$ to zero whenever the threshold is lowered.

If either τ_o falls below $\tau_{\mathbf{min}}$, or $\mathbf{C}_{\tau_o}$ becomes equal to $\tau_{\mathbf{min}}$, we accept the current estimate of τ_o as final. $\tau_{\mathbf{min}}$ is calculated as $0.005 \times (w + h)$, where $w \times h$ is the image dimension.

Figure 2.8 shows how the value of τ_o converged for three different sequence-pairs, taken from different datasets.

2.4   Complete Algorithm

The complete method is summarized in Algorithm 1.

Input: Pair of Sequences $\{\mathcal{S}_i\}$ and $\{\mathcal{S}_j\}$ of silhouettes
Output: Fundamental Matrix $\mathbf{F_{ij}}$

$\{\mathcal{H}_{\mathcal{S}i}\} \leftarrow$ Compute Convex Hull And Tangent-Tables($\{\mathcal{S}_i\}$);
$\{\mathcal{H}_{\mathcal{S}j}\} \leftarrow$ Compute Convex Hull And Tangent-Tables($\{\mathcal{S}_j\}$);
$\tau_\mathbf{o} \leftarrow$ Compute Outlier Threshold   (see Section 2.3.5);
$n_S \leftarrow \max(2\tau_\mathbf{o}, 10)$;
candidates $= \{\ \}$;

repeat
    $(\mathsf{F}, \mathsf{model}) \leftarrow$ Make Hypothesis   (see Section 2.3.3);
    Evaluate (F) (see Section 2.3.4);
    if Promising Solution ;
        candidates $\leftarrow$ candidates $\cup$ $(\mathsf{F}, \mathsf{model})$;
until ($|\mathsf{candidates}| == n_S$ $||$ maximum iterations exceeded)
$C_k \leftarrow$ Rank And Find Best $(\mathsf{k}, \mathsf{candidates})$ using inlier count ;

NonLinear Minimization And Iterative Refinement($\{C_k\}$);

return Rank And Find Best $(1, C_k)$ using inlier count ;

Algorithm 1: Computing the epipolar geometry from dynamic silhouettes.

2.4.1 Results

Datasets

Table 2.1 summarizes information about the various camera network datasets that we have collected and processed. These multi-view video streams were all acquired indoors by various researchers in fairly controlled settings, and the traditional off-line camera calibration was done using the calibration grid, LEDs etc. Most of the subjects were humans as these multi-camera networks were geared towards capturing virtual models of actors using vision-based markerless motion capture [7, 22].

Silhouettes had been extracted for all these sequences using state of the art methods (except for the MIT dataset for which the simple approach described in Appendix B-1 was good enough). Although silhouette extraction in the general case is a hard problem, fairly robust and accurate methods are now know for dealing with static backgrounds. For all the datasets described in Table 2.1, reasonably good silhouettes could be extracted, and were used for modeling dynamic scenes using a variant of shape from silhouette techniques. Using our method, these same silhouettes could be used to also recover the camera calibration.

In this section we will present the results from computing epipolar geometry using our method for these datasets. The recovered information will be used for full network calibration later on and the complete calibration and reconstruction results will be presented later in Chapters 3 and 4.

MIT Sequence

This dataset was recorded by Sand et.al. [114] for their work on capturing deformable 3D human shapes from silhouettes. He used a co-located motion capture system for the calibration and synchronized the video upto a single frame. The 4-view video footage was approximately 4 minutes long and captured at 30 frames per second. The human subject is moving around in the scene; occasionally his silhouette gets clipped in the field of view, esp. near the feet.

31
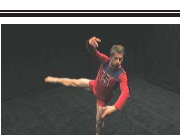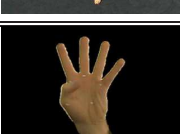
	Name	Resolution	Cameras	Frames	Pairs
	MIT (Sand et. al [114])	720×480	4	7000	5 out of 6 pairs
	DANCER (IN-RIA [42])	780×582	8	200	20 out of 28 pairs
	MAN (IN-RIA [42])	390×291	5	1000	10 out of 10 pairs
	KUNG-FU (MPI [22])	320×240	25	200	268 out of 300 pairs
	BALLET (MPI [22])	320×240	8	468	24 out of 28 pairs
	HAND (Brostow et. al [18])	640×480	9	200	25 out of 36 pairs
	CMU (Che-ung [24])	640×480	8	900	22 out of 28 pairs
	BOXER (Ballan [7])	1032×778	4	1000	6 out of 6 pairs
	BREAK-DANCER (Stark [131])	1920×1080	6	250	11 out of 15 pairs

Table 2.1: These datasets were acquired by various researchers in computer vision. We summarize the relevant information about the sequences used in our experiments. The last column reports the number of camera pairs for which the epipolar geometry was accurately recovered.
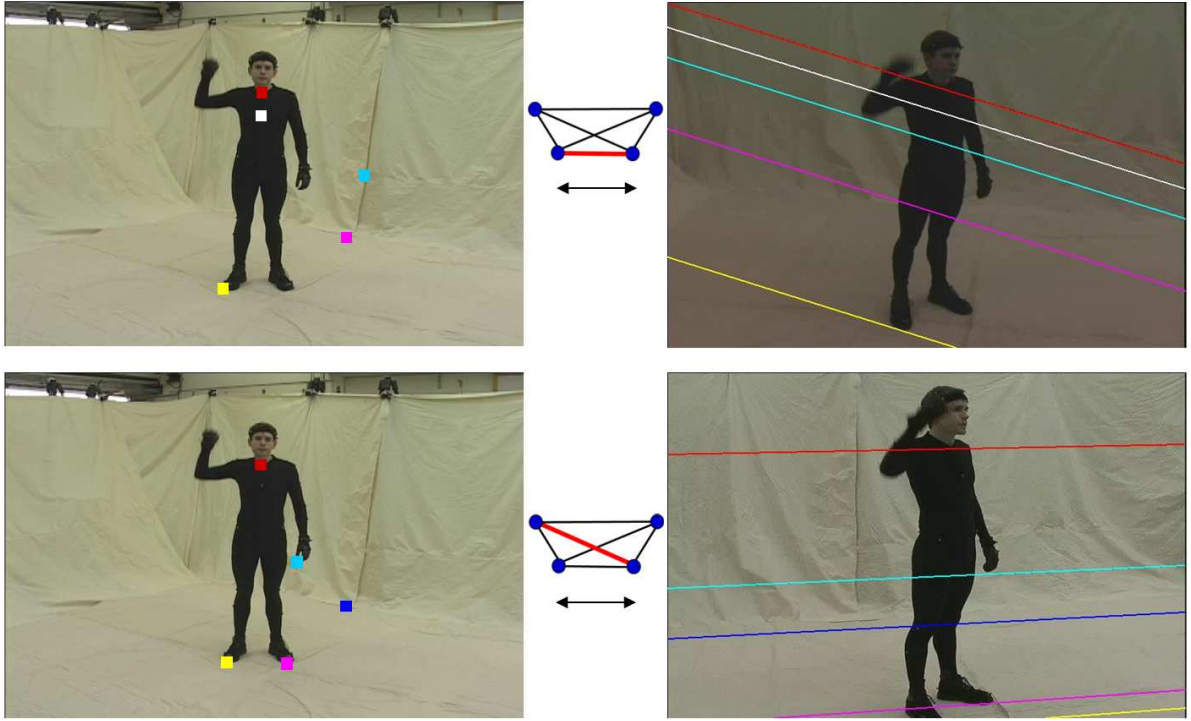
Figure 2.9: The estimated epipolar geometry for 2 of the 6 pairs in the 4-view MIT dataset.

However, this is handled robustly in our implementation.

Using the proposed approach, we computed the epipolar geometry for all pairs. The results from two pairs is shown in Figure 2.9. The epipolar geometry for one of the six pairs was unstable because in both views, the feet of the person was consistently clipped in most of the video. Since the person walks around the frontier points near the head of the person are almost planar which is a degenerate configuration for epipolar geometry estimation. Instead of using all 7000 frames that were available we chose every 5th frame and worked with about 1400 frames from video. Estimating the epipolar geometry took between about 150 seconds on an average for the six pairs. On an average, the RANSAC-based algorithm produced a good solution in about 25000 iterations but for higher reliability, multiple solutions were recovered and checked for consensus.
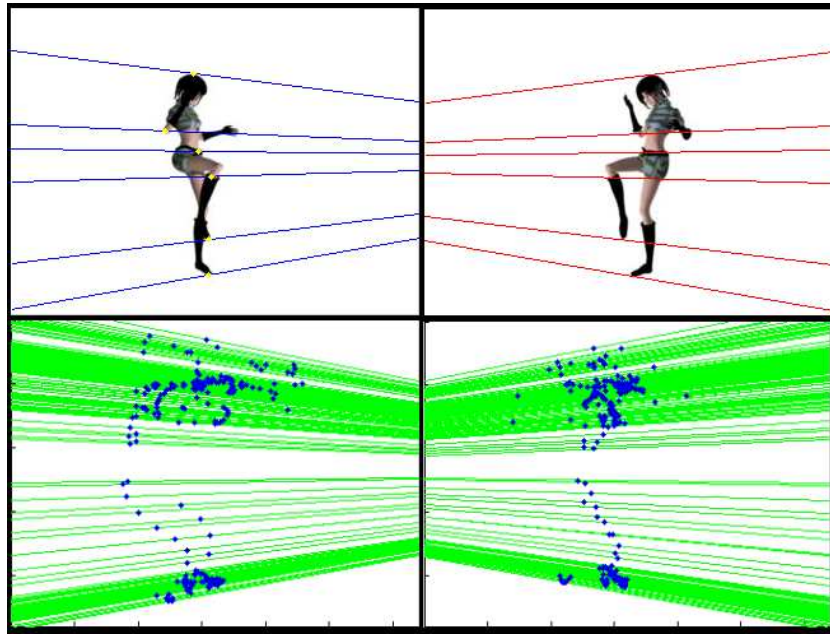
(a)
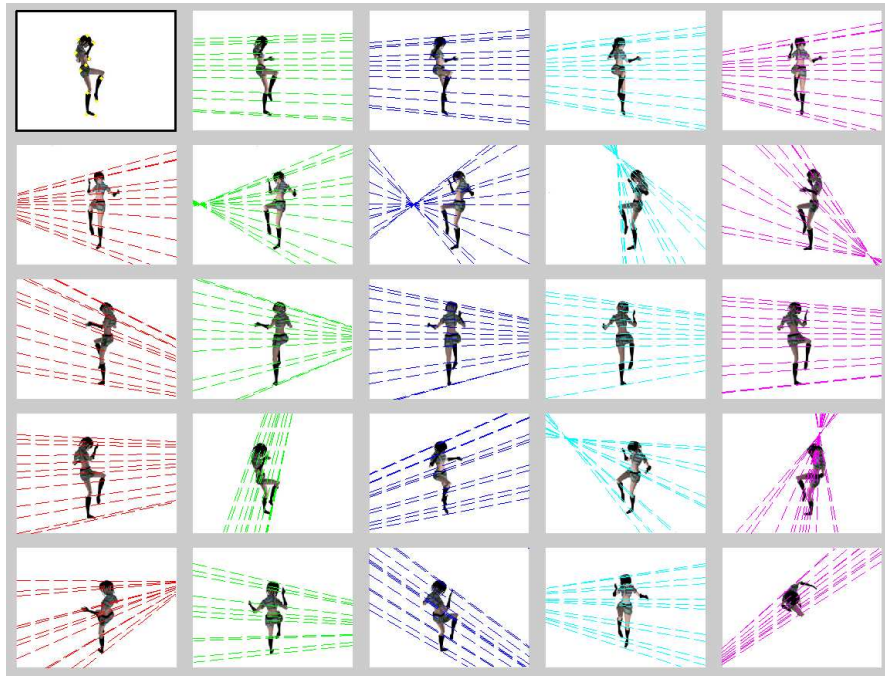
(b)

Figure 2.10: (a) The estimated epipolar geometry for one of the pairs in the 25-view KUNG-FU sequence. The extracted frontier points are also shown here (b) The estimated epipolar geometry between the first camera and all other 24 cameras.

34

Kung-Fu Sequence

It is rare to find a real dataset involving a large number of video cameras. However, we tested our method on a 25-view synthetic KUNG-FU dataset which was created by the researchers at MPI–Saarbrucken [22]. The results for a particular pair is shown in Figure 2.10(a). The top row shows the corresponding epipolar lines based on the estimated fundamental matrix while in the bottom half of the image, all the frontier point matches are displayed. The pairwise epipolar geometry for all images with respect to the first view is shown in Figure 2.10(b) and out of the 300 pairs, the epipolar geometry for 268 pairs was estimated accurately. Note that our method often fails when the cameras face each other resulting in epipoles somewhere close to the center of the image. As in this case, if the epipoles fall inside the silhouette's convex hull in most of the video frames, neither frontier points nor the epipolar tangent constraint exist. The algorithm either fails to find a solution or finds an ambiguous one which is detected and rejected. See Section 3.2.3 for the criteria that is used.

CMU 3D Room Sequence

The results from the CMU 3D-ROOM sequence in shown in Figure 2.11. Note that, the epipoles for some of the pairs coincide with the image of the camera in the respective views. This shows the accuracy of the epipole estimation for these pairs. Also, note that, the cameras are placed in a way that causes the silhouettes to often get clipped in some of the views. Finally, more results are shown in Figure 2.11 - the 8-view DANCER dataset, from the Perception Group at INRIA and the 9-view HAND dataset captured by Brostow et. al. [18] at Georgia Tech. More results from the BOXER and BREAK-DANCER sequences will be presented in the subsequent chapters.
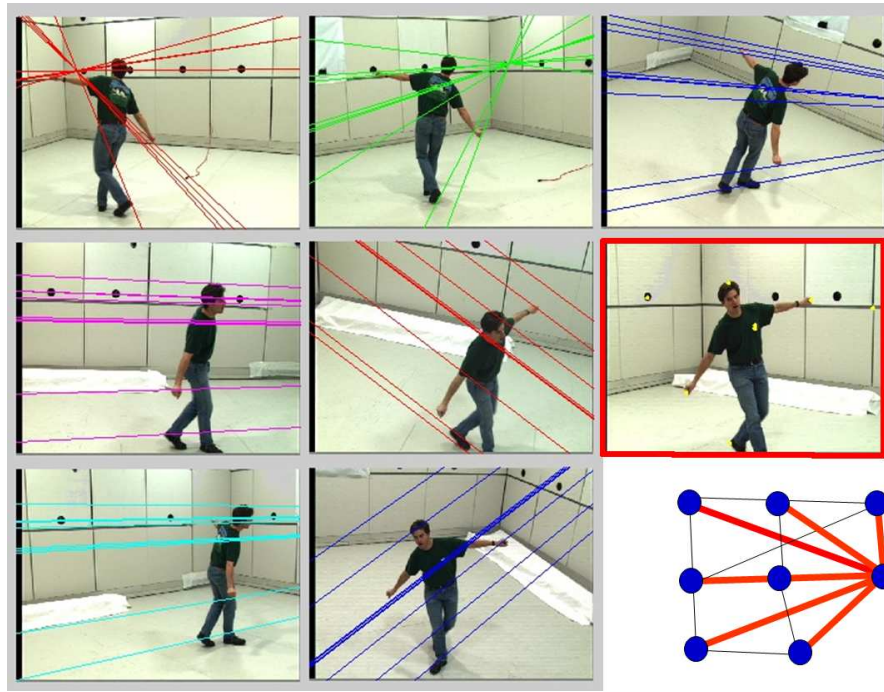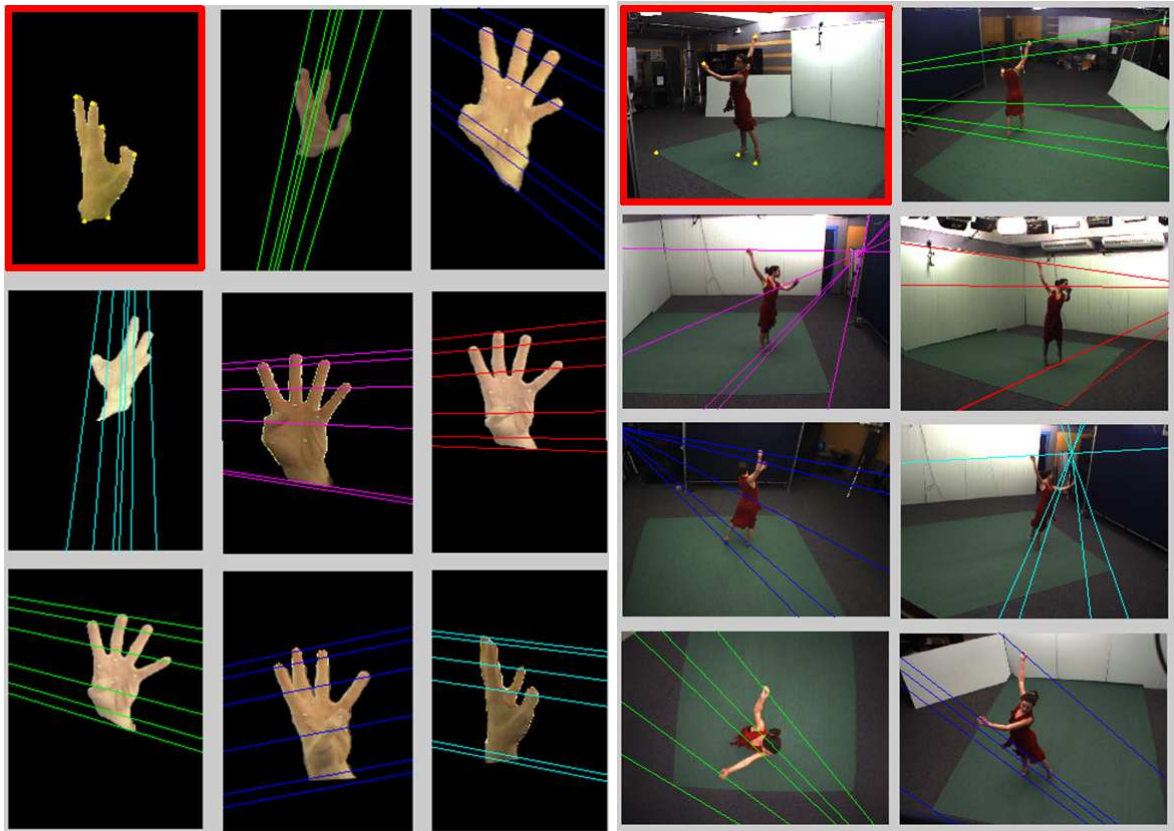
Figure 2.11: Recovered epipolar geometry for the (a) CMU (b) HAND and (c) DANCER datasets. Points are clicked in the image with a red border and epipolar lines are displayed on the other views.
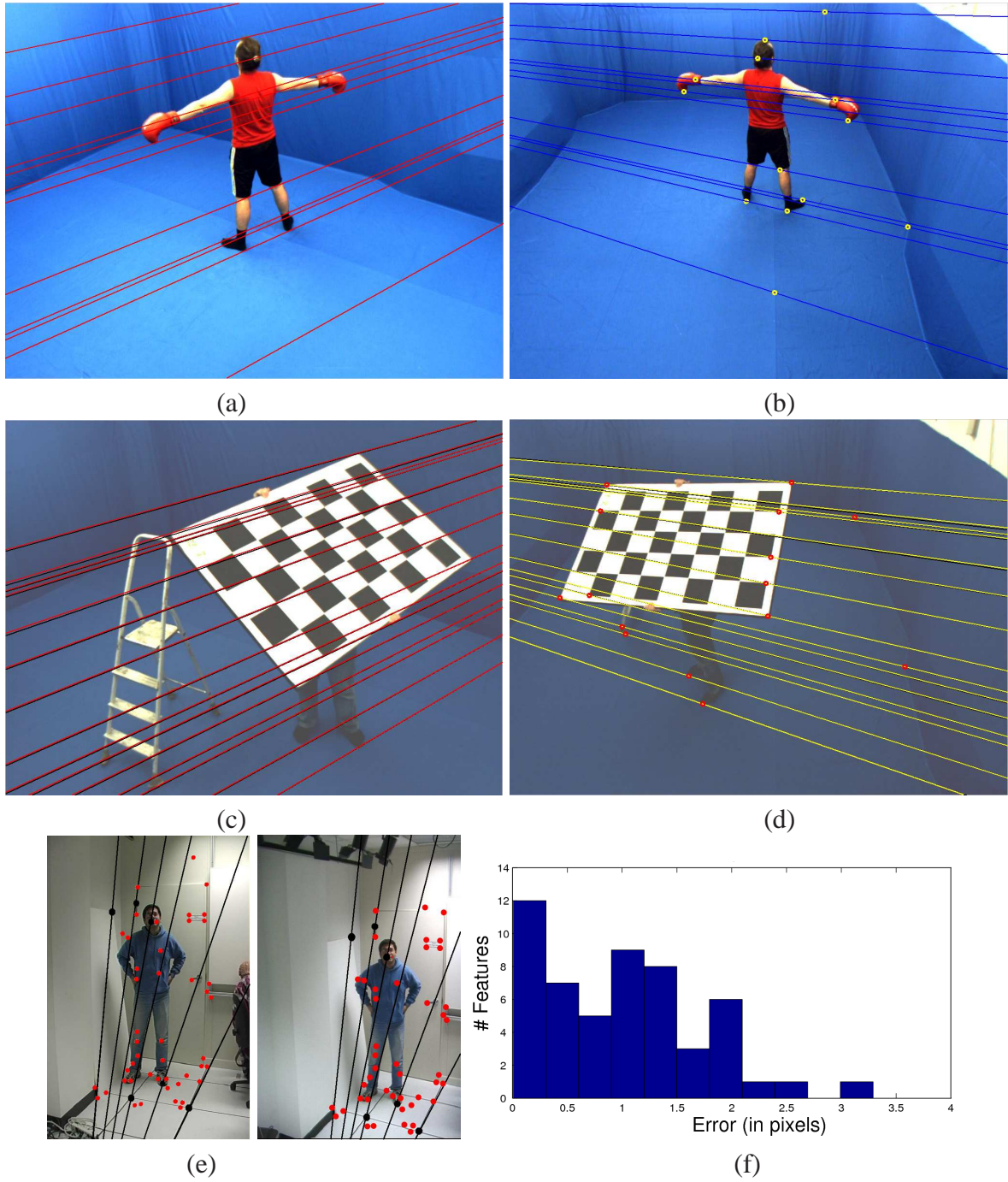
Figure 2.12: (a–b) Epipolar geometry recovered for a particular camera pair in the BOXER dataset. (c–d) Checkerboard image pair used for evaluation only. Ground truth epipolar lines are shown in black. Epipolar lines for our fundamental matrix estimate are shown in red and yellow. The image resolution is 1000×800 pixels. (e–f) For another camera pair, hand clicked points were used to verify the accuracy of the epipolar geometry estimate. The histogram shows the epipolar symmetric transfer error distribution for these points.

Evaluation

The mean residual error given by $\frac{1}{N}\sum\frac{e}{2}$, where e is defined in Equation 2.1 is reported for all estimates of the fundamental matrix. The synthetic KUNG-FU sequence reported a residual error of 0.12 pixels on average, while estimates for real datasets had a residual error of 0.25 pixels on an average with a range of $0.2 - 0.31$ pixels.

Our algorithm for epipolar geometry estimation was evaluated in two cases. First, one of the camera pair from the BOXER dataset was tested. Figure 2.12 (a–b) shows the estimated epipolar geometry using about 1000 frames of video. Subsequently, a checkerboard image pair (not used in our estimation process) was used for evaluation (shown in Figure 2.12 (c–d)). The user manually clicked 50 corresponding points and the mean symmetric residual error for these points was calculated. Our fundamental matrix estimate had an rms error of 1.21 pixels, while the error for the ground truth (derived from the checkerboard based calibration [10]) was 0.78 pixels. The relatively high residuals, in both cases, seems to be due to the error introduced by the user while clicking points.

The evaluation was done for another sequence (see Figure 2.12 (e–f)). This time, the mean symmetric residual error was 1.38 pixels. A distribution of the error is shown for the manually specified points (corresponding corner features on both the foreground as well as the background were used). Further accuracy analysis is presented in Chapter 3 after performing a full camera network calibration.

Our proposed algorithm applies RANSAC [9] in an unconventional way. Rather than using it only for robust estimation and handling outliers, we use it to explore a low dimensional bounded parameter space as well i.e. the 4D space of epipoles. This allows us to automatically adjust the random sampling budget towards detecting outliers and that for exploring the parameter space. An alternative strategy would have been to perform a deterministic, brute-force search in the 4D space of epipoles while using RANSAC only to sample the data to only deal with outliers. However, this would have the following disadvantages.

- Deterministic search for epipoles would require prior knowledge of the size of the attraction basin (i.e. convergence region), especially when the search is performed at a coarse level. This is not needed in our approach. For synthetic uncorrupted data (KUNG-FU sequence), we found a promising candidate in 1 in 6000 trials on an average. This seems to indicate that selecting the first direction in each image in approximately $\sqrt{6000} = 77$ random directions allows us to sample within the attraction basin of the true solution at least once. For higher reliability, we recover multiple solutions and then look for consensus amongst at least three. This approach was used for all the datasets in our experiments.

- If the epipoles were sampled using a deterministic strategy and RANSAC was used only to deal with erroneous silhouettes, then the number of iterations required would be significantly higher as we now show. Suppose, w is the probability that a random epipolar tangent is incident to a silhouette at a true silhouette point unaffected by silhouette noise. Note that the fraction of perfectly accurate silhouettes can be much lower than w, as an epipolar tangent to a noisy silhouette can still be an inlier as long as the noise does not corrupt the silhouette near the point of tangency. We found the range of w to be approximately $0.4 - 0.8$ in our experiments. For a fixed pair of epipoles, let us find the required number of RANSAC iterations. To compute a correct epipolar line homography, we must randomly pick three pairs of correct epipolar tangents – this has a probability of w^6. Thus, to ensure with p% confidence that the correct solution will be computed, we require the number of iterations k to be $\frac{log(1-p)}{log(1-w^6)}$. For $p = 0.95$ and $w = 0.75$ (75% inliers), $k = 15$. Now suppose epipoles were sampled deterministically on a sphere (appropriate in the calibrated case). Sampling with an angular interval of 4° between successive epipolar tangents should be sufficient. The total solid angle of a sphere is 4π steradians, which is approximately 40000 square degrees. Since each sample occupies 16 square degrees, we get 2500 samples on a sphere, and N=6.25 million

39

unique epipole pairs. Thus, the total number of iterations would be approximately equal to $kN = 93$ million.

2.5 The unsynchronized case

When the video sequences acquired by the camera network is unsynchronized, the epipolar tangent constraints which form the basis of the pairwise epipolar geometry estimation still exists up to an unknown parameter – the temporal offset Δt. We assume that the cameras are operating at a constant and known frame-rate. This is often the case with popular video cameras and is a reasonable assumption to make.

In this section we describe how the proposed algorithm can be extended to simultaneously recover both the temporal offset as well as the epipolar geometry. The main idea is to generate a random hypothesis by sampling an extra dimension – a possible range of temporal offsets in addition to the 4D space of epipoles. This algorithm typically requires more hypotheses than the synchronized case before a stable solution can be found, but a multi-resolution approach for computing the temporal offset speeds it up considerably.

2.5.1 Keyframe Selection

Directly finding the true temporal offset within a large search range will require many more hypotheses because the probability of selecting the correct temporal offset is quite low. We therefore adopt a coarse-to-fine strategy for this search. In video containing human subjects, the frontier points and epipolar tangents tend to remain stationary over a succession of frames. Although such frames are not suitable for accurate synchronization, they could be used for an initial coarse alignment of the two sequences. We will refer to these as *slow* frames. Similarly, frames in which the frontier points exhibit strong motion will be useful for accurate alignment and will be referred to as *fast* frames.

Without knowing the epipolar geometry, it is impossible to select the *slow* or *fast* keyframes accurately. Therefore, the list of keyframes are computed heuristically. We consider epipoles at infinity in four canonical directions (E, NE, N and NW) at 45° separation to each other in the image. Based on such hypothetical epipoles, we analyze the potential motion of frontier points in each sequence independently. This is used to build up list of *slow* and *fast* keyframes from the original sequences. As the RANSAC-based algorithm can search for promising epipoles locations, this information could be used to choose the hypothetical epipoles, and generate more accurate keyframes but at the cost of an extra prior step for the algorithm.

The algorithm proceeds in multiple stages. In the first stage, only the *slow* keyframes are used. A 5D random hypothesis is generated. The epipoles are sampled in the manner described earlier. For the random guess for the temporal offset, a large search range is coarsely sampled at this stage. The model verification step analyzes the error distribution in the same way as described in Section 2.3.4. It is possible that this stage estimates the epipolar geometry quite poorly, however, it helps to narrow down the search for the temporal offset. For every 40 promising candidates, a 99% confidence interval for the sample offset mean is computed, and this becomes the new sampling interval for the temporal offset. The process is continued until the search range becomes smaller than 20 frames.

In the second stage, the *fast* keyframes are used and the RANSAC-based algorithm samples from the smaller search range recovered in the first stage. Once 40 promising candidates have been found, the median of their temporal offsets is extracted.

In the final stage, all the frames are used to estimate the synchronization and epipolar geometry simultaneously. The offset is now sampled from a small interval of +/-5 frames from the estimated offset obtained from the previous stage. The distribution of promising epipoles obtained from the previous stage is used to bias the random sampling in the 4D space of epipoles. This allows us to find an accurate solution much more quickly. Although this version of the algorithm requires many more RANSAC iterations, the first two stages

are much faster as they work with smaller sets of keyframes. The stratified approach also allows us to sample epipoles from a more accurate distribution, which helps us find promising candidates more quickly in the final stage.

2.5.2 Results

We applied our techniques to the 4-view MIT sequence (courtesy Sand et.al. [114]) as the original video streams were synchronized using mocap sensors (see Figure 1.3(c) in Chapter 1). For this dataset, we extracted about 200 *slow* and *fast* keyframes from each of the four sequences. For a particular pair of cameras, we ran the basic algorithm (i.e. without using keyframes) for 5 million iterations and sampled the offset from a uniform interval (+/-125 frames) from the true offset. Figure 2.13(a) shows the distribution of promising solutions for the temporal offset. A consistent strong peak is observed near the true offset. Smaller peaks in the distribution indicate the presence of some periodicity of motion in the sequence. Figure 2.13(b) shows a typical distribution of offsets obtained in the first two stages of the algorithm. The sample median is plotted and the convergence of the search interval is also shown. We independently synchronized all six pairs, each time searching in a range of 500 frames, which was equivalent to a duration of 16.6 seconds. The recovered synchronization offsets are listed in Table 2.5.2. These individual measurements were independently estimated and can be further refined during joint synchronization of all the cameras in the network. This will be described in Chapter 3.

2.6 Conclusion

In this chapter a method to estimate the epipolar geometry from silhouettes in two video streams was described. The approach based on RANSAC [9] and robustly recover frontier point correspondence from video. First the case of synchronized video was presented. Next
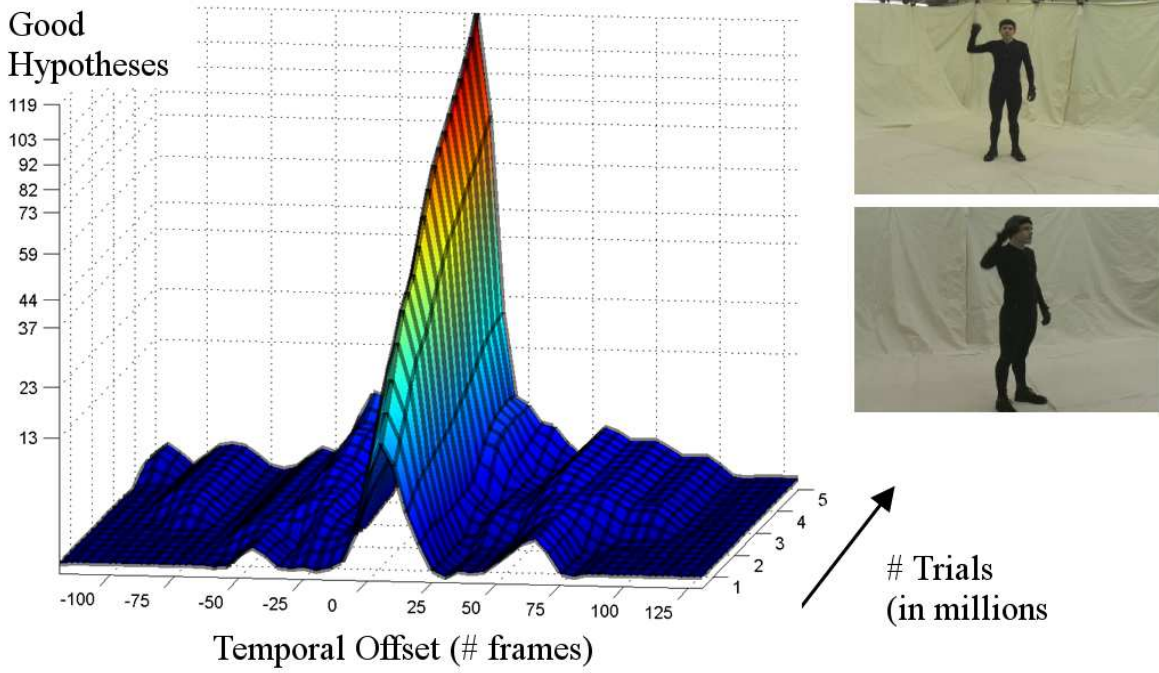
(a)
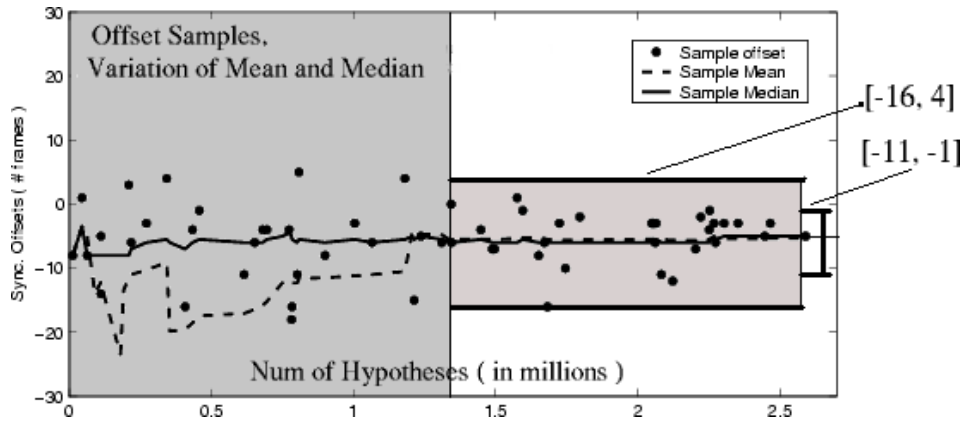
(b)

Figure 2.13: Experiment done on chosen pair from the MIT sequence. (a) A strong peak is observed in an interval that contains the real temporal offset. (b) Sample solutions for the temporal offset and the search intervals are shown plotted against the number of RANSAC trials. The median is used to estimate the temporal offset.

the algorithm was extended to simultaneously recover the epipolar geometry and the temporal synchronization of the camera pair from unsynchronized video. The pairwise information recovered here will be useful for calibration and synchronization of a full network which
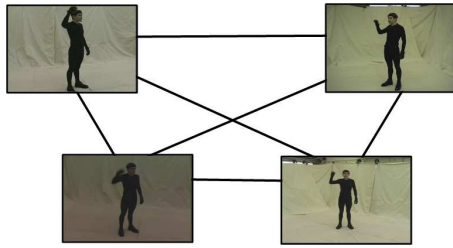
Pair	Range	t_{ij}	σ_{ij}	True ($\hat{t}_{ij}$)
e_{01}	[-13,-3]	-8.7	0.80	-8.32
e_{02}	[-11,-1]	-8.1	1.96	-8.60
e_{03}	[-12,-2]	-7.7	1.57	-7.85
e_{12}	[-5,5]	-0.93	1.65	-0.28
e_{13}	[-5,5]	0.54	0.72	0.47
e_{23}	[-6,4]	1.20	1.27	0.75

(a)   (b)

Figure 2.14: (a) The pairwise synchronization offsets computed for the 6 camera pairs in the MIT sequence are shown here. (b) The table lists the computed search interval, and the mean and variance of the estimate of the temporal offset. These are fairly close to the ground truth measurements.

is described in Chapter 3. The method is particularly useful for shape-from-silhouette systems [20, 81, 96] as visual-hulls can now be reconstructed directly from uncalibrated and unsynchronized video of an event recorded from multiple viewpoints.

CHAPTER 3
Full Calibration and Synchronization from Pairwise
Information

3.1 Introduction

In this chapter, we describe how to recover the full calibration and synchronization of a camera network from estimates of the epipolar geometry and the temporal offset of various pairs of cameras within the network. We first describe how we recover the full camera calibration from the epipolar geometries, and then discuss our method to solve the problem of network synchronization.

3.2 Camera Network Calibration

We first consider the problem of recovering full camera calibration from pairwise epipolar geometries. Given a sufficient number of links in a graph (similar to the one shown in Figure 3.1 except that the links now represent estimates of fundamental matrices), our goal is to recover the Euclidean camera matrices for all cameras in the network. An overview of our approach is described in Figure 3.2. An important step in this approach is to compute an accurate projective reconstruction of the camera network from epipolar geometries and two view matches. We start by first recovering a triplet of projective cameras from the fundamental matrices between the three views. Using an incremental approach, we add a new camera to the calibrated network by resolving a different triplet of cameras each time. Each time a new camera is added, all the parameters corresponding to the cameras and 3D points are refined
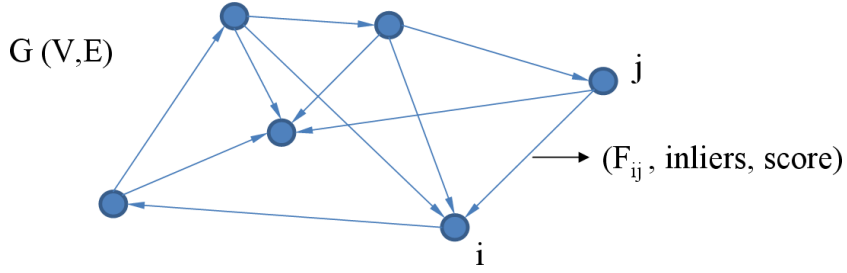
Figure 3.1: The camera network graph is shown. The edges represent pairwise epipolar geometry estimates and attributes such as an accuracy measure and set of two view correspondences (inliers).

using *projective bundle adjustment*. Finally, when a full projective reconstruction is available, standard techniques for self-calibration and Euclidean (metric) bundle adjustment are used to compute the final metric camera calibration.

3.2.1 Background

Most structure from motion techniques for uncalibrated sequences start by estimating the fundamental matrix or the trifocal tensor from two or three view correspondences. The trifocal tensor in the three-view case plays the same role that the fundamental matrix plays in the two view case. If the trifocal tensor is known, a triplet of consistent projective camera matrices for the images can be directly computed from the tensor and subsequently a projective reconstruction of the scene may be linearly computed. This is also true for the fundamental matrix in the two view case. See [55] for details.

Various approaches exist for computing projective reconstructions of the cameras and the scene simultaneously. On one hand, there exists direct factorization-based approaches for structure from motion that can compute the projective reconstruction in one step, without favoring any particular camera [134, 141]. However they typically require all the 3D points to be visible in all the cameras. Thus, most practical approaches for large-scale structure from motion incrementally computes the projective reconstruction of the cameras and the scene and our method also belongs to this category.

46
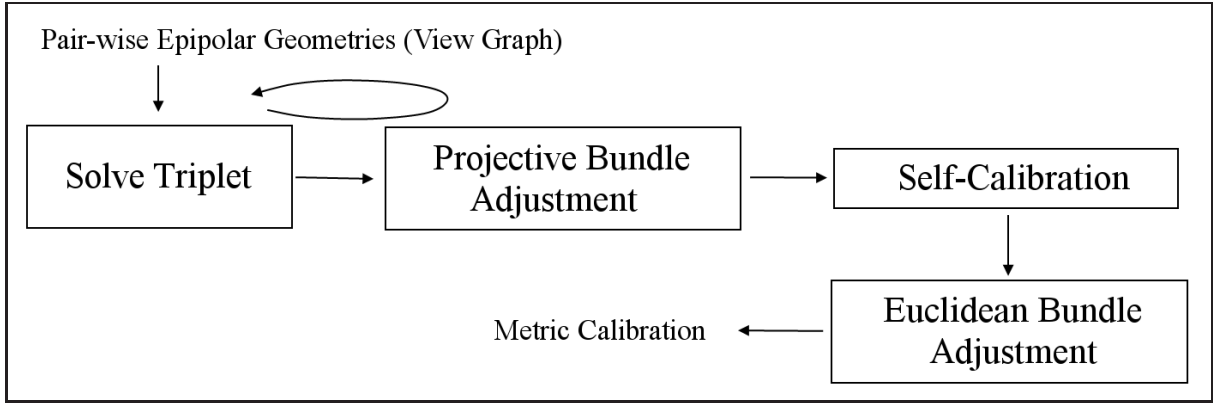
Figure 3.2: Overview of our incremental camera network calibration approach from epipolar geometries estimates. The fundamental step involves computing the projective reconstruction of a triplet of cameras. Bundle Adjustment for the projective and metric reconstructions are performed to globally refine all the parameters.

In the structure from motion pipeline developed by [106], first an initial reconstruction is done from two views. Then, one by one, the projective calibration of the other cameras is recovered within the initial reconstruction frame and the projective reconstruction of the entire sequence is incrementally computed. The camera matrices and 3D point estimates obtained in this way are used to initialize a *projective bundle adjustment*, which simultaneously refines all the camera parameters as well as the coordinates of the 3D points, by minimizing the overall reprojection error. This is a large non-linear minimization problem with potentially many parameters. However, the existence of sparse and efficient solvers makes the step computationally feasible. Self-calibration [106] is followed by *Euclidean bundle adjustment* to determine the optimal camera parameters. The Euclidean bundle adjustment is similar to the projective case, except that in the Euclidean case, the cameras are parameterized by the intrinsic and extrinsic parameters, while the 3D points are parameterized using non-homogeneous coordinates.

In our silhouette-based calibration method, frontier point correspondences do not generalize to more than two views. In a three-view case, the frontier points in the first two views do not correspond to those in the last two views. Although three view correspondences, called *triple points*, do exist on the silhouette as reported by [41, 82], they are hard to extract from

47

uncalibrated images. Thus, we are restricted to only two-view correspondences over different pairs in our camera network. We incrementally compute a full projective reconstruction of a camera network from these two-view correspondences and the corresponding fundamental matrices. Martinec et. al. [91, 92] also addressed this problem but in a different context, and proposed some solutions in parallel to our work.

Levi and Werman [86] studied the following problem: Given only a subset of all possible fundamental matrices in a camera network, when is it possible to recover all the missing fundamental matrices? They were mainly concerned with theoretical analysis, and their proposed algorithm is not suited for the practical implementation of computing projective reconstructions from sets of two-view matches in the presence of noise.

We now discuss the fundamental step in our approach, which involves recovering three consistent projective cameras, given a triplet of fundamental matrices corresponding to three views in general position.

3.2.2 Resolving Camera Triplets

Given any two fundamental matrices between three views, it is not possible to compute three consistent projective cameras. The two fundamental matrices can be used to generate canonical projective camera pairs $\{\mathbf{P}_1, \mathbf{P}_2\}$ and $\{\mathbf{P}_1, \mathbf{P}_3\}$, respectively. However, these do not correspond to the same projective frame. $\mathbf{P}_3$ must be chosen in the same projective frame as $\mathbf{P}_2$, and the third fundamental matrix is required to enforce this.

These independently estimated fundamental matrices are denoted by $\mathbf{F}_{12}$, $\mathbf{F}_{13}$, and $\mathbf{F}_{23}$, while the unknown projective cameras are denoted by $\mathbf{P}_1$, $\mathbf{P}_2$, and $\mathbf{P}_3$, respectively (see Figure 3.3). The three fundamental matrices are said to be compatible when they satisfy the following constraint:

$$\mathbf{e}_{23}^{\mathrm{T}}\mathbf{F}_{21}\mathbf{e}_{13} = \mathbf{e}_{31}^{\mathrm{T}}\mathbf{F}_{32}\mathbf{e}_{21} = \mathbf{e}_{32}^{\mathrm{T}}\mathbf{F}_{31}\mathbf{e}_{12} = 0 \qquad (3.1)$$

The three fundamental matrices available in our case are not compatible because they were
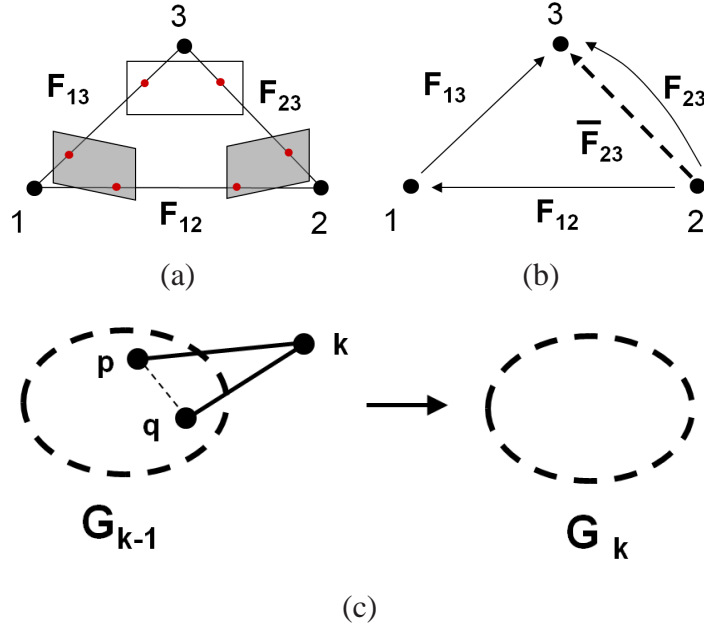
Figure 3.3: (a) Three nondegenerate views for which all the fundamental matrices have been estimated independently. (b) Family of solutions for the third fundamental matrix ($\overline{\mathbf{F}}_{23}$), compatible with the other two ($\mathbf{F}_{12}$ and $\mathbf{F}_{13}$). We look for a compatible solution closest to the measured $\mathbf{F}_{23}$. (c) New camera k is incrementally linked to a calibrated network by resolving a suitable triplet involving two cameras within G_{k-1}.

independently estimated from two-view correspondences. One linear approach for computing $\mathbf{P}_1$, $\mathbf{P}_2$, and $\mathbf{P}_3$ from three compatible fundamental matrices is described in [55]. However, it is not suitable when the fundamental matrices are not compatible, as in our case.

We now describe our linear approach to compute a consistent triplet of projective cameras. As described in [55], given $\mathbf{F}_{12}$ and $\mathbf{F}_{13}$, canonical projective cameras, $\mathbf{P}_1$ and $\mathbf{P}_2$ as well as $\mathbf{P}_3$ can be chosen as follows:

$$\mathbf{P}_1 = [\mathbf{I}|0] \quad \mathbf{P}_2 = [[\mathbf{e}_{21}]_\times \mathbf{F}_{12}|\mathbf{e}_{21}]$$
$$\mathbf{P}_3 = [[\mathbf{e}_{31}]_\times \mathbf{F}_{13}|0] + \mathbf{e}_{31}\mathbf{v}^\mathrm{T} \tag{3.2}$$

Here, $\mathbf{P}_3$ has been defined up to an unknown 4-vector $\mathbf{v}$ in Equation 3.2. By expressing $\mathbf{F}_{23}$

as a function of $\mathbf{P}_2$ and $\mathbf{P}_3$, we obtain the following relation.

$$\overline{\mathbf{F}}_{23} = [\mathbf{e}_{32}]_\times \mathbf{P}_3 \mathbf{P}_2^+ \tag{3.3}$$

The expression for $\overline{\mathbf{F}}_{23}$ is linear in $\mathbf{v}$. Hence, all possible solutions for $\overline{\mathbf{F}}_{23}$ span a 4D subspace of $\mathbb{P}^8$ [86]. We solve for $\mathbf{v}$, which produces the solution closest to the measured $\mathbf{F}_{23}$ in the 4D subspace (in Frobenius norm sense). $\mathbf{P}_3$ can now be computed by substituting this value of $\mathbf{v}$ into Equation 3.2. The resulting $\mathbf{P}_1$, $\mathbf{P}_2$, and $\mathbf{P}_3$ are fully consistent with $\mathbf{F}_{12}$, $\mathbf{F}_{13}$, and the matrix $\overline{\mathbf{F}}_{23}$ computed above.

In order to choose $\mathbf{F}_{12}$, $\mathbf{F}_{13}$, and $\mathbf{F}_{23}$ for this approach, we rank the three fundamental matrices based on an accuracy measure; the least accurate one is assigned to be $\mathbf{F}_{23}$, while the choice of the other two does not matter. This accuracy measure is described next.

The method described here works only when the camera centers for the three cameras are not collinear. This degenerate configuration can be detected by analyzing the location of the six epipoles (when all three camera centers are collinear, $\mathbf{e}_{ij} = \mathbf{e}_{ik}$ for various permutations of the three views). In our method, when a degenerate triplet is detected, we reject it and look for the next best possibility. For most camera networks (all the datasets we tried), cameras were deployed around the subject, and collinearity of camera centers was never a problem.

3.2.3 Ranking the Fundamental Matrices

In order to build up the projective reconstruction of the network from epipolar geometries, we need an automatic way to use the best fundamental matrices amongst the ones that are available. To rank the fundamental matrices based on the accuracy of their estimates, their inlier spread score s_{ij} is computed as follows:

$$s_{ij} = \sum_{(u,v)\in\mathrm{P_i}} |u - v|_2 + \sum_{(u,v)\in\mathrm{P_j}} |u - v|_2$$

50

Here P_i and P_j represent the set of 2D point correspondences in views i and j that forms the set of inliers for the corresponding fundamental matrix $\mathbf{F}_{ij}$. A higher inlier spread score indicates that $\mathbf{F}_{ij}$ is stable and accurate. The score is proportional to the inlier count, but captures the spatial distribution of 2D points that form the inlier set. Ideally, we should triangulate and analyze the spatial distribution of the 3D point cloud, however the epipolar geometry only allows a projective reconstruction where distances between points cannot be computed. However, the 2D spatial spread of the inliers in the two images is often a good indicator of how spread out the 3D points would be. Although this ranking scheme is not perfect, it is good enough for detecting the unstable and inaccurate estimates of the fundamental matrix from the first stage.

3.2.4 Incremental Construction

Our incremental approach to projective reconstruction starts by greedily choosing a set of three views for which the fundamental matrices are, relatively, the most accurate. As described in the previous section, this triplet is resolved, resulting in a partial projective reconstruction of three cameras. Next, cameras are added one at a time using the approach described next. The process stops when either all cameras have been added, or when no more cameras can be added to the network because of insufficient links (fundamental matrices).

Given G_{k-1}, a projectively calibrated camera network with $(k-1)$ cameras, we first need to choose the new camera that will be added next to this calibrated network. For this, we inspect the links (epipolar geometries) between cameras that belong to G_{k-1} and those that have not been reconstructed yet. The camera chosen for reconstruction is denoted by k, and the two cameras in G_{k-1} corresponding to the two links are denoted by p and q, respectively.

Thus for cameras p and q in G_{k-1} and k, the new view, we now reconstruct a triplet of consistent projective cameras from $\mathbf{F}_{pk}$, $\mathbf{F}_{qk}$, and $\mathbf{F}_{pq}$ (here $\mathbf{P}_k$ plays the role of $\mathbf{P}_3$). The choice of p and q is irrelevant, since the fundamental matrix corresponding to any pair within

G_{k-1} can be computed, because all projective cameras within it, are exactly known. Finally, the computed projective camera $\mathbf{P}_k$ is transformed into the projective frame of G_{k-1}, and added to the network. This produces a complete projective reconstruction of G_k, the camera network which includes the added new view.

For a network with N cameras in general position, this method will work if a sufficient number of links are present in the camera network graph. The various solvable cases are discussed in [86]. In our case, resolving the initial triplet requires three links; every subsequent view that is added requires at least two links. Thus, the minimum number of unique links that must be present in the graph is $3 + 2(N - 3) = 2N - 3$. When more links are available in the graph, our ranking procedure chooses the best ones and the less accurate links may never be used.

3.2.5 Computing the Metric Reconstruction

Every time a new camera is added, a projective bundle adjustment is done to refine the calibration of all cameras in the partial network. This prevents error accumulation during the incremental construction. Camera networks are typically small, containing 8 to 12 cameras; therefore, running the projective bundle adjustment multiple times is not an issue. Once a full projective reconstruction of the camera network has been computed, a linear self-calibration algorithm [106] is used to upgrade from a projective to a metric reconstruction.

Finally, an Euclidean bundle adjustment is done by parameterizing the cameras in terms of the intrinsic and extrinsic parameters. In all cases, we constrain the camera *skew* to be zero but impose no other parameter constraints. Depending on the exact scenario, other constraints could be enforced at this step for higher accuracy – for example, enforcing a fixed aspect ratio of pixels and enforcing the principal point to be at the center of the image. For higher accuracy, radial distortion in the images should also be modeled in the Euclidean bundle adjustment, which typically further reduces the final reprojection error. However, estimation

of radial distortion was not done in our current work, and is one way in which the calibration accuracy can be improved further.

	Name	Cameras	Frames	Pairs	Reprojection Error (final)
	MIT [114]	4	7000	5 out of 6 pairs	0.26 pixels
	DANCER (INRIA)	8	200	20 out of 28 pairs	0.25 pixels
	MAN (INRIA)	5	1000	10 out of 10 pairs	0.22 pixels
	KUNG-FU [22]	25	200	268 out of 300 pairs	0.11 pixels
	BALLET [22]	8	468	24 out of 28 pairs	0.19 pixels
	BREAK-DANCER [131]	6	250	11 out of 15 pairs	0.23 pixels
	BOXER [7]	4	1000	6 out of 6 pairs	0.22 pixels

Table 3.1: The camera network calibration was done on these datasets. The second-last column in the table, shows the number of links that were present in the camera network graph. The reprojection error obtained after the Euclidean bundle adjustment is listed in the final column.

3.2.6 Results

The full camera network calibration was performed on each one of the datasets described in the previous chapter, in Table 2.1. In each case, only a few links were missing in the camera network graph, and there were a sufficient number of accurate links (good estimates of the epipolar geometry), that allowed an accurate projective, and subsequently metric reconstruction to be computed. These results are summarized in Table 3.1. Figure 3.4 shows, for two different datasets, how the epipolar geometry estimates are more accurate after the projective reconstruction stage. All triplets of fundamental matrices at this stage are compatible, i.e. they satisfy the constraints listed in Equation 3.1.

Evaluation

We evaluated our method by comparing our calibration with ground truth data for the synthetic KUNG-FU sequence. The results are shown in Figure 3.6. Since the metric reconstruction of the camera network obtained by our method is in an arbitrary coordinate system, it first needs to be scaled and robustly aligned to the ground truth coordinate frame. The final average reprojection error in all 25 images, after the Euclidean bundle adjustment, is 0.11 pixels and the reconstructed visual hull of the Kungfu character visually looks as accurate as the visual hull computed from ground truth. The details of how these 3D models were computed is provided in Chapter 4. In Figure 3.6(c), the two models have been overlaid upon each other.

Figure 3.6(d) shows the camera network along with the reconstructed frontier points from the complete sequence. Figure 3.6(e) shows the relative error in the focal length estimates of the 25 cameras in the network, and Figure 3.6(f) shows the deviation of the principal point from the center of the image (the principal point in the ground truth data). Note that, the focal length estimation could have been more accurate if all the cameras in the bundle adjustment step were parameterized to share the same intrinsic parameters. However, in realistic scenarios, each camera will have a different set of intrinsic parameters, and therefore, we allow the

(a)

(b)
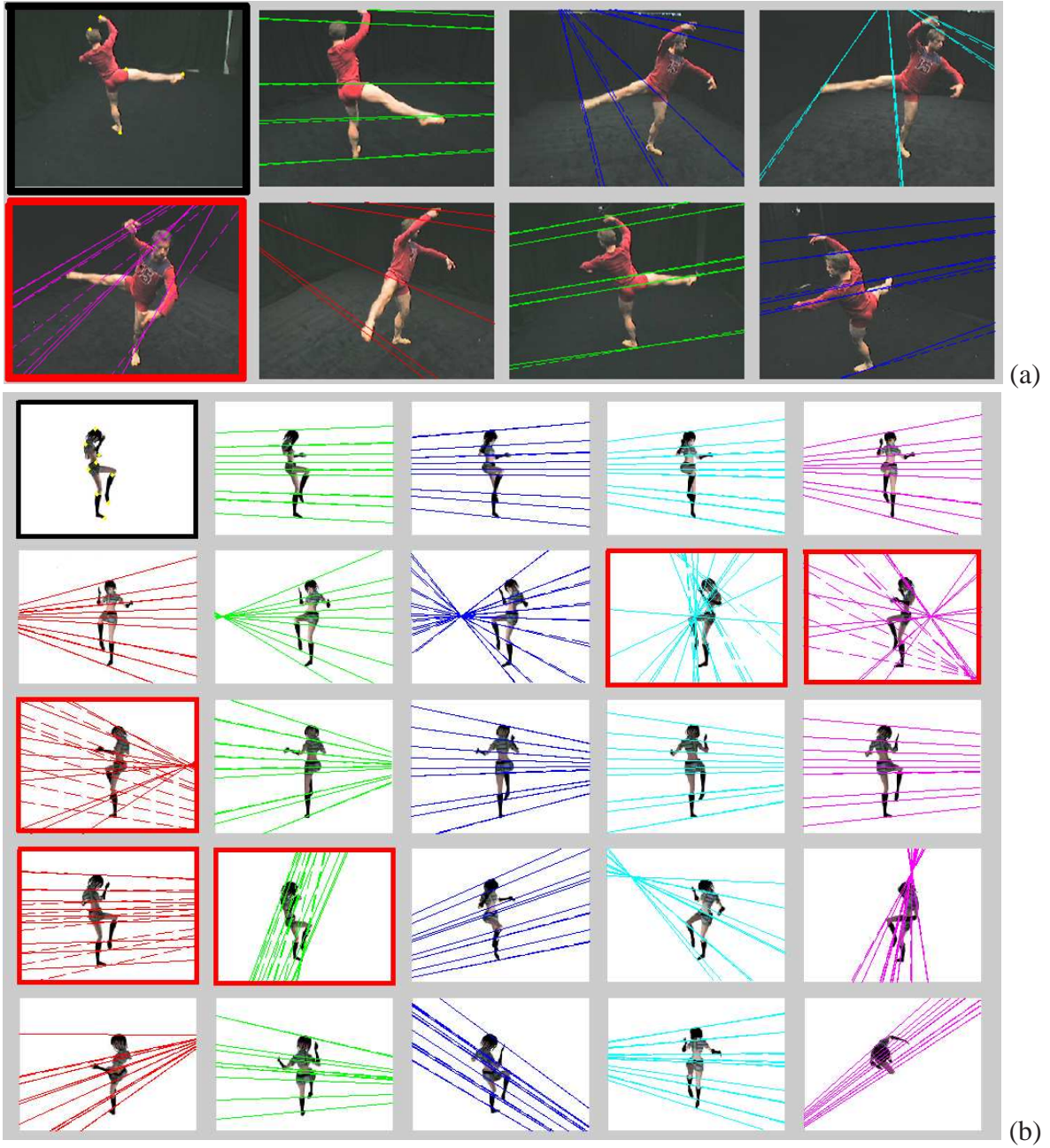
Figure 3.4: After the incremental projective reconstruction of the whole camera network, some errors in the epipolar geometry of the individual pairwise links can be corrected. The estimates for views with a red border were incorrect but have now been fixed. The dotted epipolar lines shown the incorrect epipolar geometry while the correct estimate derived from the projective calibration is shown with solid lines.

intrinsics to be different from camera to camera during the evaluation.

For the BOXER sequence with 4 cameras, the reprojection error after Euclidean bundle ad-
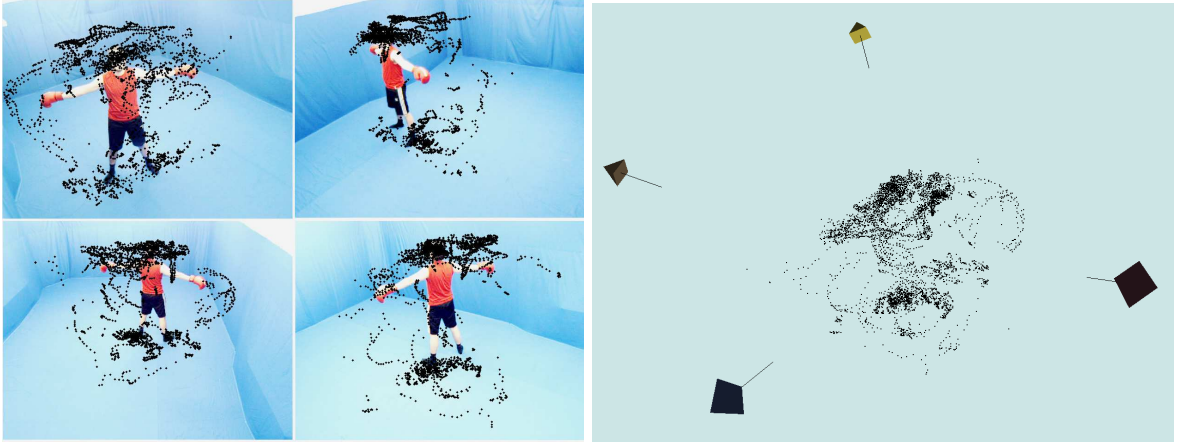
Figure 3.5: The final metric reconstruction of the camera network and 4953 reconstructed frontier points from the BOXER sequence.

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

(e)   (f)
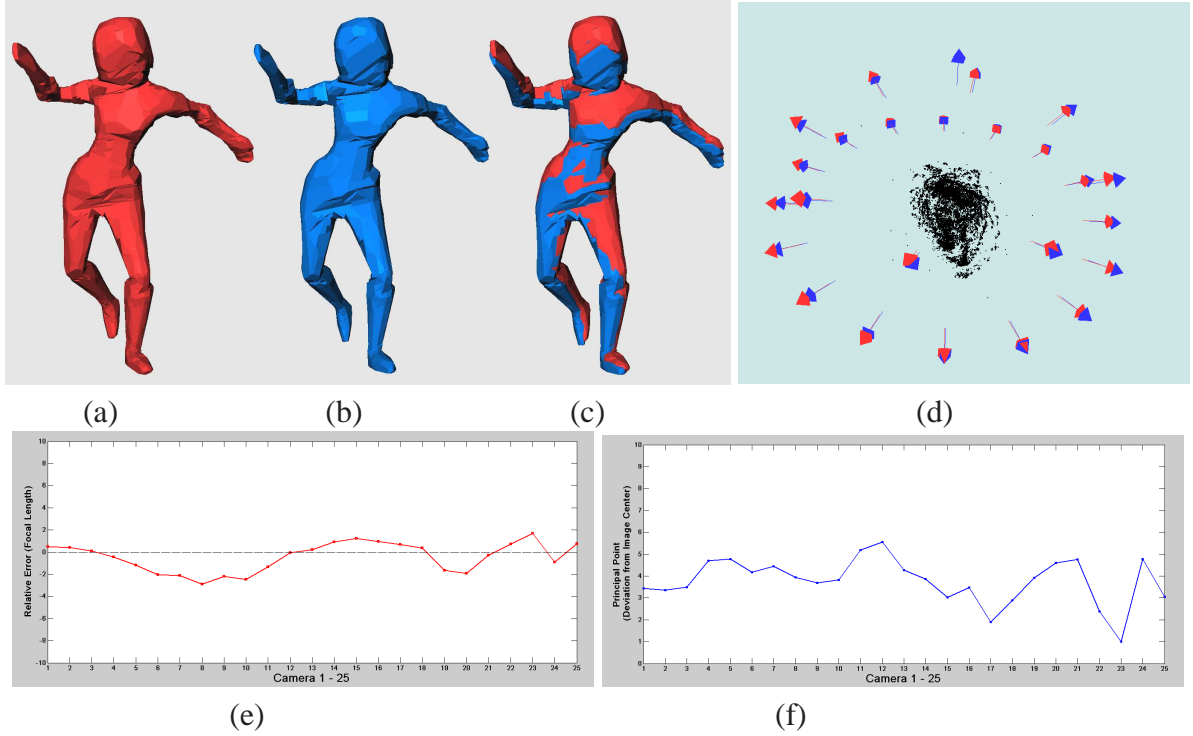
Figure 3.6: (Best seen in color) For the KUNG-FU sequence, ground truth is available. Models computed using (a) ground truth calibration and (b) the calibration recovered by our method. (c) The registered 3d models. (d) The camera network registered to the coordinate frame of the ground truth data.

justment was 0.22 pixels. The 2D feature points and reconstructed frontier points are shown in Figure 3.5 along with the metric reconstruction of the camera network. A visual hull recon-

struction of the human subject in this dataset was performed and results were visually quite accurate. See Section 4.3 for results for the BOXER and on other datasets.

The camera network calibration approach was inaccurate for the HAND sequence, although many of the epipolar geometry estimates (see Figure 2.11(b)) were fairly accurate. This was due to the lack of sufficient motion in the sequence. In this video, the hand position does not change and the motion is mostly limited to the fingers. Therefore many of the frontier points end up near the finger-tips. When the frontier points correspond to 3D points on a plane, the estimated fundamental matrix is inaccurate as this is one of the degenerate configurations for epipolar geometry estimation. When frontier point correspondences are close to a degenerate configuration, a technique such as QDEGSAC [40] could potentially alleviate the problem.

Although the pairwise epipolar geometry estimates for the HAND sequence had small residuals, the associated uncertainty of the fundamental matrix estimates were higher, and thus, an accurate projective reconstruction could not be computed. The reprojection error after the projective bundle, in this case, was greater than 0.5 pixels, and the camera intrinsics estimated through self-calibration were not good enough for initializing the Euclidean bundle adjustment. Therefore, the final calibration was inaccurate, and the reprojection error was quite high, showing that the algorithm was stuck in a local minima.

It is possible to apply our method to recover the relative pose parameters of cameras in a network, when all the intrinsics (especially the focal length) are approximately known. Arguably, this calibrated case is easier to handle than the fully uncalibrated case, and it may be possible to robustly estimate the camera extrinsics even when the pairwise epipolar geometry estimates are less accurate. In this case, the essential matrices can be computed from the fundamental matrices, and the relative camera pose for every pair can be directly recovered without an intermediate projective reconstruction. Once again, the final metric reconstruction is obtained after doing a full Euclidean bundle adjustment.
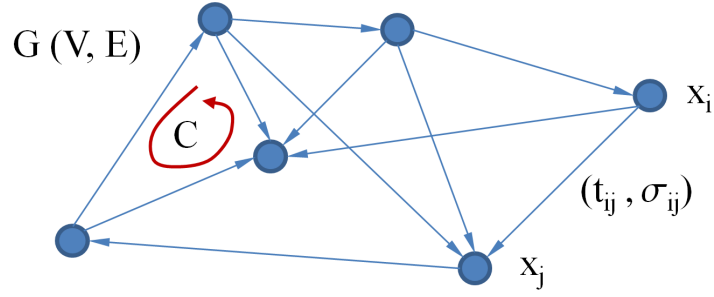
Figure 3.7: The camera network graph is shown. The edges represent pairwise measurements. When the edges represent sequence alignment offsets, they should sum to zero for all cycles such as C.

3.3 Camera Network Synchronization

The camera network synchronization problem is an instance of the general sensor synchronization problem in a network. In our case, every camera can be thought to have an independent timer, and the time differences can be measured in frame alignment offsets, since we assume that all cameras are operating at a constant and known framerate. One method for recovering such offsets was described in Chapter 2.

We represent the sensor network by a directed graph $G(V, E)$ as shown in Figure 3.7. There are N sensors, and each node $v_i \in V$, has a timer denoted by x_i. A directed edge in this network, $e_{ij} \in E$, represents an independent measurement of the time difference $x_j - x_i$, between the two timers. Each estimate t_{ij} has an associated uncertainty represented by the standard deviation σ_{ij}, that is inversely proportional to the uncertainty.

When G represents a tree, that is, it is fully connected and has $N - 1$ edges, it is possible to synchronize the whole network. When additional edges are available, each provides a further constraint, which leads to an overdetermined system of linear equations. Each edge contributes a linear constraint of the form $x_i - x_j = t_{ij}$. Stacking these equations produces a $|E| \times N$ system of linear equations. Assuming that each measurement is corrupted by independent Gaussian noise, the maximum likelihood estimate of the N timers is obtained by computing the weighted least squares solution of the linear system (each equation is multiplied

by the factor $\frac{1}{\sigma_{ij}}$). The timer estimates (the first camera is fixed at zero) are optimal, provided no outliers are present in the edges being considered.

However, it is fairly easy to detect outlier edges in the network. A consistent network should satisfy the constraint $(\sum_{e \in C} e) = 0 \,\forall$ cycles $C \in G$. For every edge $e \in E$, we check the sum of edges for cycles of length three that also contains the edge e. An outlier edge will have a significantly large number of nonzero sums, and could be easily detected and removed. This method will produce very robust estimates for complete graphs because $\frac{N \cdot (N-1)}{2}$ linear constraints are available for N unknowns. In the minimal case, a fully connected graph with at least $N-1$ edges is still sufficient to synchronize the whole network, although the estimates in this case, will be less reliable.

3.3.1 Results

Full network synchronization was performed on the MIT sequence. The sub-frame synchronization offsets from the first to the other three sequences were found to be 8.50, 8.98, and 7.89 frames, respectively, while the corresponding ground truth offsets were 8.32, 8.60, and 7.85 frames, respectively. The results are summarized in Table 3.3.1. The temporal offsets t_{ij} (and the uncertainties σ_{ij}) for the six pairs were estimated using the silhouette based approach described earlier (see Section 2.5.1 for details on how these were obtained).

3.4 Conclusions

In this chapter, a method to recover the full calibration and synchronization of the camera network was proposed. The input to the algorithm consists of estimated epipolar geometry and temporal synchronization, between many different camera pairs in the network. The calibration approach is a stratified one – first a projective reconstruction is computed, which is next upgraded to a metric one using self-calibration, and a final bundle adjustment is performed to
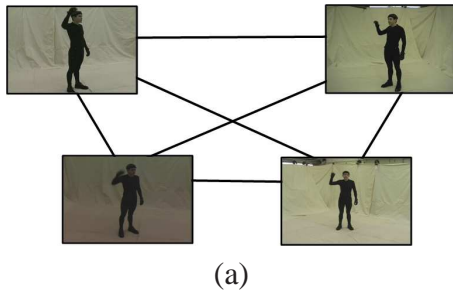
Pair	t_{ij}	σ_{ij}	$\overline{t_{ij}}$	True ($\hat{t}_{ij}$)
e_{01}	-8.7	0.80	-8.50	-8.32
e_{02}	-8.1	1.96	-8.98	-8.60
e_{03}	-7.7	1.57	-7.89	-7.85
e_{12}	-0.93	1.65	-0.48	-0.28
e_{13}	0.54	0.72	0.61	0.47
e_{23}	1.20	1.27	1.09	0.75

(a) (b)

Figure 3.8: (a) The pairwise synchronization offsets computed for the 6 camera pairs in the MIT sequence are shown here. (b) The table lists the initial estimates (t_{ij} and σ_{ij}). The final estimates obtained after full network synchronization is performed is shown in column ($\overline{t_{ij}}$). These are within $\frac{1}{3}$ of a frame i.e. $\frac{1}{100}^{th}$ of a second within the ground truth offsets.

obtain the optimal camera parameters. The recovered calibration and synchronization is used to reconstruct dynamic scenes involving humans. The 3D reconstruction methodology and results are presented in Chapter 4.

CHAPTER 4
Dynamic Scene Reconstruction

4.1 Introduction

Once the camera network is fully calibrated and synchronized, it becomes possible to combine information from multiple views, for the purpose of 3D reconstruction of the observed time-varying event. Various approaches for this task have been investigated in the past. One of the earliest successful systems [69, 70] coined the term *virtualized reality*, and used dense stero to reconstruct a dynamic scene from synchronized video streams. A special dome (see Figure 1.1(a)) comprising about 50 cameras mounted on a 5-meter diameter geodesic dome was used. The cameras recorded at a constant 30 frames per second and were synchronized using a hardware trigger. The system captured humans performing a variety of tasks on video, in real-time, and the stereo-based 3D reconstruction was subsequently computed offline.

Since then, systems such as [2, 95, 41, 96, 114] have also used silhouettes of the foreground subjects or combined silhouettes and stereo cues [22, 53, 60, 104, 131] to reconstruct time-varying events within a small area. One advantage of using silhouette based approaches is that a good reconstruction can be obtained with fewer cameras (often 8–12), provided they are suitably deployed. However, it is quite difficult to automatically extract foreground silhouettes from video in a general, uncontrolled scene due to various factors, such as image noise, complex lighting and similar appearance of the foreground and background. Nevertheless, in controlled scenes with simple background models, silhouettes can be recovered robustly and can lead to fairly convincing reconstructions.

Some of these 3D reconstruction systems were designed for real-time capture, transmis-

sion and reconstruction, while others were built for offline digitization, and could produce results of better quality and accuracy. Digitizing human actors has been of great interest, and many of these earlier systems had been designed specifically for reconstructing human actors using camera networks in well controlled environments. At a broad level, the methods for digitizing events in 3D can be classified into two categories depending on whether they employ model-based or model-free reconstruction approaches.

Model-based approaches have been particularly popular for modeling humans. They often use a parameterized human body model consisting of a skeleton, appropriate joint structure and some surface representation like a tesselated mesh. Model-based methods such as [22, 60, 131] utilize a priori knowledge about the observed scene, and formulate the reconstruction problem in terms of fitting a model to the multiple images using a suitable optimization scheme. Computing the optimal model parameters for every frame of the video allows the 3D pose of the foreground subject to be recovered in a robust fashion. Model-based methods are ideal for reconstructing human actors, as they can exploit temporal coherence, and traditionally, such methods have been preferred for capturing motion data for animations. This is often referred to as the problem of *markerless motion capture*.

On the other hand, model-free approaches are more general, and reconstruct the 3D scene without assuming any knowledge about its structure. The methods applied involve shape-from silhouette-like approaches, such as image-based visual hull [96], polyhedral visual hulls [95, 20] or stereo-based approaches [70]. Shape-from-silhouette approaches are popular because of the relative simplicity, higher robustness and computational efficiency compared to dense stereo methods. Algorithms for computing exact visual hulls were recently proposed [41, 83]. Another method [42], also based on shape from silhouette, recently proposed computing silhouette consistent shapes which had better geometric properties than the visual hull, such as local smoothness and curvature. They also allowed better viewpoint free rendering of the texture mapped 3D models that were captured. Silhouette-based approaches are also relevant

for model-based methods, as silhouettes can restrict the search for parameters of the 3D human skeleton such as joint angles, and provide strong constraints for performing model fitting. These model parameters can be estimated by maximizing the overlap between the projected human template and the original silhouettes in the multiple images, in addition to using other appearance cues.

4.2   Visual Hulls

Techniques for reconstructing shapes from silhouettes was first proposed by Baumgart [8] in 1974. Most silhouette based methods attempt to compute an approximation of the visual hull, a term that was coined by [81]. Since then a large number of visual hull algorithms have been proposed and at a high level they can be classified based on the underlying geometric representations – namely volumetric approaches such as [24, 25, 138] that compute a volumetric representation of the visual hull, and more recently, surface based approaches that directly compute tesselated meshes of the visual hull [41, 83, 95]. The surface based approaches are computationally efficient, more robust, have been shown to give higher quality results and can be used for real-time 3D reconstruction. In this section, we will briefly review recent work on exact visual hulls [41, 82, 83], and discuss the exact polyhedral visual hull (EPVH) algorithm that we build upon in Chapter 6.

The visual hull of an object is defined as the maximal shape that produces the same set of silhouettes in multiple calibrated views. The intersection of viewing cones, back-projected from silhouettes in a finite number of views, produces the visual hull of the object. The visual hull can provide a good approximation of the shape of the object when sufficient viewpoints are available. Typically 8–12 cameras, widely spaced out, can provide a reasonable reconstruction of a human. The visual hull is guaranteed to contain the true object inside of it. This is a property that is used by many multi-view reconstruction approaches. The visual hull is a projective topological polyhedron with curved edges and faces. When the contours in

(a)

Triple Point Vertex

View Edge Vertex

Cone Intersection Edge

View Edge

Planar Facet
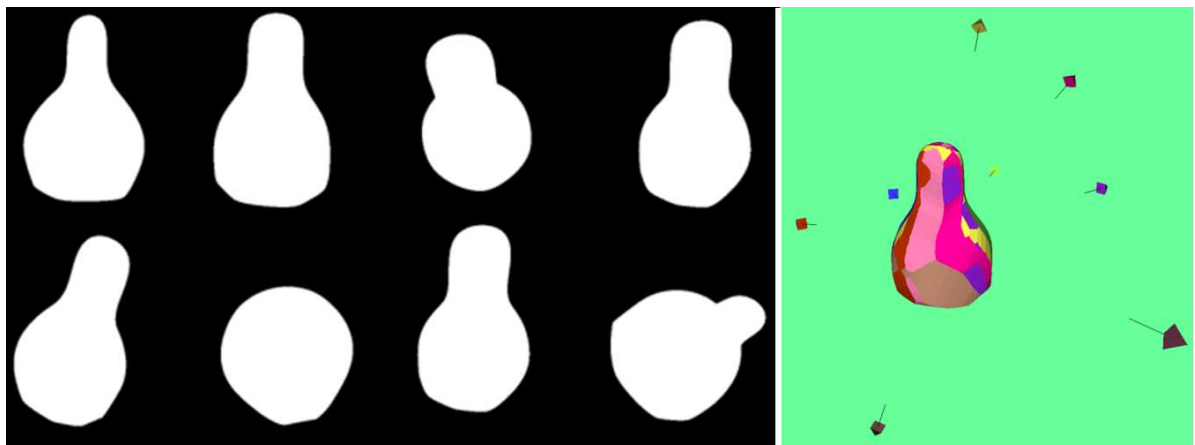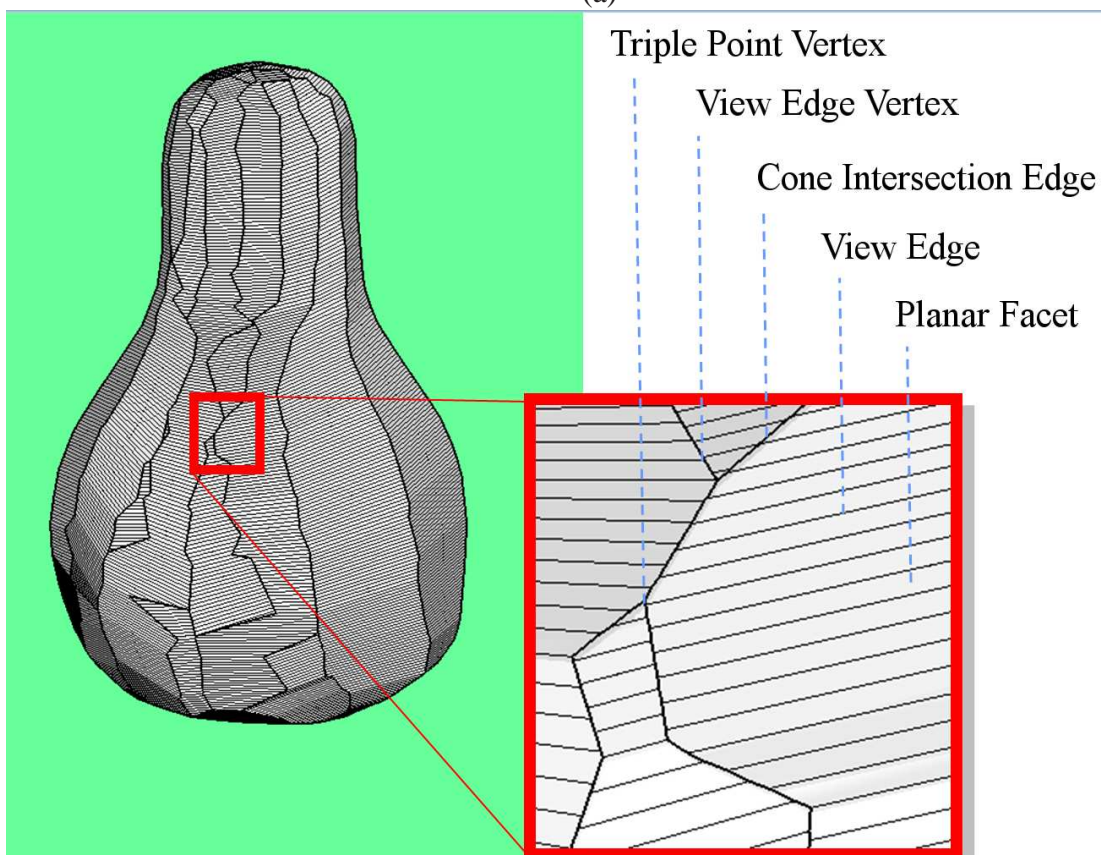
(b)

Figure 4.1: (a) The visual hull of a pear-shaped object was computed from 8 calibrated silhouette images shown here. On the right, the visual hull is shown in 3D along with the camera poses. (b) Anatomy of the Exact Visual Hull mesh. see the text and [41, 82] for more details.

the silhouette images are represented using polygons, the visual hull becomes a polyhedron. An algorithm to compute this exact visual hull was proposed first by [84], and subsequently by [41], for polygonal silhouettes. The polyhedral visual hull, computed by [41], exactly projects onto the silhouette polygons and can be computed both efficiently and robustly.

The back-projection of 2D points on the apparent contour gives rise to *viewing rays*, each of which may contribute a finite segment to the visual hull. This is called a *view edge*. At least one point on a viewing ray must touch the surface at a point on the rim or contour-generator. All view edges from a particular view-point form a ruled surface, which is called a *cone-strip*. The intersection of silhouette or viewing cones gives rise to additional vertices called *triple points* and *cone intersection edges*. The cone-intersection edges and the triple points always lie outside the true surface. The EPVH algorithm [41] which we use to reconstruct the dynamic objects in our scenes, proceeds by first recovering all view edges corresponding to all the silhouettes. Next, the cone-intersection edges and the triple points are estimated using numerically stable and efficient schemes. The results are exact and the overall method is quite robust and fast. The topological structure of the visual hull polyhedra captures key information about the unknown rims or contour generators. This will be studied in detail in Chapter 6, in the context of enforcing silhouette constraints within a multi-view stereo approach for 3D shape reconstruction.

4.2.1   Silhouette Interpolation

Visual hull methods typically treat the temporal offset between multiple video streams as an integer, and ignore sub-frame synchronization. Given a specific frame from one video stream, the closest frame in other 30 Hz video streams could be as far off as $\frac{1}{60}$ seconds in time. While this might seem small at first, it can be significant for a fast-moving person. This problem will be illustrated in Figure 4.5(d), where the visual hull was reconstructed from the closest original frames in the sequence. The gray area in the figure represents what is inside the

visual hull reconstruction, and the white area corresponds to the reprojection error (points inside the silhouette in one view, carved away from other views). Sub-frame offsets need to be considered to perfectly synchronize the motion of the arms and the legs in this case.

To deal with this problem, we propose temporal silhouette interpolation. Given two adjacent frames i and $i + 1$ from a video stream, we compute the signed distance map in each image, such that the boundary of the silhouette represents the zero level set in each case. Let us denote these distance maps by $d_i(x)$ and $d_{i+1}(x)$, respectively. Then, for a sub-frame temporal offset $\Delta \in [0, 1]$, we compute an interpolated distance map denoted $S(x) = (1 - \Delta)d_i(x) - \Delta d_{i+1}(x)$. Computing the zero level set of $S(x)$ produces the interpolated silhouette. This simple scheme, motivated by [32], robustly implements linear interpolation between two silhouettes without explicit point-to-point correspondence. However, it is approximate and does not preserve shape. Thus, it requires the inter-frame motion in the video to be relatively small.

4.3 Results

We now show the results of our camera network calibration and visual hull reconstruction on a number of different multi-view video datasets (see Table 3.1 for relevant details about these datasets). Although most of the experiments involved video streams of human subjects, both our calibration and reconstruction approaches are completely general, and can be used for reconstructing time-varying events involving any solid non-rigid shape which can be reliably segmented from the background. We tested our method on one synthetic sequence that contains 25 cameras and 8 real datasets, independently acquired by various computer vision researchers in their own laboratories, using different configuration of video cameras. We were able to recover the full calibration of the camera network in all these cases, without prior knowledge of or control over the input data. We thus show that it is possible to remotely calibrate a camera network, and reconstruct time-varying events from archived video footage

with no prior information about the cameras or the scene.

The reconstruction of the synthetic KUNG-FU dataset is shown in Figure 4.2 followed by the 8-view BALLET dataset. The original BALLET sequence was reconstructed using a model-view approach [22], but for simplicity, we only compute the visual hull from 8 views, which gives a reasonable approximation of the human shape. However, other methods proposed recently such as [7, 42, 131], could also be used for more accurate 3D reconstruction of the same event. Figures 4.3 and 4.4 show the reconstructions from the BOXER, BREAK-DANCER, DANCER and MAN sequences respectively. Some corresponding frames from the input video are shown along with the visual hull computed for that instant in time. The geometry of the camera network is also shown in 3D along with the visual hull reconstruction. Detailed results regarding the accuracy of the recovered camera calibration, are presented in Section 3.2.6.

In Figure 4.5(a), the metric 3D reconstruction for the 4-view MIT sequence is shown. The calibration and synchronization were recovered using the methods described in prior chapters of this thesis. This shows that we are able to reconstruct visual hulls from uncalibrated and unsynchronized footage. To test the accuracy of the calibration that was computed, we projected the visual hull back into the images (see Figure 4.5(b)). In the perfect scenario, the silhouettes would be completely filled in. Inaccurate calibration, poor segmentation or lack of perfect synchronization could give rise to empty regions (white pixels) in the silhouettes. Our tests gave consistent results, and the silhouettes were mostly filled, except for fast moving parts, where the re-projected visual hull is a few pixels smaller on one side of a silhouette (see the close-up in Figure 4.5(c)). This arises mostly when sub-frame synchronization offsets are ignored, or due to incorrect segmentation in the case of motion blur or shadows.

For higher accuracy, we computed visual hulls from interpolated silhouettes. The silhouette interpolation was performed using the sub-frame synchronization offsets that were computed earlier on for this sequence (see Table 3.3.1). In Figure 4.5(d), an example is shown. Given three consecutive frames, we generated the middle frame by interpolating between the

67
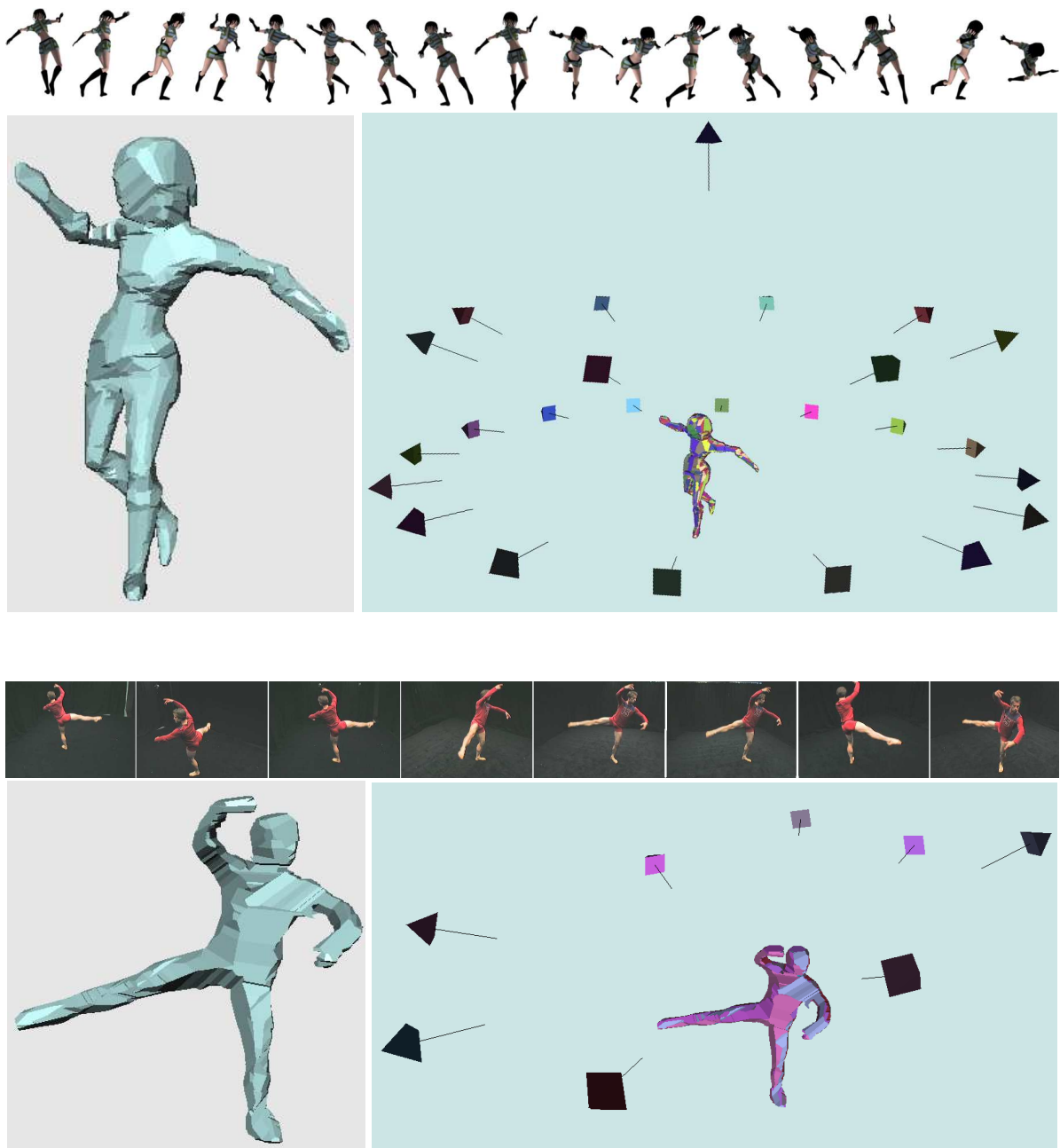
Figure 4.2: Metric 3D reconstruction of the KUNG-FU and BALLET sequence.
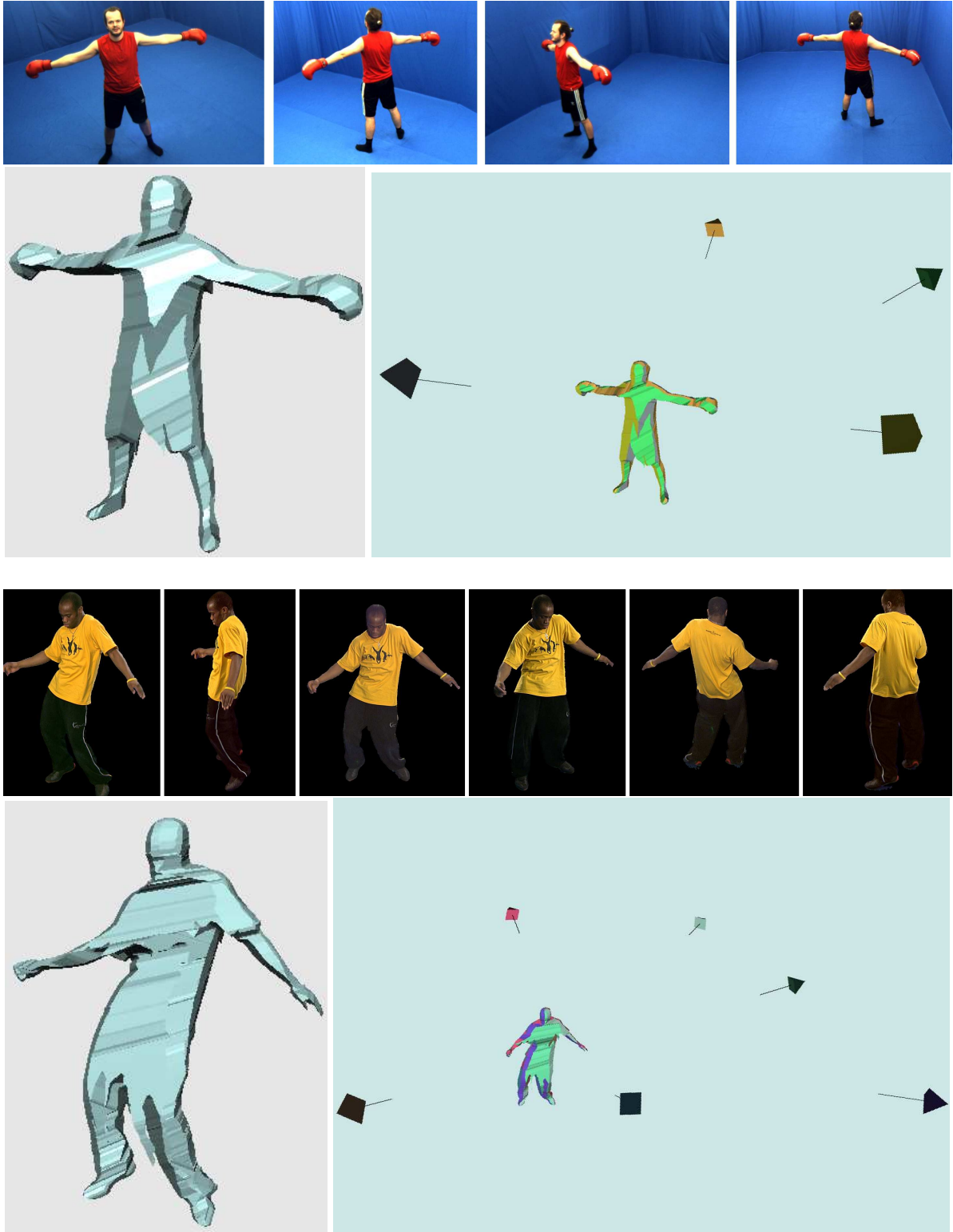
Figure 4.3: Metric 3D reconstructions of the BOXER and BREAK-DANCER sequences.
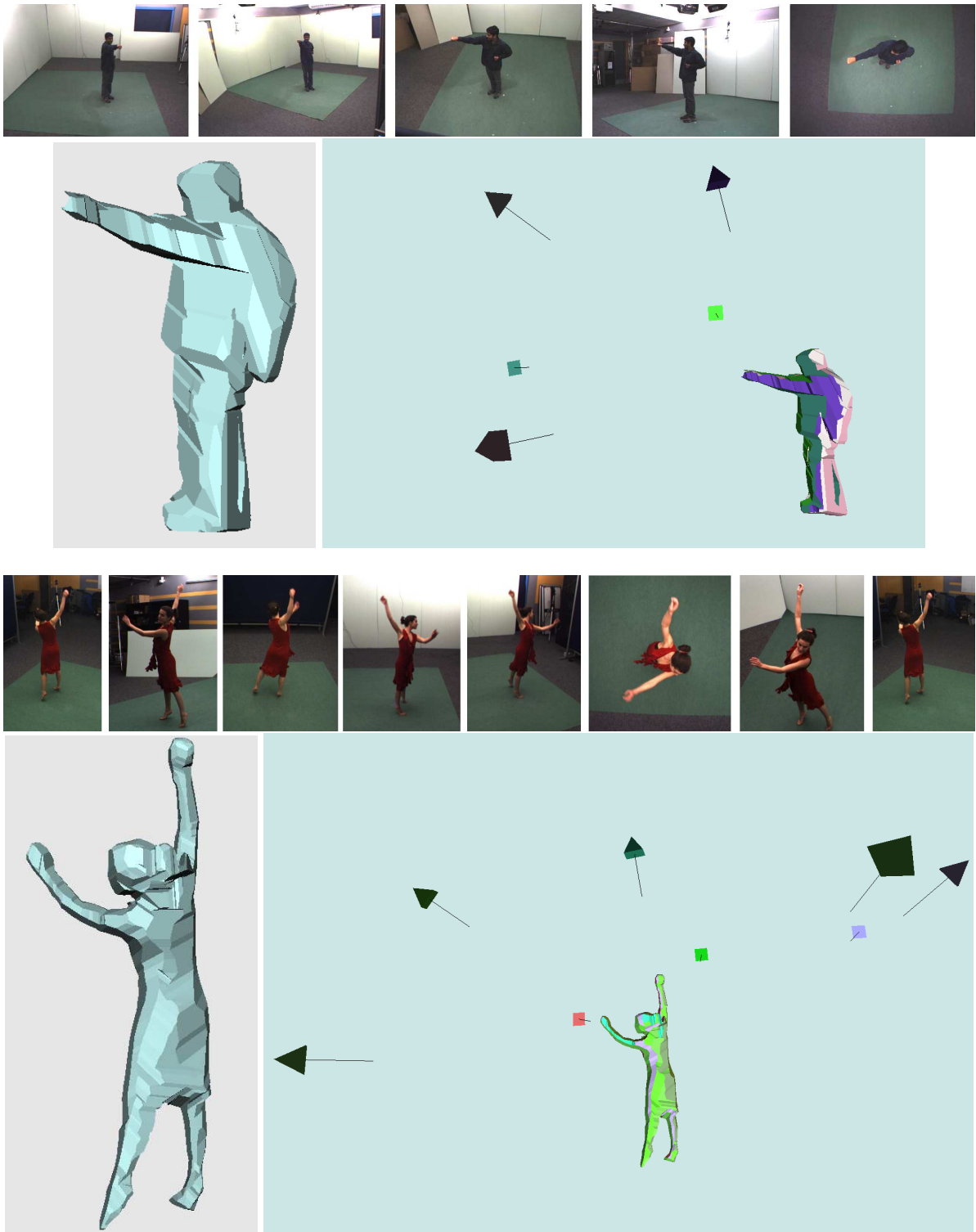
Figure 4.4: Metric 3D reconstructions of the MAN and the DANCER sequences.
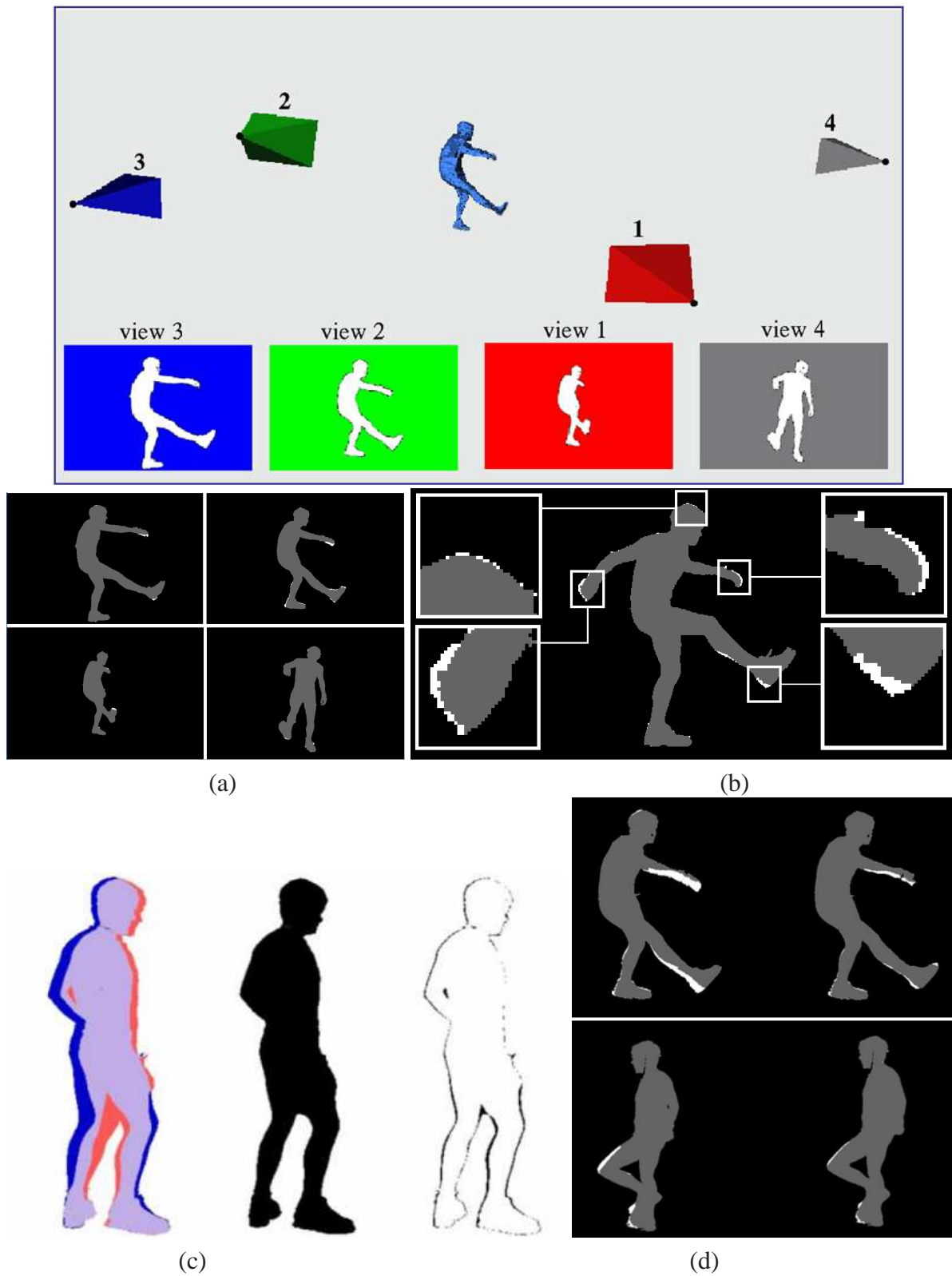
Figure 4.5: Metric 3D reconstructions of the MIT sequences. (a) The visual hull projected over the original silhouettes. (b) A close-up showing the re-projection errors. (c, d) Silhouette interpolation using the sub-frame synchronization reduces such re-projection errors.

first and the third and compared it to the true second frame. Our approximate interpolation approach works quite accurately for small motion, as would be expected in video captured at 30 frames per second. In Figure 4.5(e), the visual hull reprojection error is shown, with and without sub-frame silhouette interpolation. In the two cases, the reprojection error decreased from 10.5% to 3.4%, and from 2.9% to 1.3% of the pixels inside the silhouettes in the four views. In the future, more sophisticated shape-preserving interpolation schemes such the ones proposed by [1] will be investigated for higher accuracy.

4.4 Conclusions

This chapter concludes the first part of the thesis. We have presented a complete approach to determine the time varying 3D visual-hull of a dynamic event, from silhouettes extracted in multiple videos recorded using an uncalibrated and unsynchronized network of cameras. The key element of our approach, is a robust algorithm that efficiently computes the temporal offset between two video sequences and the corresponding epipolar geometry. The proposed method is robust and accurate and allows calibration of camera networks without the need for acquiring specific calibration data. This can be very useful for applications where sending in technical personnel with calibration targets for calibration or re-calibration is either infeasible or impractical. We have also shown that for visual-hull reconstructions from unsynchronized video streams, sub-frame silhouette interpolation can improve the quality and accuracy of the reconstructed 3D shape.

Part II

Multi-view Reconstruction of Static Objects

CHAPTER 5
Multi-view Stereo Reconstruction

5.1 Introduction

Recovering the 3D shape of a static object from multiple images has been a classical prob-
lem addressed in the computer vision literature. Given multiple images of the same scene
from different viewpoints and the calibration of the corresponding cameras, the goal is to
reconstruct a 3D model of the scene as accurately as possible. A specific case of the 3D re-
construction problem that has been investigated in detail is the problem of recovering a 3D
model of a closed, compact object. In practice, the object is often placed on a rotating turn-
table and a single camera captures images at roughly uniform intervals. This simulates a ring
of cameras surrounding the object. As discussed in Section 1.2.2, it is possible to exploit a
variety of visual cues to design a 3D reconstruction algorithm. Amongst these, the stereo and
silhouette cues are the most widely applicable and powerful for modeling solid objects. The
class of shape from silhouette methods were reviewed earlier in Section 4.2 and a robust and
efficient algorithm for computing exact polyhedral visual hulls [41] was described. In this
chapter, multi-view stereo methods will be described as well as some recent techniques that
try to fuse stereo and silhouette information together for 3D reconstruction. In this setting, the
motivation and scope of the approach proposed in the second part of this thesis will be laid
out.

5.2 Multi-view Stereo

The basic idea in multi-view stereo is to use image appearance (i.e. color or texture) to establish dense correspondence between pixels in different calibrated views. Dense binocular or multi-baseline stereo involves recovering matching pixels (pixels corresponding to the same 3D point) in two or more views. The depth of the 3D scene can then be recovered by triangulating corresponding pixels (i.e. intersecting back-projected rays) in different views. Often it is difficult to find accurate and robust correspondences for all pixels because of ambiguities in matching for textureless surfaces or the matching fails due to occlusions or non-Lambertian properties of the surface.

Stereo algorithms represent the scene structure or object's shape using disparity maps (or depth maps). Every pixel $p_1(i, j)$ in the first image has a particular disparity d with respect to the matching pixel $p_2(i + d, j)$ once the images have been rectified (i.e. they have been transformed to ensure that corresponding pixels are always on identical scanlines). In the multi-view case, a *depth-map* is commonly used to represent the correspondence or the depth from a particular viewpoint. The problem of robustly computing an accurate disparity map can be formulated as a pixel labeling problem. Some of the best stereo methods solve the pixel labeling problem by energy minimization which make use of recently proposed discrete optimization algorithms such as graph cuts [16, 14] and loopy belief propagation [155]. Later in this chapter, we will review graph-cut based energy minimization on which our 3D reconstruction method is based.

5.2.1 Photo-consistency

While stereo approaches that compute depth-maps are well suited for reconstructing general scenes, volumetric or surface based methods are more popular when it comes to reconstructing objects. Irrespective of the underlying geometric representation, all these methods use the fundamental idea of *photo-consistency* first proposed by [118] in the context of a volumetric

75

approach called *voxel coloring*.

Photo-consistency is a function that how measures the likelihood of a 3D point of being on a opaque surface in the scene. This likelihood is computed based on the images in which this 3D point is potentially visible. Suppose a 3D point $\mathbf{M}$, visible in cameras $c \in \mathbb{V}_{\mathbf{M}}$ projects to pixels in these images with colors $C_{\mathbf{M}}^{c}$ respectively, then the photo-consistency of $\mathbf{M}$ can be estimated by evaluating the color variance of the pixels in $\{C_{\mathbf{M}}^{c}\}$. An ideal Lambertian surface point will appear to have the same color in all the images and will thus have a color variance of zero in the absence of noise.

The color variance measure is an over simplificaton and not an adequate measure for real scenes. Thus instead of a single pixel, the appearance of a small neighbourhood around the projected point is often considered. Patch-based similarity measure such as normalized cross correlation (NCC), described in Appendix A-2 can be used to compare the appearance of these patches. Various ways of computing photo-consistency have been reported [17, 64, 127] and approaches for non-Lambertian surfaces such as [154] have been investigated as well.

Photo-consistency can be measured in image space or object space. Image space computations compare image patches centered at the pixels where the 3D point projects. This implicitly assumes that the images are approximately rectified or the corresponding camera pair has a small baseline. Object space computations are more general – a patch centered at the 3D point is projected into the images and the appearance of the projected patches (recovered by bilinear interpolation) are compared. Approximate knowledge of the surface orientation can be used to choose a suitable orientation for this patch. Photo-consistency cannot be computed exactly because it requires knowledge about the visibility and this in turn requires knowledge of the 3D shape. Most approaches either approximate the visibility [85, 148] or compute robust versions of the photo-consistency function that treats occlusions as outliers [149].

Volumetric 3D reconstruction methods that maximize photo-consistency roughly fall into two groups – (a) greedy carving approaches which try to recover maximal photo-consistent

shapes such as the *photo-hull* [78] or (b) energy-based global methods that first evaluate photo-consistency within a volume and then search for a surface with the highest overall photoconsistency. The greedy carving methods [78, 118] make binary hard decisions locally and cannot enforce regularization or spatial coherence. The results depend on a critical threshold parameter which is used to classify voxels as being photo-consistent or not. This is problematic because the image data is almost always ambiguous and incorrect decisions made at individual voxels cannot be corrected later.

5.2.2 Global methods

Global methods which employs energy minimization imposes regularization or spatial coherence as a soft constraint and are better suited for solving the ill-posed 3D reconstruction problem [13, 14, 38, 77, 85, 148]. These methods formulate scene reconstruction as a variational problem in which a suitable photo-consistency based energy functional is optimized. Various ways of solving the variational problem has been investigated – deformable meshes [31, 57, 66], level-sets [38] and graph-cuts [13, 77, 148].

Level-sets [119] are a popular method to minimize functionals such as $\int_S \rho(s)ds$ that can be represented as a weighted minimal surface S. An implicit function $\phi(s, t)$ is constructed such that the time-evolving surface $S(t)$ is represented by the zero level-set of $\phi(s, t)$ i.e. $\phi(S(t), t) = 0$. The surface evolves in a way to minimize the weighted surface functional. Using the Euler-Lagrange formula for this variational problem, a partial differential equation is constructed which drives the evolution of $\phi(s, t)$. Although this method minimizes a global objective function, the PDE is based on local differential operators. This technique was used by Faugeras et. al. [38] to solve the 3D reconstruction problem where the $\rho(s)$ function was based on patch-based photo-consistency.

While level-sets and deformable meshes can enforce smoothness, they are iterative and only guarantee a local minima and thus they require good initialization. On the other hand,

graph cut optimization is attractive because it can compute the global minima of the energy functional in many scenarios. However, the reconstruction results from graph-cuts often suffer from metrication errors due to the underlying discretization. This was addressed in [13] but in practice, graph cut results are never completely free of metrication errors. Although graph-cut optimization is reasonably fast, they typically require evaluating the photo-consistency over the whole volume, due to which the computational and memory requirements of this method are quite high.

5.2.3 Combining Stereo and Silhouette Cues

While the photo-consistency or stereo cue provides strong constraints on surface locations for near fronto-parallel surface patches, shape from silhouette techniques provide constraints for surface patches that are tangential to the viewing rays. Hence methods that combine the two cues show benefits from combining the complementary information.

Some of the earliest methods that combined silhouette and stereo constraints were [31, 94]. Both started by carving away voxels using silhouette information followed by carving based on photo-consistency. This is possible because the visual hull is guaranteed to contain the real surface within itself. A mesh deformation approach was proposed by [66]. It deforms the visual hull towards a photo-consistent solution by moving mesh vertices via a series of random searches along epipolar lines. Hernandez et.al. [57] too created an initial mesh from the visual hull which was then deformed to satisfy photoconsistency constraints under the effect of gradient flow forces [153]. Their mesh deformation approach models silhouette constraints as well but tends to produce a bias near the visual hull boundary. A level-set based approach that fused silhouette and photo-consistency constraints was proposed by [110]. Although these approaches show accurate results, they are susceptible to local minima and may require good initialization in the form of a visual hull that well approximates the true shape. Our work is most similar to Carved Visual Hulls [45] which also enforces some silhouette

constraints exactly and this is reviewed later in Section 5.4.2.

5.3 Energy Minimization

A large number of computer vision problems try to assign a finite set of labels to pixels based on noisy measurements. The labels could be intensity values, disparities or segmentation labels depending on the target problem. Under an optimal labeling, the assigned labels often tend to vary smoothly within the image, except at the boundaries where discontinuities typically occur. As a particular pixel label depends on the labels of its neighbours, the Markov Random Field (MRF) is a natural representation. In the MRF framework, each pixel becomes a node of the MRF and a regular grid structure is imposed on the pixels based on either a 4-connected or a 8-connected neighborhood system. In the presence of uncertainties, finding the best pixel labeling becomes an optimization problem over a MRF.

Pixel labeling problems can be posed in terms of energy minimization, where the energy function (or functional) contains two terms: a data term that penalizes solutions that are inconsistent with the observed data, and an interaction term that enforces spatial coherence over the whole solution. Thus the data energy is simply $E_d = \sum_p D_p(l_p)$ which is a sum of per-pixel data costs (the penalty associated with assiging label l_p to pixel p). The interaction (or smoothness) energy term is $E_s = \sum_{\{p,q\}\in\mathcal{N}} V_{pq}(l_p, l_q)$. Here $\mathcal{N}$ denotes the set of all unordered pairs of neighboring pixels and $V_{pq}(l_p, l_q)$ denotes the penalty of assigning labels l_p and l_q to neighboring pixels p and q respectively. Thus the energy functional that is minimized is as follows:

$$E(L) = \sum_p D_p(l_p) + \sum_{\{p,q\}\in\mathcal{N}} V_{pq}(l_p, l_q) \qquad (5.1)$$

These energy minimization approaches are well justified, as they produce the maximum a posteriori estimate of an appropriate MRF [16]. In fact the data energy is proportional to the log likelihood of the measurement noise whereas the interaction energy is proportional to the

log-likelihood of the prior as pointed out by [139].

The interaction energy is typically of the form $E_s = \sum_{\{p,q\}\in\mathcal{N}} w_{pq}.V(|l_p - l_q|)$ where $V()$ represents a monotonically increasing function of label difference. The Potts model is commonly used as it allows discontinuity preserving smooth label assignments but penalizes any pair of dissimilar labels equally. Here $E_s = \sum_{\{p,q\}\in\mathcal{N}} w_{pq}.I(|l_p \neq l_q|)$ where $I()$ is the indicator function. The choice of the interaction term is critical and under some common choices the energy minimization becomes intractable. Minimizing the Pott energy model or any other energy with discontinuity preserving smoothness terms under more than two labels is NP-Hard. However, fast approximation algorithms such as α-expansions [16] can compute a provably good approximate solution by iteratively running max-flow algorithms over the range of labels. In this thesis, we will only be concerned with binary energy functionals which can be efficiently minimized irrespective of the choice of the interaction term. Although many different approaches for energy minimization exist in the literature, we will concentrate on graph-cuts, as they guarantee finding the global minima for the class of binary energies we will be concerned with.

5.3.1 Graph-cuts based Energy Minimization

For solving the 3D reconstruction problem, we will be interested in energy minimization over a MRF where every vertex in the graph corresponds to a voxel in a bounding volume of the shape. We will be interested in binary energy functionals – i.e. we seek to label each voxel as interior or exterior. Each partition of the set of voxels corresponds to some surface and we will be interested in optimizing such a labeling under a set of constraints derived from the silhouette and stereo cues. There is a natural correspondence between such a particular pixel labeling and a cut on a flow graph G. It can be proved that the minimum cut on G produces the partition or labeling that is the global minimum of the energy function. Computing the minimum cut is equivalent to finding the maximum flow on the flow graph which can be computed
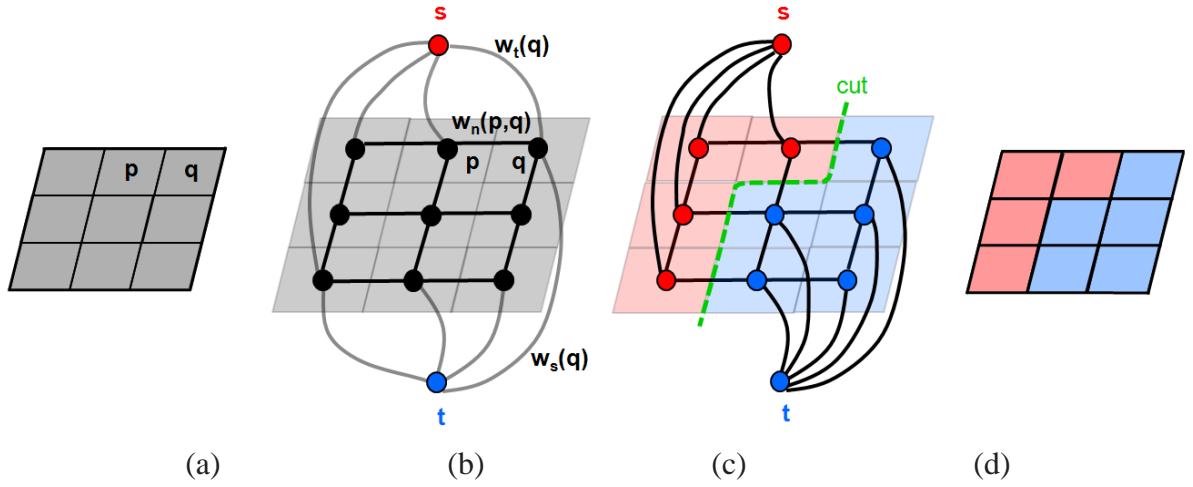
Figure 5.1: (a) a 3×3 image where p and q refer to neighboring pixels. (b) Graph construction showing the n-links, t-links and the two terminals s and t respectively. (c) Every s-t cut produces a certain labeling and (d) the minimum cost cut produces the optimal solution to the pixel labeling problem.

in polynomial time. Thus in the two labels, the global minima of the energy function can be obtained in polynomial time directly (see Appendix B-3 for an overview on max-flow/min-cut algorithms). We will now describe the construction of the graph G for an arbitrary binary energy function. The graph G will contain two kinds of vertices: *p-vertices* (these correspond to the pixels or voxels which are the nodes in the associated MRF) and *l-vertices* (these are the terminal vertices). This is illustrated in Figure 5.1. In the two label case, there will be two such l-vertices. In Figure 5.1(b) these are denoted by s and t respectively (s and t stands for source and sink respectively). All the edges present in the neighborhood system $\mathcal{N}$ of the MRF become edges in G. These edges are called *n-links*. Edges also exist between the *p-vertices* and the terminal *l-vertices*. These are called *t-links*. t-links are assigned weights based on the data terms of the energy functional, while n-links are assigned weights based on the interaction or smoothness term. To be more specific, the weight of the t-link connecting node p to s is $D_p(l_p = t)$ while the weight of the t-link between p and t gets the weight $D_p(l_p = s)$ respectively. All n-links typically get a constant weight λ which decides a trade-off between the data-penalty and smoothness penalty. The minimum cut severs some of the t-links as well

as some of the n-links. This leaves some of the p-vertices connected to the source, while the rest remain connected to the sink. This partition induced by the minimum-cut produces the most optimal binary labeling that has the minimum energy.

Graph-cut based energy minimization was initially done on 2D regular grid where the optimal labeling was often spatially coherent. However, the label boundaries were not geometrically smooth contours or surfaces. In the context of segmentation, it was found that graph-cut techniques produced large *metrication errors* because the underlying energy that was being minimized by graph-cuts did not enforce geometric smoothness of the boundaries between labels.

Boykov et. al. [13] studied cut metrics associated with graph-cuts. They showed how that the minimum cut cost can approximate any non-Euclidian metric (in R^n) using a specific type of graph construction. They proposed the *geo-cut* algorithm [13] to address the problem of *metrication errors* and showed how to use graph cuts to compute geodesic contours in 2D or minimal surfaces in 3D metric spaces in the context of object segmentation in 2D or 3D. The main advantage of geo-cut (graph-cut method) over other segmentation techniques (snakes or level-sets) lies in their ability to compute globally optimal solutions for energy functionals defined on N-D grids. Finding geodesics and minimal surfaces via level-set methods is a common approach that was popular in the past. The geo-cut algorithm can find globally optimal solutions which avoids problems with local minima. This is an important result as it shows that graph-cuts are indeed a good choice to solve the kind of variational problems we are interested in.

Although the 3D multi-view reconstruction problem is similar to 3D segmentation i.e. it involves labeling the voxels in the grid into interior and exterior ones, there are some differences. In segmentation, typically region-based information is used to associate data penalties with every voxel in the grid. In 3D reconstruction, the data-cost is derived from the photo-consistency function which gives a likelihood of the location of the surface i.e. the interface

82
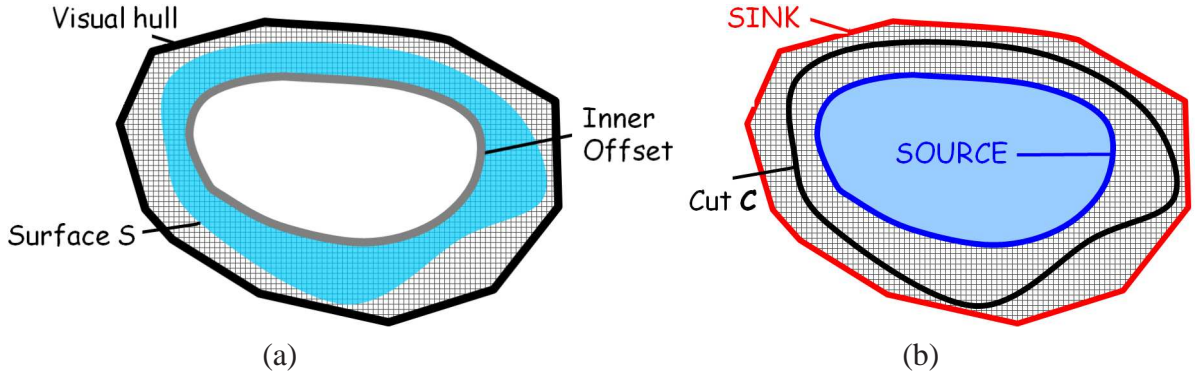
Figure 5.2: (a) A 2D slice of the volume showing the visual hull (base surface) and the inner offset surface between which the surface is assumed to lie. (b) The offset surfaces are connected to the source and the sink respectively and the minimum cost cut produces the most photo-consistent surface.

between the interior and the exterior. In the absence of region based terms, the t-links do not have any associated weights, rather the photo-consistency costs are incorporated into the weights of the n-links. In order to force the cut to go through specific parts of the volumes, hard constraints can be incorporated by including t-links with very large weights. Having a t-link with an infinitely large weight ensures that this edge is never severed by the minimum-cut. This is equivalent to saying that certain nodes in the graph or MRF have pre-determined labels i.e. they are connected to either the source or the sink.

5.4 Closely Related Work

We now review some recent multi-view stereo methods [45, 148] and analyze them in some detail as our proposed method is closely related to these.

5.4.1 Volumetric Graph Cut Stereo

The work of [148] extends the Riemannian minimal surface idea of [13] for multi-view volumetric stereo. The method uses an approximate base surface in the form of the visual hull of the scene by assuming that the true surface will be between the base surface and a parallel

inner boundary surface. In their graph-cut formulation, the nodes of these two base surfaces are connected to the source and sink respectively using very high edge-weights. This is illustrated in Figure 5.2. For computing the photo-consistency of every voxel that lies between the two base surfaces, each voxel is assumed to have the same visibility as the nearest point on the visual hull. The minimum cost cut on this graph that separates the source from the sinks corresponds to the most photo-consistent surface S which lies between the two base surfaces. This surface minimizes the energy functional $E(S) = \int_S \rho(x)dS$, where $\rho(x)$ is defined as

$$\rho(x) = 1 - exp(-tan(\frac{\pi}{4}(c(x) - 1))^2/\sigma^2)$$

and $c(x) \in [-1, 1]$ is a measure of photo-consistency. The $\rho(x)$ function is a sigmoid which maps a photo-consistency score to an individual term in the energy function ($c(x) = 1$ means high photo-consistency produces an energy term $\rho(x) = 0$). Surface smoothness is implicitly enforced since minimising $E(S)$ corresponds to finding the minimal surface with respect to a Riemannian metric. Larger values of σ produces surfaces that are less smooth but which passes through more photo-consistent points.

While the visual hull is used to generate the base surfaces for this method, the silhouette constraints are not enforced during the graph-cut optimization and thus there is no guarantee that the final solution will strictly satisfy the silhouette constraints. In fact, in most cases it won't. The approach also suffers from a *minimal surface bias*, a common problem with most global methods. Since these methods attempt to minimize a surface integral, they implicitly prefer solutions which have smaller areas. For the 3D reconstruction problem, this amounts to reconstructed surfaces that have shrunk and occupy less volume than they actually should. Also sharp protrusions or deep concavities will be missing because the imposed regularization tends to flatten the minimal surface. A ballooning term (region based term) was used by [148] in their energy functional to address this problem. The ballooning term gives a preference for larger shapes. See Figure 5.3 for an illustration. The corresponding energy functional that

84

(a)   (b)
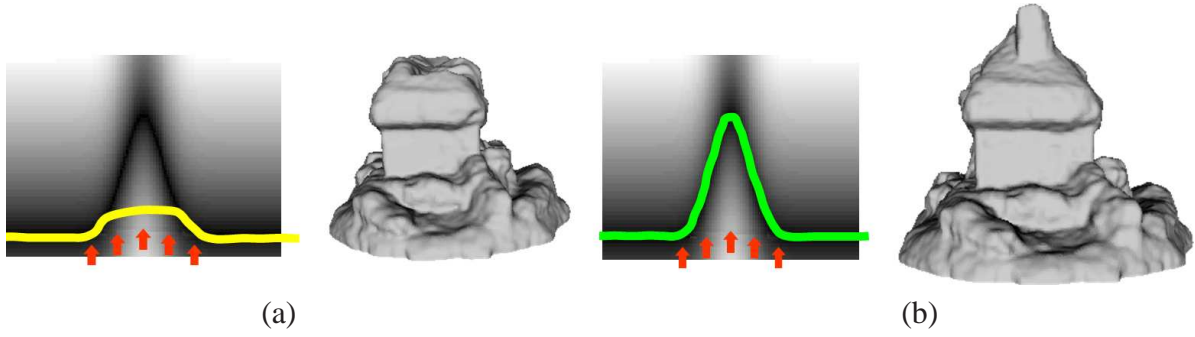
Figure 5.3: (a) Volumetric graph cuts suffer from a minimal surface bias. A slice through the photo-consistency volume is shown. (b) The problem is solved by using a ballooning term in the energy functional which is a prior for larger volumes. This illustration is taken from Vogiatzis et.al. [148].

is minimized is $E(S) = \int_S \rho(x)dS - \int_V \lambda dV$ where V is the volume enclosed by the surface. The new energy functional has an additional volumetric (also referred to as ballooning) term which favors larger shapes. However this naive regularization reduces the overall accuracy of the reconstruction because in order to recover thin protrusions, the deep concavities must be filled out as well. In our approach, silhouette constraints will be used to recover thin protrusions whereas additional visibility based information will be used to recover the concavities accurately. While silhouette constraints were used in the deformable mesh as well as level-sets based approaches, it is not obvious how to enforce them within the graph-cut based technique. This is the motivation of our work but first, the closely related work on Carved Visual Hulls [45] is reviewed next.

5.4.2   Carved Visual Hulls

The algorithm proposed by Furukawa and Ponce [45] starts by computing the exact visual hull mesh using the approach of [83]. First dynamic programming is performed on the cone-strips of the visual hull to recover segments of the rim curve. Then with the points along the rim curves held fixed, the visual hull is carved using graph-cuts (similar to [13, 148]) to globally maximize the photo-consistency of the surface. At this stage a watertight triangulated mesh

is reconstructed and this mesh is optimized further using local mesh deformation to recover more finer surface detail.

Excellent results have been reported although in all these cases the visual hull already approximates the true shape quite well. One weakness which we try to address in our work is the early commitment to rims which are identified in the first step after which they are held fixed while computing the surface during optimizing photo-consistency. In this method, when rim points are incorrectly identified, they cannot be corrected later during the graph-cut step and this early commitment can be dangerous in ambiguous situations. In contrast to this multi stage approach, our graph-cut formulation with exact silhouette constraints is designed to directly extract the rims as well as the full surface in one optimization step. Similar to [45, 57], we too perform a subsequent local refinement of our mesh using a combination of texture, silhouette and smoothness forces to recover greater degree of surface detail in our 3D models.

Both volumetric graph-cut stereo and carved visual hulls successfully use the graph-cut optimization technique but both of them require pre-computing photo-consistency on a very finely divided regular lattice grid. The cost of evaluating photo-consistency at all these nodes dominates the running time of the graph-cut optimization step. In our work, we propose an adaptive graph construction which addresses this problem. We propose a way to adaptively sample the volume that avoids computing photo-consistency where its not needed. The proposed approach allows us to reconstruct detailed geometric models from high resolution images which would have been impossible with the existing approaches because of a memory bottleneck. This adaptive framework can utilize soft silhouette constraints (when approximate silhouettes are available) but enforces them in a more general way without enforcing exactness.

5.4.3 Surface Growing Approach to Multi-view Stereo

While the two approaches described above perform some form of global optimization using a volumetric representation, recently a different approach to multi-view stereo has been explored by [44, 51, 54]. These methods first generate sparse correspondences between the multiple images. Next, a local region growing or surface growing strategy is used to iteratively build up a dense surface. Surface growing approaches use a patch-based representation of the surface. Patches or *surfels* are planar disks that are independently estimated such that their projections in the images where they are visible, match with one another. Instead of optimizing a global energy functional to enforce smoothness during reconstruction, the smoothness prior is used to make a local planarity assumption about the surface during the surface growing stage. A dense reconstruction may be obtained by multiple iterations of expanding *surfels* in the tangent plane of an existing *surfel* and refining the newly created patches.

To produce a compact watertight surface, these approaches must be combined with a surface-fitting approach such as [72]. Visibility constraints are used by [44] to perform filtering during the iterative surface growing stage. Currently, it is the best performer amongst the various multi-view stereo approaches in the Middlebury multi-view stereo benchmark [117].

5.5 Conclusions

In this chapter, we reviewed the state of the art in multi-view stereo and background in graph-cut based energy minimization and how it can help with solving the multi-view stereo problem. We discussed some of the limitations of the existing methods and motivated how silhouettes can be used to address these issues. Chapter 6 will describe our work on a graph-cut formulation for multi-view stereo that incorporates silhouette constraints exactly while Chapter 7 describes an adaptive graph construction that addresses the memory and computational bottleneck of the volumetric graph-cut algorithms.

CHAPTER 6
Multi-view Stereo with Exact Silhouette Constraints

6.1   Introduction

In this chapter, we describe a graph-cut formulation for volumetric multi-view stereo that strictly enforces silhouette constraints. This implies that in addition to being photo-consistent, the final reconstructed surface is constrained to exactly project into the original silhouettes. Existing techniques that fuse silhouettes and photo-consistency, treat these cues as soft constraints within a global optimization framework – these methods cannot ensure that the reconstructed shape be fully consistent with the original silhouettes. In our formulation, a special flow graph is constructed such that any cut that separates source and sink nodes in the flow graph, corresponds to a surface that exactly satisfies silhouette constraints. Thus silhouette consistency is guaranteed by construction and is not part of the optimization problem. Amongst the multiple cut surface candidates, the minimum cost cut corresponds to the optimal solution (the surface with minimum energy) that maximizes photo-consistency and smoothness.

Our graph-cut formulation is based on the topology of exact visual hulls [41, 83], which is based on a combinatorial mesh of cone-strips obtained by the cone-cone intersections of back-projected viewing cones from all the silhouettes. We proposed a preliminary version of this approach in [124], which was based on the concept of the rim mesh proposed earlier by Lazebnik et. al. [82] in the context of exact visual hulls. The rim mesh (similar to the idea of *epipolar nets* proposed by [31]) captures information about the combinatorial arrangement of rim curves (i.e. contour generators) on the actual surface. The location of the rim curves

is also recovered by this method. However the proposed approach is complicated and cannot handle complex geometric shapes. This is because the rim mesh for complex geometric shapes is unstable or difficult to compute. The approach described in this chapter is an extension of our previous work [124], but can handle more complex shapes. The approach presented here does not need the rim mesh, but recovers all the related information directly from the topological information encoded in the visual hull mesh. We have tested it on various real datasets involving complex shapes.

In this chapter, we start by explaining the main idea using a simple 2D example. Our formulation is based on a 2-coloring property of the visual hull which is explained next. An algorithm for 2-coloring the visual hull is then described. The 2-coloring is the primary mechanism for incorporating silhouette constraints within the graph cut step in our proposed formulation. The formulation is first presented in a form that correctly deals with only convex objects observed by multiple cameras in general position. Finally, we show how to deal with non-convex shapes which requires correctly handling T-junctions on silhouettes.

6.2   Main Idea

The basic idea behind our approach is that the visual hull polyhedron which is formed by a union of cone strips, captures important information about the geometry of the visible parts of the rim curves on the true surface. The rim curves partition the surface into patches, each of which is constrained to lie within the visual hull. Furukawa et. al. [45] first recover the geometry of the rims and then reconstructs the patches while holding the rims fixed. In contrast, the surface and the rims are recovered in a single step in our approach. We construct a new topological space (a 3-manifold) by taking subsets of the volume inside the visual hull and gluing them along certain faces of the visual hull polyhedra. The exact topology of this 3-manifold is derived from the geometry of the visual hull polyhedron. We then construct a geometric graph embedded within a discretized version of this 3-manifold. Hard silhouette
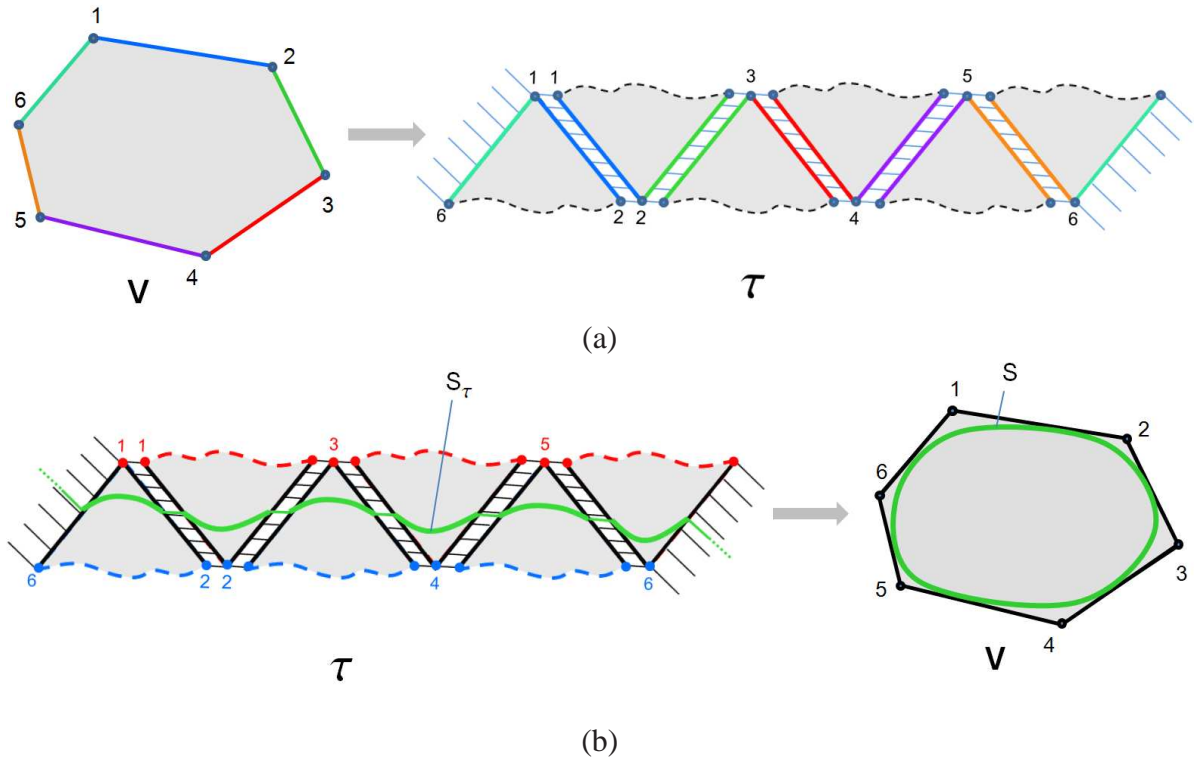
Figure 6.1: The main idea of our formulation is illustrated here (best seen in color). (a) A 2D visual hull is shown – V denotes its interior. Vertices 1–6 represent the vertices of the visual hull. A new topological space (a 2-manifold in the 2D case) can be created as follows. Multiple copies of the underlying 2D space are glued together (topologically identified) along certain edges of the visual hull polygon as shown. The small horizontal edges represent the topological identification. Note that the space wraps-around. (b) A discrete geometric graph embedding within this space is then constructed. A graph-cut problem is setup with source vertices (shown in red) and sinks vertices (shown in blue). Any s-t cut on this graph must cut through the edges (1–2, 2–3 . . .) as shown. Thus, it will map back to a surface in the original space that is guaranteed to satisfy silhouette constraints.

constraints are imposed by connecting specific vertices in the flow graph to the source and the sink on the basis of a 2-coloring scheme that is described later. Computing the minimum cut on this special flow graph will produce a 2-manifold embedded within the 3-manifold we have constructed. This minimum cost cut surface uniquely maps back to a surface embedded in the original space. We show that by construction, this surface will exactly satisfy all silhouette constraints. This idea is illustrated in Figure 6.1 using an example in 2D (flatland).

In order to describe the graph construction, we will first review relevant concepts related
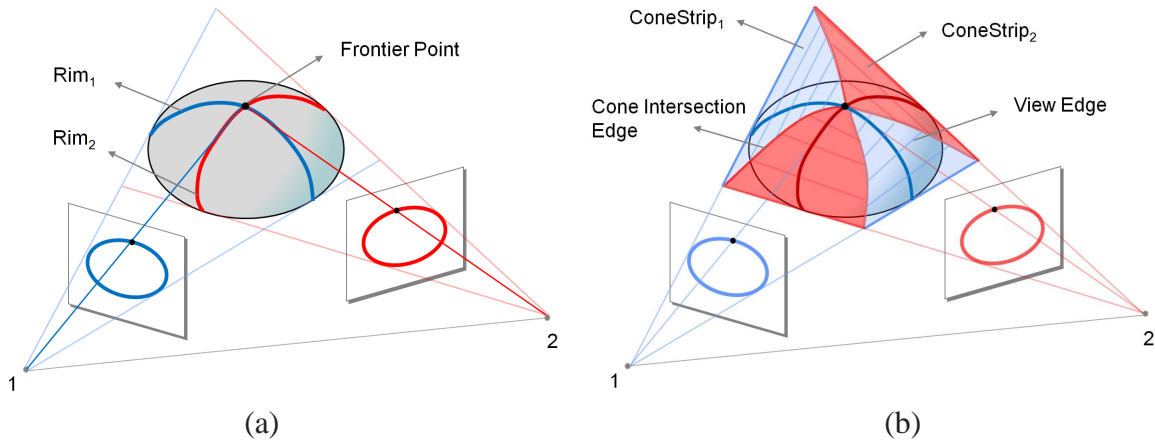
Figure 6.2: (a) Geometry of silhouettes in two views. (b) The visual hull obtained by intersection of two cones. Frontier points, rims, cone-strips and cone intersection curves are shown (best seen in color).

to the the geometry of exact visual hulls. For more details, please refer to [41, 83].

6.3 Exact Polyhedral Visual Hull

The visual hull is the maximal shape that projects consistently into a set of silhouettes, and is obtained by intersecting silhouette cones from the corresponding calibrated viewpoints. Visual rays from a camera which grazes the true surface tangentially give rise to a smooth continuous curve on the true surface called the *rim* or the *contour generator*. Its projection in the image gives rise to the apparent contour. For non-convex shapes, depending on the camera viewpoint, the *rim* can occlude itself. This gives rise to singularities on the apparent contour. Thus silhouettes of opaque, non-convex solid shapes are likely to contain T-junctions.

Rims from different cameras intersect on the surface at points called frontier points (these were discussed earlier, in the context of silhouette-based camera calibration, in Section 2.2.1). Discretization of the apparent contour or calibration error gives rise to missing frontier points on the visual hull polyhedra, due to the problem of *lost tangency* – i.e. the perfect cone tangency is lost and one cone ends up clipping the other one. This is illustrated in Figure 6.3.

The contribution from a single viewing cone to the visual hull mesh can be parameterized
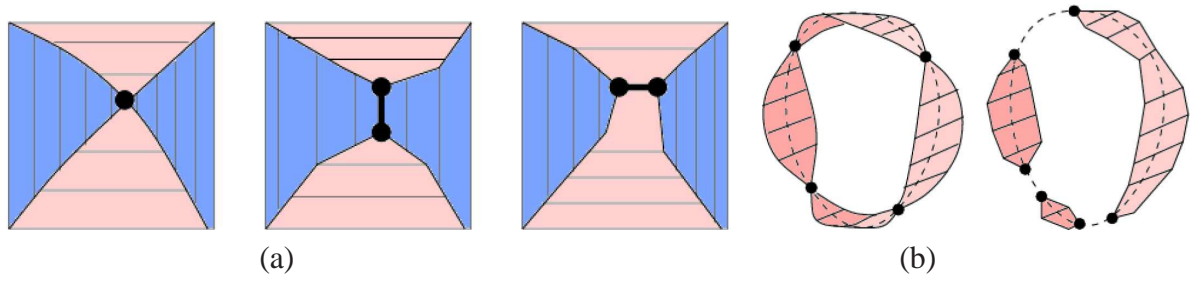
Figure 6.3: (a) Close-up of a frontier point and the problem of lost-tangency (b) An ideal cone-strip for perfect data compared to a cone-strip from the exact polyhedral visual hull in the presence of silhouette discretization and calibration noise (best seen in color).

as a ruled surface, and is called a *cone-strip*. The geometric shape of the cone-strip provides an interval constraint on the position of the associated rim curve. Intuitively, the shorter the view edge in the cone-strip, the closer the visual hull is to the true surface, locally. While the rim curve mostly lies on the cone-strip, it detaches from the cone-strip at a point on the viewing ray that corresponds to a point on the T-junction of a silhouette. The detached portion of the rim lies inside the visual hull volume, not on its surface. This portion of the rim curve is occluded by the object shape from the particular viewpoint.

When points sampled on the apparent contour are back-projected in 3D, they give rise to viewing rays, each of which may contribute a view edge segment or multiple segments to the visual hull polyhedron. At least one point on the view edge segments generated by a single viewing ray must lie on the true surface. In fact, this point lies exactly on the rim curve. When viewing cones intersect, they give rise to *cone intersection curves*. For the polyhedral visual hull [41], the cone-intersection curves are approximated by a series of *cone-intersection edges*. In fact, there are special vertices on the visual hull mesh which corresponds to the intersection of three viewing cones – these are called *triple points*. All points on the cone-intersection edges lie completely outside the true surface, although they may get infinitesimally closer to the true surface. All edges in the visual hull polyhedron are either viewing edges or cone intersection edges. Similarly, all the vertices of the visual hull polyhedra must be either triple points or vertices that form the end-points of viewing edges. Note that, all vertices of the

visual hull also lie outside the true surface, even though they may get infinitesimally close to it.

6.4   Silhouette Formation and 2-Coloring

Consider a closed manifold surface which is seen in k views. The apparent contour observed in each of these views corresponds to a continuous rim curve which is topologically a closed loop. In case of shapes with non-zero genus, the apparent contours may contain holes in which case the jth rim curve will have multiple loops. The arrangement of the closed curves that correspond to the k sets of rim curves on the surface partitions the surface into connected regions. This will be referred to as a *surface map*. See Figure 6.4 for an example. We now prove an important property of surface maps – a result that will be used in the graph construction, described later in this chapter.
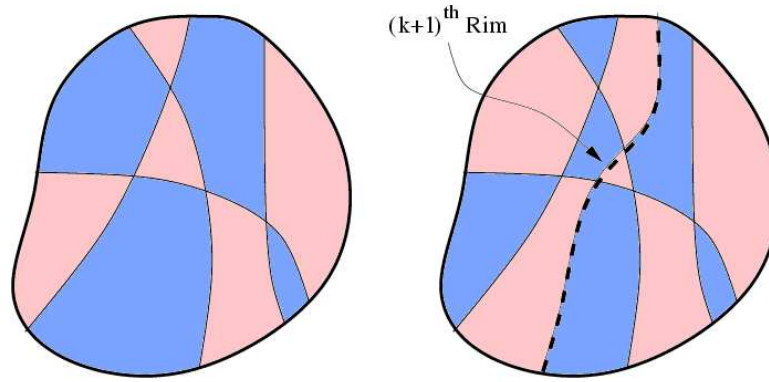
Figure 6.4: Two-coloring the surface (best seen in color). (Left) The 2-colorable surface map induced by k-rims. The color of the surface changes everytime you cross any of the rim curves. (Right) A new surface map is created after adding the $(k+1)^{(th)}$ rim (shown by a dotted line). The color of patches on one side of the new rim have been swapped and patches on the other side remain untouched. The new surface map is also 2-colorable (see proof).

Lemma 6.4.1. *A surface map M_k, induced by the rims from k views can always be 2-colored.*

Proof: We prove this by induction. The rim curve or curves (when holes are present in the silhouette) divides the surface into two parts: front and back with respect to the camera.

Let us label them with two different colors. Thus, M_1 can be *2-colored*. Assuming M_k has been *2-colored*, we must prove that M_{k+1} can also be *2-colored*. After adding the $(k+1)^{th}$ rim to M_k, we swap the colors of its front faces in the new map, M_{k+1}, but leave the back faces untouched. This will produce a 2-coloring of the newly created faces in M_{k+1}, which is consistent with that of the old faces, unchanged from M_k. Thus indeed, M_{k+1} can also be *2-colored*. $\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad\square$

The 2-colorable surface map that we described could be transferred to the visual hull surface by a projection as shown in Figure 6.5. However, when the shape of the surface and the position of the rims are unknown, the exact mapping cannot be determined. In spite of this, it is possible to generate a 2-coloring of only the visual hull vertices and the cone intersection edges, that is consistent with the surface map. This is because a view-edge provides a finite interval for the rim curve. Therefore, the two vertices at its extremities can be consistently two colored even when the exact rim point is unknown. In the next section, we will propose an algorithm for 2-coloring only the vertices of the visual hull.

Even when the surface (and the rim curves) are unknown, the geometry of the exact visual hull mesh allows us to recover the topology of the unknown 2-colorable surface map. This fact is key to the proposed approach. Later in the chapter, we show how to use the two-coloring of the visual hull vertices to setup silhouette constraints in the graph-cut optimization.

In case of a convex object, the rim curves are never occluded and lie completely on the visual hull surface (actually on the corresponding cone-strip). For non-convex objects, parts of the rim curve may detach from the visual hull and lie inside its volume. We will refer to these as the *invisible* parts of the rim. Note that, the coloring of the surface map will change across the invisible portion of the rim curves as well. However, as these invisible rims do not actually touch the visual hull surface, the location on the visual hull surface where the coloring switches, can be chosen somewhat arbitrarily. An extension to the basic approach (for convex objects) is required and is described later in Section 6.7.
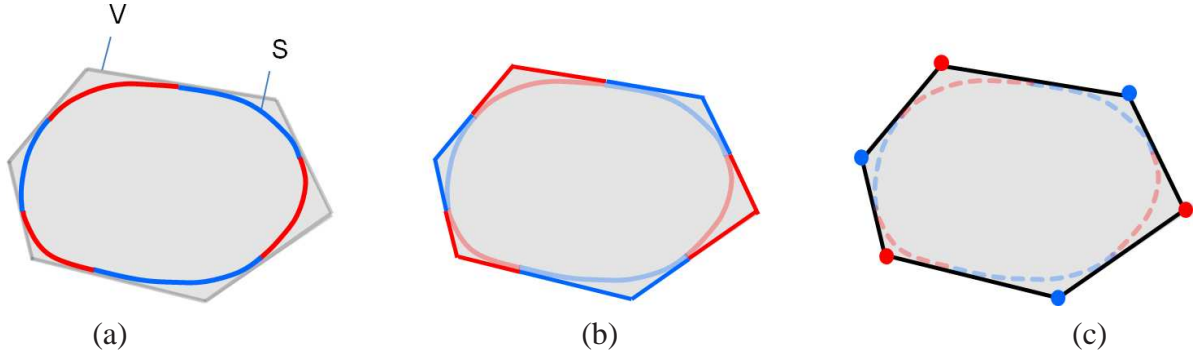
Figure 6.5: (a) 2-coloring of the surface S (best seen in color). The visual hull is denoted by V. (b) The 2-coloring can be transferred to the surface of the visual hull V. (c) When S (shown as dotted curve) and the position of the rims are unknown, only the vertices of the visual hull can be colored. The coloring of points on the viewing edges cannot be determined unless the position of the rims are known, because the coloring changes exactly at the rim point.

6.5   2-Coloring the Visual Hull

We now describe the algorithm for 2-coloring the vertices of the visual hull. It consists of the following four sequential steps, which will be described in this section.

1. Identify the *true* segments on each cone-strip.

2. Repair each individual cone strip.

3. Generate per-view orientation labels for all visual hull vertices.

4. Derive the final 2-coloring for all the visual hull vertices.

Identifying the true segments on each cone-strip

Consider the silhouette or viewing cone from a single view that contributes a cone-strip to the visual hull. Each viewing ray in this silhouette cone, contributes one or more view edges to the visual hull mesh. When a viewing ray gives rise to multiple view edge segments, only one of them contains the true rim point. This view edge segment will be referred to as the *true* segment, while the remaining ones will be referred to as *false* segments. The *true* segments can be detected by inspecting photo-consistency at points sampled on the view edge segments. The existence of photo-consistent points on a viewing edge segment indicates that
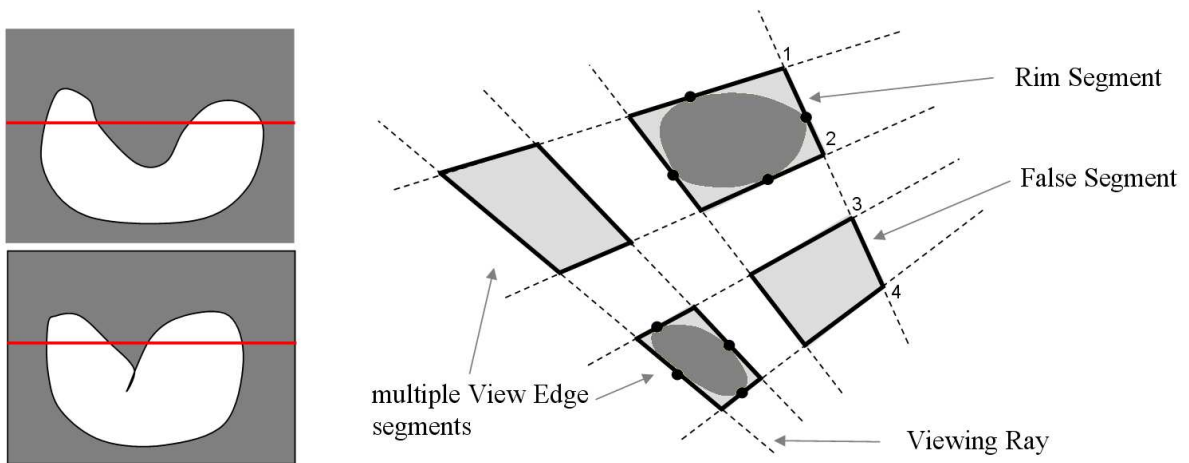
Figure 6.6: Silhouettes of an object seen in two views. A slice of the visual hull computed from two views is shown on the right (the slice in the epipolar plane corresponding to the red epipolar lines in the images). Notice how the viewing ray gives rise to multiple view edges, only one of which contains the true surface point. Such a view edge is called a *true* segment, while the other view edge generated from the same viewing ray is called a *false* segment.

the corresponding edge is likely to be a *true* segment, whereas *false* segments are unlikely to contain photo-consistent points. Although, the view edges for each viewing ray could be selected one by one, for greater robustness we globally extract the *true* segments for the whole cone-strip, using a dynamic programming approach. Specifically, we compute a shortest path on a graph that we will refer to as the *rim graph*. When multiple segments arise from a viewing ray, this optimization approach will classify exactly one of them as a *true* segment.

Furukawa et. al. [47] also used dynamic programming on the cone strip to recover the position of the rim curve. In doing so, they made an early commitment to the position of the rim curve which was held fixed during their subsequent graph-cut optimization which computes the final surface. In comparison, our approach makes little commitment at an early stage. Amongst the multiple segments, we select the one that is likely to contain the rim point. In fact, this is the only early commitment in our approach. In a generic situation, this has almost no impact on the final solution, as we get fewer false segments with more input silhouettes. This is because the additional silhouettes carve away major parts of the visual hull that contains the false segments. An early commitment to this choice of the *true* segment is
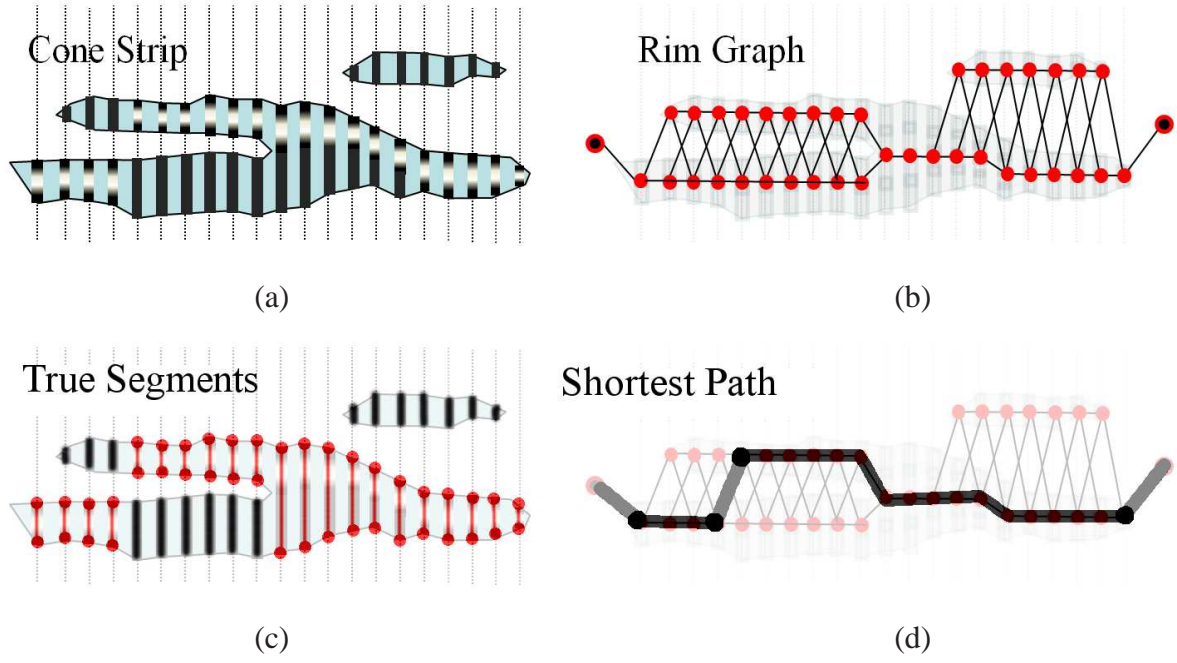
Figure 6.7: (a) A Cone Strip illustrated with photo-consistency along the view edge segments (white indicates surface points with high photo-consistency while black indicates low photo-consistency). (b) The corresponding Rim Graph. (c) The shortest path between two virtual terminals in the rim graph is computed. (d) This corresponds to the set of *true* segments on the cone-strip.

less dangerous than committing to the location of the rim curve (i.e. the position of the rim point on the *true* segment) [47]. Rather, in our approach, the rim points are recovered at the same time as the whole surface, via a single, global graph cut optimization.

The construction of the *rim graph* is now described. Each node in the *rim graph* represents a view edge segment. In addition to these nodes, two virtual terminal nodes are also created (see Figure 6.7(a,b)). Each node u is assigned a weight $w(u)$ equal to $1 - \rho(x)$ where $\rho(x)$ denotes the photo-consistency of the most photo-consistent point x on the corresponding view edge segment. To compute the photo-consistency $\rho(x)$ $(-1 \leq \rho(x) \leq 1)$, we first project a $\mu \times \mu$ patch placed at x, whose normal is orthogonal to the viewing ray, into at most k images (the normal vector is used to select cameras with small viewing angles ($\leq 60^o$). Next, the average pairwise normalized cross correlation (NCC) scores of the projected image patches is computed. In all our experiments, $\mu = 7$ and $k = 4$. Note that the underlying cone-

strip can be oriented, and this produces a cyclic ordering of the viewing rays. The edges in the *rim graph* are created by connecting nodes that correspond to view edge segments belonging to successive viewing rays. Each edge (u, v) in the rim graph is then assigned an edge weight, $w(u, v) = w(u) + w(v)$ when the view edge segments u and v are shared by a common face in the cone-strip surface. However, when u and v do not have a face in common, $w(u, v) = K$ ($K = 50$ in our experiments). The shortest path between the terminal nodes in the *rim graph* produces a set of *true* segments. Note that a larger value of K will produce a set of *true* segments, that correspond to a more continuous rim curve. The rim curve illustrated in Figure 6.7(c,d) has one discontinuity. Such a discontinuity occurs at a viewing ray, obtained by back-projecting a pixel at a T-junction on the silhouette.

Repairing each individual cone strip

Once *true* segments have been detected on the pieces of a cone-strip, the next step involves re-connecting patches of the cone-strip along a sequence of connected edges of the visual hull mesh to form a *repaired* cone strip. Figure 6.3 and Figure 6.8(a) illustrates these connecting curves using dotted lines. The edges lying on these connecting curves will be called *rim edges*, and vertices shared by them will be called *rim vertices*. Figure 6.8(b) shows some *rim edges* for four different cone-strips. First, the true segments (view edges) corresponding to the extremities of adjacent cone-strip patches are identified (such as 1–4 in Figure 6.8(a)). Next, a geodesic path is computed on the visual hull mesh between these terminal edges, for e.g. between the edges 1-2 and 3-4 respectively in the illustration.

The repaired cone-strip acts as an approximation of the true cone-strip, which when projected back into the image it was generated from, will produce the exact outline of the original silhouette. The projection of the repaired cone strip into the image, will produce the outline of the silhouette of the visual hull polyhedron (up to discretization error in the images).

We now describe how the connecting curves are computed for each cone strip C (obtained from silhouette S in image I), one by one. First, a graph G is instantiated from the visual hull
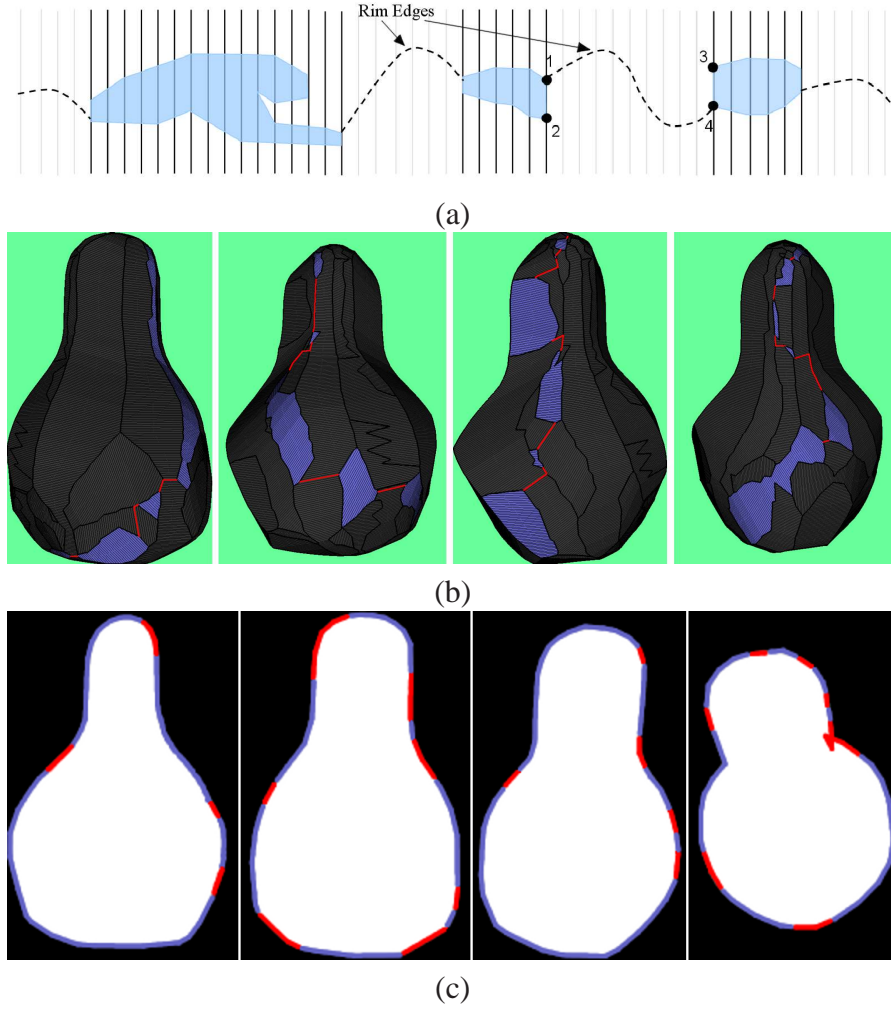
Figure 6.8: (a) Repairing a Cone Strip. Connections between disconnected cone-strip patches are computed (these are shown as dotted lines). These give rise to *rim edges*. (b) Four different examples of repaired cone-strips. The *rim edges* are shown in red. (c) The repaired cone-strips are shown projected into the images – here they are virtually identical to the original silhouette.

mesh. The nodes in G correspond to all the vertices of the visual hull mesh. The edges in G correspond to all the edges in the visual hull mesh, except for all the view edges and cone intersection edges that belong to the current cone strip C. The best way to think of this graph, is to picture the visual hull mesh itself, with all the facets of the current cone-strip removed. The connecting curves on the surface of the visual hull, that together would form the repaired cone-strip, are recovered one by one, by computing the shortest weighted path on G between extremities of adjacent cone-strip patches in C.

An edge (u, v) in G, is assigned a weight $w(u, v)$, given by:

$$w(u, v) = \sum_{t=0}^{1} \mathbf{F}(\mathbf{x}(t), S) + \epsilon, \text{with} \tag{6.1}$$

$$\mathbf{x}(t) = t\mathbf{P}(u) + (1 - t)\mathbf{P}(v) \tag{6.2}$$

where the squared distance function of a 2D point $\mathbf{x}$ in the image, with respect to the silhouette boundary S, is denoted by $\mathbf{F}(\mathbf{x}, S)$, and $\mathbf{x}$ lies on the line segment joining the 2D points where u and v project in the image, which are denoted by $\mathbf{P}(u)$ and $\mathbf{P}(v)$ respectively. Note that $\mathbf{x}$ never lies outside the silhouette S. The value of ϵ is set to a small positive constant, to ensure that all edge weights in G are non-zero. The edge-weights are made to be proportional to the proximity of the particular edge in the visual hull mesh, from a viewing ray from the silhouette S. Therefore, the shortest weighted path between adjacent cone-strip patches yields a non-intersecting connecting curve on the visual hull mesh, that lies close to the viewing rays of the silhouette cone. When the connection curves are relatively shorter, or when they lie on a convex patch of the visual hull surface, the approximation to the true rim curve is almost exact (see Figure 6.8(c)).

Generating per-view orientation labels

A fully repaired cone-strip partitions the visual hull surface into a front and back part w.r.t the corresponding camera. After the cone strips have been repaired, the next step involves associating two signature bitcodes S_a and S_b with every vertex in the visual hull. Each signature code contains n bits where n equals the number of cameras. For a vertex v, the $\mathrm{k^{th}}$ bit of S_a is set to 0 when v is back-facing w.r.t the $\mathrm{k^{th}}$ camera and set to 1 if its front-facing w.r.t the camera. However, if v was marked as a rim vertex earlier on for the $\mathrm{k^{th}}$ cone-strip, the $\mathrm{k^{th}}$ bit of S_b is set. The signatures for all vertices of the visual hull are assigned by repeating the following step for each one of the cameras.

Consider the kth camera and the corresponding cone-strip. First, the two incident vertices for each of the *true* segments of the kth cone-strip are labeled *front* or *back* depending on which one is closer to the camera center. This is shown in Figure 6.9(b). These labels are then flooded over the complete visual hull mesh, until the kth field of the signature code S_a for all vertices have been filled in. The flooding stops at a rim vertex, i.e. a vertex which lie on the rim edges of the kth cone-strip. Orientation labels stored in the bitcode S_a for various cameras are illustrated in Figure 6.9(c–e).

Deriving the final 2-coloring

Once signature codes S_a and S_b have been computed for every single vertex of the visual hull mesh, the per vertex coloring can be derived from them. This is computed by testing the parity of bits in the bitcode S_a. All vertices that have an even number of bits set (front-label) are assigned to the set of *red* vertices. The vertices with an odd number of bits set, are assigned to the set of *blue* vertices. Vertices for which at least one bit in bitcode S_b are set, are assigned to the set of *green* vertices – these correspond to *rim vertices* (see Figure 6.9(f)). The significance of these *rim vertices* will be discussed later in Section 6.7.

(a)   (b)   (c)   (d)   (e)   (f)
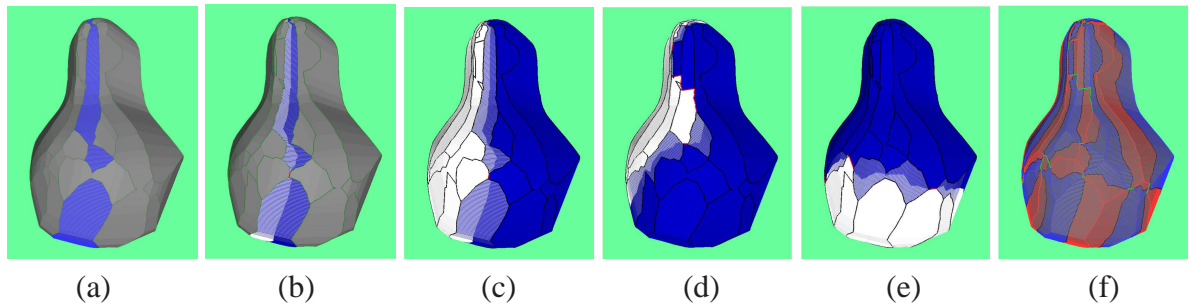
Figure 6.9: (a) An individual cone-strip is shown, the *true* segments are colored blue. (b) The repaired oriented cone-strip. Front-facing vertices are colored white while back-facing ones are colored blue. The location of the rim is unknown (the rim point shown, is chosen for the sake of illustration only). (c) The orientation labels are then flooded to all the vertices on the visual hull mesh. (d,e) Orientation labels for two other views are shown. (f) The two coloring obtained from the orientation labels is shown. The two coloring pertains to the visual hull vertices only. The rim locations (red/blue transitions) are shown only for the sake of illustration.
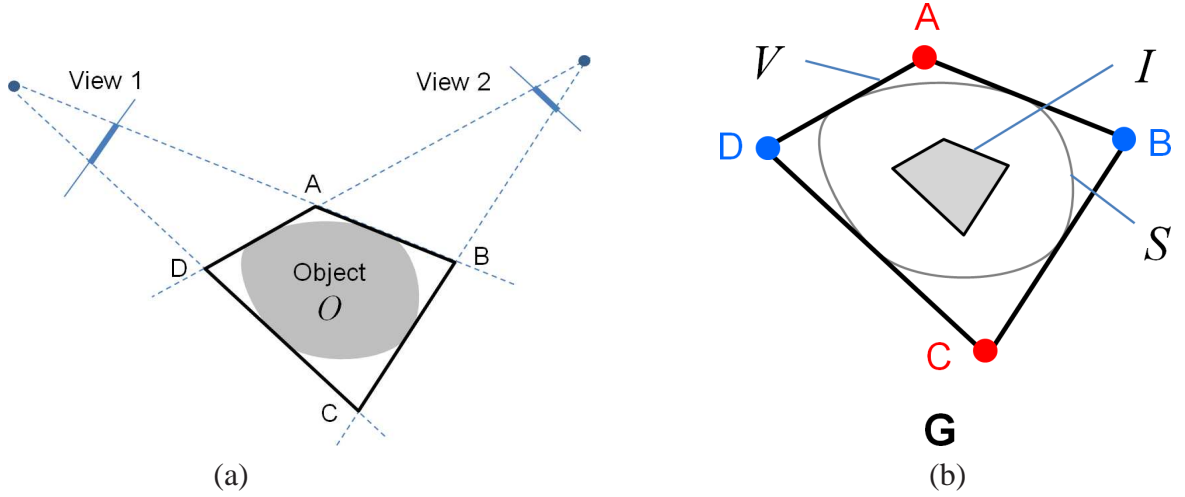
Figure 6.10: (a) In Flatland, the visual hull of object O from 2-views is denoted by polygon $ABCD$ respectively. (b) The inner offset layer is denoted by I. The space of exact silhouette consistent solutions comprise of all curves that lie between polygons V and I but also touch the four edges AB, BC, CD and DA at least once. The visual hull vertices are 2-colored (red and blue) and an underlying geometric graph embedding G is created (see Section 6.6).

6.6   Graph Construction – Formulation I

We will now describe Formulation I and the related graph construction. The various steps are illustrated in Figure 6.10 using a 2D analogy, but described for the 3D scenario.

Prior to the graph construction, the vertices of the visual hull polyhedra $\mathcal{V}$ are 2-colored as described in Section 6.5. As in our illustrations, we will refer to them as *red* and *blue* vertices respectively. Next an inner offset surface inside the visual hull is computed. The true surface is assumed to lie between the visual hull and this inner offset surface. The same assumption was made in [63, 148]. In our approach, this is implemented by first voxelizing the interior of the visual hull and then computing the signed distance transform using the approach of [97]. The maximum distance d_{max}, in this distance field is recovered and then the distance field is thresholded to detect pockets inside the volume (test if $d > T.d_{max}$) (where $T = 0.4$ was used in most of our experiments). The threshold T can be set conservatively, depending on how deep the true surface is expected to be with respect to the visual hull.

The surfaces bounding the pockets will form the inner offset surfaces – these will be

102
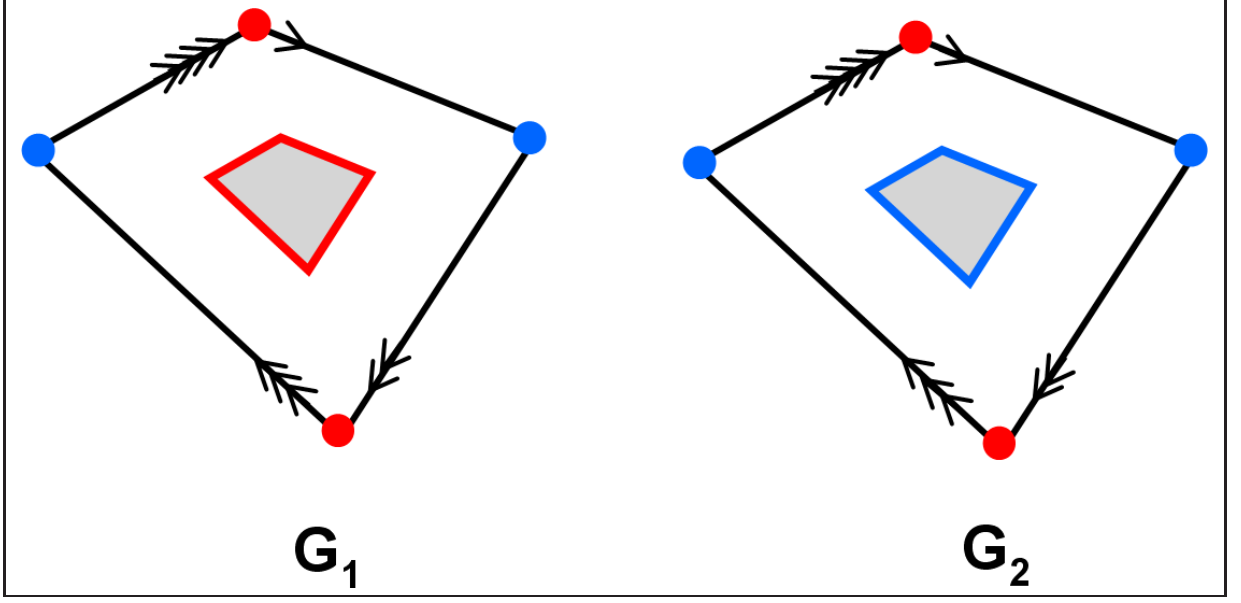
Figure 6.11: Two identical copies of G, denoted by G_1 and G_2 are created. Their inner offset nodes are labeled differently with the two colors (red and blue) as shown. Finally G_1 and G_2 are connected (glued together) by joining duplicate copies of surface nodes on G_1 and G_2 using additional edges. This gluing (topological identification) is depicted using arrows on all four edges of the polygon.

denoted collectively as $\mathcal{I}$. Points are sampled on the visual hull surface $\mathcal{O}$ and the inner offset surfaces $\mathcal{I}$. A node is then created for each of these sampled 3D points. The two surfaces are then tesselated (i.e. edges are connected between the nodes created earlier on) – the corresponding surface meshes are treated as graph embeddings denoted by $G_{\mathcal{O}}$ and $G_{\mathcal{I}}$ respectively. The graph nodes in $G_{\mathcal{O}}$ and $G_{\mathcal{I}}$ will be referred to as *surface nodes* and *inner offset nodes* respectively.

A 3D regular lattice G_v representing the interior of the visual hull (excluding the pockets) is then explicitly instantiated, and the photo-consistency of voxels on the lattice grid are evaluated (see Section 6.8.1 for the details). The graph nodes in G_v, will be referred to as *interior nodes*. Finally, additional edges must be created to connect the lattice grid G_v to $G_{\mathcal{I}}$ and $G_{\mathcal{O}}$ respectively. The connections are created by adding an edge between a *surface node* and the closest *interior node* (and similarly between an *inner offset node* with the closest *interior node*). The resulting graph is a union of G_v, $G_{\mathcal{I}}$, $G_{\mathcal{O}}$ and these new connections or edges. The
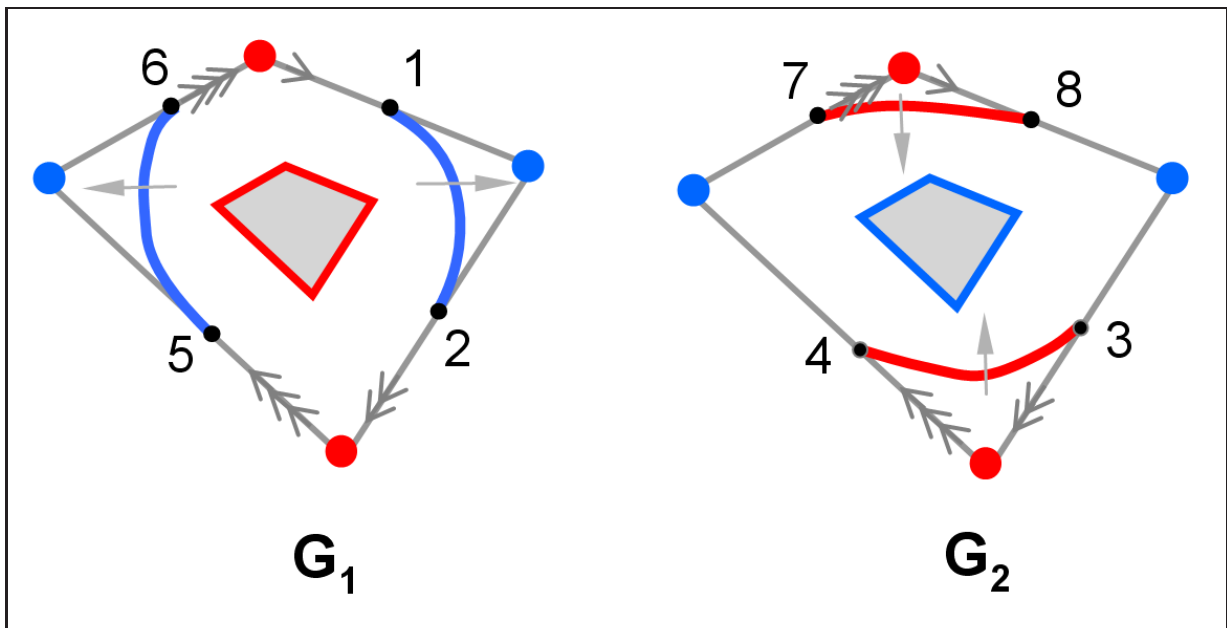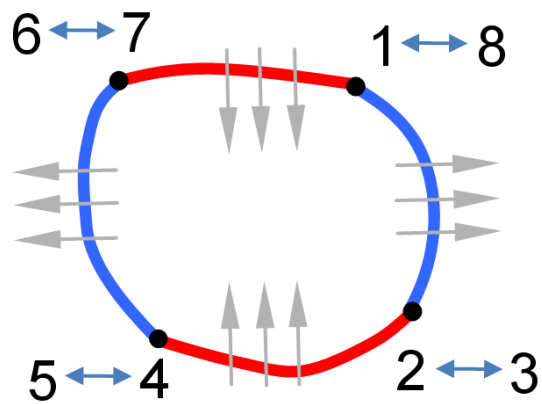
Figure 6.12: (a) A graph-cut problem is setup by connecting all red nodes to the source and all blue nodes to the sink. The resulting s-t cut is a manifold as shown. It must traverse across the four edges. Part of the cut surface is embedded in G_1 (shown in blue) and the rest of it in G_2 (shown in red). (b) Together they map back to a surface that satisfies silhouette constraints. The grey arrows indicate the direction of flow from source (red) to sink (blue).

embedding of this graph in $\mathbb{R}^3$ will be denoted by G_1.

The graph G_1 is duplicated, to form an identical copy G_2. Next, G_1 and G_2 are connected as follows. Additional edges are created between every *surface node* $v_1 \in G_1$ with its duplicate node $v_2 \in G_2$. We will refer to these as *silhouette-consistency edges*. The final flow graph denoted by $\overline{G}$, is a union of G_1, G_2. Two terminal nodes $\{s,t\}$ and the set of *silhouette-consistency edges* are added to $\overline{G}$. See Figure 6.13 for an illustration. We are now ready to describe how the various edge capacities in $\overline{G}$ are assigned.
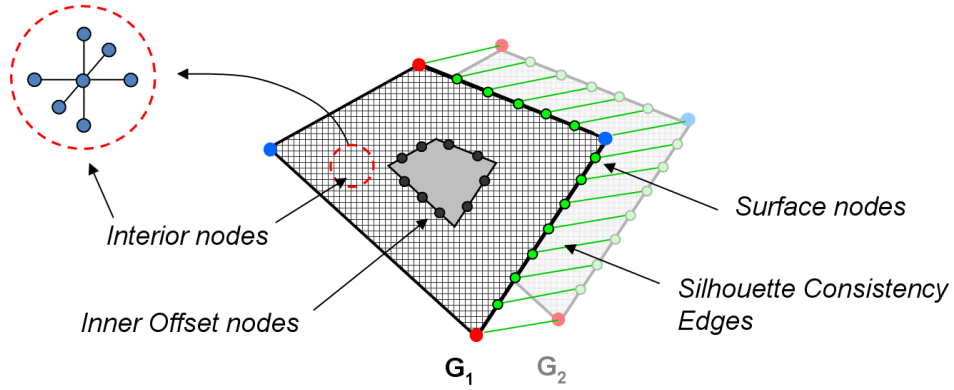
Figure 6.13: The various types of nodes in our graph embedding are shown – *interior nodes* with a 6-connected neighborhood, *inner offset nodes*, *surface nodes*. Graph embeddings G_1 and G_2 are connected via *silhouette consistency edges* shown in green.

First, all the *inner offset nodes* in G_1 are labeled *red* and the *inner offset nodes* in G_2 are labeled *blue*. Remember that the subset of *surface nodes* in G_1 and G_2 which are true vertices of the visual hull mesh have already been labeled *red* and *blue*, respectively. Next, all graph nodes labeled *red* are connected to the source terminal s, while all nodes labeled *blue* are connected to the sink terminal t. Edge capacities for all edges $(u,v) \in G$, except for *silhouette consistency edges* are derived from the photo-consistency of the 3D point corresponding to the mid-point of the two nodes u and v respectively (these nodes have explicit 3D positions).

Now a minimum cut is computed on $\overline{G}$ – let us denote this set of cut edges as C. For every edge $(u,v) \in C$, a corresponding 3D point p is created and an outward-facing normal vector is associated with it. This is done by computing the unit normal $\hat{n}$ pointing from the 3D location

of u to v or v to u depending on whether the edge came from G_1 or G_2. The exact location of p on the edge (u,v) is improved using sub-voxel refinement by computing a local parabolic fit to the photo-consistency scores. The surface normals are smoothed by performing a weighted average over the immediate neighbors on the grid. Finally, a triangulated manifold surface is fitted to these oriented points using the Poisson surface reconstruction approach of Kazhdan et. al. [72].

Lemma 6.6.1. *Any connected s-t cut on $\overline{G}$ must correspond to a surface that exactly satisfies silhouette constraints in all the views.*

Proof: Seeking a contradiction, let us suppose that a connected s-t cut C, on the flow graph $\overline{G}$, produces a surface S that does not satisfy silhouette constraints in all n views. By connected, we mean that the graph defined by the nodes and edges constituting the cut C has only one connected component. Let V denote the corresponding visual hull computed from the n silhouettes. Unless all silhouette constraints are satisfied, one of the following must be true – (1) part of S must lie outside V or (2) S is too small and does not tightly fit inside V. We now show that for both these cases, our assumption leads to a contradiction i.e. C cannot be a connected s-t cut in the flow graph $\overline{G}$.
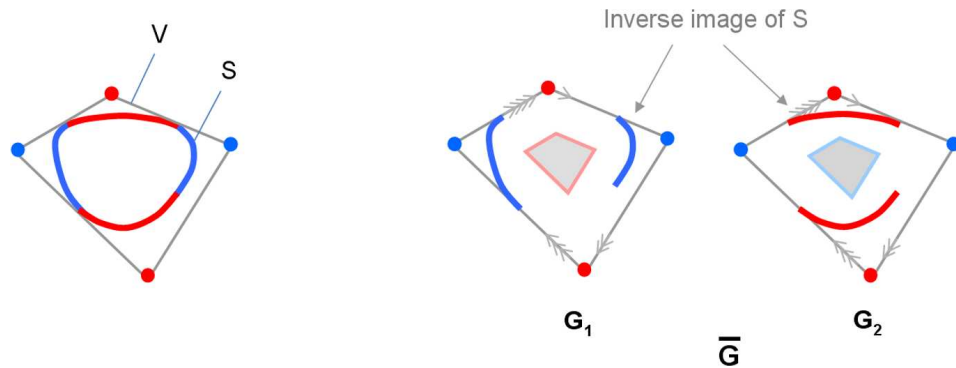
Figure 6.14: Unless silhouette constraints are satisfied by S, its inverse image embedded in $\overline{G}$, will not be a valid cut surface as there will exist in the volume at least one edge through which more flow can pass from the source to the sink.

Case 1: It follows from the graph construction described above that no point of the cut C

can ever lie outside the visual hull. This is because all vertices in $\overline{G}$ map to 3D point either in the interior or on the surface of V. Thus this case is ruled out by construction.

Case 2: If S does not touch the visual hull tightly, there must exist at least one viewing ray for which the corresponding view edges in V do not touch the surface S anywhere. This is shown in Figure 6.14. S has an inverse image C which is a manifold embedded within $\overline{G}$. C must then contain a discontinuity somewhere in the vicinity of that viewing ray. Hence, some flow can pass from the source to the sink in $\overline{G}$ through the discontinuity or gap as shown in Figure 6.14. Therefore, C is not a fully connected, valid s-t cut in $\overline{G}$.

$\square$
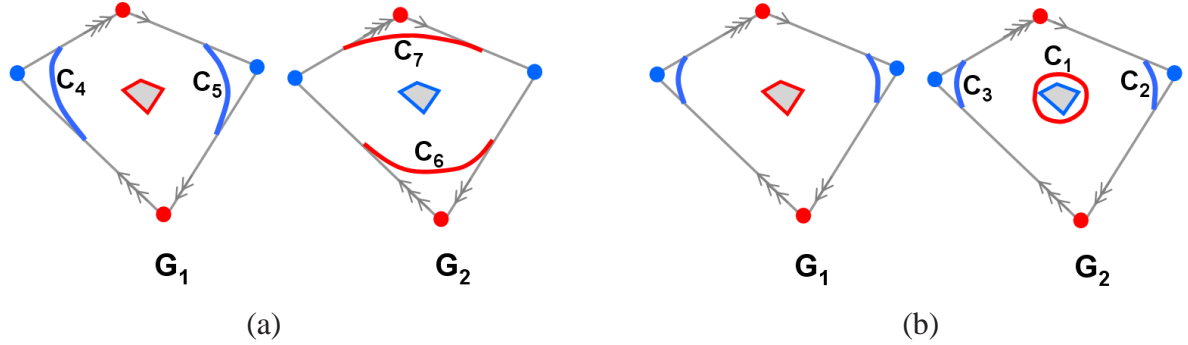
Figure 6.15: (a) desired solution. (b) degenerate solution under a minimal surface bias.

Cut surface with multiple connected components: The proof considers the case where the s-t cut is fully connected. However, the cut surface is not always guaranteed to be fully connected. It can have multiple connected components, unless the pockets detected within the interior of the visual hull are fairly large. When this is not the case, an undesirable degenerate solution may be produced. This is illustrated in Figure 6.15(b). Instead of the desired solution shown in Figure 6.15(a), the red cut surface in G_2 denoted by C_1 collapses around the interior pocket. The resulting cut, which has multiple components, does not satisfy silhouette constraints. This is once again due to the problem of the minimal surface bias which plagued the earlier volumetric energy minimization approaches. The solution we desire, is of the type shown in Figure 6.15(a). Such a solution will be obtained as long as the total cost of the cut

$(C_1 \cup C_2 \cup C_3)$ is more than the cost of the cut $(C_4 \cup C_5 \cup C_6 \cup C_7)$ as illustrated in the figure.

Input: Color Images $\{\mathcal{I}_i\}$, Binary silhouette images $\{\mathcal{S}_i\}$, Camera Calibration $\{\mathcal{P}_i\}$
Output: Triangulated 3D mesh model $\mathcal{M}$

```
𝒰 ← Compute EPVH Polyhedron ({𝒮ᵢ}, {𝒫ᵢ})   (see [41])
2-Color Visual Hull vertices (𝒰, {𝒫ᵢ})   (see Section 6.5)

{G_ℐ, G_O} ← Create Offset Surfaces (𝒰) // inner, outer layers
G_v ← Create Voxel Grid (G_ℐ, G_𝒪)         // regular grid in ℝ³
Compute Edge Costs (G_v, {ℐᵢ}, {𝒫ᵢ})      // photo-consistency

G₁ ← Connect (G_v, G_ℐ, G_𝒪)  // connect grid with offset layers
G₂ ← Duplicate Graph (G₁)
G  ← Connect (G₁, G₂)                       (see Section 6.6)

Setup Sources & Sinks (G)                   (see Section 6.6)
S₂ ← Find Minimum Cut (G)
𝓜 ← Create Mesh (S₂)  // inverse image of cut surface in ℝ³
```

return $\mathcal{M}$

Algorithm 2: A summary of Formulation I for graph-cut based volumetric stereo with exact silhouette constraints.

Note that an earlier version of our approach [124] did not suffer from this problem. However, that graph construction [124] used the rim mesh, which is instable and difficult to compute in the presence of noise and discrete silhouettes. There, each individual surface patch was reconstructed as a cut surface embedded in a subgraph representing a fraction of the visual hull volume. In the approach proposed here, these subgraphs are effectively merged into a single graph. Although this makes the construction simple and robust, the minimal surface bias cannot be completely ruled out anymore.

This minimal surface bias occurs when the inner pockets are small such that surrounding

them by a relatively low-area surface such as C_1 incurs a fairly small penalty. This problem rarely occurs when the inner pockets are relatively large. We describe a better approach to compute the inner pockets in the next chapter which avoids the minimal surface bias problem described above. Similar ideas based on the *visibility based* ballooning term have also been recently proposed by Hernandez et. al. [59] to address the same problem.

The complete graph construction approach for Formulation I is summarized in Algorithm 2. Some short-comings of this formulation will be discussed next. We will address this is Formulation II, which will extend Formulation I.

6.6.1 The Problem with T-junctions
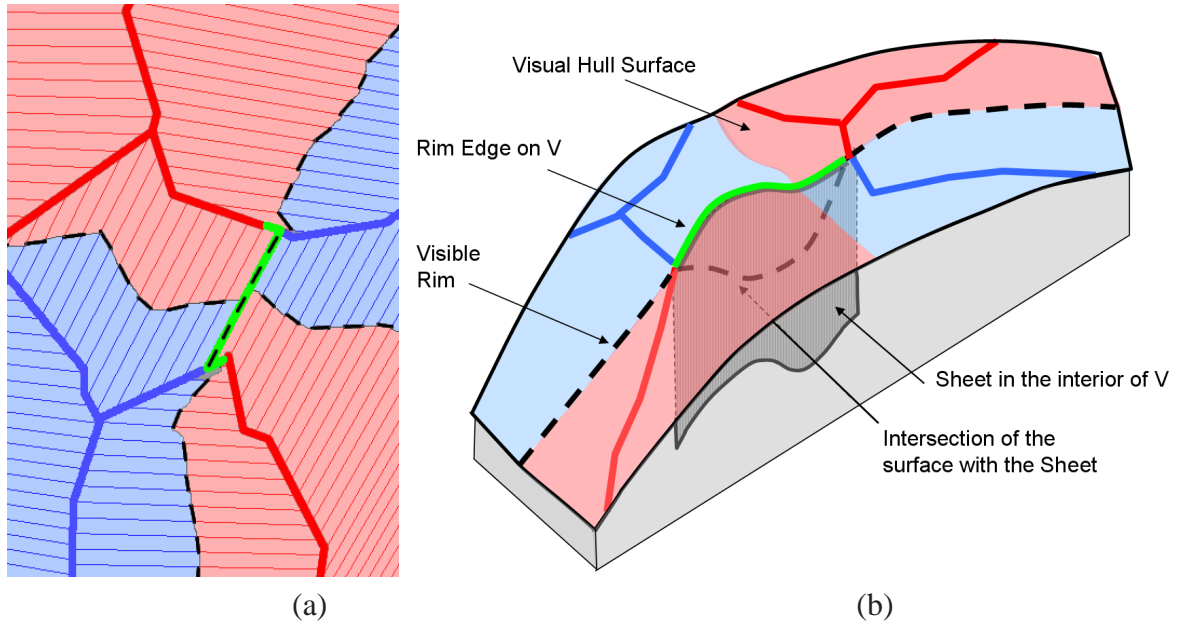
Figure 6.16: Enforcing the 2-coloring across rim-edges (best seen in color). (a) A close-up of the 2-colored visual hull. Cone intersection curves are shown in bold (red and blue). Rim edges are shown in green. Dotted curves indicate possible position of rims. (b) T-junctions give rise to self-occlusions on the rim – the part that detaches from the visual hull surface and lies within its volume – we show the surface by its intersection with the *sheet*, a swept surface dropped down from the rim edges into the volume.

Formulation I correctly handles convex objects with cameras in general position. However, it does not correctly deal with non-convex objects, for which T-junctions may occur on the

silhouettes. A T-junction arises on the silhouette, when the corresponding rim curve on the surface is occluded by the surface itself. The occluded section of the rim curve, which we will refer to as the *invisible* part of the rim does not lie on the surface of the visual hull, rather it lies within its volume. This is illustrated in Figure 6.16 using an imaginary *sheet* surface patch within the visual hull volume, which meets the visual hull surface at the set of rim edges. The *sheet surface* can be chosen arbitrarily as long as the true surface intersects it as shown by the dotted curve in the illustration.

Formulation I allows the coloring of the reconstructed surface to switch only at a silhouette consistency edge – however, that can only happen at a point on the visual hull surface, not in its interior. This causes silhouette constraints to be enforced at places where they should not be enforced such as along the rim edges shown in Figure 6.16.

Consider the region on the visual hull surface around a sequence of rim edges. When these rim edges are shared by an even number of repaired cone-strips, the vertex colors on either side of the rim edges are identical to each other. Formulation I deals with this situation correctly, because the vertex coloring across a rim edge of even multiplicity, will be identical.

However, a problem arises when a sequence of rim edges are shared by an odd number of repaired cone-strips i.e. they have odd multiplicty. In this case, the vertex colors on either side of the rim edges are opposite of each other and silhouette constraints get enforced. Such a rim edge is shown in Figure 6.16. Since, the coloring of the surface needs to switch here, the cut surface is locally pushed outwards i.e. the *invisible* part of the rim curve is constrained to lie on the visual hull surface. This problem is not catastrophic – it happens only at a few places and may cause two types of artifacts.

- The reconstructed surface juts out in the form of a thin sheet (see Figure 6.17(b)).

- Portions of the cut surface detaches from the reconstructed surface, which is recovered in the correct position and is artifact free.

In both cases, the subsequent Poisson surface reconstruction algorithm considerably reduces

these artifacts. It treats the points on the thin sheets and detached surfaces as outliers during the final surface reconstruction step and enables us to reconstruct some complex shapes with occasional artifacts as described above. We now look into handling the problem of the *invisible* rim in our formulation.

<table>
<tr><td>(a)</td><td>(b)</td></tr>
</table>
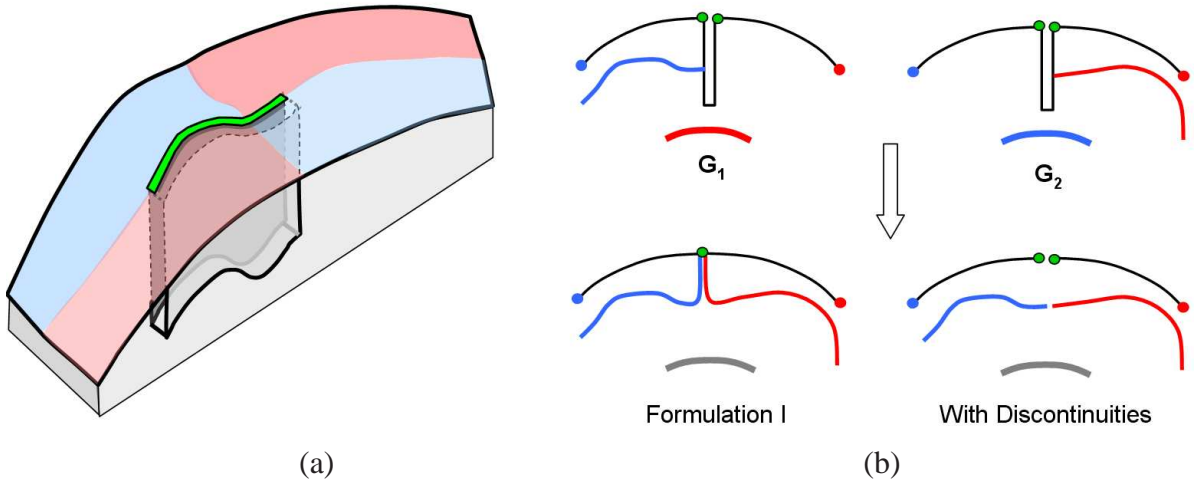
Figure 6.17: (a) Sheet surface patches are first constructed from the rim edges with odd multiplicity. The lattice edges lying on the rim sheet are removed to create a discontinuity in the volume. (b) A cross-section is shown here. Without the discontinuity, the cut surface may jut out. With the discontinuity, the cut surface can terminate on the sheet surface. The cut surface is now a manifold with a boundary.

One possible approach is to detect the problematic rim edges (the ones with odd multiplicity), and construct imaginary *sheet surfaces* from them, within the volume. The edges in the graph embedding G_v, that straddle the sheet surface patches will be removed, thereby introducing discontinuities in the interior of the lattice volume (see Figure 6.17). As a result, the minimum cut surface will become a 2-manifold with boundary. Its boundaries will lie along the sheet surfaces, and will not be constrained to lie on the visual hull surface. The inverse image of this new cut surface, will be a surface with open seams (or cracks) on it. The Poisson surface reconstruction step will typically merge these open seams. However, it will not be possible to guarantee that the two sides of the surface be geometrically aligned with each other along the seam – this could still result in noticeable artifacts on the reconstructed

surface.

Computing the Sheet Surface Patches: Our approach for generating the *sheet surfaces*, involves constructing local offset surfaces in the interior of the visual hull volume and projecting the sequence of rim edges on to this offset surface. The offset surfaces can be computed approximately using the 3D signed distance transform $D(\mathcal{O})$ of the visual hull surface $\mathcal{O}$ and the gradient of this distance transform $\nabla D(\mathcal{O})$, respectively. Note that the distance transform is computed to determine the extent of the graph embedding G_v anyway, as described in Section 6.6 – hence, this is not an extra computational burden. We densely sample points p on the rim edges, and for each one of them, we take a small step along $\nabla D(p, \mathcal{O})$ to generate an offset curve and repeat the step a few times. The gradient field $\nabla D(\mathcal{O})$ can however vanish at points within the volume [140], and therefore we resort to a linear combination of $\nabla D(\mathcal{O})$ and $-\nabla D(\mathcal{I})$ where $\mathcal{I}$ denotes the inner offset surface of the visual hull. The sheet surfaces are made to proceed only to a small depth relative to the visual hull surface. Finally, we recover the subset of voxels that contain all the points samples generated on the sheet surfaces. The relevant edges in these voxels are deleted, to create the discontinuities described earlier and this is also illustrated in Figure 6.17.

6.7   Graph Construction: Formulation II

We will now describe a more principled way to handle the *invisible rims*, but this will require computing the minimum cut on a graph which is twice as large as the one used in Formalation I. Although the memory footprint of the graph will be larger, this extension will allow handling the *invisible* rims in a more elegant fashion, and will thereby allow us to reconstruct any non-convex shape.

The main difficulty with Formulation I was that, it allows the coloring of the reconstructed surface to switch only at a *silhouette consistency edge* – this can only happen at a point on the visual hull surface. Thus, silhouette constraint are enforced at places where they should not
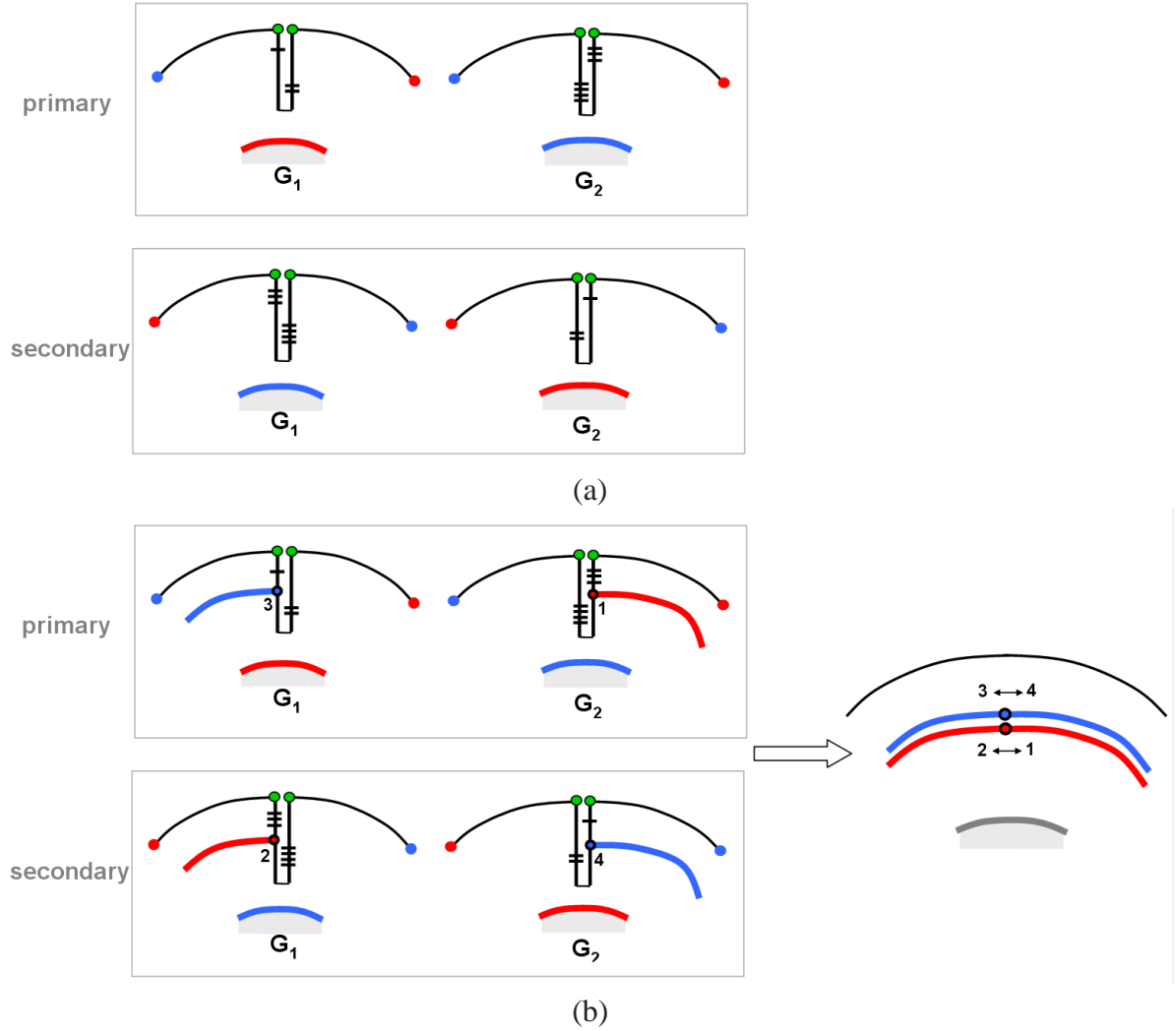
Figure 6.18: (a) In Formulation II, we duplicate $\overline{G}$ further creating two copies primary and secondary where the red/blue labels are reversed. The primary and secondary graphs are glued across the sheet surfaces as shown. (b) The cut surface now switches color at a point on the sheet surface (i.e. within the visual hull volume). The final solution is a double manifold. By symmetry the solution in the primary and the secondary map to a single surface in $\mathbb{R}^3$.

be enforced i.e. along viewing rays corresponding to T-junctions on the silhouettes. Our first solution, was to introduce discontinuities in the graph embedding in a way that would relax the silhouette constraint at the problematic rim edges.

In Formulation II, the graph $\overline{G}$ is first duplicated to create two identical copies, the *primary* and the *secondary* graphs. The red/blue vertex labels in the secondary graph are then reversed. Next, the volumes corresponding to the primary and secondary are glued together across the

sheet surfaces.
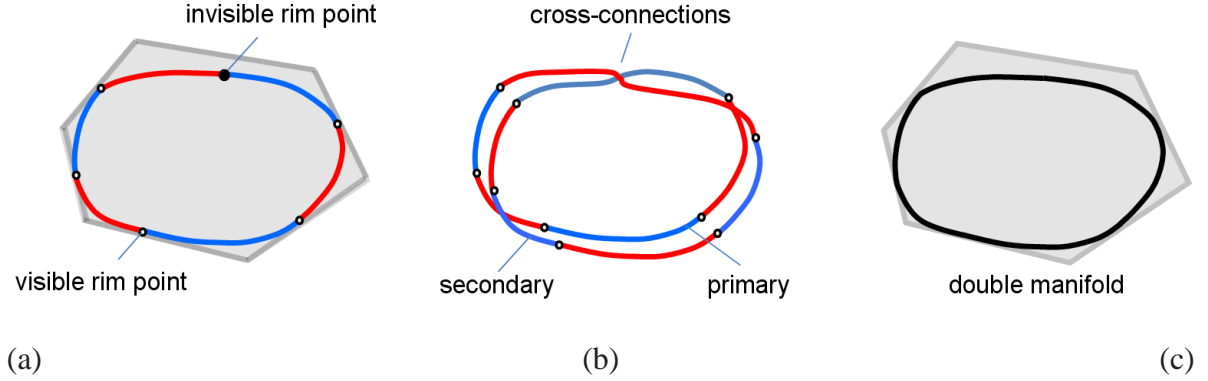
Figure 6.19: (a) 2D illustration of the visual hull and the 2-coloring it induces on the surface. (b) In Formulation II, the graph embedding is duplicated into the primary and secondary; the cut-surface is a double manifold partially embedded in the two graphs. The primary and secondary are connected through the *sheet surfaces* inside the visual hull. (c) Inverse image of the double manifold (the two copies coincide by symmetry).

The gluing or topological identification is done by creating the following edges between vertices in the primary and secondary graphs respectively. We denote the set of edges that straddle the sheet surfaces in $\overline{G}$ by S. Let (u_1,v_1) denote an edge in S that belongs to the primary graph and let (u_2,v_2) denote its copy in the secondary. First, the two edges – (u_1,v_1) and (u_2,v_2) are removed from the respective graphs. Next, two new edges are added – (u_1,v_2) and (u_2,v_1). Note that these new edges connect vertices in the primary graph with vertices in the secondary (see Figure 6.18(a)). This effectively glues together volumes in the primary and secondary along the sheet surfaces in the same way that G_1 and G_2 were glued together along the visual hull surface in Formulation I.

The cut surface on this graph has the structure of what is known as a *double manifold* in manifold theory [34] – a manifold obtained by gluing two copies of a 'manifold with boundary' along their common boundary. In our case, one copy of the cut surface lies embedded in the primary graph while another copy lies embedded in the secondary (see Figure 6.19). Their boundaries are glued together along the cross connections across the sheet surfaces while respecting the surface map of the underlying surface i.e. the red part of the cut surface in the

primary connects to the red part in the secondary and vice versa for blue . If the mincut is unique, then the two copies are geometricaly identical, by virtue of the symmetry in the graph construction described in Formulation II. However, the surface map coloring in the two copies is reversed, because the red/blue labels in the primary and secondary are opposite of each other. In general, the cross connections across the sheet surfaces (which correspond to the invisible rims) will occur in the interior of the visual hull. Note that, in practice, there may be multiple mincuts, each of which is a slight perturbation of one another and therefore the primary and the secondary cuts may not be exact copies of each other. However, the cut cost of these two surfaces will be equal.

The inverse image of this double manifold is a surface without discontinuities unlike in the extension of Formulation I (with discontinuities). More over, the invisible rims now occur in the interior of the visual hull unlike in Formulation I. For a convex shape, Formulation II will produce a double manifold with two disconnected copies of the surface that are identical in shape and superimposed on each other in 3D.

6.8   Implementation Details

In this section, we describe the photo-consistency measure that is used in the graph-cut based energy minimization approach introduced earlier in Chapter 5. While this chapter focusses on the graph construction, the actual algorithm to compute the minimum cut on the flow graph is described in Appendix B-3.

6.8.1   Computing Photo-consistency

Given color images $\{\mathcal{I}_i\}$ and the corresponding camera calibration matrices $\{\mathcal{P}_i\}$, we now describe our method to compute the photo-consistency of a 3D point M. Our photo-consistency measure is computed in image space and treats occlusions (i.e. lack of visibility) as out-
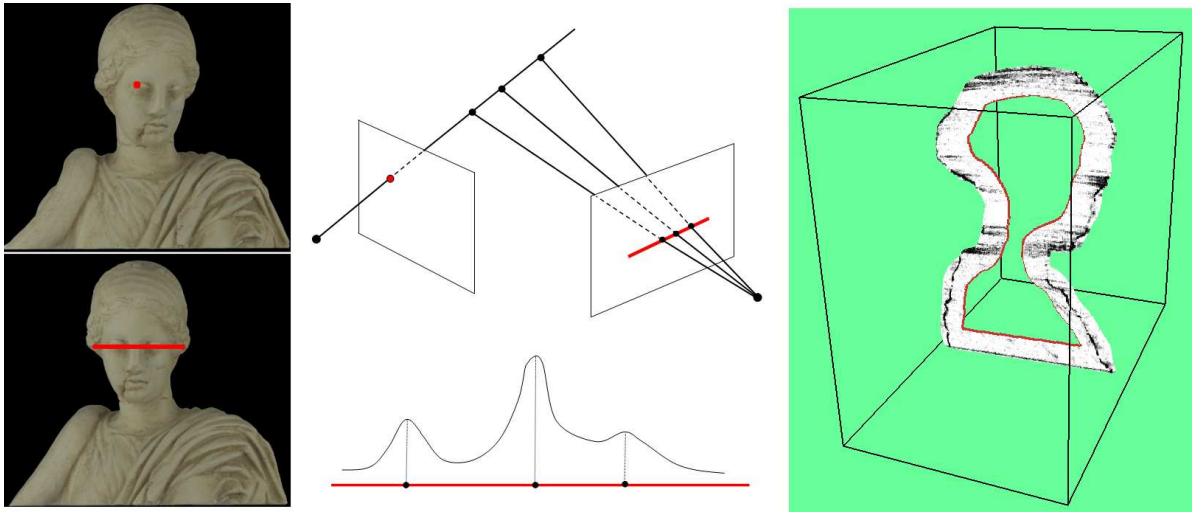
Figure 6.20: Computing multiple hypotheses for 2-view matches. These 2-view matches are triangulated and the generated 3D points are used to accumulate votes within a 3D volume. The photo-consistency measure is derived from these votes. A slice through the photo-consistency volume (interior of visual hull) is shown. Here black indicates regions of high photo-consistency.

liers during the underlying matching process. Instead of computing an absolute measure of photo-consistency by combining scores such as color-variance or normalized cross correlation (NCC) (see Appendix A-2), we compute photo-consistency through a depth-map fusion and subsequent 3D voting approach. One of the key advantages of doing this is that the photo-consistency measure is more sensitive, and shows a stronger peak near the true surface location. Similar to [149], our photo-consistency measure is robust to occlusions and non-Lambertian effects.

In standard binocular stereo, dense correspondence between two calibrated views is computed by attempting to find the best match for every pixel in both views. The search for the best match is done along the epipolar line in the second view (see Figure 6.20). Our approach for computing photo-consistency in a volume starts off similarly. However, instead of trying to find the best match for every pixel, we detect all the local maximas along the epipolar lines (one of which is most likely to be the true correspondence). The matching is evaluated using the normalized cross correlation of two $\mu \times \mu$ sub-windows (μ is typically set to 9 or 11

pixels) centered at the candidate pixels in the two views. For invariance to scale and orientation, object space patches could have been considered; however, this was not required in our experiments which mostly dealt with turntable sequences.

For each of the local maximas (potential matches) we perform sub-pixel refinement and then perform 2-view triangulation to estimate a 3D point corresponding to the hypothesized match. Each 3D point contributes a vote to the cell it falls in, within a finely divided lattice of cubical cells. Figure 6.20 shows a situation for a pixel in the first view, where there are three likely matches in the second view. Each of the three 3D points could potentially be a surface point and contributes a vote.

The photo-consistency computation proceeds as follows. For each image $\mathcal{I}_i$, we find a subset of k proximal images and compute hypothesized correspondences as described above for k pairs where $\mathcal{I}_i$ is always the first image. This is repeated for every single image in the dataset. Then the photo-consistency measure of a 3D cell $\mathbf{M}$ is set to $e^{-\lambda * \mathcal{V}(\mathbf{M})}$ where $\mathcal{V}$ denotes the votes accumulated within the cell $\mathbf{M}$ and λ is a parameter that controls the trade-off between smoothness and photo-consistency.

When computing dense correspondences over multiple image pairs, multiple votes accumulate in the cells that are near the true surface. Multiple votes for the same 3D location, coming from different image pairs reinforce each other. Although this is similar to depth-map fusion, one key difference exists. In a traditional depth map, only the best match for every pixel in the other view is stored while all other potential candidates are ignored. In our approach we generate a hypothesis for each local maxima and hence this is equivalent to generating a depth-map with multiple depth hypotheses per pixel. Thus our photo-consistency computation essentially involves fusing multiple multi-hypothesis depth-maps.

Sub-pixel refinement during the pairwise matching process is important as this increases the sharpness or accuracy of the photo-consistency bands in the volume. Without sub-pixel refinement, the photo-consistency bands are blurred and the resulting reconstruction from the

117

graph-cut step is less accurate. The sub-pixel match is identified by fitting a parabola to the matched pixel (a local maxima) and its two neighboring pixels.

6.9 Results

We first describe some experiments on synthetic datasets – on the PEAR and the BEAN datasets. The PEAR dataset had 8 silhouettes and 31 color images – these silhouettes had very few T-junctions even though the object shape is non-convex. The BEAN dataset comprised of 4 silhouettes and 28 color images, and contains T-junctions on two of its silhouettes. Screen-shots from various stages of our reconstruction approach are shown in Figure 6.21 and Figure 6.22 respectively. The results shown in these two figures were obtained using an implementation of Formulation I (with discontinuities at sheet surfaces). Some artifacts are seen on the reconstructed bean (Figure 6.22(h)), where the surface juts out abruptly near the saddle point.
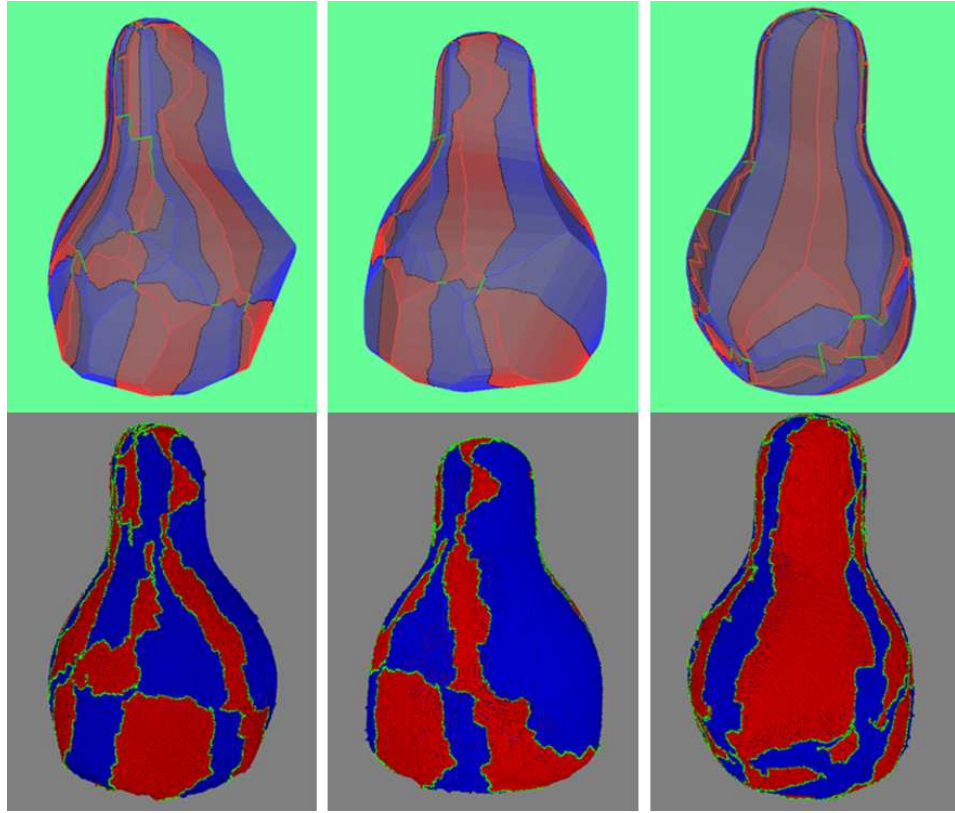
(a)
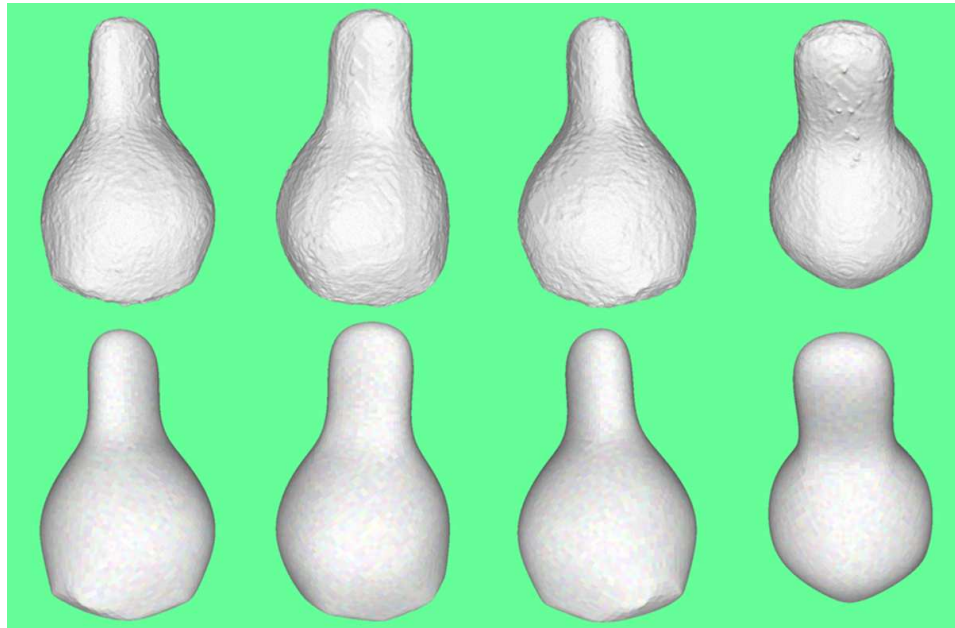
(b)

Figure 6.21: PEAR dataset reconstructed using Formulation I: (a) The two-coloring of the visual hull [Top] and the reconstructed surface [Bottom] for the pear shape. (b) [Top row] The reconstructed mesh of the pear obtained using our method. [Bottom row] The ground truth 3D shape.
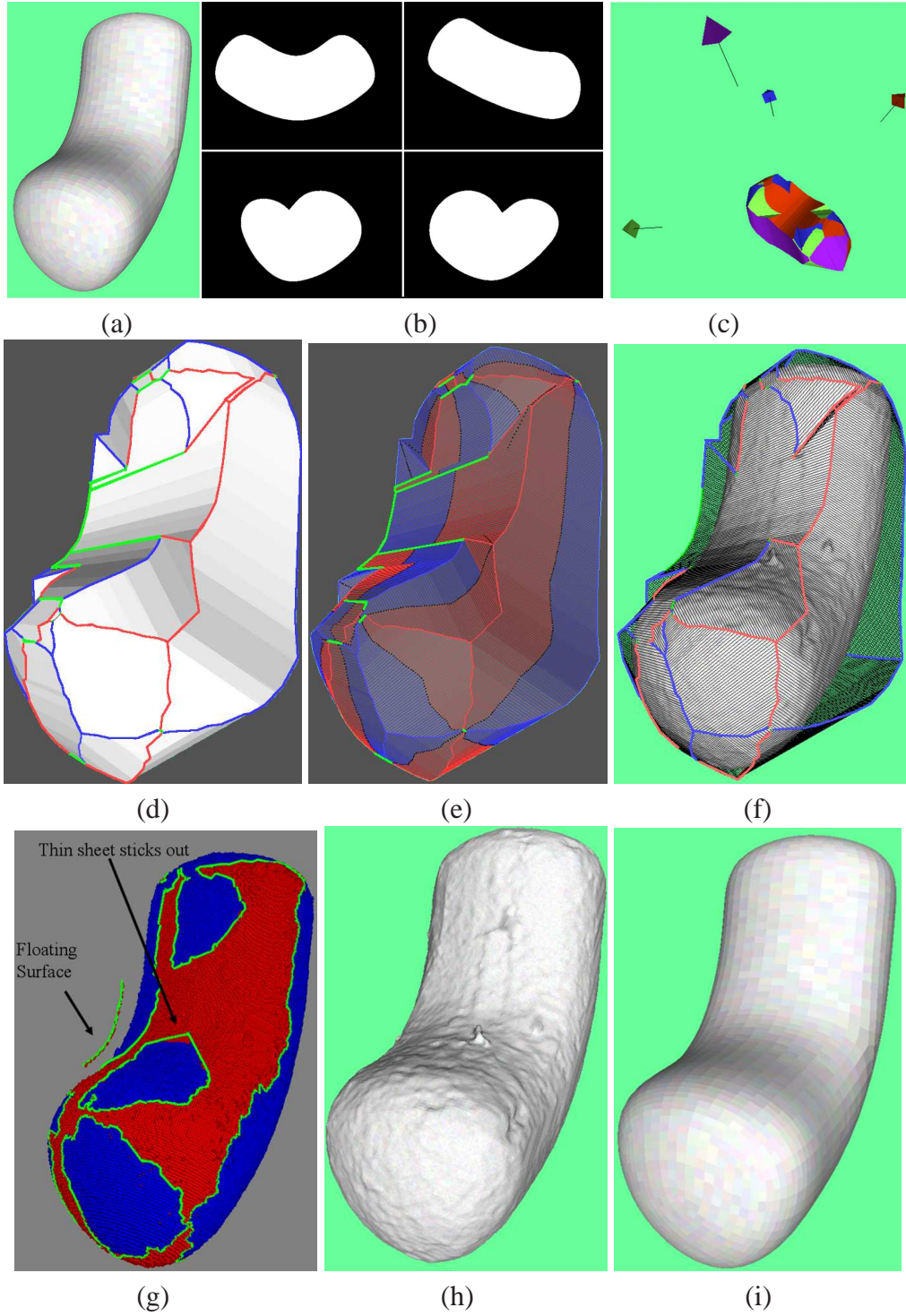
Figure 6.22: BEAN dataset reconstructed using Formulation I: (a),(b) 4 silhouettes and 28 color images (not shown). (c) The corresponding visual hull from four views. (d) Two-colored visual hull vertices. (e) A candidate surface map projected on the visual hull. (f) Recontructed surface displayed within the visual hull. (g) The two-colored surface map induced by the reconstruction. (h) The reconstructed surface mesh (i) The ground truth surface.

(a)   (b)   (c)   (d)
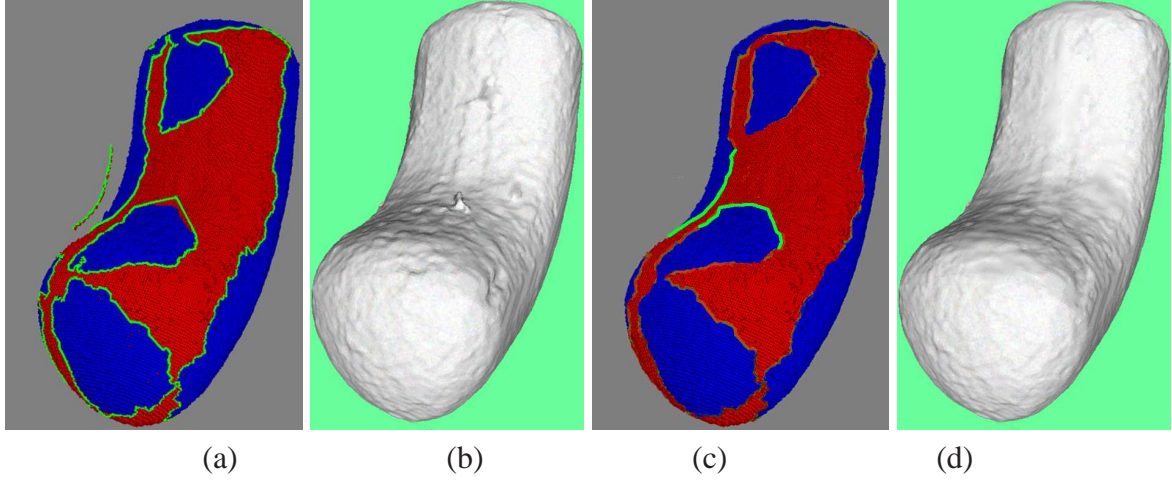
Figure 6.23: Comparison between Formulations I and II:: The two-colored surface map and the final reconstructed surface mesh is shown for (a) Formulation I (with discontinuities), and (b) Formulation II (double manifold). The artifact which appears near the saddle point on the surface, disappears in Formulation II and the invisible rim (highlighted in green) lies in the interior of the visual hull. Otherwise, the two reconstructions are almost similar. Note that only the embedding in the primary graph of the double manifold is shown here.

(a)   (b)   (c)
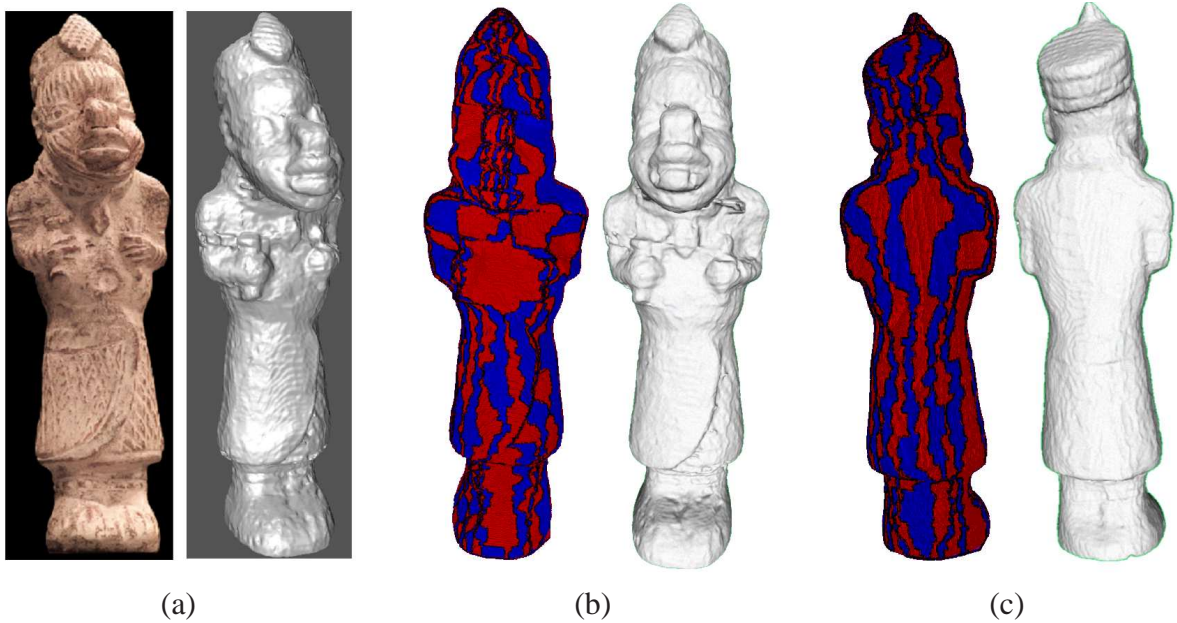
Figure 6.24: (a) one of 36 input photographs of the STATUE1 dataset and the reconstructed 3D model. (b),(c) The reconstructed model rendered from two different viewpoints shown along with the induced two coloring of the surface (surface map).

121

Figure 6.23 shows a comparison between Formulation I (with discontinuities), and Formulation II, which reconstructs a *double manifold* cut surface. The only visible difference in the two reconstructions, is near the invisible rim, close to the saddle point which is artifact free in the second formulation. The invisible rim is shown in green forms a smooth curve and lies within the visual hull. Although it is not shown here, the cut surface embedded in the primary and secondary graphs are virtually identical.

We have reconstructed various real multi-view datasets of statues that were captured using a turntable with the purpose of digitizing their 3D shape. Datasets STATUE1, STATUE2 and STATUE3 contains 33–36 high resolution images each. The images in these datasets were typically 3-4 MPixels each. The process of recovering the camera calibration and foreground silhouettes from the images for these datasets, is described in [57]. The silhouette extraction for the DINOSAUR dataset [3] was done using techniques described in Appendix B-1. Figures 6.24 and 6.25 show the reconstructed 3D models for the STATUE1 and DINOSAUR datasets along with the two-coloring induced by the surface map. More experimental results are shown in Figures 6.26, 6.27 and 6.28.

The running time in our implementations is dominated by the photo-consistency computation and computing the minimum cut on the graph, each of which takes approximately 60 - 80 minutes. While the basic shape of our reconstructed objects in accurate, these 3D models lack geometric detail. The resolution of the underlying volumetric grid influences the amount of detail that can be recovered. Our volumetric grids typically require between 2 to 15 million voxels (the voxel dimension is derived from the image resolution) and total running time are typically 2-3 hours for the turntable datasets. All the reconstructions were done on a 3 GHz processor with 2 GB RAM. The memory footprint of the graph in Formulation II was often too large to fit into RAM and hence most of the experiments were performed using Formulation I (with discontinuities).

(a)　　　　　　　　(b)　　　　　　　　(c)
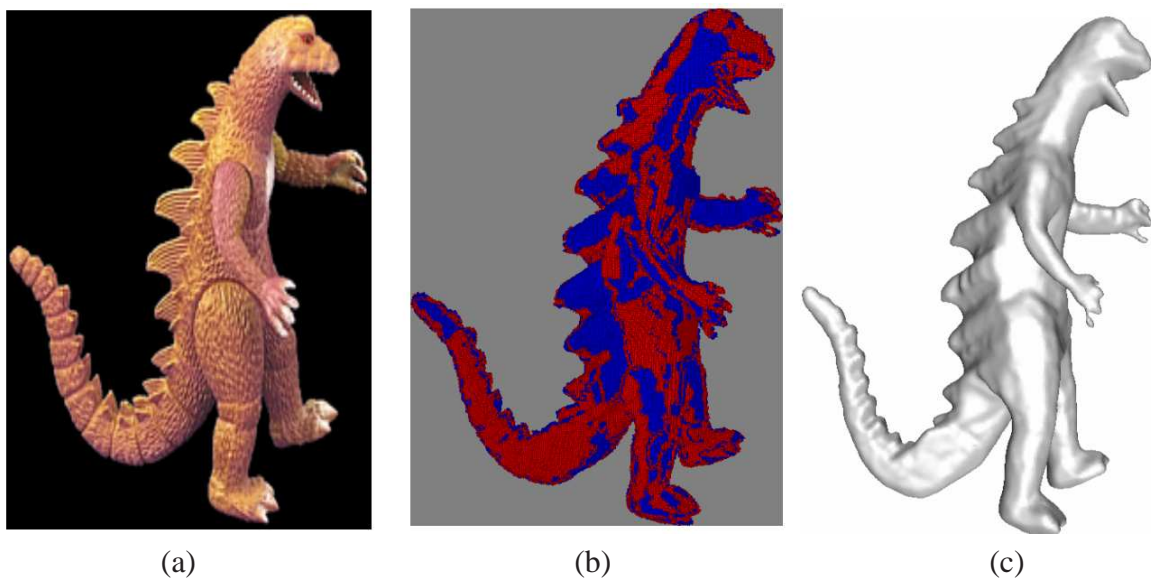
Figure 6.25: (a) one of 30 input photographs of the DINOSAUR dataset and the reconstructed 3D model. (b),(c) The reconstructed model rendered from a different viewpoints shown along with the induced two coloring of the surface (surface map).

(a)　　　　　　　　(b)　　　　　　　　(c)
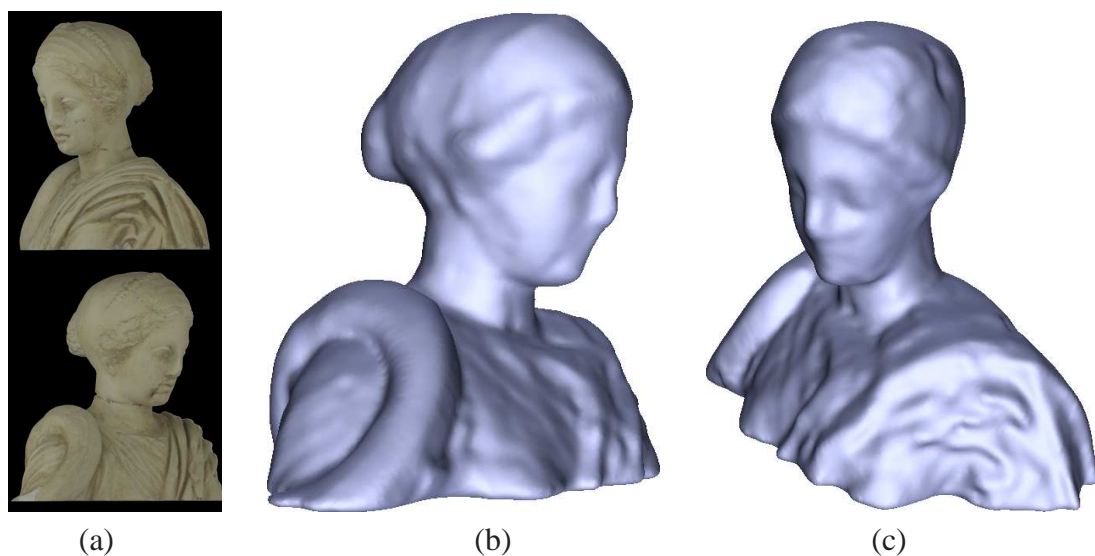
Figure 6.26: (a) Two of the 36 input photographs of the STATUE2 dataset. (b),(c) The reconstructed 3D model shown from two novel viewpoints.
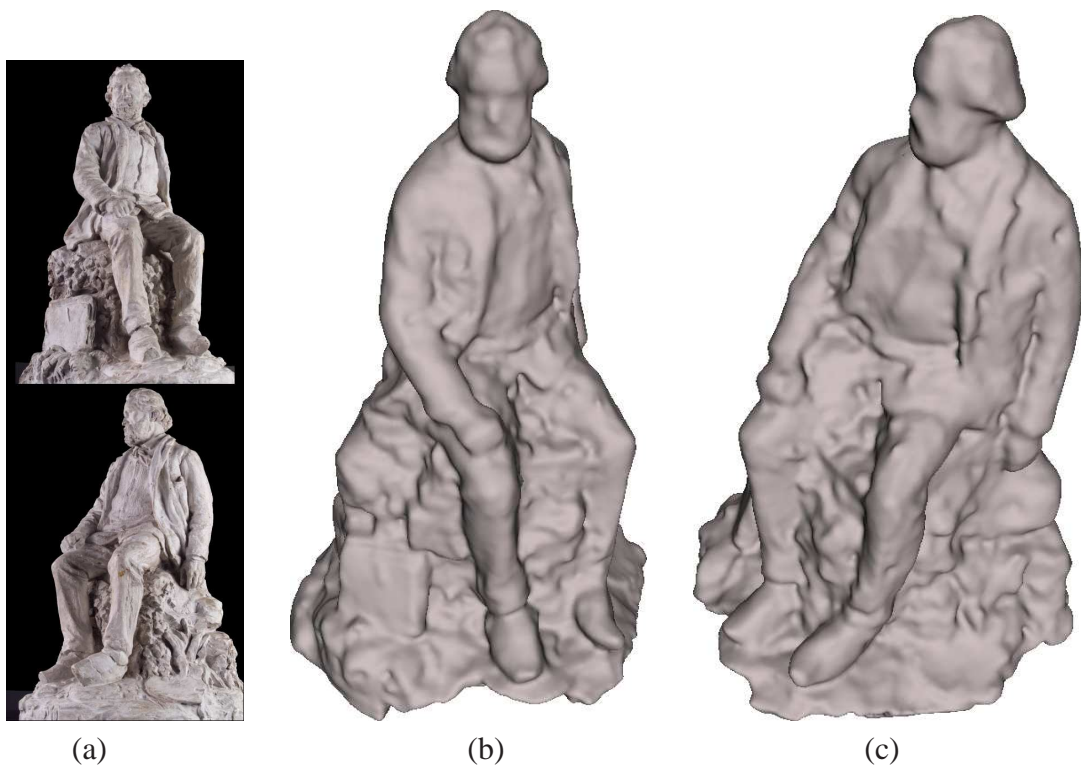
Figure 6.27: (a) Two of the 36 input photographs of the STATUE3 dataset. (b),(c) The reconstructed 3D model shown from two novel viewpoints.
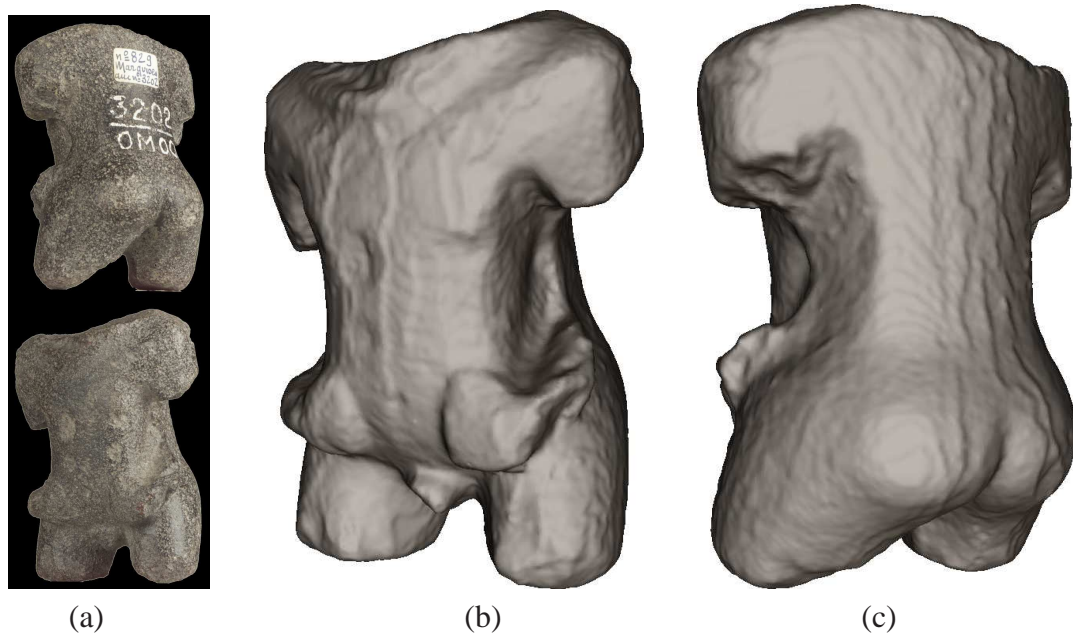
Figure 6.28: (a) Two of the 36 input photographs of the TORSO. (b),(c) The reconstructed 3D model shown from two novel viewpoints.

6.10 Conclusions

We have presented a multi-view reconstruction method that fuses the stereo cue with silhouette constraints while reconstructing static objects from image sequences. The main novelty in our approach lies in reconstructing the surface using volumetric graph cuts while enforcing exact silhouette constraints during the optimization. The special graph construction we propose, takes advantage of a two coloring property of the visual hull. As a result, the final reconstructed surface is always consistent with all the input silhouettes. We have demonstrated the approach on real and synthetic datasets using two formulations. The first formulation is only guaranteed to handle convex objects correctly but in practice is able to reconstruct non-convex shapes as well. The second formulation solves a graph-cut problem which is twice as large, but is able to correctly handle the case of T-junctions on silhouettes by computing a double manifold cut surface.

One of the weaknesses in our current approach is the method for computing the sheet surfaces within the visual hull volume, which are required by both the formulations. The current approach for computing the sheet surface is based on the discretization of the signed 3D distance function on a uniform volumetric grid. The approximations involved in this step cause a problem in computing the sheet surfaces for rim edges at sharp or thin protrusions on the visual hull surface. A better implementation of this step will be explored in the future.

The next chapter will focus on the high computational and memory overhead of the underlying volumetric graph-cut algorithm. Unfortunately the first formulation creates a graph twice as large as the graph in the original domain while the second formulation creates a graph that is four times as large. However, the second formulation contains two identical subgraphs, only one of which needs to be solved by virtue of the symmetry in the construction. The dynamic graph-cut algorithm from Kohli et. al. [74] could be used to compute the final solution once the partial solutions are available. This will be explored in the future.

CHAPTER 7
Adaptive Volumetric Graph-cut Stereo

7.1 Introduction

In this chapter, we propose an alternate formulation for multi-view 3D shape reconstruction. Here the 3D reconstruction problem is formulated as computing a minimum cut on the dual graph of a semi-regular, multi-resolution, tetrahedral mesh. This method addresses the high memory and computational requirements of the volumetric graph-cut stereo approach. The key idea is to sample the photo-consistency volume adaptively and avoid evaluating it in regions unlikely to contain any surface elements.

Contrary to the approach presented earlier, this method does not assume that the surface lies within a finite band around the visual hull or any other base surface. Instead, it uses photo-consistency to guide the adaptive subdivision of a coarse mesh of the bounding volume. This generates a multi-resolution volumetric mesh that is densely tesselated in the parts likely to contain the unknown surface. The graph-cut on the dual graph of this tetrahedral mesh produces a minimum cut corresponding to a triangulated surface that minimizes a global surface cost functional. Our method makes no assumptions about topology and can recover deep concavities when enough cameras observe them. Our formulation also allows silhouette constraints to be enforced during the graph-cut step (when they are available), to counter its inherent bias for producing minimal surfaces. Local shape refinement via surface deformation is used to recover details in the reconstructed surface. Reconstructions of the Multi-View Stereo Evaluation benchmark datasets and other real datasets show the effectiveness of our method.
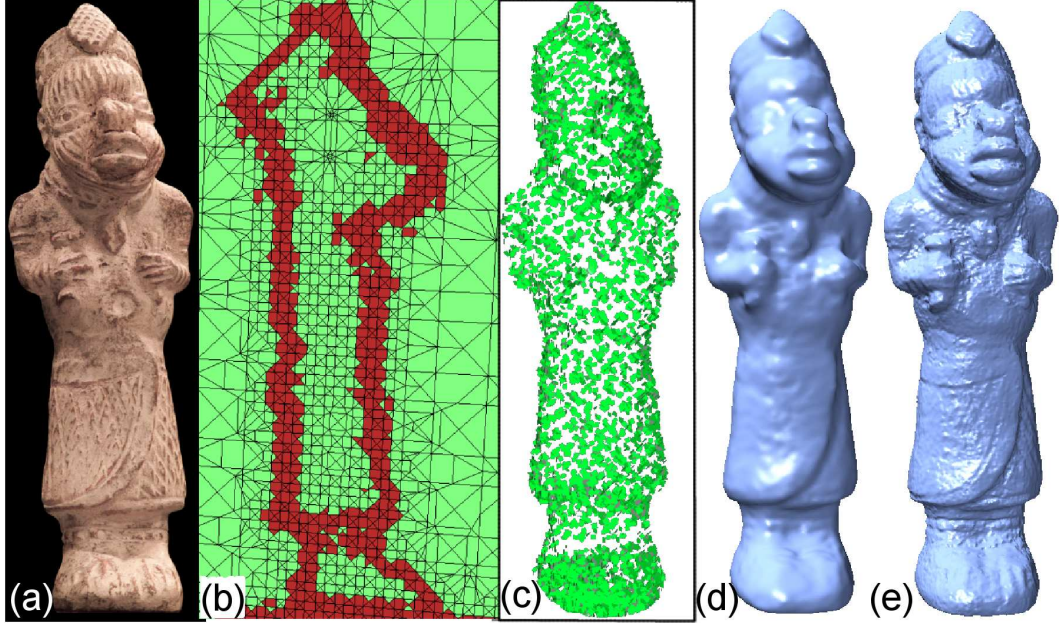
Figure 7.1: (STATUE1 dataset) (a) One of 36 input images. (b) A slice through the adaptive tetrahedral mesh showing the photo-consistent region in red (dark). (c) Quasi-dense patches produced during mesh refinement. (d) The 3D model obtained from graph-cut optimization. (e) The final refined 3D mesh.

7.2 Graph-cut on CW-complex

Let us assume that we are given a volumetric mesh M of the bounding volume with its set of cells and faces denoted by C and F respectively and that some of its cells have been labeled as interior and exterior to the unknown surface. The surface reconstruction problem can then be formulated as finding the most photo-consistent surface embedded within M, which separates the set of interior cells C_{in} from the exterior ones denoted by C_{out}. This can be achieved by minimizing a surface cost functional $\int_S \phi(s)\, ds$, where $\phi(s)$ represents the image discrepancy of an infinitesimal area ds of the unknown surface. In the discrete case, the energy functional becomes $\sum_S \phi(s)$ where S is a set of polygonal faces constituting a candidate surface.

The discrete optimization can be formulated as a binary graph-cut problem [85] on the dual graph of M denoted by $G(V, E)$. See Figure 7.2 for a 2D illustration. The vertices in V are dual to the cells of M while the directed edges in E are dual to the oriented cell boundaries

127
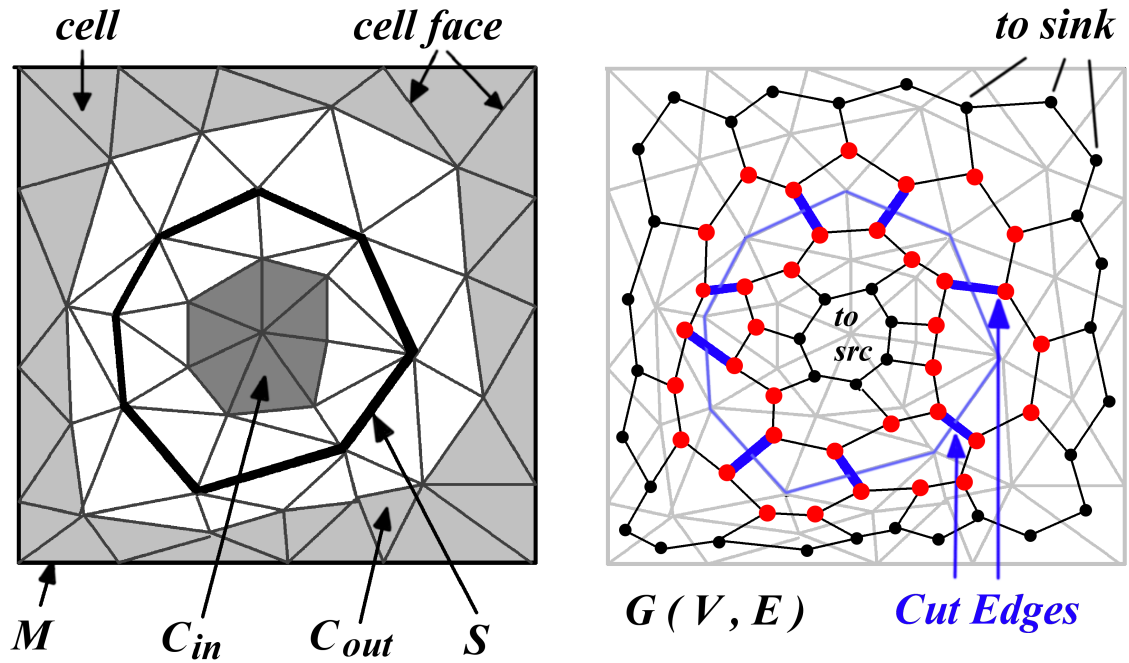
Figure 7.2: 2D illustration of the graph-cut formulation on the dual graph G of a volumetric mesh M (C_{in} and C_{out} are interior and exterior cells respectively). The value of a cut on G is equal to the cost of a surface S embedded within M.

(faces) of M. The capacity of an edge in E can be derived from the photo-consistency cost of its dual polygonal face. The main difference from the approach presented in Chapter 6 is that in this method, the photo-consistency is evaluated directly on tiny surface elements (cell faces) which are embedded within the volumetric mesh. The vertices in V representing cells in C_{in} and C_{out} are connected to the source and sink vertices in the flow graph using edges with infinite capacities. The minimum cut on G can be computed in low-order polynomial time and corresponds to a surface which gives a global minimum of the surface cost functional.

We choose M to be a tetrahedral mesh, motivated by their popularity in the mesh generation literature [103] and the fact that a minimum cut on its dual graph directly produces a triangulated surface. The rest of the chapter first describes how to build a suitable adaptive, tetrahedral mesh M using recursive subdivision. Later, we describe how additional cues such as visibility of photo-consistent patches and silhouette constraints (when available) can

be incorporated into the optimization framework leading to more efficient and accurate 3D reconstruction.

7.2.1 Limitations

Graph cuts on CW-complexes (duals of volumetric meshes) were first used by [73] for optimizing surface functionals. Later such methods for volumetric stereo were proposed [15, 85] with the advantage over [148] that these did not require initialization via the visual hull. However, to achieve a high quality reconstruction, this method must uniformly tessellate the volume into millions of tiny tetrahedra. Hence the computational and memory requirements of this algorithm is extremely high and it is not practical for reconstructing detailed 3D models which becomes possible when high-resolution images are available. The use of a uniform CW-complex creates this prohibitive memory bottleneck, thus limiting them to complexes with coarser resolutions.

Hornung and Kobbelt [63] proposed a multi-resolution approach also based on complexes (dual graph embedding of a uniform voxel grid). Although they propose a multi-resolution approach to address the memory bottleneck, their method relies on a good visual hull for initialization. A limitation of their coarse to fine approach is that it can result in sub-optimal solutions and thus the main advantage of the graph-cut based energy minimization framework is lost. Running the graph-cut at multiple resolutions in a coarse to fine manner involves making an early commitment at each of those stages. If the solution of the coarse graph-cut problem misses any fine structures on the surface, it may be impossible to recover them in subsequent iterations.

Both these formulations suffer from the inherent minimal surface bias which in the past has been addressed by introducing a naive ballooning term in the energy function. However, this introduces its own problems (see Section 5.4.1). These method do not try to use additional information that could be used – namely multi-view visibility information and silhouette cues.

These limitations will be addressed in our work.

7.3 Key Ideas

Existing graph-cut based reconstruction methods [85, 148] first densely sample voxels on uniform grids to build a graph embedding and then evaluate photo-consistency at all these voxels. The photo-consistency evaluation step is much more expensive than solving the actual graph-cut optimization, particularly when using robust patch-based similarity measures such as NCC. The key idea in our approach is that we adaptively sample the photo-consistency volume and construct a CW-complex whose resolution adapts according to the photo-consistency – thus the cells are finer at places where surfaces are more likely to exist (as indicated by high photo-consistency) and coarser elsewhere.

To construct such an adaptive CW-complex, we start with a coarse, uniform tetrahedral mesh. A novel photo-consistency driven mesh refinement strategy is then used to decide which elements of the coarse mesh need refinement. The strategy avoids subdividing and evaluating photo-consistency in those parts of the volume where surface elements are unlikely to exist. The mesh refinement is done using a recursive subdivision scheme on a coarse, regular, tetrahedral mesh and adaptively refining the most photo-consistent regions until the desired level of tessellation is reached in the photo-consistent parts.

For textured surfaces, this provides huge memory savings as bands of photo-consistency in the volume are usually extremely thin. Textureless surfaces however, tend to create wider bands of photo-consistency; such regions need to be finely sampled in our mesh and the savings are somewhat reduced there. High resolution is critical for accurate 3D reconstruction and our method aims at maximizing sampling density wherever needed. Along the lines of [63, 85], the presence of 12 different oriented faces in the mesh (as opposed to 6 in a uniform grid) reduces the discretization or metrication error in the cut surface. This photo-consistency driven mesh refinement strategy is now described.

7.3.1 Photo-consistency driven tetrahedral mesh refinement

Our base mesh denoted by M_0 is a body-centered cubic (BCC) lattice which comprises the nodes of a 3D grid along with the centers of all the cubic cells (see Fig. 7.3(a)). It can be thought of as two interlaced cubic lattices. Edges are added between every node in the first grid and its eight diagonal neighbors in the second grid.
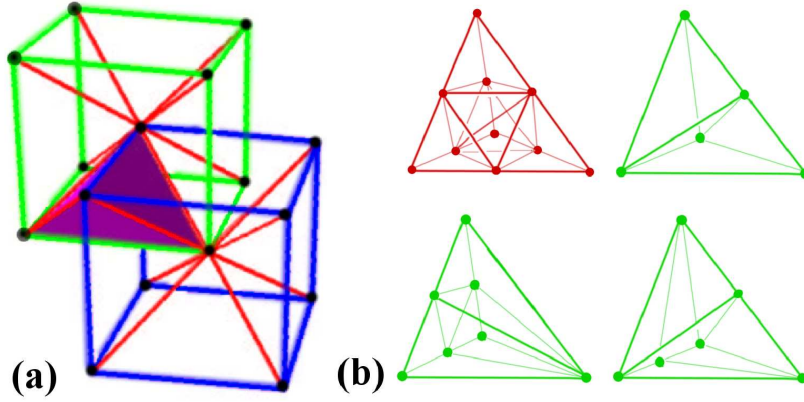
Figure 7.3: (a) Tetrahedral cell in a BCC lattice (two interlaced grids with diagonal edges added). (b) (Top-left) Red-Refinement (1:8) subdivision. (Rest) Green Refinement (1:2, 1:4) subdivision.

We choose a simple red-green mesh refinement strategy [103] to obtain a semi-regular mesh from M_0. The mesh obtained after i subdivision steps will be denoted by M_i and its tetrahedral cells and triangular faces by C_i and F_i respectively. A subset of cells in C_i which lie in the photo-consistent region, referred to as the *active* region will be denoted by A_i. The refined mesh M_{i+1} is obtained by applying red-refinement to the cells in A_i and green-refinement to the cells in $C_i - A_i$.

A tetrahedron is red-refined into eight tetrahedra as shown in Figure 7.3(b) by bisecting its six edges. The shortest of the three possible diagonal edges internal to the tetrahedron must be chosen to make the eight new tetrahedra geometrically similar to the original cell. Green tetrahedra which share faces with red tetrahedra require between one to five edge-splits. Similar to [103], we reduce the various cases of green refinement to the three shown in

131

Figure 7.3(b). Green tetrahedra are not geometrically similar to the original BCC tetrahedra and are never subdivided any further.

A photo-consistency measure $g(X) : R^3 \rightarrow R$ which computes the likelihood of the 3D point X of being a true surface point is used to find the *active* set $A_{i+1} \subset C_{i+1}$, excluding cells created by green refinement. When the unknown surface passes through a tetrahedral cell, some of its four faces must contain points with a high measure of photo-consistency. We refer to these as *crossing* faces.

If none of the faces of a cell contain any photo-consistent points, that cell cannot contain a piece of the surface. We assume that the surface is larger than the smallest tetrahedral cell and thus cannot be fully contained within any of the cells. We do not refine such cells any further and avoid sampling in their interior.

Assuming that the unknown object is large enough not to be completely contained inside a single tetrahedron, a cell must have at least one *crossing* face in order to be labeled *active*. To determine A_{i+1}, we evaluate $g(X)$ on the faces of cells created by red-refinement of A_i and determine the subset of *crossing* faces. Then each *crossing* face labels its two neighboring tetrahedral cells as *active*. This is illustrated in Figure 7.4.
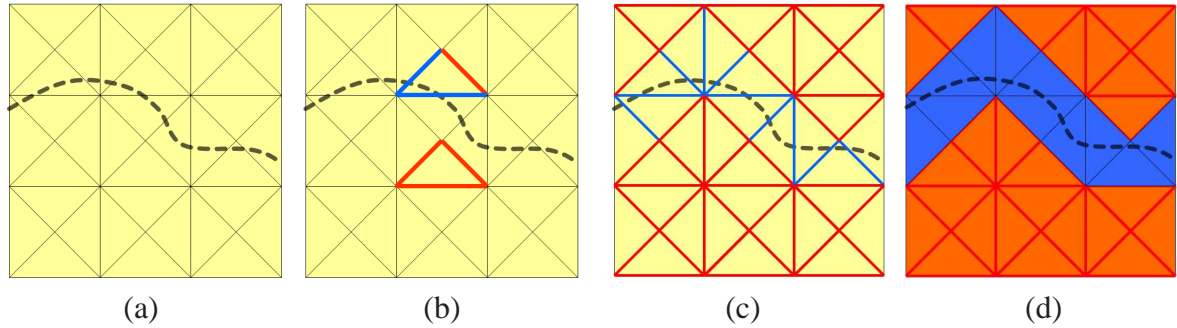
Figure 7.4: A 2D illustration of the tetrahedral refinement scheme. (a) Unknown surface embedded within coarse tetrahedral grid (each triangle represents a tetrahedral cell). (b) Check for photo-consistency on the cell faces. (c) Crossing faces are shown in blue, rest are in red. (d) The set of active cells are shown in blue – these will be subdivided in the next iteration.

132

7.3.2 Computing Photo-consistency

To determine whether a face f is a *crossing* face, we sample a triangular lattice on it as shown in Figure 7.3(c). The spacing between samples in the lattice is selected to prevent aliasing by ensuring that no pixels are skipped when the lattice is projected onto the images. At each lattice position X, we use the normalized cross correlation (NCC) of the image projections of patches placed at X to measure its likelihood of being on the surface. Since the mesh is initially coarse, its faces may not be parallel to true surfaces. In this case, it would be undesirable to compute NCC on the faces themselves. To overcome this, we place multiple patches with different orientations at each point X. The patches and the set of images used for the computation are determined as follows.

At X, we place patches at multiple orientations, each denoted by unit vector n_X. For all points, n_X is chosen from 14 canonical orientations sampled along the corners and faces of a unit cube. For a given orientation n_X, we choose the best k cameras such that the angle between n_X and the direction vector from X towards the camera is at most $60°$. Let us denote this subset of cameras by $P(X)$. If X is a true surface point and n_X is a fair approximation of the surface normal on the unknown surface, then the projection of that patch should have a high matching score for the subset of visible cameras $\subset P(X)$.

Since we are only interested in determining whether a point could potentially belong on a surface or not, we use a simple computation for the photo-consistency to reduce computational complexity. We simply place a 1D $1 \times \mu$ grid along the intersection line of the patch and the underlying face f (see Figure 7.5). This direction is given by $n_X \times n_f$, where n_f is the normal of the face. This $1D$ grid is now projected into each of the cameras in $P(X)$ and pairwise NCC scores are computed for all such pairs.

The photo-consistency score for each camera in $P(X)$ is computed by averaging the best k' NCC scores with the other $(k\text{-}1)$ cameras ($k' = \max\{k/2, 3\}$) allowing for matching to succeed despite some occlusion. The score of the best overall camera is retained as the score

133
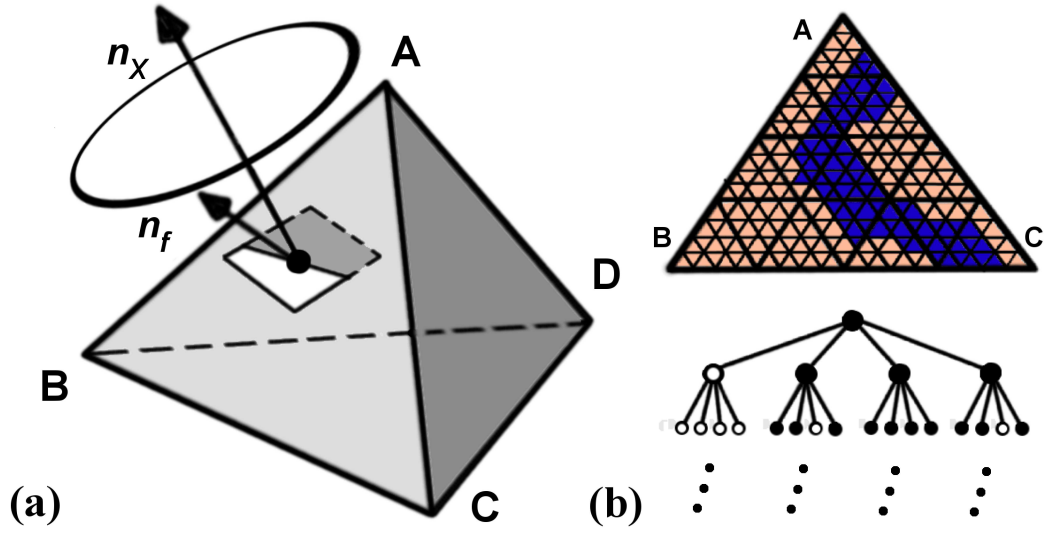
Figure 7.5: (a) Testing if face ABC with face normal n_f is a *crossing* face; Test patch at P with unit normal n_X for photo-consistency. (b) Computation performed on a triangular lattice (photo-consistent points are blue (dark)) with the results stored in a quad-tree associated with face ABC. Nodes corresponding to *crossing* faces are black in the quad-tree.

for the patch at X with orientation n_X. Points with score larger than a global threshold T are labeled photo-consistent. Finally, if a face contains at least 20% photo-consistent points, or at least 20 points if 20% corresponds to a number below 20, we declare it to be a *crossing* face.

This computation could be repeated for every face at every subdivision level during mesh refinement. However, this would be highly redundant since during each subdivision level, a large face f splits into four new faces f_1, f_2, f_3 and f_4 whose photoconsistency measures have already been computed to decide whether f was a *crossing* face.

The solution then is to perform the computation recursively for face f only once and store the results in a quad-tree associated with f (see Figure 7.3(b)). Concretely, the root node of the quad-tree corresponds to f while the four children correspond to the faces $\{ f_i \mid 1 \leq i \leq 4\}$ obtained by bisecting the three edges of f and connecting the mid-points. At each tree node we store: (1) the number of photo-consistent samples (those with matching score $> T$) on the triangle lattice, (2) the total sample-count and (3) the best oriented point for f along-with the set of cameras it correlated on.

All such oriented points form a *quasi-dense* reconstruction of the scene (see Figure 7.1(c)) and is next used to detect interior and exterior cells. When f is split during subdivision, the four new faces inherit the children of f's quad tree and can be immediately tested for being *crossing* faces.

7.3.3 Computing Cell Costs

Multiple iterations of mesh refinement produces a set of highly tesselated *active* cells. We will now try to classify some of the remaining cells into sets C_{in} and C_{out}. Since the visual hull contains the true shape, any cell which falls outside the visual hull can be automatically labeled as exterior. However, green tetrahedra contained within the visual hull i.e. the ones which were not fully subdivided could potentially belong to regions either interior or exterior to the surface (eg. a deep concavity).

The set of quasi-dense oriented surface points recovered during the photo-consistency driven mesh refinement (Section 7.3.1) allows us to determine which green tetrahedra are part of the true interior. An oriented point p that was photo-consistent in k' views must have been visible from each of those cameras. Hence we path-trace rays from p to all of the camera centers and vote for each cell that the ray intersects along the way. This can be done efficiently by walking along the ray within the tetrahedral mesh and performing ray-triangle intersections. Finally amongst all the green tetrahedra contained within the visual hull, the ones which received votes lower than the 10th percentile are labeled interior, while the ones with votes above the 75th percentile are labeled exterior. Since labeling cells as interior and exterior imposes hard constraints in the reconstruction, we apply the labels conservatively and leave ambiguous cells undecided ie. we re-label them *active*.

7.4 Approach

Our complete approach summarized in Algorithm 3 begins with tetrahedral mesh generation described in Section 7.3, followed by the first graph-cut on its dual. This is followed by a second graph cut after interior and exterior sets are augmented by enforcing silhouette constraints and a final local refinement. The graph construction and approach for incorporating silhouette constraints are described next. Finally the method for local shape refinement is briefly described.

Input: images $\{I\}$, cameras $\{P\}$, bounding-box B
Output: polygonal mesh model H

$M_0 \leftarrow$ `BuildBCCMesh`(B);
$Q = \{\ \}$;
for $i \leftarrow 0$ **to** m-1 **do**
    patches $\leftarrow$ `ComputeMatchingCost`(F_i);
    $A_i \leftarrow$ `FindActiveCells`(M_i, F_i);
    $M_{i+1} \leftarrow$ `MeshRefine`(M_i);
    $Q \leftarrow Q \cup$ patches ;
end
$C_{in}, C_{out} \leftarrow$ `MarkInteriorExterior`(M_m, Q) ;
$G \leftarrow$ `SetupGraphCut`(M_m, C_{in}, C_{out}) ;
$[S_1, C_{in}^1, C_{out}^1] \leftarrow$ `FindMinCut`(G) ;
foreach *camera* j *in* $\{P\}$ **do**
    $K_j \leftarrow$ `RenderSilhouettes`(S_1, P_j) ;
end
$C_{in}^a = C_{in}^1;\quad C_{out}^a = C_{out};$
foreach *camera* j *in* $\{P\}$ **do**
    $C_{in}^a, C_{out}^a \leftarrow$ `EnforceSilhouettes`(I_j, K_j) ;
end
$G' \leftarrow$ `SetupGraphCut`(M_m, C_{in}^a, C_{out}) ;
$S_2 \leftarrow$ `FindMinCut`(G') ;
$H \leftarrow$ `RefineShape`(S_2);

Algorithm 3: The Complete Algorithm.

7.4.1 Graph Construction

Having generated a tetrahedral mesh M and sets of voxels, C_{in} and C_{out} we then construct G, the dual graph of M. Vertices in G dual to cells in C_{in} and C_{out} are connected to the *source* and *sink* respectively for the graph-cut. Edge capacities in G are derived from the dual oriented faces in M. Unlike in Section 7.3.1 where 1D patches were used for speed, the goal here is to minimize a true surface cost functional. To this end, a 2D $\mu \times \mu$ grid, placed on each face f, is projected into the images and their pairwise NCC scores are combined. We pick the best k cameras at an angle of at most $60°$ from the surface normal of f. Each of these is chosen as a reference view (as in Sec. 7.3.1) and correlated with the other k-1 views; the best k' ($k' = \max\{\frac{k}{2}, 3\}$) scores out of these truncated to [0,1] are averaged. The best average score is assigned as the final score ω_f of f. Eq 7.1 shows how ω_f maps to the edge weight $\phi(f)$ where a_f is the area of face f.

$$\phi(f) = \left(1 - \exp\left(-\tan\left(\frac{\pi}{2}(\omega_f - 1)\right)^2 / \sigma^2 \right) \right) . |a_f| + \lambda . |a_f| \tag{7.1}$$

As explained in [148], minimizing the surface functional $\sum_S \phi(s)$ over surfaces S embedded in the volume is equivalent to finding the minimal surface with respect to a Riemannian metric [13] where higher values of σ and lower values of λ produce a more photo-consistent but less smooth surface and vice-versa.

7.4.2 Enforcing Silhouette Constraints

Variational surface reconstruction approaches have a bias for smaller shapes, as surfaces with a lower total cost are preferred over a more accurate surface which has lower cost per unit area but higher total cost. The energy can be regularized by including a *ballooning* term [85, 148] which acts as a prior for larger shapes. While this can recover protrusions, it also pushes the concave parts outwards thereby significantly reducing the accuracy of the final result. While

[58] proposes visibility-based *intelligent* ballooning to address this issue, it only reduces the graph cut bias and preserves concavities better but does not guarantee consistency with the silhouettes. We address this in a different way by enforcing hard constraints in the graph-cut derived from both visibility as well as silhouettes constraints.
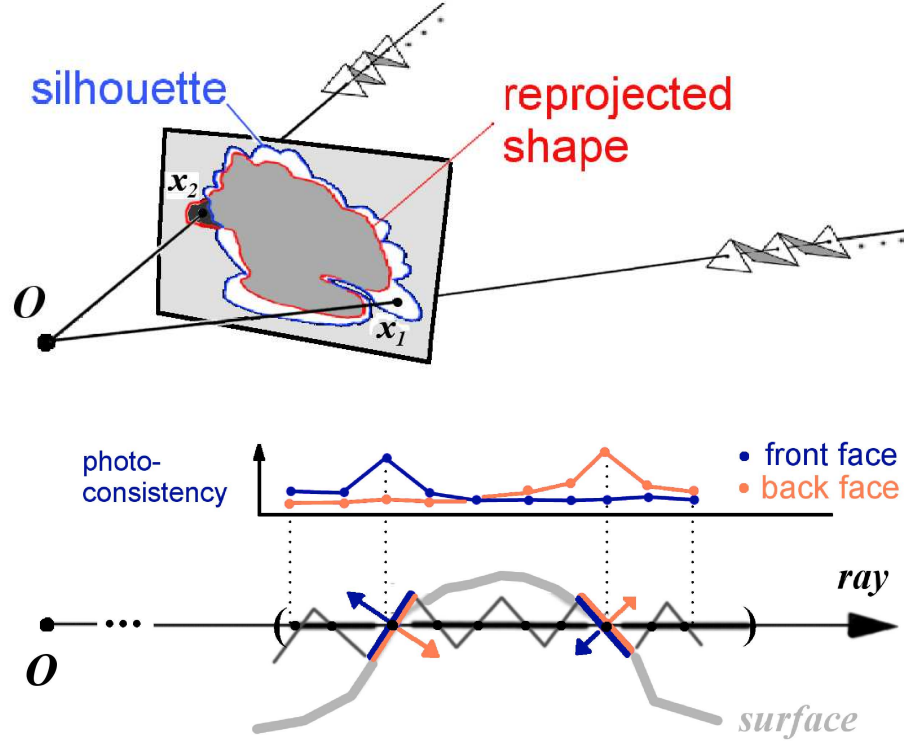
Figure 7.6: Top: the original silhouette $\mathcal{S}$ and re-projected silhouette $\mathcal{S}_r$ (O is the camera center). x_1 and x_2 indicate re-projection errors (see text for details). Bottom: for x_1, we inspect photo-consistency costs on front and back-faces for all triangles in M which are intersected by the ray back-projected from x_1.

Figure. 7.6 shows $\mathcal{S}_r$ the re-projected silhouette overlaid on the original silhouette $\mathcal{S}$. The re-projection errors are in pixels such as x_1 which fall inside S but outside $\mathcal{S}_r$ and x_2 which fall inside $\mathcal{S}_r$ but outside $\mathcal{S}$. Consider the rays r_1 and r_2 backprojected from x_1 and x_2 and the cells they intersect. The ray r_2 should not meet surface because x_2 is outside the silhouette $\mathcal{S}$, therefore all cells intersected by r_2 can be safely labeled added to C_{out}. On the other hand, r_1 must intersect the surface at least twice. Thus at least one of the cells that r_1 passes through

must be an interior cell. For such rays in every view, we intend to mark at least one such cell as interior and add it to our set of interior voxels.

We adopt a two-step approach. First, by computing the minimum cut on G as described above, we obtain (a) a triangulated surface (b) a partition of all tetrahedral cells into C_{in}^1 (interior cells) and C_{out}^1 (exterior cells). The triangulated surface is then re-projected into all the images and the sets of erroneous pixels (such as x_1 and x_2) are determined. Pixels such as x_2 add cells to the set C_{out}. Pixels such as x_1 are processed to mark some additional cells in M as interior; these are added to C_{in}^a, the augmented set of interior cells. The candidate cell is chosen as follows. We first find the sequence of tetrahedral cells in M that ray r_1 cuts through and sort them by distance from the reference camera. Cells in this sequence that fall outside the visual hull are excluded, leaving groups of contiguous tetrahedral cells each of which we will refer to as a *segment*. For each *segment*, we orient the triangles (faces of the cells) consistently with respect to the ray.

Let us first consider the simpler case when r_1 intersects the surface twice (see Fig. 7.6). This ray must meet a triangle f_f whose *front-face* is photo-consistent before it meets a triangle f_b whose *back-face* is photo-consistent. A few tetrahedral cells within such a depth-interval can be chosen as interior. More specifically, we look for the maxima of front-face photo-consistency and find the next significant peak in back-face photo-consistency (within 0.8 of the global maximum) for faces along r_1 in the same *segment* to determine a conservative depth interval for the interior. We then pick the center-most cell in this depth interval and add it to C_{in}^a. This step is highly redundant and we pick candidates (a hard constraint) only when we are sure about a cell being interior. We skip pixels with multiple *segment*s per ray and let a more favorable view enforce silhouette constraints there. In our experiments processing only a few pixels was sufficient to recover all the protrusions. It is better to enforce a minimal number of additional hard constraints for silhouette consistency since performing this step exhaustively increases the likelihood of including incorrect constraints.

A second minimum-cut is now computed on the same graph G but with the augmented interior set C_{in}^a as *source* and augmented exterior set C_{out}^a as *sink*. This new triangulated surface satisfies silhouette constraints upto a few pixels (the actual value depends on the cell resolution of M and is typically in the range of 1-5 pixels in the images. An analogy can be drawn between our approach and the graph-cut based Grab-cut [112] segmentation method, where iterative graph-cut optimization is performed while the user interactively provides additional hard constraints. In a similar fashion, we use silhouettes for generating reliable hard constraints (automatically in our case) as described above and perform a second graph-cut iteration to correct the shortcomings of the first one.

7.4.3 Local Surface Refinement

Finally, local optimization is used to refine the shape locally to remove discretization errors introduced by the graph cut reconstruction. The triangulated minimum-cut surface mesh is iteratively deformed and remeshed during local refinement. This is similar to the approaches of Furukawa et. al. [47] and Hernandez et. al. [57]. Vertices of the mesh are displaced by a combination of smoothness, silhouette and texture forces. The smoothness force is computed by the approach of [150] which prevents the surface from shrinking while silhouette forces are computed as described in [47]. To compute the texture force, we use the normal vector at a vertex to determine a set of k cameras and compute a photo-consistency score (see Section 7.4.1) at multiple offsets from the current vertex location along its surface normal. A red-green 2D mesh subdivision scheme [103] is used to remesh the model after bisecting edges which project to more than 2 pixels in the best reference view.
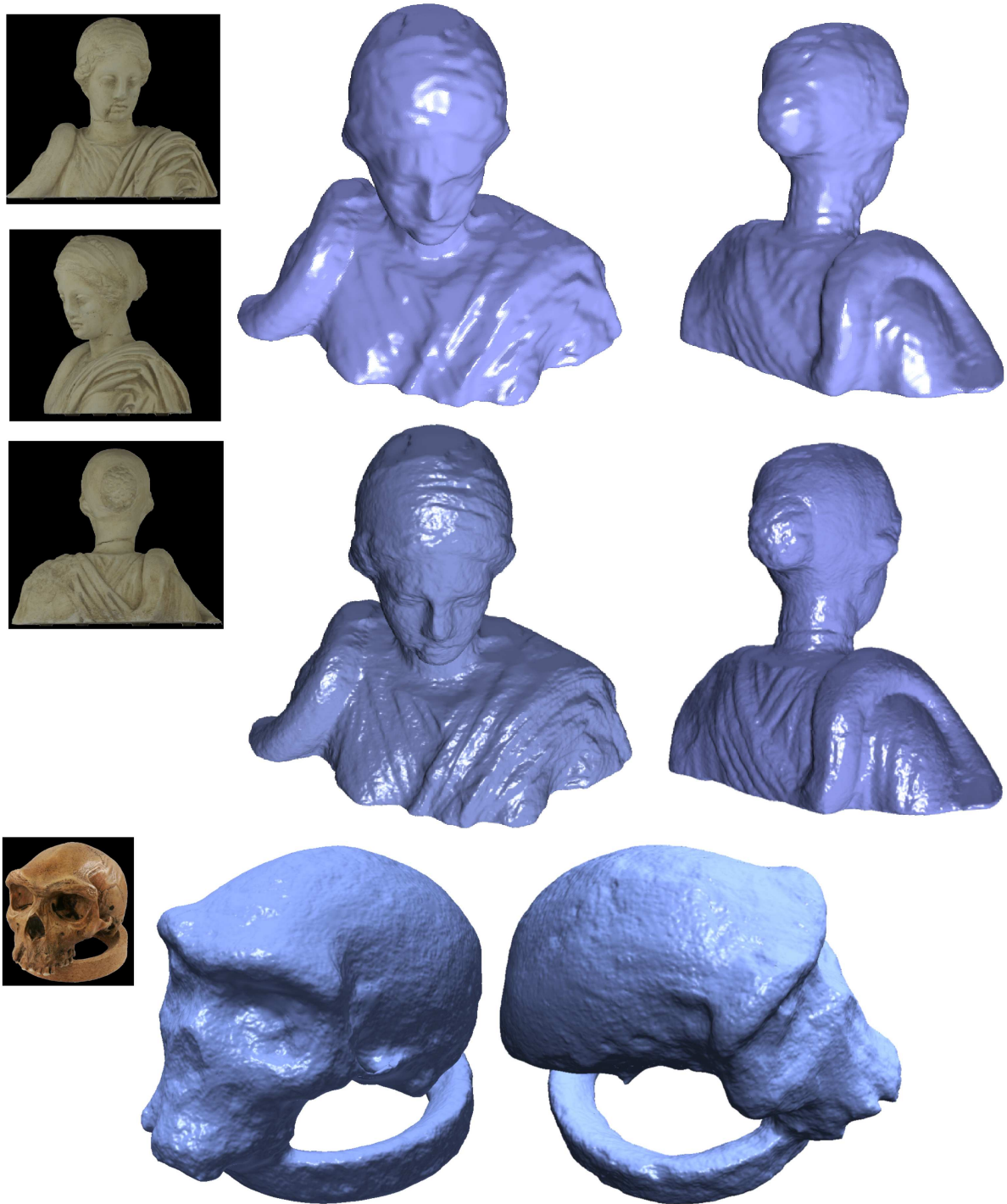
Figure 7.7: (Top) STATUE3 dataset: Three of the input images. The reconstructed surface from the graph-cut step is shown on the top row while the final 3D model after refinement is displayed in the middle row. (Bottom) SKULL dataset: Two views of the final model.

141
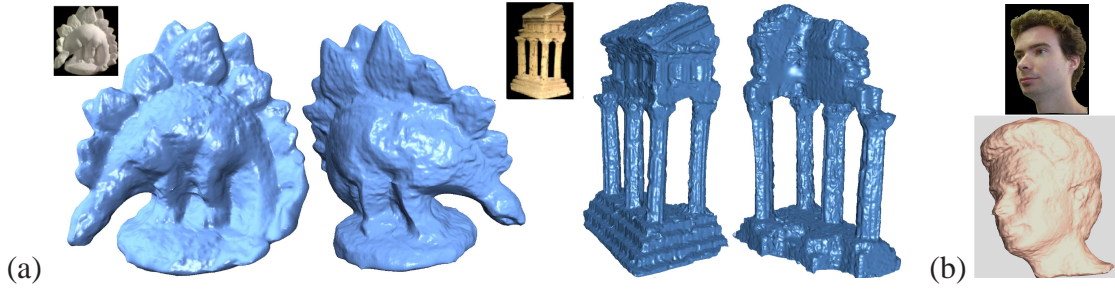
Figure 7.8: (a) Middlebury Multi-view Stereo Evaluation benchmarks: (left) DINORING, (right) TEMPLERING. (b) HEAD dataset reconstructed from a set of 21 images lacking perfect color calibration.
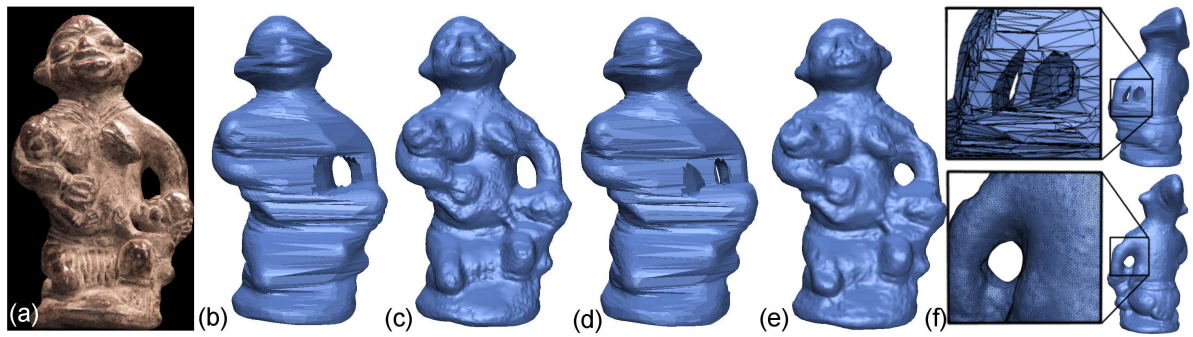
Figure 7.9: STATUE2 dataset: (a) One of the input images (note that in our experiments we leave this image out). (b) Visual Hull from all 36 images. (c) Our result using all 36 images. (d) Visual Hull from 26 images (10 out of 14 views which see the gap between arm and body are left out) has genus 3. (e) Our model using these 26 images has the correct topology (genus 1). (f) Zoomed-in rear view of (top) visual hull (bottom) our result using these 26 views.

7.5 Results

We have reconstructed several real multi-view datasets using our approach as shown in Figures 7.1, 7.7 – 7.9. Datasets STATUE1, STATUE2 and STATUE3 contain 36 images (6 Mpixels) each, captured using a turntable. The HEAD dataset contains 21 640×480 images without good color calibration while the SKULL dataset contains 24 2000×2000 images.

We have evaluated our approach using the Multi-View Stereo Evaluation (the reconstructions are shown in 7.8(a)). This evaluation provides metrics on the accuracy and completeness of the reconstruction. The accuracy metric is the distance d such that 90% of the reconstructed is within d from the ground truth surface. Completeness is defined as the percentage of the

Middlebury Evaluation	TEMPLE RING (47 views)		DINORING (48 views)	
	Accuracy (mm)	Completeness (%)	Accuracy (mm)	Completeness (%)
Furukawa [45]	0.58	98.5	0.42	98.8
Furukawa2 [44]	0.55	99.1	0.33	99.6
Gargallo [48]	0.88	84.3	0.60	92.9
Goesele [50]	0.61	86.2	0.46	57.8
Goesele2 [51]	0.42	98.2	0.46	96.7
Habbecke [54]	0.66	98.0	0.43	99.7
Hornung [63]	0.58	98.7	0.79	95.1
Kolev [76]	0.79	96.0	0.53	96.9
Kolmogorov [77]	1.86	90.4	2.81	86.0
Merrell [101]	0.83	88.0	0.84	83.1
Merrell2 [101]	0.76	85.2	0.73	73.1
Pons [110]	0.60	99.5	0.55	99.0
Sinha [108]	**0.79**	**94.9**	**0.69**	**97.2**
Sormann [130]	0.69	97.2	0.81	95.2
Strecha [133]	0.86	97.6	1.21	92.4
Tran [140]	1.12	92.3	1.12	92.0
Vogiatzis [148]	0.76	96.2	0.49	96.7
Zach [158]	0.58	99.0	0.67	98.0

Table 7.1: Table lists the accuracy and completeness scores of various multi-view stereo methods on the Middlebury benchmark. The accuracy and completeness were computed using thresholds of 95% and 1.25mm, respectively. See text for details.
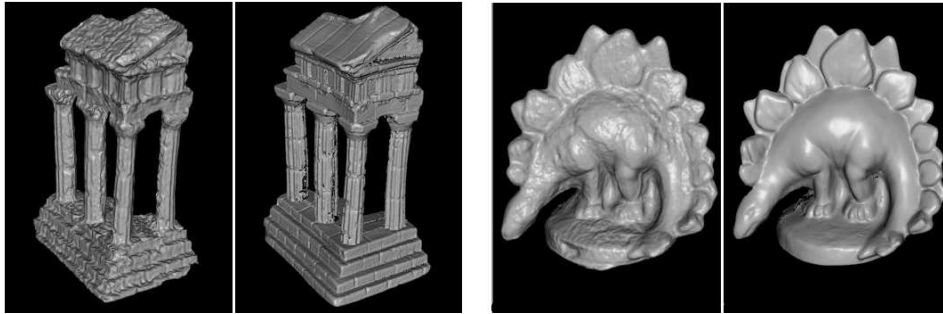
Figure 7.10: (a) Middlebury Multi-view stereo evaluation benchmarks: (left) DINORING, (right) TEMPLERING. The 3D models obtained by our method in comparison to ground truth. calibration.

ground truth surface within $1.25mm$ of the model. The accuracy and completeness of our reconstruction for the 47-view *templeRing* dataset were $0.79mm$ and 94.9% respectively. The same metrics for the 48-view DINORING dataset were $0.69mm$ and 97.2%. See Table 7.1 for

a comparison with other methods.

Fig. 7.9 illustrates the results of an experiment performed to demonstrate that our method is not limited by the topology of the base surface. While the visual hull built from all 36 images has the correct topology, the visual hull built after omitting 10 images (the separation between the arm and body is observed in these) has genus three. Our method still recovers a model with the correct topology (see Fig. 7.9(e,f)). The critical parameters of our algorithm are as follows. The patch size μ is typically 11 pixels while the photo-consistency threshold T is chosen in the range of 0.4-0.7 (a fraction between 0 and 1). A lower T is more conservative and retains more cells as *active*. The surface functional parameters of σ is set to 0.1 in all our experiments and λ is varied between 1 to 10. The stopping criterion for recursive mesh refinement is based on the size of the finest cells in the images; we typically stop when this is in the range of 1 to 3 pixels. Our method requires a smaller fraction of graph vertices compared to approaches which construct uniform grid graphs in the interior of the visual hull. Our mesh typically has between 2-10 million cells and total running time are typically 1 to 2 hours for each reconstruction.

7.6   Conclusions

We have presented a multi-view reconstruction method that addresses the high memory and computational requirements of volumetric graph-cut stereo, by performing a graph-cut on the dual of an adaptive volumetric mesh (a CW-complex) created by photo-consistency driven recursive mesh subdivision. It does not need any initialization, and is not restricted to a specific surface topology which is a limitation with methods that use a base surface for initialization. Our graph-cut formulation also incorporates visibility and silhouette constraints to counter the bias for minimal surfaces, and recovers highly detailed 3D model.

CHAPTER 8
Conclusion

8.1 Summary

In this dissertation, I have studied how silhouettes extracted from images and video can help with two fundamental problems of computer vision - namely, multi-view camera calibration and 3D surface reconstruction from multiple images.

First, I presented an automatic method for calibrating a network of cameras that works by analyzing only the motion of silhouettes in the multiple video streams. This is particularly useful for automatic reconstruction of a dynamic event using a camera network in a situation where pre-calibration of the cameras is impractical or even impossible. Our key contribution is a novel RANSAC-based algorithm that simultaneously computes the epipolar geometry and synchronization of a camera pair from only the silhouettes of moving objects. The approach starts by independently computing the epipolar geometry and synchronization for various camera pairs in the network. In the next stage, the calibration and synchronization of the complete network is recovered. We remotely calibrated about ten different camera networks that researchers have setup in their own labs for acquiring models of human actors and other applications in computer vision and graphics. We did this only from the archived multi-view video streams that were previously captured by these camera networks. No additional data capture was required in order to run our calibration approach. This demonstrates the effectiveness of the proposed method.

In the second part of the dissertation, I addressed some shortcomings of existing volumetric multi-view stereo approaches. First, I proposed an improved multi-view stereo formulation

that allows for robust and accurate fusion of the silhouette and stereo cues. I showed that it is possible to enforce exact silhouette constraints within the graph-cut optimization step of the volumetric graph-cut stereo algorithm. Hence the reconstructed surface can be guaranteed to be consistent with the original silhouettes. I also described an alternate multi-view stereo formulation involving an adaptive graph construction, which addresses the high memory and computational overhead of the underlying approach. The proposed method does not need any initialization and is not restricted to a specific surface topology. Using the method, accurate and detailed 3D models have been reconstructed from high-resolution images.

8.2 Directions for Future Work

I conclude this dissertation with some discussions on directions for future research.

8.2.1 Camera Network Calibration

The approach that was developed in the first part of this dissertation made it possible to reconstruct dynamic scenes and events from uncalibrated and unsynchronized archived video. All the necessary information was recovered from the silhouettes of moving objects. In our work, both camera intrinsics as well as extrinsics were assumed to be unknown. However, in some scenarios it is reasonable to assume that the intrinsics are known ahead of time. Then instead of estimating the epipolar geometry, one could directly estimate the relative pose. The approach to recover relative pose using only silhouettes should be further investigated, as this scenario may frequently occur in the context of camera network calibration.

We used a visual hull approach to reconstruct the dynamic scene from the archived video streams. However, much work remains to be done in the area of dynamic scene reconstruction. Some progress in this direction has already been made by Ballan et. al. [7], Furukawa et.al. [46], Stark et. al. [131] etc. Although we show the benefit of accurately synchroniz-

ing the video streams up to sub-frame accuracy, our silhouette interpolation scheme is simple and introduces errors when the inter-frame motion is quite large. In such scenarios, shape preserving silhouette interpolation such as [1] should be used.

Arguably silhouette extraction (i.e. background segmentation) is a difficult problem to solve in a general setting. Although our method is robust to outliers in silhouette extraction, it still requires a reasonably high percentage of accurate silhouettes which is harder to guarantee in realistic outdoor scenes. Future work should address this weakness, to make the technique less reliant on high quality silhouettes. To deal with real world scenarios, the technique needs to be extended to deal with silhouettes that are occluded by other static objects in the scene.

An interesting direction to pursue for camera network calibration in the outdoor world is to establish correspondence between high level objects in the images instead of trying to recover accurate point correspondences between them. A wide range of object detectors for specific object classes such as faces, pedestrians, cars etc. have been recently developed. This gives rise to the question – can the correspondences between image regions (without exact pixel to pixel correspondence) generated by running trained detectors on multi-view sequences produce enough constraints to recover the calibration of the camera network ? Given sufficient data, to what degree of accuracy can the calibration be computed ?

A related strategy for camera network calibration that would be interesting to investigate is whether model-based methods can be used. Both monocular and multi-view model-based techniques for motion capture and pose estimation of a class of objects (most research has been focussed on humans) have gradually improved over the last decade. The multi-view methods, such as [7, 22] require pre-calibrated cameras. An interesting question that comes up here is – can the camera network calibration and the pose of the human be recovered simultaneously ? This would be a significant step towards making such multi-camera systems easy to deploy and use in the real world.

8.2.2 Multi-view reconstruction

Although we have proposed two alternate formulations of multi-view stereo that address different issues in existing approaches, ideally they should be combined together. As our formulation that enforced exact silhouette constraints doubles the size of the graph problem, it can benefit from our alternate formulation that creates a sparser graph embedding. Although we only show how to enforce silhouette constraints within the graph-cut based energy minimization framework, it should also be possible to enforce the same silhouette constraints during mesh refinement, which is used for local shape refinement in the final stage of the reconstruction pipeline.

One of the important limitations of our work on multi-view stereo is that only closed objects are handled. However, there is also immense interest in the vision community in acquiring detailed models of open scenes. The proposed methods do not directly extend to such scenarios, and this is an important direction for future work. Some techniques for addressing this problem have been explored in recent work by Furukawa et. al. [46]. As segmentation techniques get better, it may be worth investigating whether silhouette constraints can be exploited for reconstructing surfaces in open scenes as well. The key difference there is that the surface cannot be assumed to be a closed one that partitions the 3D volume into an interior and exterior region.

Although volumetric energy minimization based approaches, such as the ones we proposed give high quality results, they do not scale well to large scenes. Combining sparse 3D reconstruction and structure from motion techniques with the advantages of energy minimization is an interesting direction of future work. Some promising results have been shown by [79]. Techniques similar to our adaptive graph construction, could possibly be used for energy minimization based depth map fusion.

Appendix A: Camera Models and Multi-view Geometry

This appendix briefly covers the theory of camera models, multi-view geometry, and image similarity metrics.

A-1 Camera Models and Multi-view Geometry

A-1.1 Pinhole Camera Model

The pin-hole or perspective camera model (shown in Figure 8.1) is commonly used to explain the geometry of image formation. Given a fixed center of projection C (the pin-hole or the
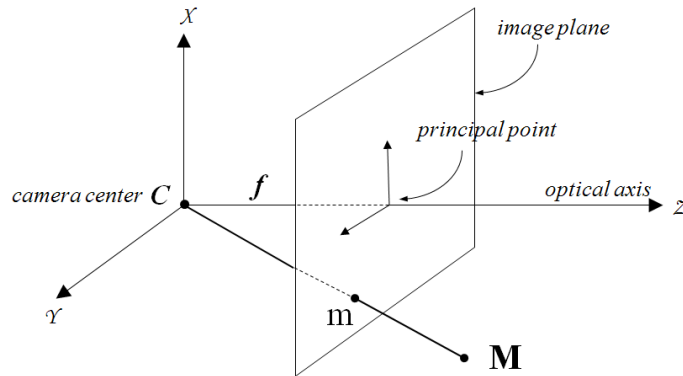
Figure 8.1:

camera center) and an image plane, every 3D point **M** (X,Y,Z) other than the *camera center* itself, maps to a 2D point **m** (x,y) on the image plane, which is assumed to be at a distance f from C. They are related as follows:

$$x = f\frac{X}{Z} \qquad y = f\frac{Y}{Z}$$

149

Using homogeneous coordinates for 2D and 3D points, this can be written in matrix form.

$$
\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}
\tag{A-1}
$$

The line through C, perpendicular to the image plane is called *optical axis* and it meets the image plane at the *principal point*. Often, 3D points are represented in a different world coordinate system. The coordinates of the 3D point in the camera coordinate system $\mathbf{M_c}$, can be obtained from the world coordinates $\mathbf{M_w}$, as follows:

$$
\mathbf{M_c} = \mathbf{R M_w} + \mathbf{t}
$$

Here $\mathbf{R}$ represents a 3×3 rotation matrix and $\mathbf{t}$ represents a translation vector for 3D points. This can be written in matrix form.

$$
\mathbf{M_c} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0^T} & \mathbf{1} \end{bmatrix} \mathbf{M_w} \qquad \text{i.e.} \qquad \mathbf{M_c} = \mathbf{T_w}\,\mathbf{M_w}
\tag{A-2}
$$

In images, the image coordinate system is typically not centered at the principal point and the scaling along each image axes can vary. So the coordinates of the 2D image point undergoes a similarity transformation, represented by $\mathbf{T_c}$. Substituting these into the perspective projection equation, we obtain:

$$
\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \mathbf{T_c} \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \mathbf{T_w} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}
\tag{A-3}
$$

or simply,

$$m = \mathbf{P}M$$

where $\mathbf{P}$ is a 3×4 non-singular matrix called the *camera projection matrix*. This matrix $\mathbf{P}$ can be decomposed as shown in Eq. A-4, where $\mathbf{K}$ is called the *camera intrinsics matrix*, while $\mathbf{R}$ and $\mathbf{t}$ together represents the *camera extrinsics* i.e. the relative pose of the camera in the 3D world.

$$\mathbf{P} = \mathbf{K}[\mathbf{R} \mid \mathbf{t}] \quad \text{where} \quad \mathbf{K} = \begin{bmatrix} f_x & s & p_x \\ 0 & f_y & p_y \\ 0 & 0 & 1 \end{bmatrix} \tag{A-4}$$

The intrinsics $\mathbf{K}$ is often parameterized by f_x, f_y, s, p_x and p_y (Eq. A-4), where f_x and f_y are the focal length f measured in pixels in the x and y directions respectively, s is the skew and (p_x, p_y) is the principal point in the image.

Thus, in the general case, an Euclidean perspective camera can modeled in matrix form with five intrinsic and six extrinsic parameters (three for the rotation and three for the translation) which defines the transformation from the world coordinate system, to the coordinate system in the image. Real cameras deviate from the pin-hole model due to various optical effects, amongst which, the most pronounced is the effect of radial distortion. Radial distortion is often corrected by warping the image with a non-linear transformation. Thus, the undistorted image coordinates $(\tilde{x}, \tilde{y})$ can be obtained from the distorted image coordinatese (x, y) as follows:

$$\tilde{x} = x_c + (x - x_c)\mathrm{L}(r) \tag{A-5}$$

$$\tilde{y} = y_c + (y - y_c)\mathrm{L}(r) \tag{A-6}$$

$$\mathrm{L}(r) = (1 + \kappa_1 r + \kappa_2 r^2 + \ldots) \tag{A-7}$$

Here κ_1, κ_2 etc. are the coefficients of radial distortion, (x_c, y_c) is the center of radial distortion

in the image and r is the radial distance from the center of distortion.

A-1.2   Epipolar Geometry
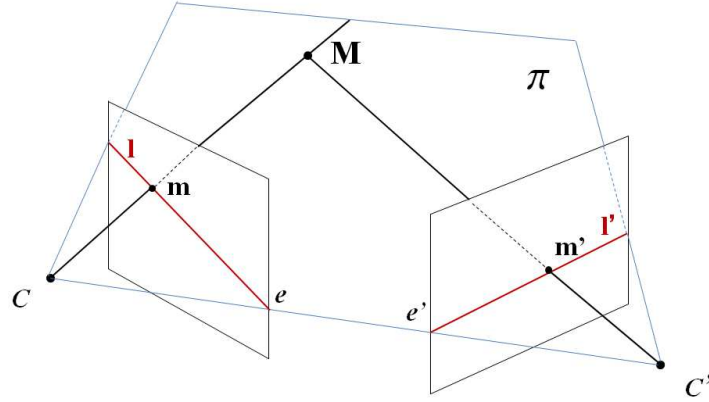
Figure 8.2: For every pixel $\mathbf{m}$, the corresponding pixel in the second image $\mathbf{m}'$ must lie somewhere along a line $\mathbf{l}'$. This property is referred to as the *epipolar constraint*. See text for details.

The epipolar geometry captures the geometric relation between two images of the same scene. When a 3D point $\mathbf{M}$ projects to pixels $\mathbf{m}$ and $\mathbf{m}'$ in the two images, $\mathbf{m}$ and $\mathbf{m}'$ are said to be in correspondence. For every point in the first image, the corresponding point in the second image is constrained to lie along a specific line called the *epipolar line*. Every plane such as π that contains the *baseline* i.e. the line joining the two camera centers, must intersect the two image planes in corresponding epipolar lines, such as $\mathbf{l}$ and $\mathbf{l}'$, respectively. All epipolar lines within an image, intersect at a special point called the *epipole*. Algebraically,

$$\mathbf{m}'^{\mathrm{T}}\mathbf{F}\mathbf{m} = 0 \qquad\qquad \text{(A-8)}$$

where $\mathbf{F}$ is called the *fundamental matrix* and has rank two. Points $\mathbf{m}$ and $\mathbf{m}'$ can be transferred to the corresponding epipolar lines in the other image, using the following relations.

$$\mathbf{l} = \mathbf{F}^{\mathrm{T}}\mathbf{m}' \qquad \mathbf{l}' = \mathbf{F}\mathbf{m}$$

The epipoles are also the left and right null-vector of the fundamental matrix:

$$\mathbf{F}\mathbf{e} = 0 \qquad \mathbf{F}^{\mathrm{T}}\mathbf{e}' = 0 \tag{A-9}$$

Since $\mathbf{F}$ is a 3×3 matrix unique up to scale, it can be linearly computed from 8 pair of corresponding points in the two views using Equation A-8, which is often called the *epipolar constraint*. This is known as the *8-point algorithm*. However, when the rank constraint is enforced, $\mathbf{F}$ can be computed from 7 pairs of correspondences using the non-linear *7-point algorithm*. Refer to [55] for the details.

Any pair of cameras denoted by camera matrices $\mathbf{P}$ and $\mathbf{P}'$, results in a unique fundamental matrix. Given a fundamental matrix $\mathbf{F}$, the camera pairs are determined up to a projective ambiguity (a projective transformationf of 3 space). Thus, given $\mathbf{F}$, there exists a four parameter family of canonical camera pairs corresponding to $\mathbf{F}$. These are given by:

$$\mathbf{P} = [\mathbf{I} \mid \mathbf{0}] \qquad \mathbf{P}' = [[\mathbf{e}']_\times \mathbf{F} + \mathbf{e}'\mathbf{v}^{\mathrm{T}} \mid \lambda \mathbf{e}'] \tag{A-10}$$

Epipolar Line Homography
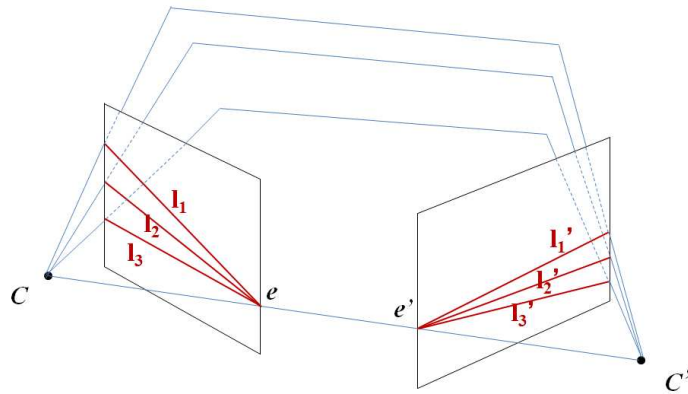
Figure 8.3: The pencil of epipolar lines forms a 1D projective space.

There exists a 1D homography that relates the pencil of epipolar lines in one view (a 1D

projective space) with the pencil of epipolar lines in the other view (see Figure 8.3. This homography has three degrees of freedom. Two degrees of freedom for each of the epipoles account for the total seven degrees of freedom of the fundamental matrix. The epipolar line homography can be used to transfer epipolar lines as follows:

$$l' = \mathbf{F}[e]_\times l \qquad l = \mathbf{F}^T[e']_\times l'$$

A-1.3 Projective Reconstruction and Self-Calibration

Without special knowledge about the contents of the scene, it is impossible to recover the position, orientation, and scale of a 3D scene reconstructed from images. When the camera intrinsics are unknown, there is a higher degree of ambiguity in the reconstruction – it can only be determined up to a projective transformation of 3 space. By recovering the whole projective structure starting from only point correspondences in multiple views, one is able to compute a *projective reconstruction* of the cameras and the scene. Note that this can be done without any knowledge of the camera intrinsics, and makes it possible to reconstruct 3D scenes from uncalibrated sequences.

   This projective cameras and scene differs from the actual cameras and scene (often referred to as a Euclidean or *metric reconstruction*) by a projective transformation – a 4×4 homography. There exists classical techniques to transform a projective reconstruction to a metric one by computing this unknown homography – this is called *auto-calibration* or *self-calibration*. Please refer to [55, 106] for more details.

A-2   Similarity Measures

The *correspondence problem* in computer vision involves finding, for every pixel in one image, the corresponding pixel in the other image. As individual pixel values are not distinctive
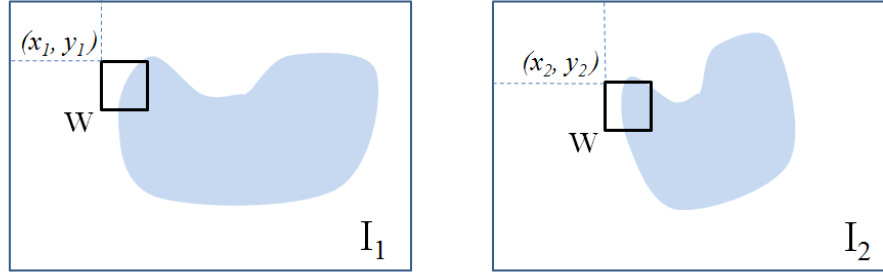
Figure 8.4: Many computer vision algorithms require a similarity metric between image patches that evaluate the similarity of their appearance. W denotes an image patch ($k \times k$ pixels) in the two images at locations (x_1, y_1) and (x_2, y_2) respectively.

enough, similarity is often computed using a patch (typically a $k \times k$ square window) around a pixel. Different patch-based similarity functions based on difference and correlation measures are used for this task. Depending on the degree of invariance required, level of image noise and computational requirements, one of the following similarity measures are typically used.

The Sum of Absolute Differences (SAD) is given by the expression:

$$\sum_{(u,v)\in W} |I_1(x_1 + u, y_1 + v) - I_2(x_2 + u, y_2 + v)| \tag{A-11}$$

The Sum of Squared Differences (SSD) is given by the expression:

$$\sum_{(u,v)\in W} (I_1(x_1 + u, y_1 + v) - I_2(x_2 + u, y_2 + v))^2 \tag{A-12}$$

The Normalized Cross Correlation (NCC) is given by the expression:

$$\frac{\sum_{(u,v)\in W} I_1(x_1 + u, y_1 + v).I_2(x_2 + u, y_2 + v)}{\sqrt{\sum_{(u,v)\in W} I_1^2(x_1 + u, y_1 + v). \sum_{(u,v)\in W} I_2^2(x_2 + u, y_2 + v)}} \tag{A-13}$$

The Zero-mean Normalized Cross Correlation (ZNCC) is given by the expression:

$$\frac{\sum_{(u,v)\in W} (I_1(x_1 + u, y_1 + v) - \overline{I_1}).(I_2(x_2 + u, y_2 + v) - \overline{I_2})}{\sqrt{\sum_{(u,v)\in W} (I_1(x_1 + u, y_1 + v) - \overline{I_1})^2. \sum_{(u,v)\in W} (I_2(x_2 + u, y_2 + v) - \overline{I_2})^2}} \tag{A-14}$$

SAD and SSD produce a value of zero for identical patches, but are not normalized as the scores depend on the patch size and the appearance in the images. NCC and ZNCC produces values in the range [-1,+1] with +1 for identical patches. Contrary to SAD, SSD and NCC, ZNCC is invariant to intensity offsets and should be used when brightness change is expected in the images, although it is the most expensive to compute. Note that in the presence of noise, similar *textureless* patches will have high similarity under SAD and SSD, while ZNCC will in general produce a low similarity score as it factors out the average and tries to correlate two random signals. For more details, and other non-parametric similarity measures discussed in the context of stereo matching, please refer to [61].

Appendix B: Miscellaneous Topics

This appendix provides a brief introduction to multiple unrelated topics that are relevant to this thesis. First, the problem of silhouette extraction in images and video is covered. Next, a brief description of *sparse bundle adjustment* is included. Finally, network flow algorithms are introduced, as they form the basis of the graph cut optimization technique used throughout this thesis.

B-1 Silhouette Extraction

The methods developed in this thesis assume that silhouettes of objects can be automatically extracted from images and video. This is called the *foreground* or *background segmentation* problem. It is quite a challenging problem in the general setting and continues to be an area of ongoing research [30, 56, 65, 112, 132]. First, we briefly describe our method for automatically extracting silhouettes of dynamic foreground objects from video. These silhouettes are used for both camera calibration and modeling the dynamic scene. Next, we briefly cover methods for recovering high quality silhouettes of static objects in multiple calibrated images. Such silhouettes are then used for enforcing silhouette constraints for reconstructing more accurate 3D models.

B-1.1 Silhouette Extraction in Video

The simplest and fastest *background segmentation* methods assume that a background image from the viewpoint of a static camera is available. When a background image is not explicitly available, it can be generated by computing a median image (the median from the sequence

of intensity values at each pixel is computed). When the video sequence is sufficiently long and foreground objects move constantly, the generated median image contains little trace of the foreground.

Each pixel in the target image is then classified as foreground or background based on the intensity difference with the corresponding pixel in the background image. Simple thresholding does not work well because the background image is seldom static. The intensities of background pixels vary due to image noise, presence of shadows cast by the foreground object and time varying illumination. More over, the appearance and color of the foreground may be similar to the background resulting in mis-classification. However, in controlled scenes (a *blue-screen* is sometimes used), such a simple background subtraction approaches can produce silhouettes of acceptable quality. Some post-processing is required to clean up the noisy silhouettes. This is done via local morphological operations on the image such as *dilation* and *erosion* [67], and connected component analysis of foreground blobs in the segmented image.

A more flexible approach involves using a per-pixel intensity distribution to model the appearance of the dynamic background. The per-pixel distribution is modeled as a mixture of Gaussians, whose parameters must be estimated using a prior training sequence, where only the background is observed. Statistical modeling for such *adaptive background segmentation* techniques are described in detail in [56, 132]. Methods such as [65, 151] explicitly model intensity variation in the background due to cast shadows and can thereby compute more accurate silhouettes.

Per-pixel classification methods are fast but fail to guarantee spatially coherent foreground segments. Image morphology only addresses this partially – the silhouette boundaries can get considerably eroded by successive *erosion* and *dilation* operations. A more powerful technique for solving the background segmentation problem uses a global approach based on energy minimization in a Markov Random Field (MRF) framework. The MRF framework was introduced in Chapter 5.3, and has been widely used for low level pixel labeling problems

such as stereo, segmentation and image restoration. The framework encourages smoothness (spatial coherence) in the labeling. In our case, the binary labels represent foreground and background respectively. Such binary energies can be efficiently minimized using graph cuts. The per pixel data term in the energy is derived from its likelihood of being foreground (or background), whereas the commonly used Pott model is used for the smoothness energy.

To summarize, the silhouettes used in our experiments were obtained by a variety of techniques. For recovering silhouettes from video in controlled scenes, we used simple background subtraction with morphological operations. For some data sets, graph-cut based background segmentation was used. Note that the data sets used in our experiments were captured by other researchers in fairly controlled interior scenes with relatively simple backgrounds. The simple background segmentation approaches described here were sufficient for our purpose. As more powerful automatic segmentation techniques are developed [36, 74, 112], it will become possible to use the techniques proposed in this thesis on more complex outdoor scenes with dynamic backgrounds.

B-1.2 Silhouette Extraction in Images

The multi-view stereo method proposed in this thesis assumes that calibrated images of an object captured from multiple viewpoints were available along with their silhouettes. The data sets used in our experiments were acquired in a special manner. The object was placed on a turn-table and a static camera was used to capture images as the turntable was rotated. In some cases, the silhouettes were extracted manually [45]. While this is a tedious step, many semi-interactive techniques for silhouette extraction of objects exist in the literature. Many of the earlier techniques, based on snakes [71], required good initialization and were susceptible to the problem of local minima. Recently, some practical, interactive image segmentation methods [30, 112] have been developed. These techniques are in general easier to use and provides more control to the user, as compared to snake-based methods.

In the system developed by [57], a simple background (screen) of relatively constant color was used during the turntable acquisition process. The silhouettes were then automatically extracted using a variety of segmentation techniques – a) a color histogram approach which performs a binary classification of pixels based on appearance (similar to the techniques described in the previous section), b) a level set algorithm using the Mumford Shah model [33, 145] and c) the JSEG algorithm [36]. The details of each of these techniques are described in [57]. An interesting approach that combines silhouette extraction with volumetric 3d reconstruction was proposed by [21].

B-2 Sparse Bundle Adjustment

An indispensable part of any structure from motion pipeline is *bundle adjustment* – a method to refine the complete scene structure (3D point set) and camera parameters simultaneously, by minimizing the reprojection error of the structure in all the images, given a set of correspondences in multiple views. It involves solving a large global optimization problem, where all camera parameters and structure parameters are refined simultaneously. This is typically the final step of a 3D reconstruction pipeline and involves nonlinear minimization of an objective function of the following form:

$$\sum_{i,j} d(\mathbf{P}^i \mathbf{X}_j, \mathbf{x}_j^i)^2$$

Here $d(\mathbf{m}, \mathbf{m}')$ is the reprojection error between the 2D homogeneous points $\mathbf{m}$ and $\mathbf{m}'$ in the image, $\mathbf{P}^i$ denotes the ith projection matrix in the sequence, $\mathbf{X}_j$ denotes the jth 3D homogeneous point that is observed in the ith image at pixel $\mathbf{x}_j^i$. Although thousands of parameters need to be simultaneously optimized in a typical problem instance, the Jacobian has a sparse structure, since the parameters of different cameras do not interact with each other. A similar form of *bundle adjustment* is used to refine the Euclidean (metric) structure and calibration and this is sometimes referred to as *Euclidean bundle adjustment*.

In practice, *bundle adjustment* requires good initialization and assumes that the measurements are free of outliers. The minimization of the nonlinear objective function is done using the iterative Levenberg-Marquardt method [90]. In fact, a sparse version of the Levenberg Marquardt method is used in bundle adjustment. It exploits the sparse structure of the Jacobian and every iteration of the algorithm involves solving a sparse linear system. A detailed description of the algorithm is provided in [55], and forms the basis of the implementation used in this thesis.

B-3   The Min-cut/Max-flow problem

A *flow network* $G(V, E)$, is defined as a fully connected directed graph where each edge (u, v) ϵ E has a positive capacity $c(u, v) \geq 0$. Two special vertices in a flow network are designated the *source* s, and the *sink* t, respectively. A *flow* in G is a real-valued function $f : V \times V \rightarrow R$ that satisfies the following properties:

- Capacity Constraint: $\forall$ u,v ϵ V, $f(u, v) \leq c(u, v)$.

- Skew Symmetry: $\forall$ u,v ϵ V, $f(u, v) = -f(v, u)$.

- Flow Conservation: $\forall$ u ϵ $(V\text{-}\{s, t\})$, $\sum_{v \epsilon V} f(u, v) = 0$.

The value of a flow is defined as $|f| = \sum_{v \epsilon V} f(s, v)$ ie. the total flow out of the source in G. The max-flow problem is to find the flow of maximum value on G.

A *s-t cut* or simply *cut* of G, is a partition of V into S and $T = V - S$, such that s ϵ S and t ϵ T. For a given flow f, the *net flow* across the cut (S, T) is defined as $f(S, T) = \sum_{x \epsilon S} \sum_{y \epsilon T} f(x, y)$. Using a similar notation, the capacity of a cut (S, T) is defined as $c(S, T) = \sum_{x \epsilon S} \sum_{y \epsilon T} c(x, y)$. A *minimum cut* of a flow network is a cut whose capacity is the least over all the s-t cuts of the network. An example of a flow network is shown in Figure 8.5.
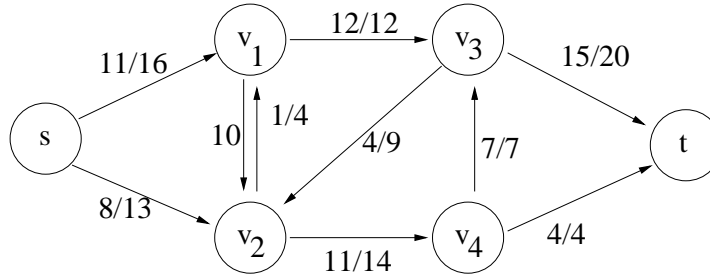
Figure 8.5: (a) The figure (taken from Cormen et. al. [28] shows a flow network $G(V, E)$ with a valid flow f. The values on the edges are $f(u, v)/c(u, v)$. The current flow has value 19. Note that this is not a maximum flow.

Theorem 1. *The max-flow min-cut theorem : If f is a flow in a flow network $G = (V, E)$ with source s and sink t then the value of the maximum flow is equal to the capacity of a minimum cut. Refer to Cormen et. al. [28] for the proof.*

The intuition behind the proof is as follows. The maximum flow must saturate edges in the flow network such that no further flow can be pushed. These saturated edges must lie on one of the min-cuts. This result allows one to compute the minimum cut of a flow network by first solving for the max-flow, for which polynomial time algorithms exist.

The single-source single-sink max-flow problem described above is a specific case of the more general multiway cut problem, where there are k terminals and a multiway cut is a minimum set of edges which separates each terminal from all the others. It has been shown that if $k \geq 3$, the problem is NP-Hard. However, in this thesis, we will only be concerned with the case $(k = 2)$, which can be exactly solved in polynomial time. A brief description of these algorithms is now presented.

B-3.1 Algorithms for Computing Max-flow

The polynomial algorithms for the single-source single-sink max-flow problem can be divided into two classes, algorithms based on the Ford Fulkerson method [39] and those based on the *push-relabel* method [52]. The two contrasting approaches are described below.

The intuitive idea behind the Ford-Fulkerson method is that starting with zero flow ie.

$f(u, v) = 0$ for all $u, v \, \epsilon \, V$, the flow can be gradually increased by finding a path from s to t, along which more flow can be sent. Such a path is called an *augmenting path*, and once it has been found, the flow can be augmented along this path. The process if repeated, must end after a finite number of iterations, after which no *augmenting paths* between s and t exist anymore. A typical algorithm of this type maintains for a given flow f, the *residual graph* of G, called G_f whose topology is identical to G but whose edge capacities stores the residual capacity of all the edges, given that there is already some flow in them. The search for an *augmenting path* at the ith iteration is done on the current *residual graph* G_{f_i}. Once an *augmenting path* is found, the maximum amount of flow that can be sent down it, f_{incr} must saturate at least one of the edges of this path. The new flow at the end of the iteration will be $f_i + f_{incr}$.

The running time complexity of different algorithms will in general vary depending on how the *augmenting path* is chosen. Dinic algorithm [37] that uses breadth-first search to find the shortest paths from s to t on the *residual graph*, has an overall worst case running time of $O(n^2m)$, n being the number of nodes and m being the number of edges.

In contrast to the Ford-Fulkerson method where augmenting the flow operates on the complete *residual graph*, the *push-relabel* algorithms operate locally on a vertex at a time, inspecting only its neighbours. Unlike the Ford-Fulkerson method, the flow conservation property is not satisfied during the algorithm's execution. The intuitive idea here is to associate a notion of height along with all the nodes in the network. The height of the source and sink are fixed at $|V|$ and 0 respectively, and at the beginning, all other vertices are at height 0. The algorithm starts by sending flow down from the source and the amount of flow sent, saturates all the outgoing edges. All intermediate nodes have a buffer or a reservoir that can store excess flow. Nodes with positive excess flow are said to be overflowing nodes. Overflowing nodes try to push the excess flow downhill. However, when an overflowing node finds the edges to its neighbours at the same height as itself saturated, it increments its own height, a process which is called "relabeling". This allows it to get rid of the excess flow. The algorithm terminates

when none of the nodes in V are overflowing. Often, excess flow accumulated in the interior nodes is sent back to the source by relabeling these nodes with height beyond $|V|$.

The generic *push-relabel algorithms* thus have two basic operations – *push* flow and *re-label* an overflowing node, and Cormen et. al. [28] proves that a generic *push-relabel* style algorithm has a $O(n^2m)$ worst case running time, and there are certain $O(n^3)$ algorithms in this class. Refer to [39, 52] for the details of the algorithms, relevant data structures and practical trade-offs in implementations.

The max-flow implementation used in various parts of this thesis uses a variant of the *augmenting path* based method that was experimentally shown to be efficient for grid graphs that are common in computer vision [14]. Grid graphs are sparse, have uniform connectivity at all vertices and a large number of connections to the source and sink nodes. The main difference in [14], lies in the method for computing the *augmenting paths*. Generally, these paths are recomputed on the *residual graph* from scratch, but this is a costly operation on large grid graphs as the breadth first search visits all vertices. The main improvement proposed by [14], was to reuse search trees in a way, such that subsequent *augmenting paths* could be computed efficiently. This algorithm has a worst case time complexity of $O(n^2m|C|)$ where $|C|$ is the maximum capacity, but has been shown to be faster than *push-relabel* based implementations, on a variety of problem instances.

Appendix C: Publications

The work presented in this dissertation was also published in the following papers:

- S.N. Sinha, P. Mordohai and M. Pollefeys, *Multi-View Stereo via Graph Cuts on the Dual of an Adaptive Tetrahedral Mesh*, Intl. Conf. on Computer Vision, (ICCV) Rio de Janiero, Oct 2007.

- S.N. Sinha and M. Pollefeys, *Multi-view Reconstruction using Photo-consistency and Exact Silhouette Constraints: A Maximum-Flow Formulation*, Intl. Conf. on Computer Vision, (ICCV) Beijing, China, October 2005.

- S.N. Sinha and M. Pollefeys, *Visual-Hull Reconstruction from Uncalibrated and Unsynchronized Video Streams*, 2nd Intl. Symposium on 3D Data Processing, Visualization and Transmission (3DPVT), Greece, September, 2004.

- S.N. Sinha and M. Pollefeys, *Calibrating a network of cameras from live or archived video*, In Proc. of Advanced Concepts for Intelligent Systems, 2004.

- S.N. Sinha and M. Pollefeys, *Synchronization and Calibration of Camera Networks from Silhouettes*, In Proc. of Intl. Conf. on Pattern Recognition, (ICPR) Cambridge, UK, Aug 2004.

- S.N. Sinha, M. Pollefeys and L. McMillan, *Camera Network Calibration from Dynamic Silhouettes*, In Proc. of IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), Washington D.C., June 2004.

BIBLIOGRAPHY

[1] Marc Alexa, Daniel Cohen-Or, and David Levin. As-rigid-as-possible shape interpolation. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 157–164, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[2] J. Allard, E. Boyer, J-S Franco, C. Mnier, and B. Raffin. Marker-less real time 3d modeling for virtual reality. In *Proceedings of IPT'04 Symposium*, Ames, USA, May 2004.

[3] Geoff Cross Andrew W. Fitzgibbon and Andrew Zisserman. Automatic 3d model construction for turn-table sequences. In R. Koch and L. VanGool, editors, *Proceedings of SMILE Workshop on Structure from Multiple Images in Large Scale Environments*, volume 1506 of *Lecture Notes in Computer Science*, pages 154–170. Springer Verlag, June 1998.

[4] Ben Appleton and Hugues Talbot. Globally minimal surfaces by continuous maximal flows. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(1):106–118, 2006.

[5] Kalle Astrom, Roberto Cipolla, and Peter Giblin. Generalised epipolar constraints. *Int. J. Comput. Vision*, 33(1):51–72, 1999.

[6] Patrick T. Baker and Yiannis Aloimonos. Calibration of a multicamera network. *cvpr*, 07:72, 2003.

[7] Luca Ballan and Guido Maria Cortelazzo. Multimodal 3d shape recovery from texture, silhouette and shadow information. In *3DPVT '06: Proceedings of the Third International Symposium on 3D Data Processing, Visualization, and Transmission (3DPVT'06)*, pages 924–930, Washington, DC, USA, 2006. IEEE Computer Society.

[8] Bruce Guenther Baumgart. *Geometric modeling for computer vision*. PhD thesis, Stanford University, 1974.

[9] R. C. Bolles and M. A. Fischler. A ransac-based approach to model fitting and its application to finding cylinders in range data. In *Proc. of the 7th IJCAI*, pages 637–643, Vancouver, Canada, 1981.

[10] J. Bouguet. Matlab camera calibration toolbox, 2000.

[11] Edmond Boyer. On using silhouettes for camera calibration. In *ACCV (1)*, pages 1–10, 2006.

[12] Edmond Boyer and Marie-Odile Berger. 3d surface reconstruction using occluding contours. *Int. J. Comput. Vision*, 22(3):219–233, 1997.

[13] Yuri Boykov and Vladimir Kolmogorov. Computing geodesics and minimal surfaces via graph cuts. In *ICCV '03: Proceedings of the Ninth IEEE International Conference on Computer Vision*, page 26, Washington, DC, USA, 2003. IEEE Computer Society.

[14] Yuri Boykov and Vladimir Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(9):1124–1137, 2004.

[15] Yuri Boykov and Victor Lempitsky. From photohulls to photoflux optimization. *BMVC*, 3:1149–1158, 2006.

[16] Yuri Boykov, Olga Veksler, and Ramin Zabih. Fast approximate energy minimization via graph cuts. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23(11):1222–1239, 2001.

[17] Adrian Broadhurst, Tom Drummond, and Roberto Cipolla. A probabilistic framework for space carving. In *ICCV*, pages 388–393, 2001.

[18] Gabriel J. Brostow, Irfan Essa, Drew Steedly, and Vivek Kwatra. Novel skeletal representation for articulated creatures. In *ECCV04*, pages Vol III: 66–78, 2004.

[19] M. Brown and D. G. Lowe. Unsupervised 3d object recognition and reconstruction in unordered datasets. In *3DIM '05: Proceedings of the Fifth International Conference on 3-D Digital Imaging and Modeling*, pages 56–63, Washington, DC, USA, 2005. IEEE Computer Society.

[20] Chris Buehler, Steven J. Gortler, Michael F. Cohen, and Leonard McMillan. Minimal surfaces for stereo. In *ECCV (3)*, pages 885–899, 2002.

[21] Neill Campbell, George Vogiatzis, Carlos Hernndez, and Roberto Cipolla. Automatic 3d object segmentation in multiple views using volumetric graph-cuts. In *In British Machine Vision Conference*, 2007.

[22] Joel Carranza, Christian Theobalt, Marcus A. Magnor, and Hans-Peter Seidel. Free-viewpoint video of human actors. In *SIGGRAPH '03: ACM SIGGRAPH 2003 Papers*, pages 569–577, New York, NY, USA, 2003. ACM.

[23] Yaron Caspi, Denis Simakov, and Michal Irani. Feature-based sequence-to-sequence matching. *Int. J. Comput. Vision*, 68(1):53–64, 2006.

[24] G.K.M. Cheung, S. Baker, and T. Kanade. Shape-from-silhouette of articulated objects and its use for human body kinematics estimation and motion capture. In *CVPR*, pages I: 77–84, 2003.

[25] Kong Man Cheung, Simon Baker, and Takeo Kanade. Shape-from-silhouette across time part i: Theory and algorithms. *International Journal of Computer Vision*, 62(3):221 – 247, May 2005.

[26] R. Cipolla, K. E. Astrom, and P. J. Giblin. Motion from the frontier of curved surfaces. In *ICCV '95: Proceedings of the Fifth International Conference on Computer Vision*, page 269, Washington, DC, USA, 1995. IEEE Computer Society.

[27] Roberto Cipolla and Peter Giblin. *Visual Motion of Curves and Surfaces*. Cambridge University Press, 2000.

[28] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Second Edition*. The MIT Press, September 2001.

[29] Antonio Criminisi, Ian D. Reid, and Andrew Zisserman. Single view metrology. *International Journal of Computer Vision*, 40(2):123–148, 2000.

[30] Antonio Criminisi, Toby Sharp, and Andrew Blake. Geos: Geodesic image segmentation. In *ECCV (1)*, pages 99–112, 2008.

[31] Geoffrey Cross and Andrew Zisserman. Surface reconstruction from multiple views using apparent contours and surface texture. In *Confluence of computer vision and computer graphics*, pages 25–47, Norwell, MA, USA, 2000. Kluwer Academic Publishers.

[32] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 303–312, New York, NY, USA, 1996. ACM.

[33] Mumford D. and Shah J. Optimal approximations by piecewise smooth functions and associated variational problems. *Communications on Pure and Applied Mathematics*, 42(5):577–685, 1989.

[34] R.J. Daverman and R.B. Sher. *Handbook of Geometric Topology*. North-Holland, December 2001.

[35] Andrew Delong and Yuri Boykov. A scalable graph-cut algorithm for n-d grids. In *CVPR*, 2008.

[36] Yining Deng and B.S. Manjunath. Unsupervised segmentation of color-texture regions in images and video. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(8):800–810, 2001.

[37] E.A. Dinic. Algorithm for solution of a problem of maximum flow in networks with power estimation. *Soviet Math. Dokl.*, 11:1277–1280, 1970.

[38] Olivier D. Faugeras and Renaud Keriven. Complete dense stereovision using level set methods. In *ECCV '98: Proceedings of the 5th European Conference on Computer Vision-Volume I*, pages 379–393, London, UK, 1998. Springer-Verlag.

[39] L. Ford and D. Fulkerson. Flows in networks. *Princeton Univ. Press*, 1962.

[40] Jan-Michael Frahm and Marc Pollefeys. Ransac for (quasi-)degenerate data (qdegsac). In *CVPR '06: Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 453–460, Washington, DC, USA, 2006. IEEE Computer Society.

[41] Jean-Sébastien Franco and Edmond Boyer. Exact polyhedral visual hulls. In *Proceedings of the Fourteenth British Machine Vision Conference*, pages 329–338, September 2003. Norwich, UK.

[42] Jean-Sébastien Franco, Marc Lapierre, and Edmond Boyer. Visual shapes of silhouette sets. In *Proceedings of the 3rd International Symposium on 3D Data Processing, Visualization and Transmission, Chapel Hill (USA)*, 2006.

[43] Henry Fuchs, Gary Bishop, Ruzena Bajcsy, Sang Wook Lee, Hany Farid, and Takeo Kanade. Virtual space teleconferencing using a sea of cameras. In *Proc. First International Conference on Medical Robotics and Computer Assisted Surgery*, pages 161–167, 1994.

[44] Y. Furukawa and J. Ponce. Accurate, dense, and robust multi-view stereopsis. In *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, pages 1–8, 2007.

[45] Yasutaka Furukawa and Jean Ponce. Carved visual hulls for image-based modeling. *ECCV*, 2006.

[46] Yasutaka Furukawa and Jean Ponce. Dense 3d motion capture from synchronized video streams. In *CVPR*, 2008.

[47] Yasutaka Furukawa, Amit Sethi, Jean Ponce, and David Kriegman. Robust structure and motion from outlines of smooth curved surfaces. *PAMI*, 28(2):302–315, February 2006.

[48] P. Gargallo, E. Prados, and P.F. Sturm. Minimizing the reprojection error in surface reconstruction from images. In *ICCV*, pages 1–8, 2007.

[49] Ovidiu Ghita, Paul F. Whelan, and John Mallon. Computational approach for depth from defocus. *Journal of Electronic Imaging*, 14(2), 2005.

[50] Michael Goesele, Brian Curless, and Steven M. Seitz. Multi-view stereo revisited. In *CVPR (2)*, pages 2402–2409, 2006.

[51] Michael Goesele, Noah Snavely, Brian Curless, Hugues Hoppe, and Steven M. Seitz. Multi-view stereo for community photo collections. In *ICCV*, pages 1–8, 2007.

[52] A V Goldberg and R E Tarjan. A new approach to the maximum flow problem. In *STOC '86: Proceedings of the eighteenth annual ACM symposium on Theory of computing*, pages 136–146, New York, NY, USA, 1986. ACM.

[53] Markus Gross, Stephan Würmlin, Martin Naef, Edouard Lamboray, Christian Spagno, Andreas Kunz, Esther Koller-Meier, Tomas Svoboda, Luc Van Gool, Silke Lang, Kai Strehlke, Andrew Vande Moere, and Oliver Staadt. blue-c: a spatially immersive display and 3d video portal for telepresence. *ACM Trans. Graph.*, 22(3):819–827, 2003.

[54] M. Habbecke and L. Kobbelt. A surface-growing approach to multi-view stereo reconstruction. In *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, pages 1–8, 2007.

[55] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*, volume 23. Cambridge University Press, New York, NY, USA, 2005.

[56] Eric Hayman and Jan-Olof Eklundh. Statistical background subtraction for a mobile observer. *Computer Vision, IEEE International Conference on*, 1:67, 2003.

[57] Carlos Hernández. *Stereo and Silhouette Fusion for 3D Object Modeling from Uncalibrated Images Under Circular Motion*. PhD thesis, Ecole Nationale Supŕieure des Télécommunications, May 2004.

[58] Carlos Hernández, Francis Schmitt, and Roberto Cipolla. Silhouette coherence for camera calibration under circular motion. *PAMI*, 29(2):343–349, February 2007.

[59] Carlos Hernández, George Vogiatzis, and Roberto Cipolla. Probabilistic visibility for multi-view stereo. In *CVPR*, pages xx–xx, 2007.

[60] Adrian Hilton and Jonathan Starck. Multiple view reconstruction of people. In *3DPVT '04: Proceedings of the 3D Data Processing, Visualization, and Transmission, 2nd International Symposium*, pages 357–364, Washington, DC, USA, 2004. IEEE Computer Society.

[61] Heiko Hirschmuller and Daniel Scharstein. Evaluation of cost functions for stereo matching. In *CVPR*. IEEE Computer Society, 2007.

[62] Derek Hoiem, Alexei A. Efros, and Martial Hebert. Automatic photo pop-up. In *ACM SIGGRAPH*, August 2005.

[63] Alexander Hornung and Leif Kobbelt. Hierarchical volumetric multi-view stereo reconstruction of manifold surfaces based on dual graph embedding. In *CVPR (1)*, pages 503–510, 2006.

[64] Alexander Hornung and Leif Kobbelt. Robust and efficient photo-consistency estimation for volumetric 3d reconstruction. In *ECCV (2)*, pages 179–190, 2006.

[65] Thanarat Horprasert, David Harwood, and Larry S. Davis. A statistical approach for real-time robust background subtraction and shadow detection. In *ICCV*, 1999.

[66] J. Isidoro and S. Sclaroff. Stochastic refinement of the visual hull to satisfy photometric and silhouette consistency constraints. In *ICCV*, pages 1335–1342, 2003.

[67] Anil K. Jain. *Fundamentals of Digital Image Processing (Prentice Hall Information and System Sciences Series).* Prentice Hall, September 1988.

[68] Tanuja Joshi, Narendra Ahuja, and Jean Ponce. Structure and motion estimation from dynamic silhouettes under perspective projection. In *ICCV*, pages 290–295, 1995.

[69] Takeo Kanade, Peter Rander, and P. J. Narayanan. Virtualized reality: Constructing virtual worlds from real scenes. *IEEE MultiMedia*, 4(1):34–47, – 1997.

[70] Takeo Kanade, Hideo Saito, and Sundar Vedula. The 3d room: Digitizing time-varying 3d events by synchronized multiple video streams. Technical Report CMU-RI-TR-98-34, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, December 1998.

[71] Michael Kass, Andrew Witkin, and Demetri Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1:321–331, 1988.

[72] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Fourth Eurographics Symposium on Geometry Processing*, pages 61–70, June 2006.

[73] D. Kirsanov and S.J. Gortler. A discrete global minimization algorithm for continuous variational problems. Technical report, Harvard University, july 2004.

[74] Pushmeet Kohli and Philip H. S. Torr. Effciently solving dynamic markov random fields using graph cuts. In *ICCV '05: Proceedings of the Tenth IEEE International Conference on Computer Vision*, pages 922–929, Washington, DC, USA, 2005. IEEE Computer Society.

[75] Pushmeet Kohli and Philip H. S. Torr. Dynamic graph cuts for efficient inference in markov random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(12):2079–2088, 2007.

[76] Kalin Kolev, Maria Klodt, Thomas Brox, Selim Esedoglu, and Daniel Cremers. Continuous global optimization in multiview 3d reconstruction. In *EMMCVPR*, pages 441–452, 2007.

[77] Vladimir Kolmogorov and Ramin Zabih. Multi-camera scene reconstruction via graph cuts. In *ECCV 2002: Proceedings of the 7th European Conference on Computer Vision-Part III*, pages 82–96, London, UK, 2002. Springer-Verlag.

[78] K. Kutulakos and S. Seitz. A theory of shape by space carving. *International Journal of Computer Vision*, 38(3):199–218, 2000.

[79] P. Labatut, J.P. Pons, and R. Keriven. Efficient multi-view reconstruction of large-scale scenes using interest points, delaunay triangulation and graph cuts. In *ICCV*, pages 1–8, 2007.

[80] Patrick Labatut, Renaud Keriven, and Jean-Philippe Pons. Fast level set multi-view stereo on graphics hardware. In *3DPVT*, pages 774–781. IEEE Computer Society, 2006.

[81] A. Laurentini. The visual hull concept for silhouette-based image understanding. *PAMI*, 16(2):150–162, February 1994.

[82] Svetlana Lazebnik, Edmund Boyer, and Jean Ponce. On computing exact visual hulls of solids bounded by smooth surfaces. In *CVPR*, pages I:156–161, 2001.

[83] Svetlana Lazebnik, Yasutaka Furukawa, and Jean Ponce. Projective visual hulls. *Int. J. Comput. Vision*, 74(2):137–165, 2007.

[84] Svetlana Lazebnik, Amit Sethi, Cordelia Schmid, David J. Kriegman, Jean Ponce, and Martial Hebert. On pencils of tangent planes and the recognition of smooth 3d shapes from silhouettes. In *ECCV (3)*, pages 651–665, 2002.

[85] Victor Lempitsky, Yuri Boykov, and Denis Ivanov. Oriented visibility for multiview reconstruction. In *ECCV*, pages 226–238, 2006.

[86] Noam Levi and Michael Werman. The viewing graph. *CVPR*, 01:518–522, 2003.

[87] Marc Levoy, Kari Pulli, Brian Curless, Szymon Rusinkiewicz, David Koller, Lucas Pereira, Matt Ginzton, Sean Anderson, James Davis, Jeremy Ginsberg, Jonathan Shade, and Duane Fulk. The digital michelangelo project: 3D scanning of large statues. In *Proceedings of ACM SIGGRAPH 2000*, pages 131–144, July 2000.

[88] M. Lhuillier and L. Quan. A quasi-dense approach to surface reconstruction from uncalibrated images. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(3):418–433, 2005.

[89] M.I.A. Lourakis and A.A. Argyros. The design and implementation of a generic sparse bundle adjustment software package based on the levenberg-marquardt algorithm. Technical Report 340, Institute of Computer Science - FORTH, Heraklion, Crete, Greece, August 2004.

[90] Donald W. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *SIAM Journal on Applied Mathematics*, 11(2):431–441, 1963.

[91] D. Martinec and T. Pajdla. Consistent multi-view reconstruction from epipolar geometries with outliers. In *SCIA*, pages 493–500, 2003.

[92] D. Martinec and T. Pajdla. 3d reconstruction by gluing pair-wise euclidean reconstructions, or 'how to achieve a good reconstruction from bad images'. In *3DPVT*, pages 25–32, 2006.

[93] Jiri Matas and Ondrej Chum. Randomized ransac with t, *d* test. *Image Vision Comput.*, 22(10):837–842, 2004.

[94] Y. Matsumoto, K. Fujimura, and T. Kitamura. Shape-from-silhouette/stereo and its application to 3-d digitizer. In *DCGI '99: Proceedings of the 8th International Conference on Discrete Geometry for Computer Imagery*, pages 177–190, London, UK, 1999. Springer-Verlag.

[95] W. Matusik, C. Buehler, and L. Mcmillan. Polyhedral visual hulls for real-time rendering. *Proceedings of Eurographics Workshop on Rendering,*, pages 115–126, 2001.

[96] Wojciech Matusik, Chris Buehler, Ramesh Raskar, Steven J. Gortler, and Leonard McMillan. Image-based visual hulls. In Kurt Akeley, editor, *Siggraph 2000, Computer Graphics Proceedings*, pages 369–374. ACM Press / ACM SIGGRAPH / Addison Wesley Longman, 2000.

[97] Sean Mauch. *Efficient Algorithms for Solving Static Hamilton-Jacobi Equations*. PhD thesis, California Institute of Technology, 2003.

[98] Sean Mauch. *Efficient Algorithms for Solving Static Hamilton-Jacobi Equations*. PhD thesis, California Institute of Technology, Pasadena, CA, 2003.

[99] A. Melkman. On-line construction of the convex hull of a simple polygon. *Information Proc. Letters*, 25(11), 1987.

[100] Paulo R. S. Mendonça, Kwan-Yee K. Wong, and Roberto Cipolla. Epipolar geometry from profiles under circular motion. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23(6):604–616, 2001.

[101] Paul Merrell, Amir Akbarzadeh, Liang Wang, Philippos Mordohai, Jan-Michael Frahm, Ruigang Yang, David Nistér, and Marc Pollefeys. Real-time visibility-based fusion of depth maps. In *ICCV*, pages 1–8, 2007.

[102] MeshLab. Mesh editing software.

[103] Neil Molino, Robert Bridson, Joseph Teran, and Ronald Fedkiw. A crystalline, red green strategy for meshing highly deformable objects with tetrahedra. In *12th International Meshing Roundtable*, pages 103–114, 2003.

[104] Jan Neumann and Yiannis Aloimonos. Spatio-temporal stereo using multi-resolution subdivision surfaces. *Int. J. Comput. Vision*, 47(1-3):181–193, 2002.

[105] Sylvain Paris, François X. Sillion, and Long Quan. A surface reconstruction method using global graph cut optimization. *Int. J. Comput. Vision*, 66(2):141–161, 2006.

[106] Marc Pollefeys, Luc Van Gool, Maarten Vergauwen, Frank Verbiest, Kurt Cornelis, Jan Tops, and Reinhard Koch. Visual modeling with a hand-held camera. *Int. J. Comput. Vision*, 59(3):207–232, 2004.

[107] Marc Pollefeys, Frank Verbiest, and Luc J. Van Gool. Surviving dominant planes in uncalibrated structure and motion recovery. In *ECCV '02: Proceedings of the 7th European Conference on Computer Vision-Part II*, pages 837–851, London, UK, 2002. Springer-Verlag.

[108] Sudipta N. Sinha:Philippos Mordohai:Marc Pollefeys. Multi-view stereo via graph cuts on the dual of an adaptive tetrahedral mesh. *ICCV 2007, IEEE 11th International Conference on Computer Vision*, pages 1–8, October 2007.

[109] Jean-Philippe Pons, Renaud Keriven, and Olivier Faugeras. Multi-view stereo reconstruction and scene flow estimation with a global image-based matching score. *Int. J. Comput. Vision*, 72(2):179–193, 2007.

[110] Jean-Philippe Pons, Renaud Keriven, and Olivier D. Faugeras. Modelling dynamic scenes by registering multi-view image sequences. In *CVPR (2)*, pages 822–827, 2005.

[111] John Porrill and Stephen Pollard. Curve matching and stereo calibration. *Image Vision Comput.*, 9(1):45–50, 1991.

[112] C. Rother, V. Kolmogorov, and A. Blake. "grabcut": interactive foreground extraction using iterated graph cuts. *ACM Trans. Graph.*, 23(3):309–314, 2004.

[113] Sébastien Roy and Ingemar J. Cox. A maximum-flow formulation of the n-camera stereo correspondence problem. In *ICCV '98: Proceedings of the Sixth International Conference on Computer Vision*, page 492, Washington, DC, USA, 1998. IEEE Computer Society.

[114] Peter Sand, Leonard McMillan, and Jovan Popović. Continuous capture of skin deformation. In *SIGGRAPH '03: ACM SIGGRAPH 2003 Papers*, pages 578–586, New York, NY, USA, 2003. ACM Press.

[115] Ashutosh Saxena, Sung H. Chung, and Andrew Y. Ng. 3-d depth reconstruction from a single still image. *International Journal Computer Vision*, 76(1):53–69, January 2008.

[116] Daniel Scharstein and Richard Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *Int. J. Comput. Vision*, 47(1-3):7–42, 2002.

[117] Steven M. Seitz, Brian Curless, James Diebel, Daniel Scharstein, and Richard Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *CVPR '06: Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 519–528, Washington, DC, USA, 2006. IEEE Computer Society.

[118] Steven M. Seitz and Charles R. Dyer. Photorealistic scene reconstruction by voxel coloring. In *CVPR '97: Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition (CVPR '97)*, page 1067, Washington, DC, USA, 1997. IEEE Computer Society.

[119] J. A. Sethian. *Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science*. Cambridge University Press, June 1999.

[120] Sudipta Sinha, Jan-Michael Frahm, Marc Pollefeys, and Yakup Genc. Feature tracking and matching in video using programmable graphics hardware. *Machine Vision and Applications*, 2006.

[121] Sudipta N. Sinha and Marc Pollefeys. Calibrating a network of cameras from live or archived video. In *In Proc. of Advanced Concepts for Intelligent Systems*, volume 0, Los Alamitos, CA, USA, 2004.

[122] Sudipta N. Sinha and Marc Pollefeys. Synchronization and calibration of camera networks from silhouettes. In *ICPR '04: Proceedings of the Pattern Recognition, 17th International Conference on (ICPR'04) Volume 1*, pages 116–119, Washington, DC, USA, 2004. IEEE Computer Society.

[123] Sudipta N. Sinha and Marc Pollefeys. Visual-hull reconstruction from uncalibrated and unsynchronized video streams. *3dpvt*, 0:349–356, 2004.

[124] Sudipta N. Sinha and Marc Pollefeys. Multi-view reconstruction using photo-consistency and exact silhouette constraints: A maximum-flow formulation. In *Proc. of ICCV*, pages 349–356, Beijing, China, October 2005.

[125] Sudipta N. Sinha, Marc Pollefeys, and Leonard McMillan. Camera network calibration from dynamic silhouettes. *cvpr*, 01:195–202, 2004.

[126] Sudipta N. Sinha, Drew Steedly, Richard Szeliski, Maneesh Agrawala, and Marc Pollefeys. Interactive 3d architectural modeling from unordered photo collections. In *SIGGRAPH Asia '08: ACM SIGGRAPH Asia 2008 papers*, pages 1–10, New York, NY, USA, 2008. ACM.

[127] Gregory G. Slabaugh, W. Bruce Culbertson, Thomas Malzbender, Mark R. Stevens, and Ronald W. Schafer. Methods for volumetric reconstruction of visual scenes. *Int. J. Comput. Vision*, 57(3):179–199, 2004.

[128] Noah Snavely, Steven M. Seitz, and Richard Szeliski. Photo tourism: Exploring photo collections in 3d. In *SIGGRAPH Conference Proceedings*, pages 835–846, New York, NY, USA, 2006. ACM Press.

[129] D. Snow, P. Viola, and R. Zabih. Exact voxel occupancy with graph cuts. In *CVPR*, pages 345–353, 2000.

[130] Mario Sormann, Christopher Zach, Joachim Bauer, Konrad F. Karner, and Horst Bischof. Watertight multi-view reconstruction based on volumetric graph-cuts. In *SCIA*, volume 4522 of *Lecture Notes in Computer Science*, pages 393–402. Springer, 2007.

[131] Jonathan Starck and Adrian Hilton. Surface capture for performance-based animation. *IEEE Computer Graphics and Applications*, 27(3):21–31, 2007.

[132] Chris Stauffer and W.E.L. Grimson. Adaptive background mixture models for real-time tracking. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 2:2246, 1999.

[133] Christoph Strecha, Rik Fransens, and Luc J. Van Gool. Combined depth and outlier estimation in multi-view stereo. In *CVPR (2)*, pages 2394–2401, 2006.

[134] P. Sturm and B. Triggs. A factorization based algorithm for multi-image projective structure and motion. In *ECCV*, pages 709–20, Cambridge, England, apr 1996. Springer-Verlag.

[135] P.F. Sturm and S.J. Maybank. On plane-based camera calibration: A general algorithm, singularities, applications. In *CVPR*, pages I: 432–437, 1999.

[136] Steve Sullivan and Jean Ponce. Automatic model construction and pose estimation from photographs using triangular splines. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20(10):1091–1097, 1998.

[137] Tomáš Svoboda, Daniel Martinec, and Tomáš Pajdla. A convenient multi-camera self-calibration for virtual environments. *PRESENCE: Teleoperators and Virtual Environments*, 14(4):407–422, August 2005.

[138] Richard Szeliski. Rapid octree construction from image sequences. *CVGIP: Image Underst.*, 58(1):23–32, 1993.

[139] Richard Szeliski, Ramin Zabih, Daniel Scharstein, Olga Veksler, Vladimir Kolmogorov, Aseem Agarwala, Marshall F. Tappen, and Carsten Rother. A comparative study of energy minimization methods for markov random fields. In *ECCV (2)*, pages 16–29, 2006.

[140] S. Tran and L. Davis. 3d surface reconstruction using graph cuts with surface constraints. In *ECCV*, pages 219–231, 2006.

[141] B. Triggs. Factorization methods for projective structure and motion. In *CVPR*, San Francisco, June 1996.

[142] Bill Triggs, Philip McLauchlan, Richard Hartley, and Andrew Fitzgibbon. Bundle adjustment – A modern synthesis. In W. Triggs, A. Zisserman, and R. Szeliski, editors, *Vision Algorithms: Theory and Practice*, LNCS, pages 298–375. Springer Verlag, 2000.

[143] Roger Y. Tsai. A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses. *Radiometry*, pages 221–244, 1992.

[144] Sundar Vedula, Simon Baker, Robert Collins, Takeo Kanade, and Peter Rander. Three-dimensional scene flow. In *ICCV '99: Proceedings of the International Conference on Computer Vision-Volume 2*, page 722, Washington, DC, USA, 1999. IEEE Computer Society.

[145] Luminita A. Vese and Tony F. Chan. A multiphase level set framework for image segmentation using the mumford and shah model. *International Journal of Computer Vision*, 50(3):271–293, 2002.

[146] VICON. Motion capture software.

[147] B. Vijayakumar, D. J. Kriegman, and J. Ponce. Structure and motion of curved 3d objects from monocular silhouettes. In *CVPR '96: Proceedings of the 1996 Conference on Computer Vision and Pattern Recognition (CVPR '96)*, page 327, Washington, DC, USA, 1996. IEEE Computer Society.

[148] G. Vogiatzis, P. H. S. Torr, and R. Cipolla. Multi-view stereo via volumetric graph-cuts. In *CVPR '05: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 2*, pages 391–398, Washington, DC, USA, 2005. IEEE Computer Society.

[149] George Vogiatzis, Carlos Hernández Esteban, Philip H. S. Torr, and Roberto Cipolla. Multiview stereo via volumetric graph-cuts and occlusion robust photo-consistency. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(12):2241–2246, 2007.

[150] J. Vollmer, R. Mencl, , and H. Môller. Improved laplacian smoothing of noisy surface meshes. In *Computer Graphics Forum*, volume 18(3), pages 131–138, 1999.

[151] Y. Wang, K.F. Loe, and J.K. Wu. A dynamic conditional random field model for foreground and shadow segmentation. *PAMI*, 28(2):279–289, February 2006.

[152] K.Y.K. Wong and R. Cipolla. Structure and motion from silhouettes. In *ICCV*, pages II: 217–222, 2001.

[153] C. Xu and J. Prince. Gradient vector flow: A new external force for snakes. In *Proceedings of Computer Vision and Pattern Recognition (CVPR '97)*, pages 66–71. IEEE, June 1997.

[154] Ruigang Yang, Marc Pollefeys, and Greg Welch. Dealing with textureless regions and specular highlights-a progressive space carving scheme using a novel photo-consistency measure. In *ICCV '03: Proceedings of the Ninth IEEE International Conference on Computer Vision*, page 576, Washington, DC, USA, 2003. IEEE Computer Society.

[155] Jonathan S. Yedidia, William T. Freeman, and Yair Weiss. Generalized belief propagation. In *NIPS*, pages 689–695, 2000.

[156] Anthony J. Yezzi and Stefano Soatto. Structure from motion for scenes without features. In *CVPR (1)*, pages 525–532, 2003.

[157] Tianli Yu, Narendra Ahuja, and Wei-Chao Chen. Sdg cut: 3d reconstruction of non-lambertian objects using graph cuts on surface distance grid. In *CVPR (2)*, pages 2269–2276, 2006.

[158] Christopher Zach, Thomas Pock, and Horst Bischof. A globally optimal algorithm for robust tv-l1 range image integration. In *ICCV*, pages 1–8, 2007.

[159] Gang Zeng, Sylvain Paris, Long Quan, and Franois Sillion. Accurate and scalable surface representation and reconstruction from images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(1):141–158, 2007.

[160] Li Zhang, Brian Curless, and Steven M. Seitz. Spacetime stereo: Shape recovery for dynamic scenes. *cvpr*, 02:367, 2003.

[161] Zhengyou Zhang. Determining the epipolar geometry and its uncertainty: A review. *International Journal of Computer Vision*, 27(2):161–195, March 1998.

[162] Zhengyou Zhang. A flexible new technique for camera calibration. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(11):1330–1334, 2000.