Visual Hull Construction, Alignment and Refinement for Human Kinematic Modeling, Motion Tracking and Rendering

Kong Man (German) Cheung

CMU-RI-TR-03-44

A Dissertation submitted in partial fulfillment of the

requirements for the degree of

Doctor of Philosophy in ROBOTICS

at the

Robotics Institute

Carnegie Mellon University

Pittsburgh, Pennsylvania 15213

October 2003

Abstract

The abilities to build precise human kinematic models and to perform accurate human motion tracking are essential in a wide variety of applications such as ergonomic design, biometrics, anthropological studies, entertainment, human computer interfaces for intelligent environments, and surveillance. Due to the complexity of the human bodies and the problem of self-occlusion, modeling and tracking humans using cameras are challenging tasks. In this thesis, we develop algorithms to perform these two tasks based on the shape estimation method Shape-From-Silhouette (SFS) which constructs a shape estimate (known as Visual Hull) of an object using its silhouettes images.

In the first half of this thesis we extend the traditional SFS algorithm so that it can be used effectively for human kinematic modeling and motion tracking. Though popular and easy to implement, traditional SFS has two serious disadvantages which greatly limit its use in human related applications. First of all, SFS involves time-consuming testing steps which make it inefficient in real-time applications. Moreover, building detailed human body models using SFS is difficult unless we use a large number of cameras because Visual Hull built from small number of silhouette images is coarse. We address the first problem by proposing a fast testing/projection algorithm for voxel-based SFS algorithms. To deal with the second problem, we combine silhouette information over time to effectively increase the number of cameras without physically adding new cameras. We first propose a new Visual Hull representation called Bounding Edges. We then analyze the ambiguity problem of aligning two Visual Hulls. Based on the analysis, we develop an algorithm to

align Visual Hulls over time using stereo and an important property of the Shape-From-Silhouette principle. This temporal SFS algorithm combines both geometric constraints and photometric consistency to align Colored Surface Points of the object extracted from the silhouette and color images. Once the Visual Hulls are aligned, they are refined by compensating for the motion of the object. The algorithm is developed for both rigid and articulated objects.

In the second half of this thesis we show how the improved SFS algorithms are used to perform the tasks of human modeling and motion tracking. First we build a system to acquire human kinematic models consisting of precise shape (constructed using the rigid object temporal SFS algorithm) and joint locations (estimated using the SFS algorithm for articulated objects). Once the kinematic models are built, they are used to track the motion of the person in new video sequences. The tracking algorithm is based on the Visual Hull alignment idea used in the temporal SFS algorithms. Finally we demonstrate how the kinematic model and the tracked motion data can be used for image-based rendering and motion transfer between two people.

Acknowledgments

I wish to sincerely thank my advisors Takeo Kanade and Simon Baker for their guidance, encouragement, support, inspiration, patience, persistence and enthusiasm during my time at CMU. Their advice, idea and suggestions on my research and thesis writing is invaluable. I am especially grateful to have the opportunity to work with both of them. I would also like to thank the other members of my thesis committee: Gray Bradski, Bob Collins and Steve Seitz for their comments, suggestions on this thesis.

I would like to dedicate this thesis to my family: my mom So Lan LAM for her love, my sisters and brothers: Lai Kuen, Lai Kwan, Wai Kwan, Wai Ngor, Kwong Sang and Kong Hou for their support, encouragement and more importantly their sacrifices which made it possible for me to pursue my dreams.

I would like to express my gratitude to Shigeyuki Baba, Ralph Gross, Iain Matthews, Hideo Saito and Sundar Vedula for their help with the Virtualized Reality Laboratory (3D Room), to Louise Ditmore and Suzanne Lyons Muth for their help on the administrative issues. Special thanks should be given to Simon Baker, Erin Bridges, Takeo Kanade and Yoky Matsuoka for kindly agreed to be the models for my experiments.

Finally I would like to thank my friends from King's College, HKU and HKUST back in Hong Kong, my friends Frank A., Nima H., John L., Jeremy L., Tim R., Jim W. here in Pittsburgh for keeping me entertained and sane throughout these years and Tasty Restaurant for their superb authentic Chinese food.

Table of Contents

List of Figures

List of Tables

List of Video Clips

Clips from Chapter 3 (http://www.cs.cmu.edu/~german/research/Thesis/Video/Chapter3)

- **Realtime-SFS-reconstruction-fitting.mpg**: reconstruction/fitting results of the real-time 3D human voxel reconstruction and fitting system.
- **Realtime-SFS-system.mpg**: footage of the real-time 3D human voxel reconstruction and fitting system, the user-interface and the results captured during an experiment.

Clips from Chapter 5 (http://www.cs.cmu.edu/~german/research/Thesis/Video/Chapter5)

- **Torso.mpg**: input images from one of the cameras and alignment/refinement results of the Torso sequence.
- **Pooh.mpg**: input images from one of the cameras and alignment/refinement results of the Pooh sequence.
- **Dinosaur-Banana.mpg**: input images from one of the cameras and alignment/refinement results of the Dinosaur-Banana sequence.

Clips from Chapter 6 (http://www.cs.cmu.edu/~german/research/Thesis/Video/Chapter6)

- **Synthetic-joints-leftleg.mpg**: input images from one of the cameras and segmentation/alignment/joint estimation results for the left hip and left knee joints of the synthetic data set.
- **Pooh-Dinosaur.mpg**: input images from one of the cameras, alignment/segmentation/temporal refinement results of the Pooh-Dinosaur sequence (two separate and independently moving rigid objects).
- **SubjectE-joints-rightarm.mpg**: input images from one of the cameras and segmentation/alignment/joint estimation results for the right shoulder and right elbow joints of SubjectE.

- **SubjectG-joints-rightleg.mpg**: input images from one of the cameras and segmentation/alignment/joint estimation results for the right hip and right knee joints of SubjectG.

Clips from Chapter 7 (http://www.cs.cmu.edu/~german/research/Thesis/Video/Chapter7)

- **SubjectS-joints-leftarm.mpg**: input images from one of the cameras and segmentation/alignment/joint estimation results for the left shoulder and left elbow joints of SubjectS.
- **Subject-EGS-kinematicmodels.mpg**: 3D fly-around views of the kinematic models of SubjectE, SubjectG and SubjectS.

Clips from Chapter 8 (http://www.cs.cmu.edu/~german/research/Thesis/Video/Chapter8)

- **Synthetic-track.mpg**: motion tracking results of the PUNCH and KICK sequences from the synthetic data set.
- **SubjectG-track.mpg**: motion tracking results of the STILLMARCH, AEROBICS and KUNGFU sequences of SubjectG.
- **SubjectS-track.mpg**: motion tracking results of the THROW sequence of SubjectS.
- **SubjectE-track.mpg**: motion tracking results of the SLOWDANCE and STEP-FLEX sequences of SubjectE.

Clips from Chapter 9 (http://www.cs.cmu.edu/~german/research/Thesis/Video/Chapter9)

- **SubjectE-rendered-PUNCH.mpg**: results of using IBAMRA to render SubjectE performing the PUNCH motion.
- **SubjectS-rendered-THROW.mpg**: comparing images rendered using IBAMRA with real images of SubjectS performing the THROW motion.
- **SubjectE-transfer-THROW.mpg**: transferring the THROW motion (originally performed by SubjectS) to SubjectE.
- **SubjectS-transfer-KUNGFU.mpg**: transferring the KUNGFU motion (originally performed by SubjectG) to SubjectS.
- **SubjectG-transfer-STEP-FLEX.mpg**: transferring the STEP-FLEX motion (originally performed by SubjectE) to SubjectG.

Chapter 1

Introduction

Human kinematic modeling, motion tracking and rendering are difficult problems because of the complexity of the human body. Despite the difficulties, these problems have received a great deal of attention recently due to the large number of applications. Having a precise 3D kinematic (shape and joint) model of an individual human is very useful in a variety of different situations. For example, they can be used in the garment/furniture manufacturing industry to make clothes/furniture that are tailored to body shape and motion range of the individual. A collection of such models can be used to generate valuable statistics of kinematic information (such as arm length, shape, etc.) of people from different races for anthropological studies. Likewise, accurate human motion tracking is essential in a wide variety of applications. For example, in intelligent environments such as smart offices or households [SKB+98, Coe98, LZG98], tracking motion and recognizing gestures is a natural way for the computer to understand the action and intention of humans. In the field of machine surveillance and security, it is important for computers to be able to observe suspicious people and track their actions over time. For sports science and medicine, the ability to track the body parts of athletes is critical for improving their performance during competition or for injury rehabilitation. Last but not the least, the entertainment industry is another area where there is an increasing need for better human modeling, motion track-

ing and rendering algorithms. Accurate human kinematic models, precise motion capture data and photo-realistic motion rendering are all essential components for making animated virtual characters more human-like in both games development and motion picture production.

Although there are laser-scanning systems for high precision human body shape acquisition, most of these systems are expensive and they do not estimate the important joint information of the body. Similarly, commercial marker-based motion capture systems are invasive and difficult to use. In applications such as security/surveillance and human-computer interaction, these systems are not applicable because placing markers on the person is either impossible or undesirable. In view of this, the study of non-invasive, vision-based human modeling/tracking systems is vital. There are many advantages of using a vision-based approach. For example, cameras are low-cost, easily reconfigurable and non-invasive. Moreover, camera images contain both shape and color (texture) information of the person. Also instead of using two separate systems for human modeling and motion tracking, one multi-cameras system can be used for both tasks.

Over the past few years researchers have proposed a variety of vision-based systems to capture the 2D and 3D shapes of human body parts [FHPB00, FGDP02] or track simple human motion (such as walking, running or simple arm motions) using single or multiple camera systems [MG01]. While some of these systems work well, there is much room for improvement, especially in the areas of automatic human joint information acquisition and precise tracking of complex human motion such as dancing, fighting, or moves made by athletes.

Among existing systems, silhouette information has been used extensively together with other cues such as edges, feature points and color textures to locate and track humans [CA96, WADP97, CA98, BK99]. Silhouette images are used because they are easily obtainable in most situations and contain valuable body shape information. In particular, many human shape modeling/motion tracking systems (such as [MTG97, KM98] and more

recently [CKBH00, Mat01, MHTC01]) use the silhouette-based shape estimation method called Shape-From-Silhouette (SFS) to construct 3D estimates of body shape. Shape-From-Silhouette is also known as as Visual Hull (VH) construction [Lau91, Lau94].

Shape-From-Silhouette has been a popular shape estimation algorithm for years. Though easy to implement, SFS has its own limitations. Existing SFS methods involve time-consuming testing steps which hinder their use in real-time applications. Moreover, shape estimation by SFS is coarse if there are only a few silhouette images. This means that it is difficult to obtain very detailed human body shapes using traditional SFS unless we use a large number of cameras or we can combine silhouettes captured across time. The traditional SFS formulation assumes that all of the silhouette images are captured either at the same time or while the object is static. This assumption is violated when the object moves or changes shape. Hence in previous SFS-based human modeling/tracking systems, SFS has been applied to each time instant sequentially and independently and little work has been done in extending SFS across time. In order to use SFS more effectively for human applications, the traditional SFS algorithms have to be improved to overcome these limitations.

With the above motivation, the goal of this thesis is to investigate the shortcomings of existing Shape-From-Silhouette algorithms and develop improved algorithms to apply better to the problems of human articulated body modeling, motion tracking and rendering.

1.1 Thesis Outline

The remainder of this thesis is organized as follows. In Chapter 2 we give a background review of Shape-From-Silhouette. The review includes the definitions of Visual Hulls, common methods to represent them, and a discussion of the use of Visual Hulls in human related applications. In Chapter 3 we quantitatively analyze the effect of silhouette noise on the accuracy of SFS and propose a fast voxel-based SFS algorithm called SPOT using the

results of the analysis. A practical real-time system is built based on SPOT to reconstruct 3D human voxel models. Different body parts modeled by simple ellipsoidal shells are then used to fit the reconstructed voxels.

In Chapter 4 we introduce a new Visual Hull representation called Bounding Edge. The relation of Bounding Edges with the Second Fundamental Property of Visual Hulls is discussed together with a comparison of the other Visual Hull representations. In the next two chapters, we propose algorithms to perform Shape-From-Silhouette across time. Chapter 5 focuses on rigid objects. We first study the ambiguity problem of aligning two Visual Hulls and show how colors can be used to break the ambiguity. Then we propose a temporal SFS algorithm which extracts points (Colored Surface Points) on the surface of the object and combines shape information and color stereo to align and refine Visual Hulls. In Chapter 6 the rigid object temporal SFS algorithm is extended to articulated objects by iteratively segmenting the Colored Surface Points and estimating the motion of the rigid parts of the articulated object. Both chapters include extensive synthetic and real experimental results.

Chapters 7, 8 and 9 focus on applying the temporal SFS algorithms to human-related problems. In Chapter 7 a step-by-step system is proposed to acquire a full kinematic model of a person (including 3D shape of body parts and joint locations). In Chapter 8 the kinematic model is used to perform motion capture of the same person in new video sequences using an image-based articulated object tracking algorithm very similar to the temporal SFS algorithms. We include experimental results and demonstrate that the tracking algorithm works well for both simple and complex motions. Chapter 9 investigates the problem of rendering the acquired human articulated model based on image-based techniques, together with potential use of the rendering algorithm in applications such as motion transfer and editing. Finally, in Chapter 10 we enumerate the contributions of this thesis and discuss several possible future directions.

Chapter 2

Shape-From-Silhouette and Visual Hulls

As its name implies, Shape-From-Silhouette (SFS) is a method of estimating the shape of an object from its silhouette images. The concept of using silhouettes for 3D shape reconstruction was first introduced by Baumgart in 1974. In his PhD thesis [Bau74], Baumgart estimated the 3D shapes of a baby doll and a toy horse from four silhouette images. Since then, various different variations of the Shape-From-Silhouette methods have been proposed. For example, Aggarwal et al. [MA83, KA86] used volumetric descriptions to represent the reconstructed shape. Potmesil [Pot87], Noborio et al. [NFA88] and Ahuja et al. [AV89] all suggested using octree data structure to speed up SFS. Pujari derived the optimal positions and directions to take silhouette images for 3D shape reconstruction in [SP91]. Szeliski built a non-invasive 3D digitizer using a turntable and a single camera with Shape-From-Silhouette as the reconstruction method [Sze93]. In summary, SFS has become a popular 3D reconstruction method for static objects.

2.1 Basic Principle

The concept of reconstructing the shape of an object from its silhouette images is explained in Figure 2.1. In Figure 2.1(a) a head-shaped object casts silhouettes on the image planes

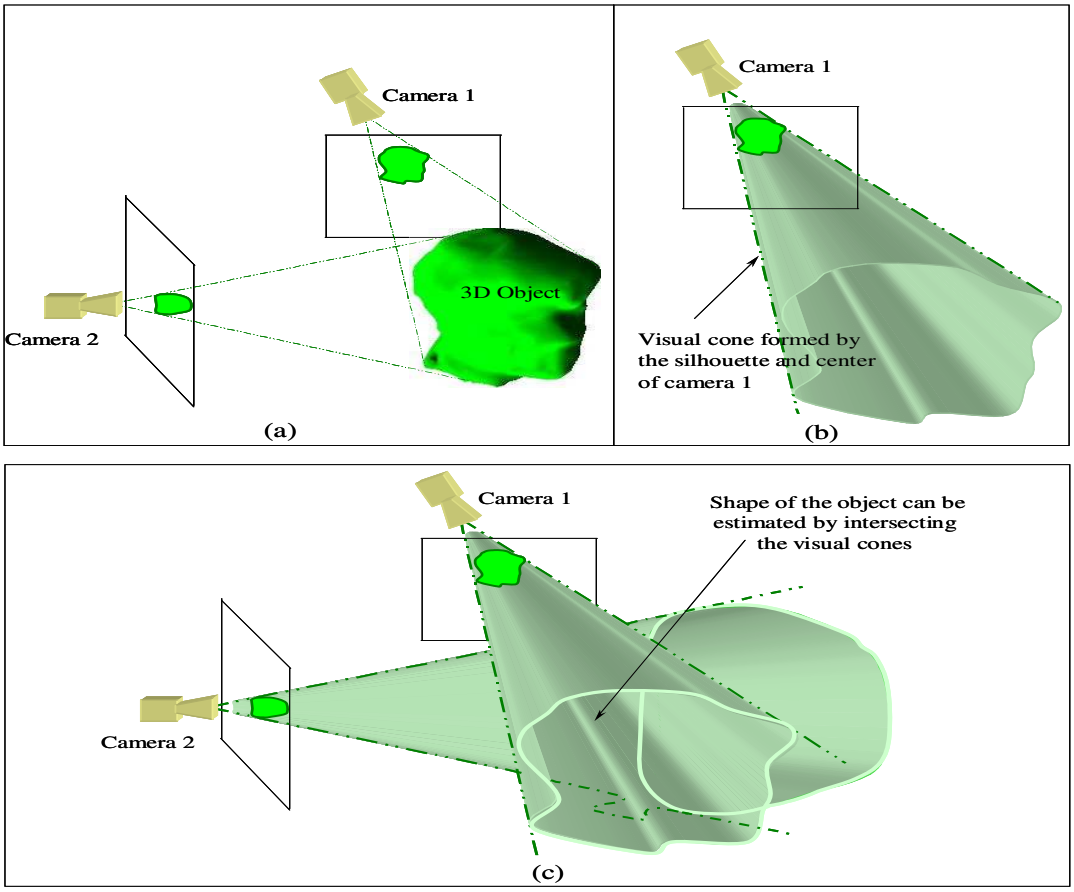


Figure 2.1: (a) A head-shaped object casts silhouettes on two cameras. (b) The visual cone formed by the silhouette image and the center of camera 1. (c) The shape of the object is estimated by intersecting all of the visual cones. The Visual Hull of a general 3D object contains curved surface patches making it difficult to represent and visualize.

of two cameras. For each camera and its silhouette image, there is a bounding volume that contains the object. This bounding volume, which is also called the visual cone, is constructed by projecting the silhouette into 3D space through the center of the camera as shown in Figure 2.1(b). Since each visual cone provides an upper bound on the object, the position and approximate shape of the object can be estimated by intersecting the visual cones from all the cameras as shown in Figure 2.1(c).

2.2 Visual Hulls

The term Visual Hull (VH) has been used in a general sense by researchers for over a decade to denote the shape estimated from the Shape-From-Silhouette principle: the intersection of the visual cones formed by the silhouettes and camera centers (Figure 2.1(c)). The term was first coined in 1991 by Laurentini [Lau91] who also published a series of subsequent papers studying the theoretical aspects of Visual Hulls of 3D polyhedral [Lau94, Lau95] and curved objects [Lau99]. Before discussing the different ways of representing and constructing Visual Hulls, we first define the problem scenario as follows.

2.2.1 Problem Scenario and Notation

Suppose there are K cameras positioned around a 3D object O. Let $\{S_j^k;\ \ k = 1; \cdots\cdots, K\}$ be the set of silhouette images of the object O obtained from the K cameras at time t_j. An example scenario is depicted in Figure 2.2 with a head-shaped object surrounded by four cameras at time t_1. It is assumed that the cameras are calibrated with $\Pi^k() : \boldsymbol{R}^3 \rightarrow \boldsymbol{R}^2$ and C^k being the perspective projection function and the center of camera k respectively. In other words $p = \Pi^k(P)$ are the 2D image coordinates of a 3D point P in the k^{th} image. As an extension of this notation, $\Pi^k(A)$ represents the projection of a volume A onto the image plane of camera k. Assume we have a set of K silhouette images $\{S_j^k\}$ and projection functions $\{\Pi^k\}$. A volume A is said to *exactly explain* $\{S_j^k\}$ if and only if its projection onto the k^{th} image plane coincides exactly with the silhouette image S_j^k for all $k \in \{1, \cdots\cdots, K\}$, i.e. $\Pi^k(A) = S_j^k$. If there exists at least one non-empty volume which explains the silhouette images exactly, we say the set of silhouette images is consistent, otherwise we call it inconsistent. Normally a set of silhouette images obtained from an object is consistent, unless there are camera calibration errors or silhouette image noise.

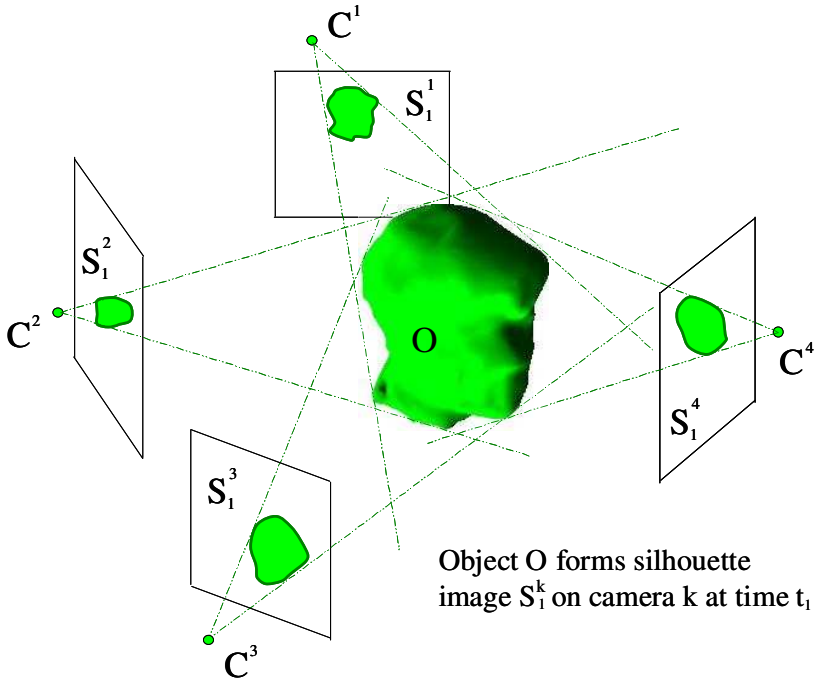


Figure 2.2: An example Shape-From-Silhouette problem scenario: a head-shaped object O is surrounded by four cameras at time t_1. The silhouette images and camera centers are represented by S_j^k and C^k respectively.

2.2.2 Definitions of Visual Hull

In this section we present two different ways (each of which has its pros and cons) to define Visual Hulls. Although these two definitions are seemingly different, they are in fact equivalent to each other. The proof of equivalence is given in Appendix A.

Visual Hull Definition I: Intersecting Visual Cones

The Visual Hull H_j with respect to a set of consistent silhouette images $\{S_j^k\}$ is defined to be the intersection of the K visual cones, each formed by projecting the silhouette image S_j^k into the 3D space through the camera center C^k.

This first definition, which is the most commonly used one in the SFS literature, defines the Visual Hull as the intersection of the visual cones formed by the camera centers and the silhouettes. Though this definition provides a direct way of computing the Visual Hull

from the silhouettes (see Section 2.3.1), it lacks information and intuition about the object (which forms the silhouettes). We therefore also use a second definition:

Visual Hull Definition II: Maximally Exactly Explains
The Visual Hull H_j with respect to a set of consistent silhouette images $\{S_j^k\}$ is defined to be the largest possible volume which exactly explains *$\{S_j^k\}$ for all $k = 1, \cdots\cdots, K$.*

Generally for a consistent set of silhouette images $\{S_j^k\}$, there are an infinite number of volumes (including the object O itself) that *exactly explain* the silhouettes. Definition II defines the Visual Hull H_j as the largest one among these volumes. Though abstract, this definition implicitly expresses one of the useful properties of Visual Hull: the Visual Hull provides an upper bound on the object which forms the silhouettes. To emphasize the importance of this property, we state it as the first fundamental property of Visual Hulls.

2.2.3 First Fundamental Property of Visual Hulls

First Fundamental Property of Visual Hulls (1^{st} FPVH):
The object O that formed the silhouette set S_j^k lies completely inside the Visual Hull H_j constructed from S_j^k.

The 1^{st} FPVH is very important as it gives us useful information on the object O, especially in applications such as robotic navigation or obstacle avoidance. The upper bound given by the Visual Hull gets tighter if we increase the number of distinct silhouette images. Asymptotically if we have every possible silhouette images of a *convex* object, the Visual Hull is exactly equal to the object. If the object is not convex, the Visual Hull may or may not be equal to the object.

2.3 Representation and Construction

2.3.1 Two-Dimensional Surface Based Representation

For a consistent set of silhouette images, its Visual Hull can be (according to Definition I) constructed by intersecting the visual cones directly. By doing so, the Visual Hull is represented by 2D surface patches obtained from intersecting the surfaces of the visual cones. For illustration purpose, an example in two-dimensions is given in Figure 2.3(a) in which the Visual Hull is constructed by intersecting the 2D visual wedges. Although simple in 2D and there exists fast algorithms for computing cones intersection for 3D *polyhedral* objects [BMM01], direct cone intersection representation is difficult to use for general 3D objects. The Visual Hull of a general 3D object consists of curved and irregular surface patches which are difficult to represent using simple geometric primitives. The computational complexity and numerical instability of intersecting surfaces with lines and planes in 3D are also reasons why researchers approximate general 3D objects to polyhedral shape when intersecting visual cones [BMM01].

The example in Figure 2.1(c) illustrates that it is difficult to express, represent or even visualize the surface patches of the Visual Hull of a general 3D object. Recently Buehler et al. [BMM01] proposed an approximate way to compute Visual Hull directly using the visual cone intersection method by approximating all 3D objects as having polyhedral shapes. Since polyhedral objects produce polygonal silhouette images, their Visual Hulls consist of only planar surface patches which can be more readily computed and represented than curved surface patches.

2.3.2 Three-Dimensional Volume Based Representation

Since it is difficult to intersect visual cones of general 3D objects, other more effective ways have been proposed to construct 3D Visual Hulls from the silhouette images. The approach which is used by most researchers [Pot87, NFA88, AV89, Sze93] is volume based

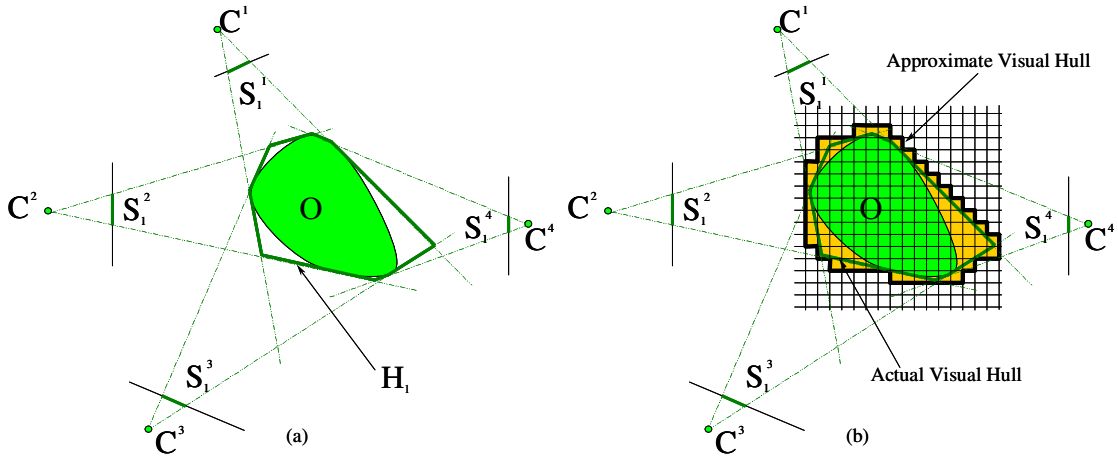


Figure 2.3: A two dimensional example of constructing the Visual Hull H_1 from the silhouettes $\{S_1^k\}$ and camera centers $\{C^k\}$: (a) by direct intersection of visual wedges, (b) by voxel-based approximation. The orange-shaded region (bounded by thick black lines) represents the approximate Visual Hull while the polygon (outlined in green) denotes the true one. The former is significantly larger than the latter.

construction. One version of this approach, also known as voxel-based SFS is given as follows:

Standard Voxel-based Shape-From-Silhouette Algorithm

1. Divide the space of interest into $N \times N \times N$ discrete voxels v_n, $n = 1, \cdots\cdots N^3$.
2. Initialize all the N^3 voxels as *inside* voxels.
3. For $n = 1$ to N^3 {
   For $k = 1$ to K {
   (a) Project v_n into the k^{th} image plane by the projection function $\Pi^k()$;
   (b) If the projected area $\Pi^k(v_n)$ lies completely outside S_j^k,
   then classify v_n as *outside* voxel;
   }
   }
4. The Visual Hull H_j is approximated by the union of all the *inside* voxels.

Voxel-based SFS uses the same principle of visual cone intersection. However, the Visual Hull is represented by 3D volume elements ("voxels") rather than 2D surface patches. The space of interest is divided into discrete voxels which are then classified into two categories: *inside* and *outside*. The union of all the *inside* voxels is an approximation of the Visual Hull. For a voxel to be classified as *inside*, its projection (step 3(a)) on each and every one of the K image planes has to be inside or partially overlap with the corresponding silhouette images. If the projection of the voxel is totally outside any of the silhouette images, it is classified as *outside*. Figure 2.3(b) gives a 2D example of this voxel-based method. The area of interest is divided into 16 by 16 squares. The convex polygon represents the true Visual Hull while the shaded region denotes the approximate Visual Hull obtained using the 2D version of the standard voxel-based SFS algorithm. The approximate 2D VH is significantly larger than the true one. This is one of the disadvantages of using discrete voxels to represent Visual Hulls.

2.4 Silhouette Extraction

As crucial as it is to represent and construct Visual Hulls effectively, accurate silhouette extraction is also of great importance to the process of Shape-From-Silhouette. In this section, we describe a simple background subtraction algorithm which serves as the core component of extracting silhouettes for all of the real data sequences in this thesis. Naturally there are other silhouette extraction algorithms in the literature, such as [SB96, HHD99, EHD99, IBL00, RT00], which can also be used.

One difficult problem of background subtraction is to remove shadows. In our algorithm, two techniques are used to tackle this problem: (1) use color information to distinguish shadow and non-shadow pixels and (2) use different thresholds based on the type of the pixels. The algorithm is summarized as follows with $\boldsymbol{I}_{BG}(u, v)$ and $\boldsymbol{I}_{RT}(u, v)$ representing the color vector of the $(u, v)^{th}$ pixel of the background and runtime images respectively:

Real-time Background Subtraction Algorithm

1. Calculate the intensity difference

   $$I_{DIFF}(u,v) = \|\boldsymbol{I}_{RT}(u,v) - \boldsymbol{I}_{BG}(u,v)\|.$$

2. If $I_{DIFF}(u,v) > T^U$,

   then set the $(u,v)^{th}$ pixel as a silhouette pixel, stop.

3. If $I_{DIFF}(u,v) < T^L$,

   then set the $(u,v)^{th}$ pixel as a non-silhouette pixel, stop.

4. Calculate the color difference

   $$\theta(u,v) = \cos^{-1}\left[\frac{\boldsymbol{I}_{RT}(u,v)\cdot\boldsymbol{I}_{BG}(u,v)}{\|\boldsymbol{I}_{RT}(u,v)\|\|\boldsymbol{I}_{BG}(u,v)\|}\right].$$

   If $\theta(u,v) > T^C$,

   then set the $(u,v)^{th}$ pixel as a silhouette pixel,

   else set the $(u,v)^{th}$ pixel as a non-silhouette pixel.

Here, $\|\boldsymbol{I}(u,v)\|$ represents the norm of a vector $\boldsymbol{I}(u,v)$ and $\cdot$ is the dot product operator. The algorithm basically consists of three tests. Steps 2 and 3 test the intensity difference between the run-time and background pixels. If the intensity difference is very large (compared to an up threshold constant T^U), then the pixel is treated as a silhouette pixel. If the intensity difference is very small (compared to a low threshold constant T^L), then the pixel is classified as a background pixel. For pixels with intensity differences between T^L and T^U, a third test (Step 4) is performed to determine if it is a shadow pixel. The value θ is the angle between the vectors $\boldsymbol{I}_{RT}$ and $\boldsymbol{I}_{BG}$ in the RGB color domain and hence is a measure of the color difference between the run-time and background pixels. For a shadow pixel, the color difference between run-time and background should be small because the difference lies mainly in the intensity values. Hence by comparing θ with a color threshold constant T^C, most of the shadow pixels can be removed.

The second technique we employed in our background subtraction process is to use different threshold constants for different regions of the image. The background image

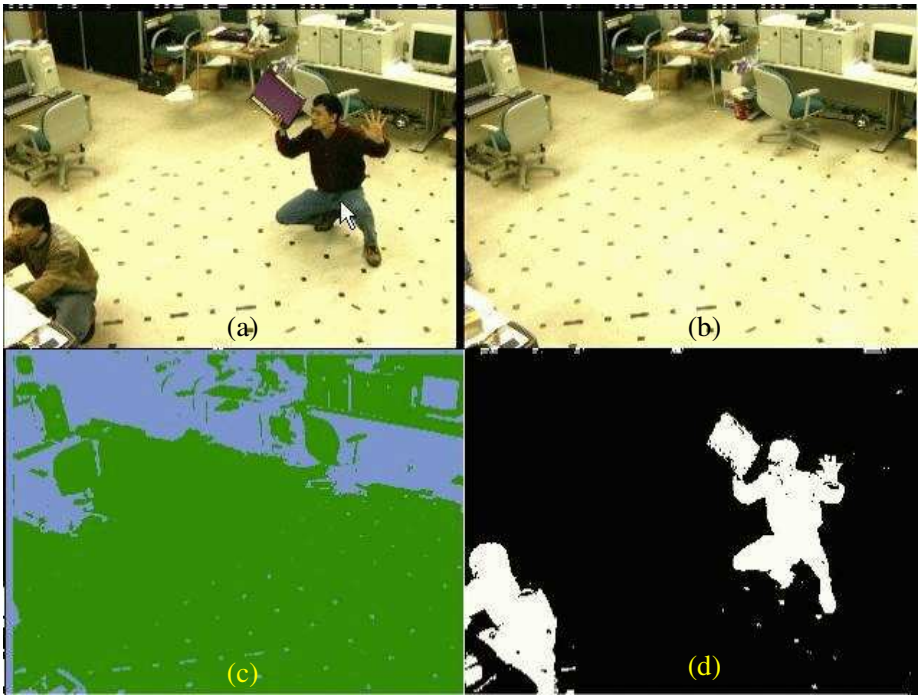


Figure 2.4: Example images of the background subtraction algorithm : (a) Run-Time image, (b) Background image, (c) Segmented background image, (d) Extracted foreground silhouette.

is automatically segmented into different regions using color information. For example, Figure 2.4(c) shows the segmentation of the background image in Figure 2.4(b) into two regions: floor and non-floor. The rationale behind this is that different types of regions have different color statistics and shadow probabilities. For example, the floor region has a higher probability of having shadows than the non-floor region and therefore different thresholds are used. This region-based approach is more flexible than the simplest method of using only one set of thresholds for the whole image. Moreover, compared to the method of having thresholds for *each* pixel, our approach is more practical and accurate. The number of thresholds in our approach is not high and hence we can determine each of them manually. On the other hand, in the pixel-based method, the large number of thresholds are usually determined using pixel color variances and cannot be fine tuned individually. Figure 2.4(d) shows the foreground silhouette image extracted from the images in Figures 2.4(a) and (b). Note that the shadows cast by the legs are removed completely.

The above real-time background subtraction algorithm was first introduced and used in [CKBH00] for the voxel-based real-time SFS system to be discussed in Chapter 3. It is easy to implement and fast enough for extracting silhouettes at 30 frames per second. However, as with all other background subtraction methods, our algorithm is not perfect and the extracted silhouette contain errors and noise (see Section 3.1 for the study of the effect of noisy silhouette on voxel-based SFS). For the real data sequences used in Chapter 4 through Chapter 8, since real-timeness is not a requirement, the silhouettes are cleaned up using connected component analysis and morphological operations [Jai89] after being extracted by our real-time background subtraction algorithm. Manual inspection is then used to make sure no significant part of the foreground silhouette is missing. Alternatively, other non real-time background subtraction methods [SB96, RT00] can be used.

2.5 Advantages and Disadvantages

Estimating shape using SFS has many advantages. First of all, silhouettes are readily and easily obtainable, especially in indoor environment where the cameras are static and there are few moving shadows. The implementation of most Visual Hull construction methods is also relatively straightforward, especially when compared to other shape estimation methods such as multi-baseline stereo [OK93] or space carving [KS00]. Moreover, from the First Fundamental Property of Visual Hull, SFS gives us an upper bound on the shape of the object. This inherently conservative property is particularly useful in applications such as obstacle avoidance in robot manipulation and visibility analysis in navigation where an upper bound on the shape of the object is preferred to a lower bound. All these advantages have prompted a large number of researchers to apply SFS to solve other computer vision and graphics problems beyond 3D reconstruction. Examples include human related applications such as virtual human digitization [MTG97], body shape estimation [KM98], motion tracking/capture [DF99, BL00] and image-based rendering [BMMG99].

On the other hand, SFS suffers from a number of limitations and inadequacies. Existing SFS methods involve time-consuming testing steps which hinder their use in real-time applications. SFS is also sensitive to errors in silhouette extraction and camera calibration, making it less desirable to use when the silhouette images are very noisy. Moreover, the Visual Hull obtained from SFS is only an approximation of the actual object shape. The approximation can be very coarse when there are only a few cameras, posing a disadvantage for SFS in applications such as detailed shape acquisition and realistic re-rendering of objects. In this thesis, we show how to overcome some of these limitations and apply SFS to construct detailed human body models for motion capture and rendering.

Chapter 3

Real-time Shape-From-Silhouette

In order to use SFS in real-time applications, the computational bottleneck of existing methods has to be identified and replaced by faster alternatives. The behavior of the alternatives under noisy silhouette images has to be studied in order to maintain reasonable shape estimates. In this chapter, we develop a fast voxel-based SFS algorithm called SPOT (Sparse Pixel Occupancy Test) which takes into account the effect of noisy silhouette images. A real-time system for reconstructing 3D volumetric models of people performing arbitrary motion is built based on this fast algorithm. Note that some of the material presented in this chapter first appeared in the paper [CKBH00].

3.1 Analysis of Voxel-based SFS with Noisy Silhouettes

Visual hull construction using the voxel representation is by far the most popular SFS method used by researchers because it is easily implementable and gives reasonable results in applications where approximate but complete shape information of the object is required. The speed of the standard voxel-based SFS algorithm described in Section 2.3.2 depends heavily on two procedures : (1) voxel projection (step 3(a) of the algorithm) and (2) silhouette overlap testing (step 3(b) of the algorithm). These two procedures, together

with the quality of the silhouette images, also determine the accuracy of the estimated Visual Hull. Silhouette images generated in real-time are always noisy. Figure 3.1 shows a foreground image and its noisy silhouette image obtained using the real-time background subtraction method described in Section 2.4. Although post-processing steps, such as morphological operations and connected component analysis [Jai89] can be applied to clean up the silhouette, they are time consuming computations. The same real-time limitation also prohibits us from using probabilistic approaches such as those proposed for voxel coloring in [DV99, BDC01], or the graph cuts approach (which does not require hard labeling of the voxels) proposed by Snow et al. in [SVZ00]. Thus the aim of this section is to propose fast implementations of procedures (1) voxel projection and (2) silhouette overlap testing, while understanding the effect of silhouette noise on the implementations. To characterize the quality of the silhouette images, hereafter let ξ be the probability that during silhouette extraction, a non-silhouette pixel is wrongly marked as a silhouette pixel. Likewise, let η represent the probability that a silhouette pixel is wrongly marked as a non-silhouette pixel. The noise is assumed to be independent between pixels. There are two ways of determining ξ and η : theoretically by assuming certain error probability distribution functions or experimentally by checking wrongly marked pixels in a large set of noisy silhouettes. For example, after tunning the threshold constants in our real-time background subtraction method (see Section 2.4), ξ and η are found experimentally to be 0.021 and 0.043 respectively.

One way of finding the projected voxel $\Pi^k(v_n)$ is to project the eight vertexes of v_n onto the k^{th} image plane and compute the convex hull of the projected points. Suppose on average, there are Z pixels inside the convex hull. A straightforward way to implement silhouette overlapping is to check all Z pixels and if at least Z_ε of them lie inside the silhouette image S_j^k, the voxel is said to be inside S_j^k. Due to silhouette noise, this implementation causes voxel misclassification. There are two types of voxel misclassification: False Acceptance (FA) which means an *outside* voxel is misclassified as *inside* and False Rejection

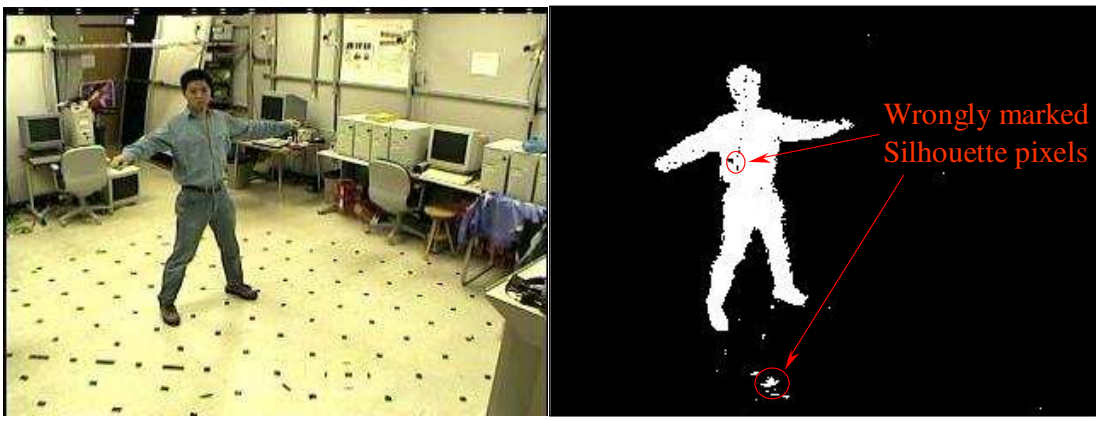


Figure 3.1: An example of a foreground image and its silhouette extracted by real-time background subtraction method described in Section 2.4. There are wrongly marked pixels in the silhouette image due to noisy original and background images.

(FR) which means an *inside* voxel is misclassified as *outside*. The probabilities $P(FR)$ and $P(FA)$ depend on ξ, η, Z and Z_ε. For voxels that are either totally outside or totally inside the silhouette (i.e. non-boundary voxels), $P(FR)$ and $P(FA)$ are found to be

I. False Acceptance :

$$P(FA) = \left[\sum_{i=Z_\varepsilon}^{Z} \binom{Z}{i} \xi^i (1-\xi)^{Z-i} \right]^K . \tag{3.1}$$

II. False Rejection :

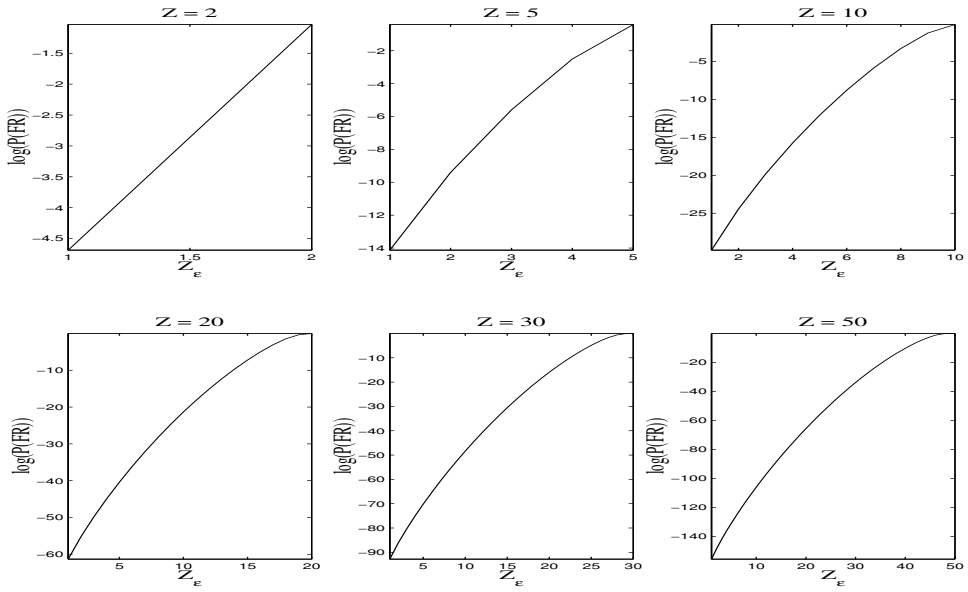
$$P(FR) = p \sum_{j=0}^{K-1} (1-p)^j ; \qquad p = \sum_{i=Z-Z_\varepsilon+1}^{Z} \binom{Z}{i} \eta^i (1-\eta)^{Z-i} . \tag{3.2}$$

The exponential of K (which represents the total number of cameras used) in Equation (3.1) is due to the fact that an *outside* voxel has to be misclassified as *inside* in *all* the K images for a FA to happen. In Equation (3.2), p is the probability that an *inside* voxel is classified as *outside* in *one* image. The summation over j comes from the fact that an

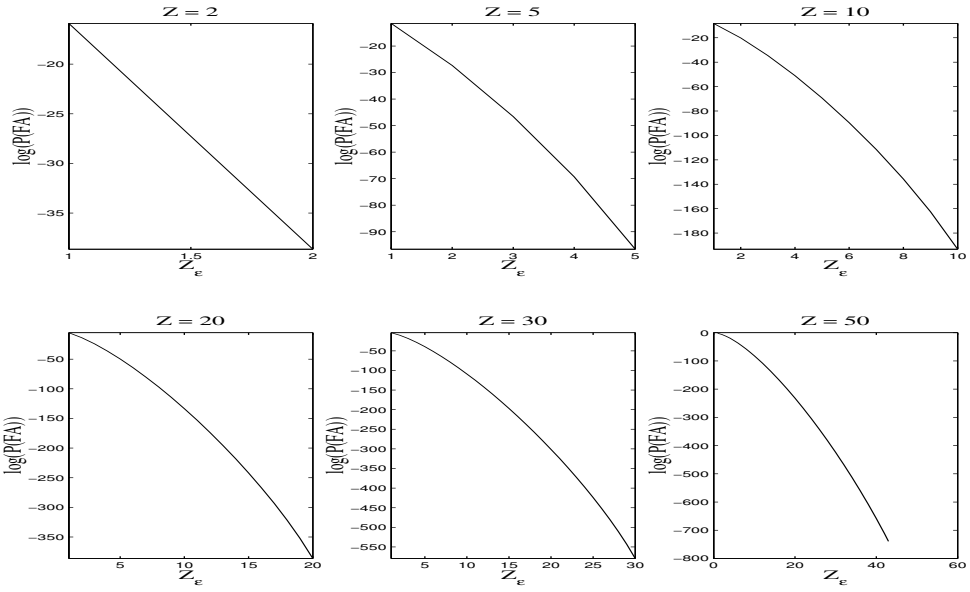
inside voxel is misclassified as outside once it is misclassified in one image and the rest of the images are not tested. Note that these equations are applicable only to totally inside or totally outside voxels. With slight modifications, similar analysis can be applied to boundary voxels although they are not considered here as the number of them are relatively fewer than the number of totally inside or totally outside voxels.

For fixed Z, ξ and η, the False Rejection probability $P(FR)$ increases with Z_ε while the False Acceptance probability $P(FA)$ decreases when Z_ε increases. Using the values $\xi = 0.021$ and $\eta = 0.043$ (the values determined experimentally), graphs of $\log(P(FR))$ and $\log(P(FA))$ versus Z_ε for different values of Z are shown in Figure 3.2(a) and (b) respectively. The optimal Z_ε for a particular Z is chosen by considering the total error probability $P(FA) + P(FR)$. The graph of $\log(P(FA) + P(FR))$ against Z_ε for different Z is plot in Figure 3.3(a). Clearly, different values of Z have different optimal Z_ε. For example the optimal Z_ε is 1 for $Z = 2$ while the optimal Z_ε is 7 when Z is 30. The graphs of optimal Z_ε and optimal $\log(P(FA) + P(FR))$ against Z are plot in Figure 3.3(b)(c). These graphs are useful in determining the optimal silhouette testing rule and the corresponding error probabilities.

Besides serving as guidelines for choosing the optimal Z_ε, the graphs in Figure 3.3(b)(c) also give us insight into deciding the voxel size of the system. Intuitively the smaller the voxel size, the better the shape approximation (see Figure 2.3(b) as an example) because the discretization error is smaller. Niem verified this conjecture in [Nie97] by analyzing effect of voxel size on shape reconstruction error *without considering noisy silhouettes*. However, the graph in Figure 3.3(c) shows that when there is noise in the silhouettes, the smaller the voxel size, the smaller the number of projected pixels Z which implies the larger the misclassification probabilities. Therefore to pick an optimal voxel size for a voxel-based SFS system, there is a trade-off between lowering discretization errors (favoring smaller voxels) and minimizing misclassification probabilities (favoring larger voxels).

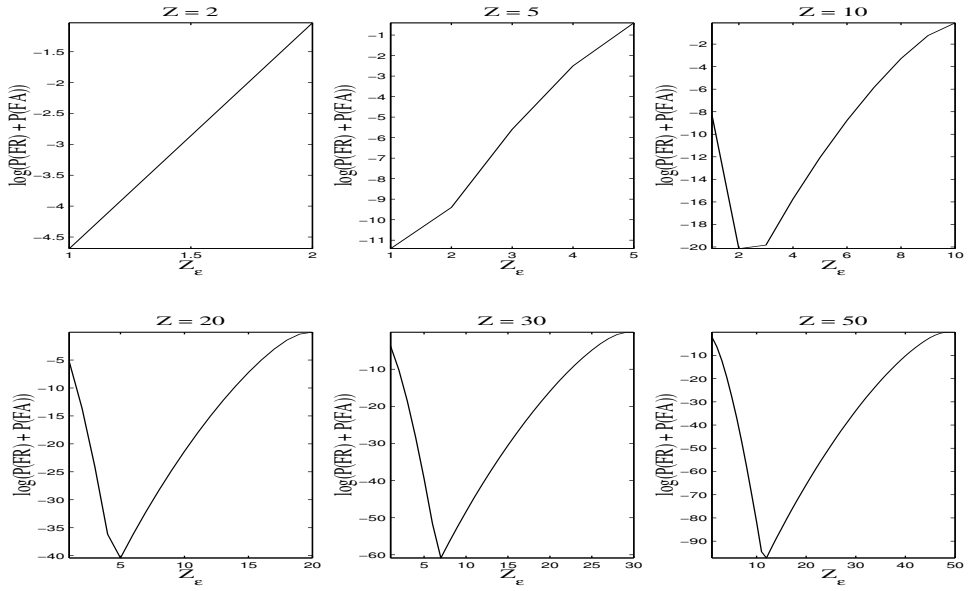


(a) Graph of log(P(FR)) against Z_ε for different Z

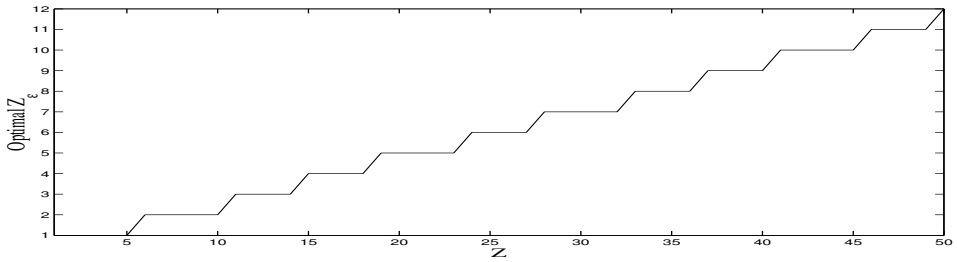


(b) Graph of log(P(FA)) against Z_ε for different Z

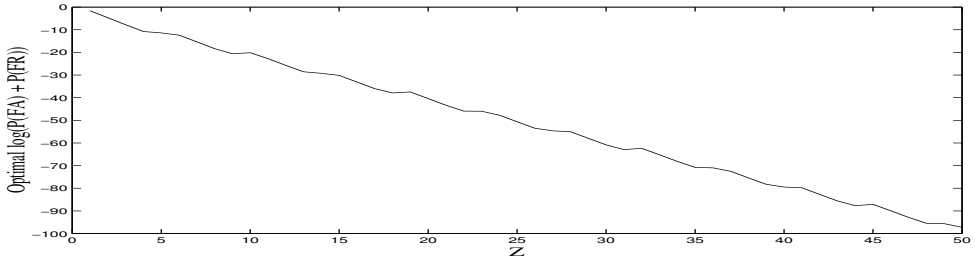
Figure 3.2: (a) Graphs of log($P(FR)$) vs. Z_ε for different Z, (b) Graphs of log($P(FA)$) vs. Z_ε for different Z.



(a) Graph of log(P(FR) + P(FA)) against Z_ε for different Z



(b) Graph of Optimal Z_ε against Z



(c) Graph of Optima log(P(FA)+P(FR)) against Z

Figure 3.3: (a) Graphs of $\log(P(FA) + P(FR))$ vs. Z_ε for different Z, (b) Graph of optimal Z_ε vs. Z, (c) Graph of optimal $\log(P(FA) + P(FR))$ vs. Z.

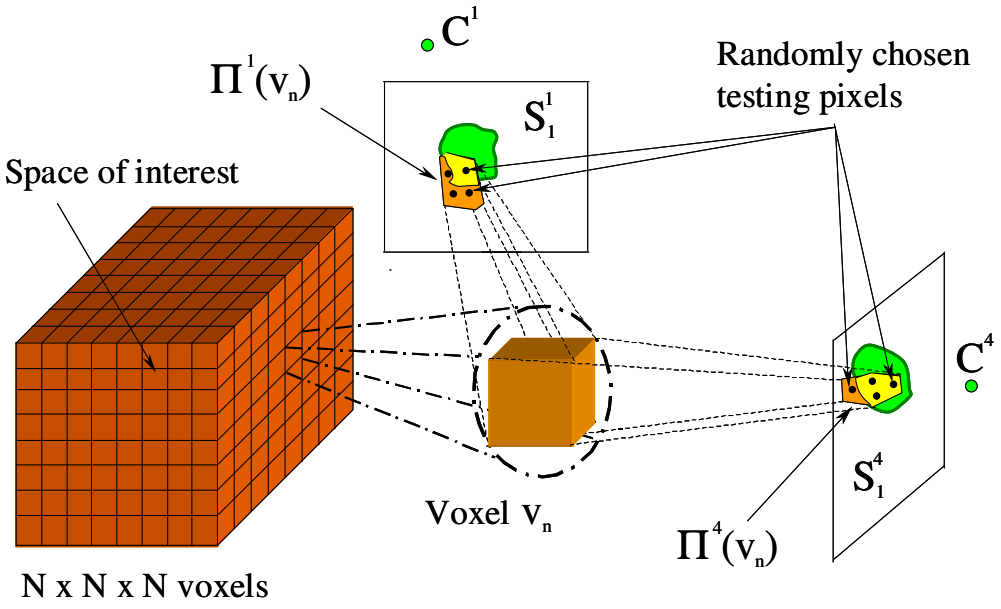


Figure 3.4: The SPOT-modified voxel-based Shape-From-Silhouette.

3.2 A Fast Voxel-based SFS Algorithm: SPOT

To increase the speed of the algorithm, assume that we pre-compute the convex hull of each projected voxel and store the pixel locations in a lookup table. To apply the straightforward implementations of procedures (1) voxel projection and (2) silhouette overlap testing, we need $O(N^3Z)$ memory and $O(N^3Z)$ testing operations for a volume of $N \times N \times N$ voxels. This high memory requirement and number of tests make this straightforward approach too slow for real time applications. Here we propose faster and more efficient implementation. For each image and each voxel, instead of testing all Z pixels, Q random pixels are chosen within the Z-pixel convex hull. If at least Q_ε of these chosen pixels lie inside the silhouette image S_j^k, the voxel is said to be inside S_j^k. We name this approach Sparse Pixel Occupancy Test (SPOT) and is stated formally as below. The SPOT-modified voxel-based SFS algorithm is also illustrated in Figure 3.4.

Sparse Pixel Occupancy Test (SPOT) Algorithm

1. For voxel v_n and camera k, randomly choose Q points inside the projection of $\Pi^k(v_n)$, represented by $q(l)$ where $l = 1, \cdots\cdots, Q$.
2. Initialize *incount* to 0.
3. For $l = 1$ to Q {
   If $q(l)$ is a silhouette pixel
   then increments *incount*.
   }
4. If $incount < Q_\varepsilon$
   then classify v_n as *outside* voxel,

SPOT can be incorporated into the standard voxel-based SFS easily by replacing the projection and testing steps (Steps 3(a) and (b) in the standard voxel-based SFS algorithm) by SPOT. There are two advantages of using SPOT. It is $\frac{Z}{Q}$ times faster and the memory requirement is $\frac{Z}{Q}$ times less than the straightforward approach. The downside is the misclassification probabilities also increase when we reduce the number of testing pixels.

One question that remains is how do we choose Q and Q_ε accordingly? Since we have assumed the pixels are independent of each other and the Q points are chosen randomly, the notion of testing Q out of Z pixels is theoretically equivalent to testing a projected area with Q pixels. In other words, Equations (3.1) and (3.2) and the graphs in Figures 3.2 and 3.3 are valid for analyzing SPOT if we replace Z by Q. The choice of Q is a compromise between speed, memory and accuracy. In real-time applications where speed is more important than quality, a low value of Q is used to trade accuracy for speed. For example, if we use $Q = 5$ for a 10-pixel projected voxel (i.e. $Z = 10$), we gain a factor of 2 in both speed and memory but the total misclassification probability $P(FA)+P(FR)$ also increases from 1.7e-9 to 1.1e-5 (see Figure 3.3(c)). Once Q is fixed, Q_ε is chosen according to Figure 3.3(b).

3.3 Real-time 3D Voxel Reconstruction of Human Motions

To show how SPOT is used in practical situations, a real-time system [CKBH00] is built to reconstruct and track human motions over time. The system consists of two parts: 3D voxel reconstruction and ellipsoid fitting.

3.3.1 Surface Voxel Reconstruction

The first part of the system consists of synchronized cameras placed evenly around the moving human subject. Each camera is connected to an individual local computer. The camera captures images continuously and the real-time background subtraction algorithm described in Section 2.4 is applied by each camera's local computer to extract the silhouettes. The synchronized silhouette images are then transferred, via network, from the local computers to a main host computer where SPOT-modified SFS is used to build the 3D voxel model of the person. Once the model is reconstructed, the surface voxels are extracted and used for ellipsoid fitting in the second part of the system.

3.3.2 Ellipsoid Fitting

After obtaining the 3D surface voxel data of the humans, ellipsoids are fit to the data in real-time. Six ellipsoidal shells (to model the head, the body, the two arms and the two legs of a human body) F_f where $f = 1, \cdots\cdots, 6$ are used. Each ellipsoidal shell F is represented by nine parameters: the center, the orientation and the sizes (lengths of the axes) denoted by $(x_{F_f}, y_{F_f}, z_{F_f})$, $\boldsymbol{R}_{F_f}$ and $(\alpha_{F_f}, \beta_{F_f}, \gamma_{F_f})$ respectively. The whole fitting process has to be fast enough for real-time processing. To achieve this goal, we use a two-step approach based on the Expectation-Maximization (EM) paradigm [DLR77].

The first step of the fitting process is to segment the voxel data by using a proximity criterion. A surface voxel v is assigned to belong to the ellipsoidal shell F_f (i.e. $v \in F_f$) if the voxel is closest to F_f (as compared to other ellipsoidal shells). In order to calculate the

distance between the voxel and the shells, the shell parameters estimated from the previous frame are used. Note that this proximity criterion is fast and easy to compute.

After the segmentation process, moment analysis is used to estimate the ellipsoidal shell parameters from the centers (x_v, y_v, z_v) of the surface voxels v which are belong to F_f. For example, the zeroth, first and second moments of x_v are defined as

$$M_0 = \sum_{v \in F_f} 1; \quad M_x = \frac{1}{\boldsymbol{M}_0} \sum_{v \in F_f} x_v; \quad M_{xx} = \frac{1}{\boldsymbol{M}_0} \sum_{v \in F_f} x_v x_v. \tag{3.3}$$

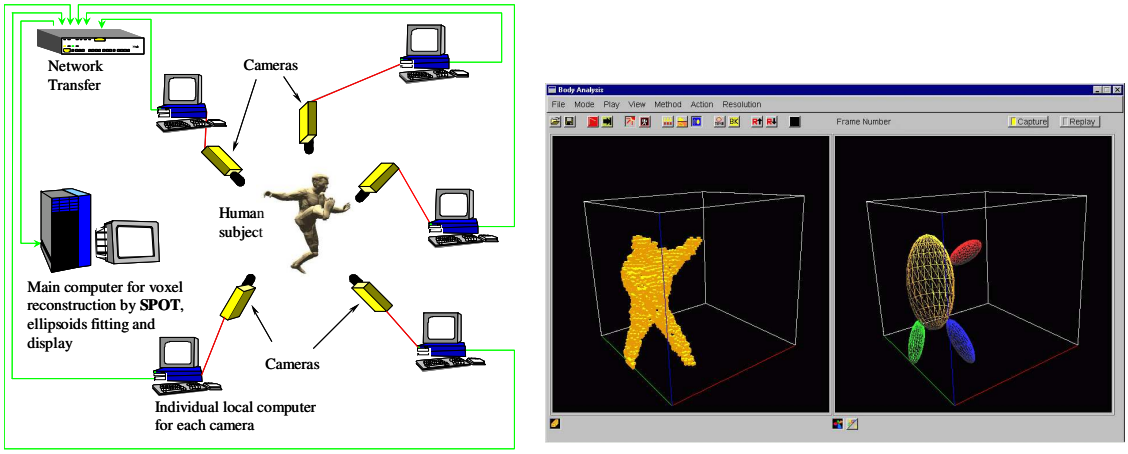
Similarly, moments of y_v and z_v (such as $M_y, M_{xy}, M_{yz}, M_{zz}$ etc.) can also be calculated using Equations (3.3). The ellipsoidal shell parameters are now estimated as

$$(x_{F_f} \;\; y_{F_f} \;\; z_{F_f}) = (M_x \;\; M_y \;\; M_z)\,, \tag{3.4}$$

$$\boldsymbol{M} = 4 \begin{bmatrix} M_{xx} & M_{xy} & M_{xz} \\ M_{yx} & M_{yy} & M_{yz} \\ M_{zx} & M_{zy} & M_{zz} \end{bmatrix} = \boldsymbol{R}_{F_f} \begin{bmatrix} \alpha_{F_f}^2 & 0 & 0 \\ 0 & \beta_{F_f}^2 & 0 \\ 0 & 0 & \gamma_{F_f}^2 \end{bmatrix} \boldsymbol{R}_{F_f}^T\,, \tag{3.5}$$

with $\boldsymbol{M}$ being the moment matrix. $\boldsymbol{R}_{F_f}$ and $(\alpha_{F_f}, \beta_{F_f}, \gamma_{F_f})$ can be obtained easily by performing an eigen-decomposition on the (3 by 3) matrix $\boldsymbol{M}$.

To perform the segmentation for the current frame, we need the estimated ellipsoidal shell parameters from the previous frame. To initialize the shells when the system is first started, we used an activation procedure. When the system is first started, only one ellipsoidal shell F_1 is activated for the fitting. All the voxel data is used to estimate the parameters of the first ellipsoid F_1 and the error of fitting is calculated. If this error is larger than a threshold this means that one ellipsoid shell is not enough to model the voxel data. In this case, the second ellipsoidal shell F_2 is activated so that two shells are used to fit the data. As the human moves his arms and legs around, the shells are activated sequentially to fit different parts of the body until all six of them are activated.



(a) (b)

Figure 3.5: (a) The system architecture of the real-time human motion model reconstruction system. (b) A screen shot of the user interface.

One problem of our two-step fitting approach is that the joint and size constraints of the human body are not incorporated. Hence fitting fails when the body parts are too close together which causes problems for the segmentation. For the same reason this algorithm cannot handle appearing and disappearing body parts very well.

3.3.3 System Architecture and Performance

The architecture of our system is illustrated in Figure 3.5(a). Using a simple user interface, the user can display and observe, in real time and from any view-point, the 3D models and the fit ellipsoidal shells. Figure 3.5(b) is a snapshot of the interface with the voxel models being shown on the left and the fit ellipsoidal shells on the right.

The following summarizes various aspects and parameters of the system.

1. Five cameras are used and each of them is connected to a 266 MHz PC. They are calibrated by planar calibration patterns with the method described in [Zha99]. The main

computer is equipped with dual Pentium II 400 MHz processors. The computers are connected together by a 100Mb/s hub.

2. The space of interest is a cube of size 2m × 2m × 2m divided into 64 × 64 × 64 (i.e. $N = 64$) voxels with a resolution of about 3cm for each voxel.

3. The images are captured at a resolution of 320x240 pixels. The silhouette extraction method in Section 2.4 gives silhouette error probabilities of $\eta = 0.043$ (the probability that a silhouette pixel is wrongly marked as a non-silhouette pixel) and $\xi = 0.021$ (the probability that a non-silhouette pixel is wrongly marked as a silhouette pixel).

4. The average number of pixels in a projected voxel is 10 (i.e. $Z \approx 10$). To obtain the desired speed, only two pixels (i.e. $Q = 2$) are chosen for each voxel in the silhouette overlap test. The optimal Q_ε is 1 according to Figure 3.3(b). This corresponds to a total misclassification probability $P(FA) + P(FR)$ of 0.0092, compared to a value of 1.78e-9 if we set $Q = 10$.

5. Table 3.1 summaries the approximate time required for each step. With the display function turned off, the system outputs results of higher than 15 frames per second.

Figure 3.6 shows frames selected from a movie clip **Realtime-SFS-reconstruction-fitting.mpg*** illustrating the reconstruction/fitting results of our system. Within each frame, the upper left picture is the run-time image captured by one of the five cameras and the lower left picture depicts its silhouette generated by the background subtraction method. The upper right picture shows the reconstructed voxels of the person using SPOT while the lower right picture represents the fit ellipsoids. There are two frames delay between the run-time image/silhouette computation and the reconstructed voxels/ellipsoids due to the pipeline processing of the system. Another movie clip **Realtime-SFS-system.mpg** shows

*All movie clips of this chapter can be found at http://www.cs.cmu.edu/~german/research/Thesis/Video/Chapter3/

Step	Approximate Time Required
Sending/Receiving Silhouette Images	15 ms
Image Acquisition	10 ms
Voxel Reconstruction without Surface extraction	35 ms
Voxel Reconstruction with Surface extraction	40 ms
Silhouette Generation	15 ms
Ellipsoid Fitting	10 ms
Model Display	100 ms

Table 3.1: The approximate processing time for each step in our system.

real-time footage of the actual system, the user-interface and the results captured during an experiment.

3.4 Related Work

Our real-time system for 3D Voxel reconstruction of human motion is partially inspired by the arts project TRACE of Penny et al. [PSB99]. While they use only one computer and focus on the artistic form of the resulting voxels (without fitting any model to the voxels), our system studies the accuracy of the reconstruction and fits simple ellipsoidal models to the reconstructed voxels. The application-based papers [BL00, BL01] by Laurentini following his series of theoretical papers [Lau94, Lau95, Lau97, Lau99] have a similar goal as our real time human motion reconstruction system. However, they concentrate on the motion capture/fitting aspects from the reconstructed voxels, thus are not real-time and there is no silhouette noise analysis. Niem performed error analysis of voxel-based reconstruction methods in [Nie97]. He focused on local shape error caused by the discrete resolution of the silhouettes, the voxels, the finite number of views, and to some extent camera calibration parameters. His analysis does not include silhouette image noises but

provide guidelines for choosing the number of viewpoints and the voxel size to obtain a given shape accuracy. Moreover, Prock and Dyer proposed a real-time voxel coloring algorithm using an octree data structure and a nearest neighbor search approach to maintain the details of the reconstructed voxels [PD98].

Since we introduced the SPOT algorithm [CKBH00], Buehler et al. have derived another fast algorithm for building and rendering Visual Hulls of 3D *polyhedral objects* using the visual cones intersection method [MBR+00, BMM01]. Their algorithm produces surface-based (mesh) Visual Hulls and is particularly suited for rendering. On the other hand, Mikic et al. built a system similar to ours (Section 3.3) to perform human posture estimation [MHTC01]. They extended our system by introducing a better articulated body model and voxel fitting strategy. They use fixed-size ellipsoids to model the human body and enforce joint constraints using the Extended Kalman Filter framework. Although they have not achieved real-time performance for their fitting procedure (20s per frame), their strategy is more systematic than our naive way of fitting ellipsoidal shells to the voxels.

3.5 Discussion

Although we have demonstrated that voxel-based SFS can be used to reconstruct 3D human motions in real time, it is difficult to achieve very accurate human motion tracking from the reconstructed voxels. There are two major reasons behind this difficulty. First, since only a small number of cameras (4-6) are used, the reconstructed voxel model is very coarse and lacks enough information for precise voxel fitting and motion tracking. Secondly the shape (ellipsoids) and joint information (for the system in [MHTC01]) of the human model used is not tailored to the particular human subject to be tracked. The inaccuracy in the kinematic information increases the errors in tracking for the high degree-of-freedom human body. To solve these two problems, in the next few chapters we first extend the traditional SFS so that silhouette images over time can be combined to build a detailed human model. We then extend this approach to acquire kinematic information of the human subject.

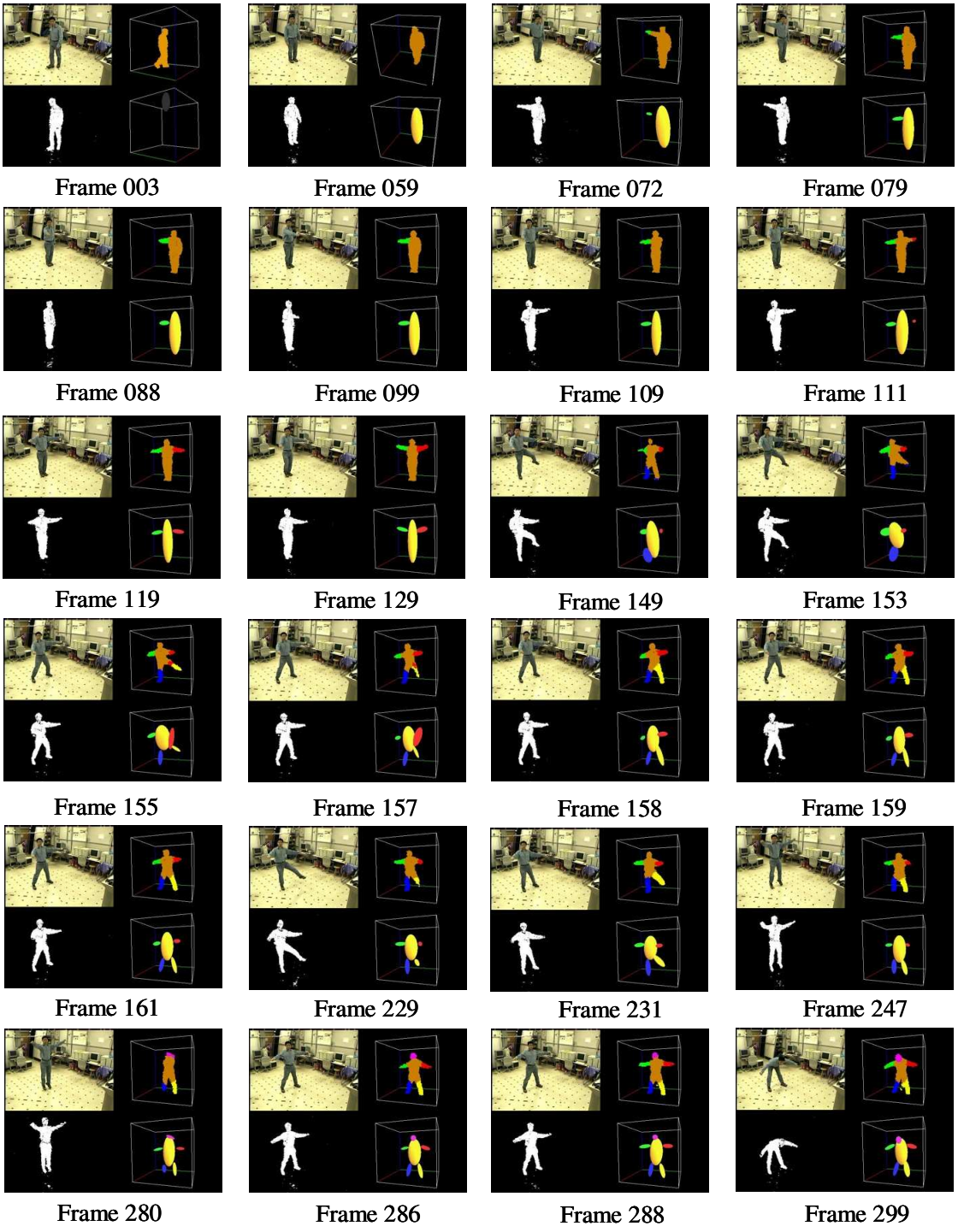


Figure 3.6: Twenty four selected frames from the movie clip **Realtime-SFS-reconstruction-fitting.mpg** illustrating our real-time human motion reconstruction system [CKBH00].

Chapter 4

A New Visual Hull Representation: Bounding Edge

In Chapter Two we presented two common ways to represent Visual Hulls: two-dimensional surface patches and three-dimensional discrete voxels. In this chapter, we propose a new representation for Visual Hulls using a one-dimensional entity called a Bounding Edge (BE). We first give the definition of Bounding Edges and explain how they can be constructed from the silhouette images. We then discuss the Second Fundamental Property of Visual Hulls which is closely related to the definition of a Bounding Edge. Finally we compare the advantages and disadvantages of the three Visual Hulls representations (discrete voxels, surface patches and Bounding Edges).

4.1 Definition of Bounding Edge

Consider a set of K silhouette images $\{S_j^k\}$ at a given time instant t_j. Let u_j^i be a point on the boundary of the silhouette image S_j^k. By projecting u_j^i into 3D space through the camera center C^k, we get a ray r_j^i. A Bounding Edge E_j^i is defined to be the part of r_j^i such that the projection of E_j^i onto the l^{th} image plane lies completely inside the silhouette S_j^l for all

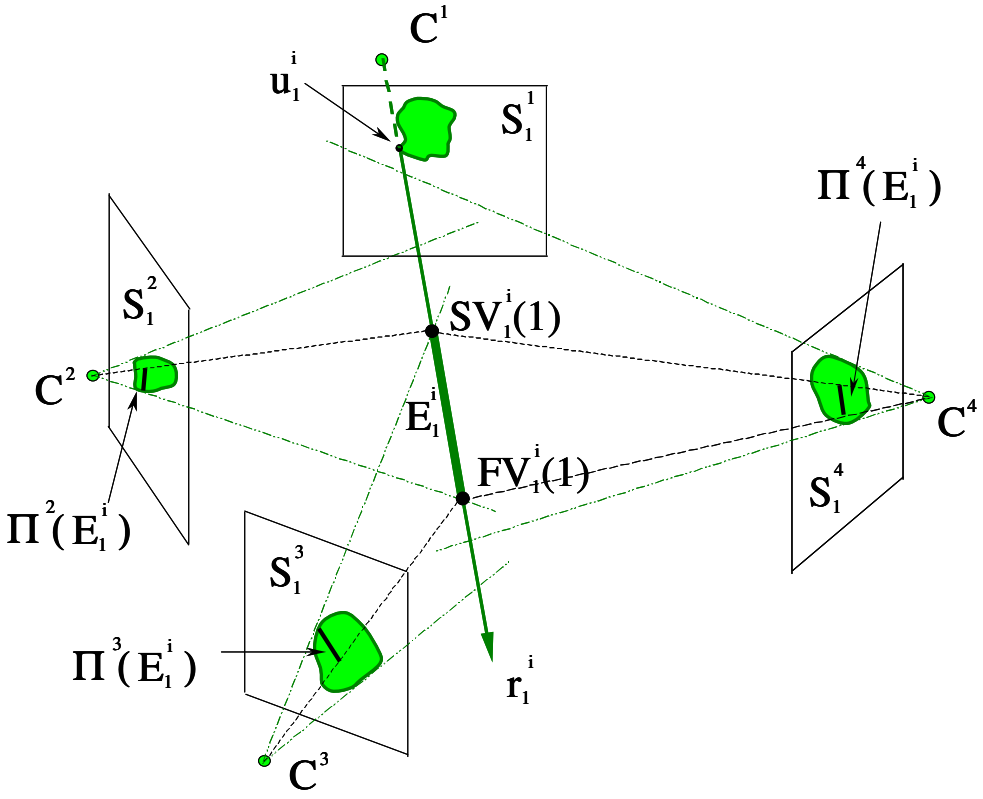


Figure 4.1: The Bounding Edge E_1^i is obtained by first projecting the ray r_1^i onto S_1^2, S_1^3, S_1^4 and then re-projecting the segments overlapped with the silhouettes back into the 3D space. E_1^i is the intersection of the reprojected segments.

$l \in \{1, \cdots\cdots, K\}$. Mathematically the condition can be expressed as

$$E_j^i \subset r_j^i \quad \text{and} \quad \Pi^l(E_j^i) \quad \subset \quad S_j^l \quad \forall \quad l \in \{1, \cdots\cdots, K\}\,. \tag{4.1}$$

Figure 4.1 illustrates the definition of a Bounding Edge at t_1. A Bounding Edge can be computed easily by first projecting the ray r_j^i onto the $K-1$ silhouette images S_j^l, $\quad l = 1, \cdots\cdots, K; \quad l \neq k$, and then re-projecting the segments which overlap with S_j^l back into 3D space. The Bounding Edge is the intersection of the reprojected segments. Note that the Bounding Edge E_j^i is not necessarily a continuous line. It may consist of several segments if any of the silhouette images are not convex. An example is shown in Figure 4.2 where the Bounding Edge contains two segments. Hereafter, a Bounding Edge E_j^i is denoted by a set of *ordered* 3D vertex pairs as follows:

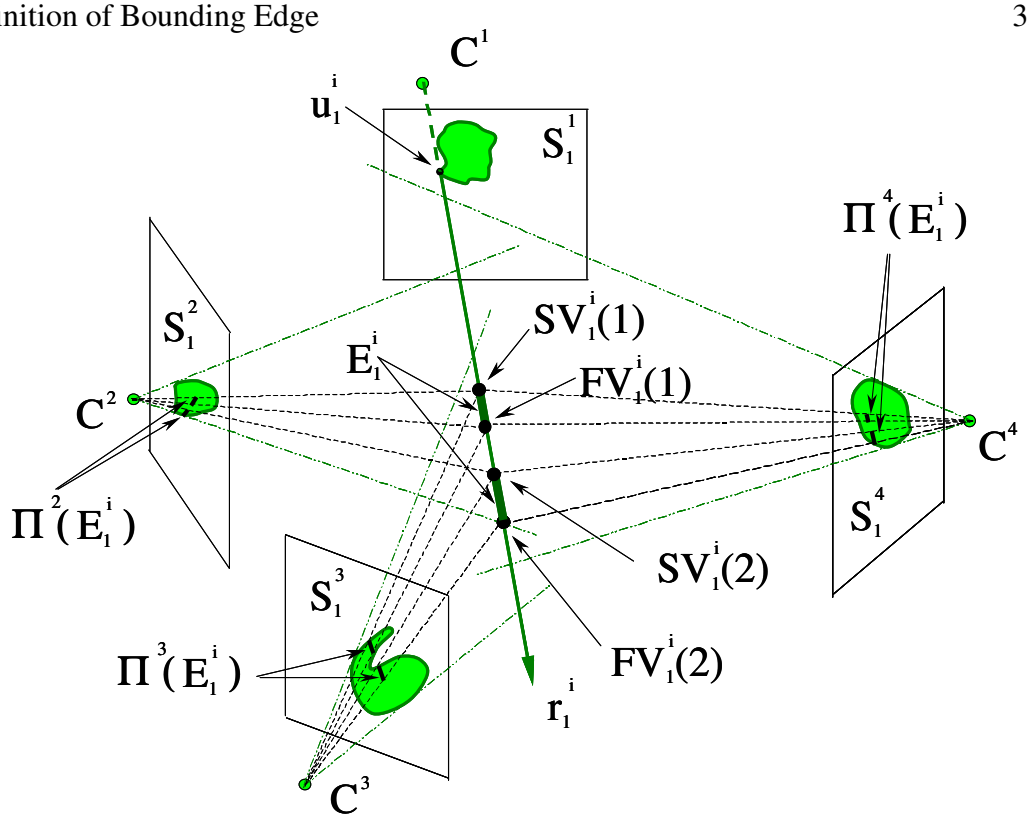


Figure 4.2: A situation where the Bounding Edge E_1^i consists of more than one segment when one or more of the silhouettes are not convex. In this case E_1^i contains two segments $\left(SV_1^i(1), FV_1^i(1)\right)$ and $\left(SV_1^i(2), FV_1^i(2)\right)$.

$$E_j^i = \left\{ \left(SV_j^i(m), FV_j^i(m)\right); \quad m = 1, \ldots, M_j^i \right\} , \tag{4.2}$$

where $SV_j^i(m)$ and $FV_j^i(m)$ represent the start vertex and finish vertex of the m^{th} segment of the Bounding Edge respectively and M_j^i is the number of segments that E_j^i is comprised of. By sampling points on the boundaries of all the silhouette images $\{S_j^k; \quad k = 1, \cdots\cdots, K\}$, we can construct a list of L_j Bounding Edges that represents the Visual Hull H_j.

An example of using a set of Bounding Edges to represent the Visual Hull of a real object is shown in Figure 4.3. Figure 4.3(b) shows six silhouette images of a toy dinosaur placed on a bunch of bananas (Figure 4.3(a)). Bounding Edges are extracted from the silhouette images and are shown from three different viewpoints in Figure 4.3(c). As a comparison, a voxel-based model is also built using the six silhouettes and is shown from different viewpoints in Figure 4.3(d).

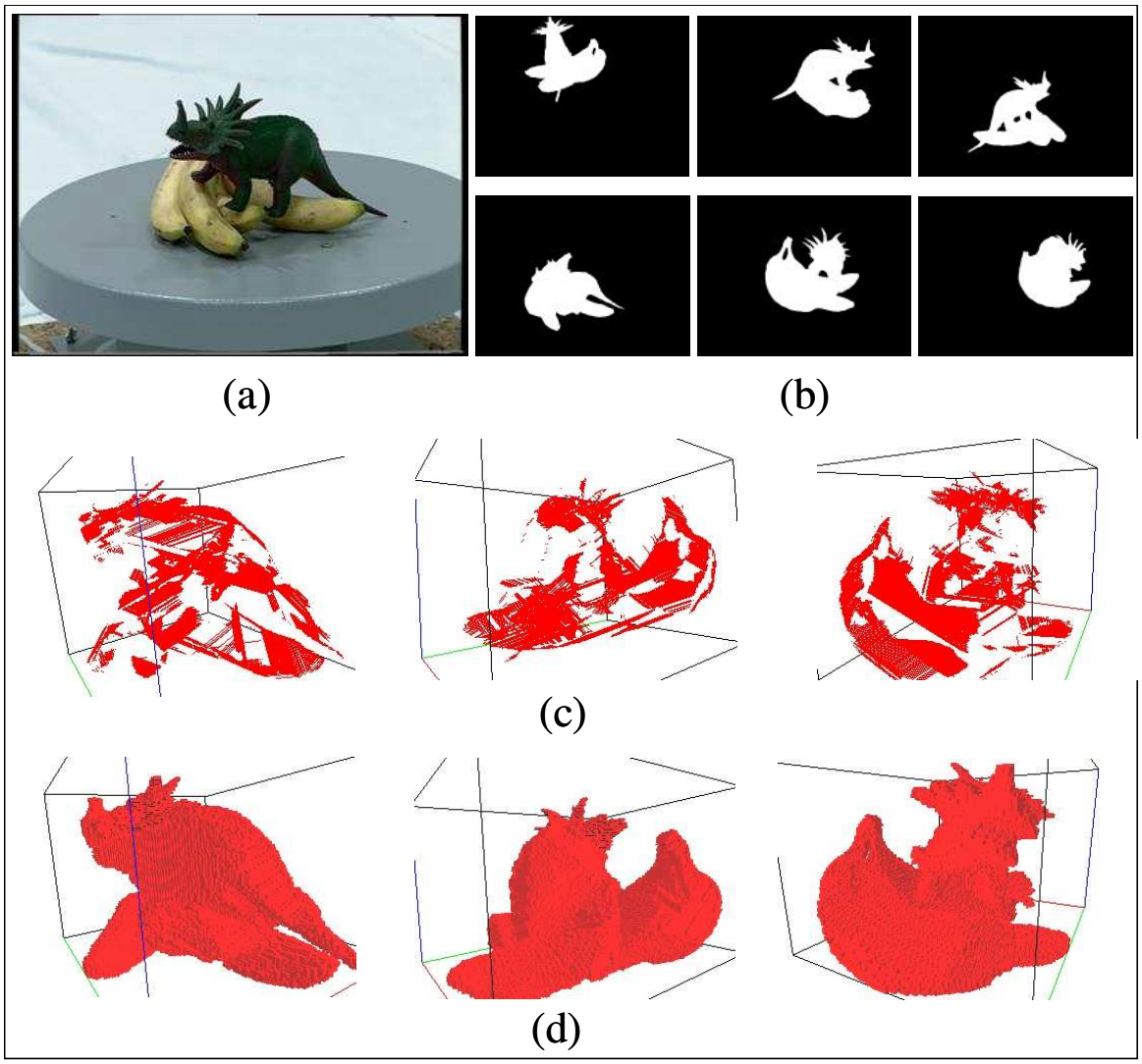


Figure 4.3: (a) A toy dinosaur placed on a bunch of bananas. (b) Six silhouette images of the dinosaur and the bananas captured from six different cameras. (c) Three different views of the Bounding Edges extracted from sampled boundary points of the six silhouette images. (d) Three different views of a voxel model reconstructed using the the standard voxel-based SFS algorithm as discussed in Section 2.3.2. Each side of the voxel is about 1.5cm long and the dimensions of the toy dinosaur are about 30cm by 14cm by 15cm.

4.2 Second Fundamental Property of Visual Hull

The most important property of the Bounding Edge representation is that its definition captures one aspect of Shape-From-Silhouette very naturally. To be precise, we state this property as the Second Fundamental Property of Visual Hulls as follows:

Second Fundamental Properties of Visual Hulls (2^{nd} FPVH)
Each Bounding Edge of the Visual Hull touches the object (that formed the silhouette images) at at least one point.

We have to emphasize that the Bounding Edges *touches* the object (tangentially) rather than *intersects* objects. Notice that segment originating from an interior point of the silhouette intersects (rather than touches) the object at at least one point. The advantage of the Bounding Edges touching the object rather than intersecting it is that the visibility of the points on the Bounding Edges can be easily determined using only the silhouette images. The details of determining visibility of points on Bounding Edges are discussed in Section 5.4.

The 2^{nd} FPVH allows us to use Bounding Edges to store and represent the one aspect of the shape information of the object that can be extracted from a set of silhouette images. Despite being an important property, the 2^{nd} FPVH is often overlooked by researchers who usually focus on the 1^{st} FPVH (Section 2.2.3) when using SFS for shape estimation. In the next chapter, we will see how the 2^{nd} FPVH can be combined with stereo to locate points on the surface of the object.

4.3 Related Work

In their image-based Visual Hull rendering work [BMMG99, MBR+00, Mat01], Matusik et al. proposed a ray-casting algorithm to render objects using silhouette images. Their way of intersecting the casting rays with the silhouette images is similar to the way our Bound-

ing Edges are constructed. However, there are two fundamental differences between their approach and the definition of Bounding Edge. First, our Bounding Edges are originated only from points on the boundary of the silhouette image while their casting rays can originate from anywhere, including any point inside the silhouette. Second, their casting rays do not embed the important 2^{nd} FPVH as Bounding Edges do. In a separate paper [BMM01], Matusik et al. also proposed a fast way to build polyhedral Visual Hulls. They based their idea on visual cone intersection but simplified the representation and computation by approximating the actual silhouette as polygons (i.e. any curved part of the silhouette is approximated by straight lines) which is equivalent to approximating the 3D object as polyhedral shape. Due to this approximation, their results are not the exact surface-based representation discussed in Section 2.3.1 except for true polyhedral objects. Nevertheless their idea of calculating silhouette edge bins can be applied to speed up the construction of Bounding Edges. Lazebnik et al. [LBP01] independently proposed a new way of representing Visual Hulls. The edge of the "Visual Hull mesh" in their work is theoretically equivalent to the definition of a Bounding Edge. However, they compute their edges after locating frontier and triple points whereas we compute Bounding Edges directly from the silhouette images.

4.4 Discussion

To compare the merits of using the three Visual Hull representations, we define two important attributes: exactness and completeness. A VH representation is said to be exact if the geometric entities of the representation actually lie on the exact boundary of the Visual Hull. The representation is said to be complete if the geometric entities give us a completely *closed* boundary of the Visual Hull. An exact representation is not necessarily complete and vice versa.

Inherently the surface representation is an exact and complete representation because

the surface patches form an exact and closed boundary of the Visual Hull. On the other hand, the voxel representation is inexact but complete as the collection of all surface voxels forms a closed boundary to the Visual Hull, although they are not guaranteed to lie on the exact boundary of the Visual Hull. By the way they are constructed, Bounding Edges all lie on the exact boundary of the Visual Hull and therefore they are exact. However, since the boundary of the silhouette is only sampled at a finite collection of points, the Bounding Edge representation does not form a completely closed boundary. In other words, Bounding Edges are exact but incomplete representation of a 3D Visual Hull. Asymptotically if we sample the silhouette boundary infinitely, the Bounding Edge representation will be complete and equivalent to the surface representation. Similarly the voxel representation will be exact and equal to the volume bounded by the surface representation if the voxel size is infinitely small.

There are pros and cons of using each Visual Hull representation under different situations. In terms of low complexity, the exact surface representation is the least desirable representation as it is computationally the most expensive. For real-time Visual Hull construction, the voxel representation outperforms the other two representations due to the fast voxel-based VH construction methods. In applications such as Visual Hull alignment (to be discussed in the next chapter) where accurate shape information of the object is needed (i.e. exactness is more important than completeness), the Bounding Edge representation is preferred because it is exact, computationally less expensive than the surface representation and its definition embeds the 2^{nd} FPVH naturally. For applications such as obstacle avoidance or object re-rendering from a different viewpoint (see the image-based motion rendering application in Chapter 9 as an example), the Bounding Edge representation is not suitable as it is not complete. As a summary, Table 4.1 lists some of the properties of the three Visual Hull representations : surface patch, discrete voxel and Bounding Edge.

Properties	**Surface Patches**	**Discrete Voxel**	**Bounding Edge**
Basic Geometric Entities	2D Surface	3D Volume	1D Line Segment
Completeness	complete	complete	incomplete
Exactness	exact	inexact	exact
Computational Complexity	high (intersect 3D planes and surfaces)	low (test overlap of 2D points with 2D silhouettes)	moderate (intersect 2D lines with 2D silhouettes)
natural relationship with the 1^{st} FPVH	yes	yes	no
natural relationship with the 2^{nd} FPVH	yes	no	yes
Applications		Real-time shape estimation [CKBH00]	Visual Hull Alignment and Refinement [CBK03]

Table 4.1: Table comparing the surface patch, discrete voxel and Bounding Edge representations of Visual Hulls.

Chapter 5

Visual Hulls Across Time: Rigid Objects

Shape estimation by SFS can be very coarse when there are only a few silhouette images, especially for complex-shaped objects such as the dinosaur and bananas example in Figure 4.3. To illustrate the coarseness of the voxel model (which was built using only six silhouette images) in Figure 4.3(d), we colored the voxels by re-projecting their centers onto the color images. The model is re-displayed in Figure 5.1(b) with colors. Although this model approximates the shape of the objects well, a lot of the details, such as the legs and the horns of the dinosaur are missing.

Better shape estimates can be obtained using SFS if the number of distinct silhouette images is increased. For example, the model shown in Figure 5.1(c) is constructed using 36 silhouette images and that in Figure 5.1(d) is constructed using 66 silhouette images. It can be seen that the models built using 36 and 66 silhouette images are much better than the one built using only 6 silhouettes in Figure 5.1(b).

The number of distinct silhouettes can be increased in one of two ways: across space or across time. By across space, we mean increasing the number of cameras used. The across space approach, though simple and straightforward, may not be feasible in many practical situations due to financial (buying more cameras) or physical (system setup and camera calibration) limitations. In this chapter we propose another way of increasing the

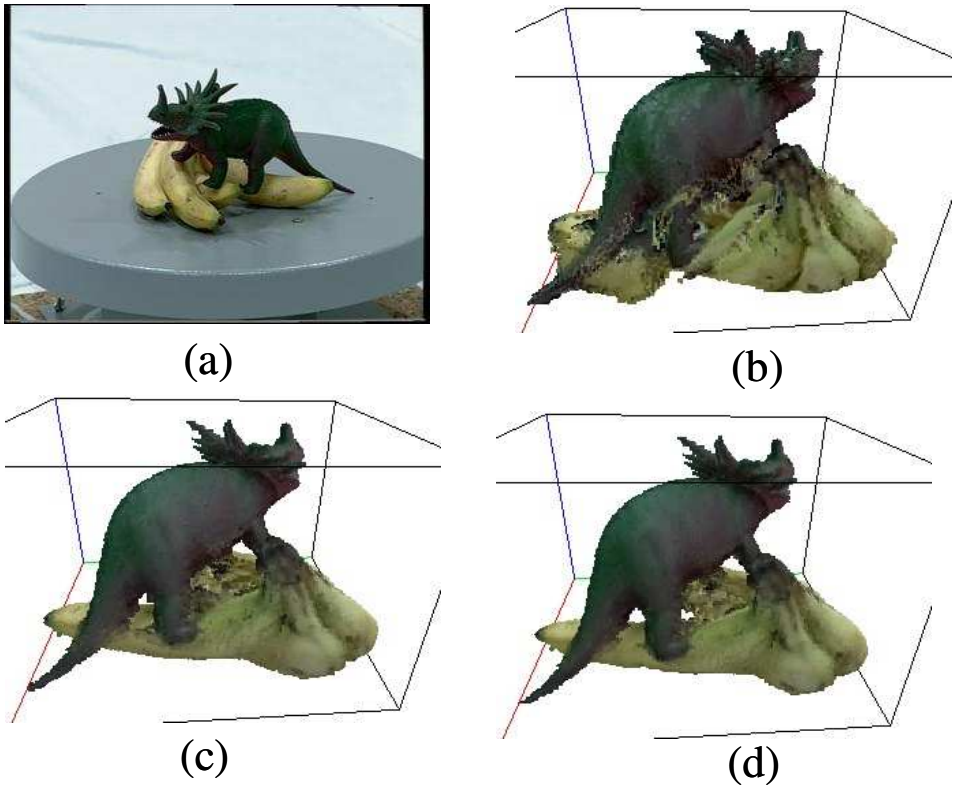


Figure 5.1: (a) An image of the toy dinosaur and bananas. (b) The 3D colored Visual Hull voxel model reconstructed using six silhouette images of the dinosaur/bananas. Some shape details such as the legs and the horns of the dinosaur are missing in this model. (c) Voxel model reconstructed using 36 silhouette images. Much better shape estimation is obtained. (d) Voxel model reconstructed using 66 silhouette images. An even better shape estimate is obtained.

number of silhouette images by capturing a number of silhouettes from each camera as the object moves across time and then using all the silhouette images (after compensating for the motion of the object) for reconstruction. For example, for a system with K cameras and J frames of images, the effective number of cameras would be increased to JK. This is equivalent to adding an additional $(J-1)K$ physical cameras to the system.

There are two tasks to constructing Visual Hulls across time: (1) estimating the motion of the object between successive time instants and (2) combining the silhouette images at different time instants to get a refined shape of the object. In this chapter, we assume the object of interest is rigid, but the motion of the object between frames is totally arbitrary and unknown. We refer to the task of computing the rigid transformation as Visual Hull

Alignment and the task of combining the silhouette images at different times as Visual Hull Refinement. Both tasks will be discussed in details in the rest of this chapter. While we focus on rigid objects here, in the next chapter we will extend the same alignment and refinement idea to articulated objects.

5.1 Visual Hull Alignment

To combine silhouette images across time, the motion of the object between frames is required. For static objects, the problem may be simplified by putting the object on a *precisely calibrated* turn-table so that the motion is known in advance [Sze93]. However for dynamic objects whose movement we do not have control or knowledge of, we have to estimate the unknown motion before we can combine the silhouette images across time. To be more precise, we state the Visual Hull Alignment Problem as follows:

Visual Hull Alignment by Silhouette Images:
Suppose we are given two sets of consistent silhouette images $\{S_j^k;\ \ k = 1; \cdots\cdots, K;\ \ j = 1, 2\}$ of a rigid object O from K cameras at two different time instants $t_1 and t_2$. Denote the Visual Hulls for these silhouette sets by $H_j, j = 1, 2$. Without loss of generality, assume the first set of images $\{S_1^k\}$ are taken when the object is at position and orientation of $(\boldsymbol{I}, \underline{0})$ while the second image set $\{S_2^k\}$ is taken when the object is at $(\boldsymbol{R}, \boldsymbol{t})$. The problem of Visual Hull alignment is to find $(\boldsymbol{R}, \boldsymbol{t})$ such that there exists an object O which exactly explains the silhouettes at both times t_j and the relative position and orientation of O is related by $(\boldsymbol{R}, \boldsymbol{t})$ from t_1 to t_2. Moreover, we say that the two Visual Hulls H_1 and H_2 are *aligned consistently with transformation* $(\boldsymbol{R}, \boldsymbol{t})$ if and only if we can find an object O such that H_1 is the Visual Hull of O at orientation and position $(\boldsymbol{I}, \underline{0})$ and H_2 is the Visual Hull of O at orientation and position $(\boldsymbol{R}, \boldsymbol{t})$.

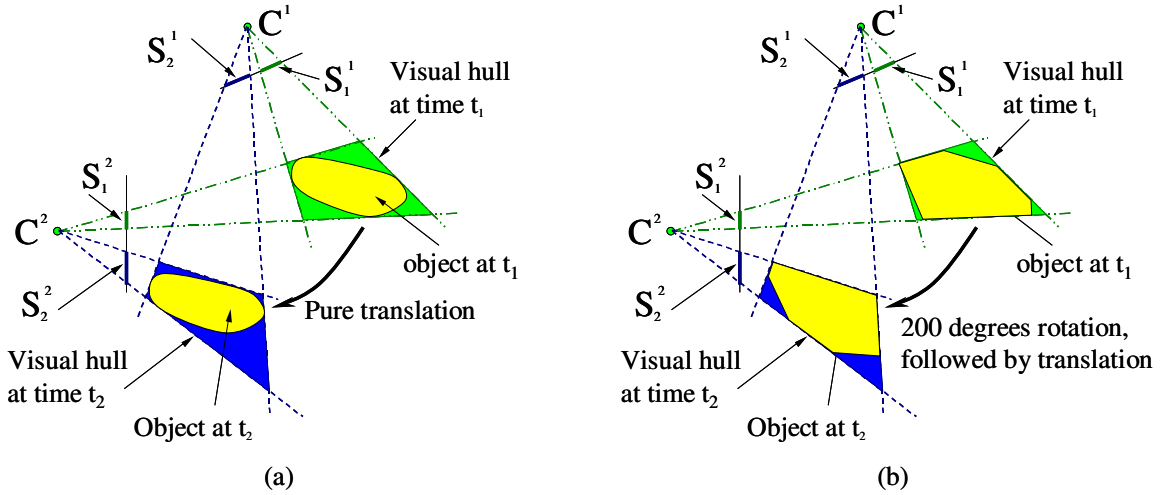


Figure 5.2: A 2D example showing the ambiguity issue of aligning Visual Hulls. Both cases in (a) and (b) have the same silhouette image sets at times t_1 and t_2 but they are formed from two different objects with different motion.

Since it is assumed that the two sets of silhouette images are consistent and come from the same object, there always exists at least one set (the true solution) of object O and motion $(\boldsymbol{R}, \boldsymbol{t})$ that exactly explains both sets of silhouette images. Now we study whether aligning two Visual Hulls can be solved uniquely or not.

5.2 VH Alignment Ambiguity And Geometrical Constraints

Aligning two Visual Hulls using silhouette images alone is inherently ambiguous. This means that in general the solution is not unique and there exists more than one set of $(\boldsymbol{R}, \boldsymbol{t})$ which satisfies the alignment criterion. A 2D example is shown in Figure 5.2. In the figure, both (a) and (b) have the same silhouette image sets (and hence the same Visual Hulls) at times t_1 and t_2. However, in (a), the silhouettes are formed by a curved object with a pure translation between t_1 and t_2, while in (b), the silhouettes are created by a polygonal object with both a rotation (200 degrees) and a translation between t_1 and t_2.

The motion ambiguity in Visual Hull alignment is a direct result of the indeterminacy in the shape of the object. Although the alignment solution is not unique, there are constraints

on the motion and the shape of the object for a consistent alignment. We first discuss the geometrical constraints for aligning two 2D Visual Hulls and later extend them to 3D.

5.2.1 Geometric Constraints for Aligning 2D Visual Hulls

To establish the constraints for aligning two 2D Visual Hulls, we begin with the following two lemmas which express some fundamental properties of 2D Visual Hulls.

Lemma 5.1:

For a *closed* and *connected* 2D object, its Visual Hull from K silhouette images is a *convex* polygon with at most $2K$ Bounding Edges. Conversely, any convex polygon with $M \geq 4$ edges can be thought of as a Visual Hull formed from K silhouettes of some closed and connected 2D object where $K = \lceil \frac{M}{2} \rceil$.

Proof: See Appendix B.1 ■

Lemma 5.2:

Each edge of the 2D polygonal Visual Hull H of an object O has to touch the object O at at least one point. Conversely any closed and connected 2D object O which satisfies the two conditions: (1) $O \subseteq H$, and (2) O touches each edge of H at at least one point, is an object which forms the silhouettes of H.

Proof: See Appendix B.2 ■

Lemma 5.1 establishes the fact that the Visual Hull of any closed and connected 2D object must be a convex polygon. Lemma 5.2 is essentially the 2D version of the 2^{nd} FPVH. Now if we let E_j^i be the edges of H_j, $T_{(\boldsymbol{R},\boldsymbol{t})}(A)$ be the entity after applying transformation of $(\boldsymbol{R}, \boldsymbol{t})$ to A and $T^{-1}_{(\boldsymbol{R},\boldsymbol{t})}()$ denotes the inverse transformation, the geometric constraints for aligning two 2D Visual Hulls are expressed in Lemma 5.3 as follows:

Lemma 5.3:

Given two 2D Visual Hulls H_1 and H_2, the necessary and sufficient condition for them to be

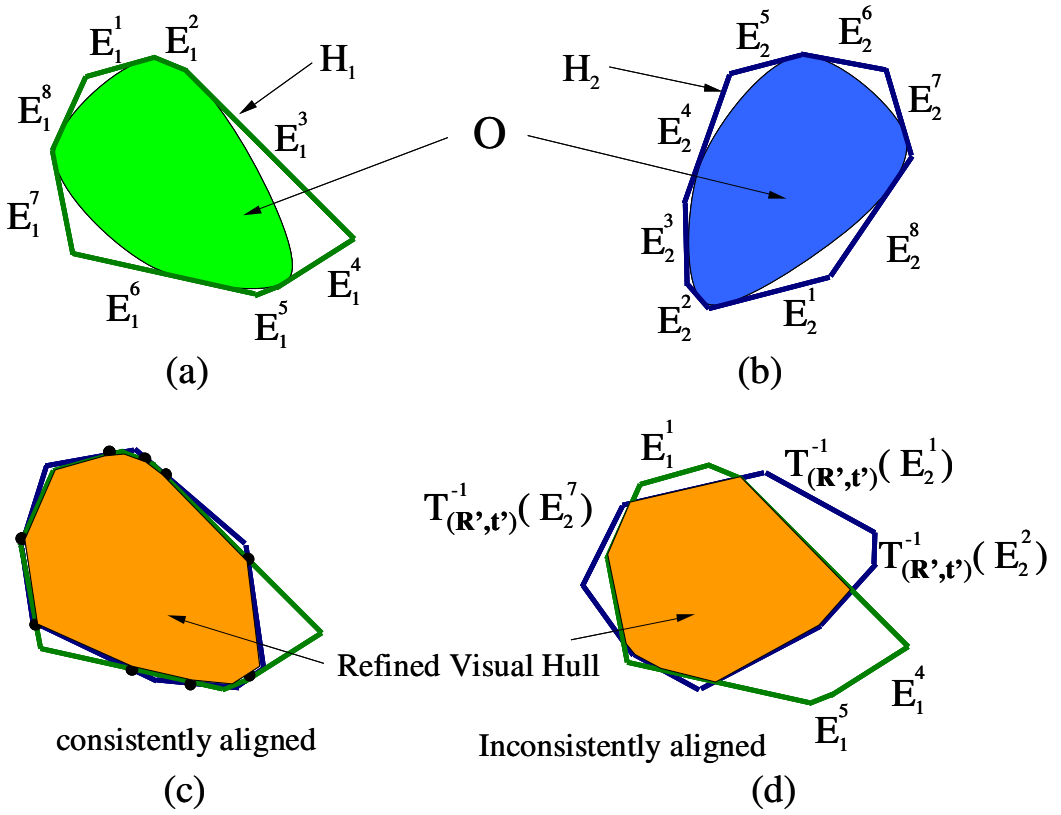


Figure 5.3: (a)(b) Two Visual Hulls of the same object at different positions and orientations. (c) All edges satisfy Lemma 5.3 when the alignment $(\boldsymbol{R}, \boldsymbol{t})$ is consistent, (d) edges E_1^1, E_1^4, E_1^5, $T^{-1}_{(\boldsymbol{R}',\boldsymbol{t}')}(E_2^1)$, $T^{-1}_{(\boldsymbol{R}',\boldsymbol{t}')}(E_2^2)$, $T^{-1}_{(\boldsymbol{R}',\boldsymbol{t}')}(E_2^7)$ all violate Lemma 5.3 when the Visual Hulls are not aligned consistently.

aligned consistently with transformation $(\boldsymbol{R}, \boldsymbol{t})$ is given as follows : No edge of $T_{(\boldsymbol{R},\boldsymbol{t})}(H_1)$ lies completely outside H_2 and no edge of H_2 lies completely outside $T_{(\boldsymbol{R},\boldsymbol{t})}(H_1)$.

Proof: See Appendix B.3 ■

Figure 5.3(a)(b) shows examples of two 2D Visual Hulls of the same object. In (c), the alignment is consistent and all edges from both Visual Hulls satisfy Lemma 5.3. In (d), the alignment is inconsistent and the edges E_1^1, E_1^4, E_1^5, $T^{-1}_{(\boldsymbol{R}',\boldsymbol{t}')}(E_2^1)$, $T^{-1}_{(\boldsymbol{R}',\boldsymbol{t}')}(E_2^2)$, $T^{-1}_{(\boldsymbol{R}',\boldsymbol{t}')}(E_2^7)$ all violate Lemma 5.3. Lemma 5.3 provides a good way to test if an alignment is consistent or not, at least in 2D.

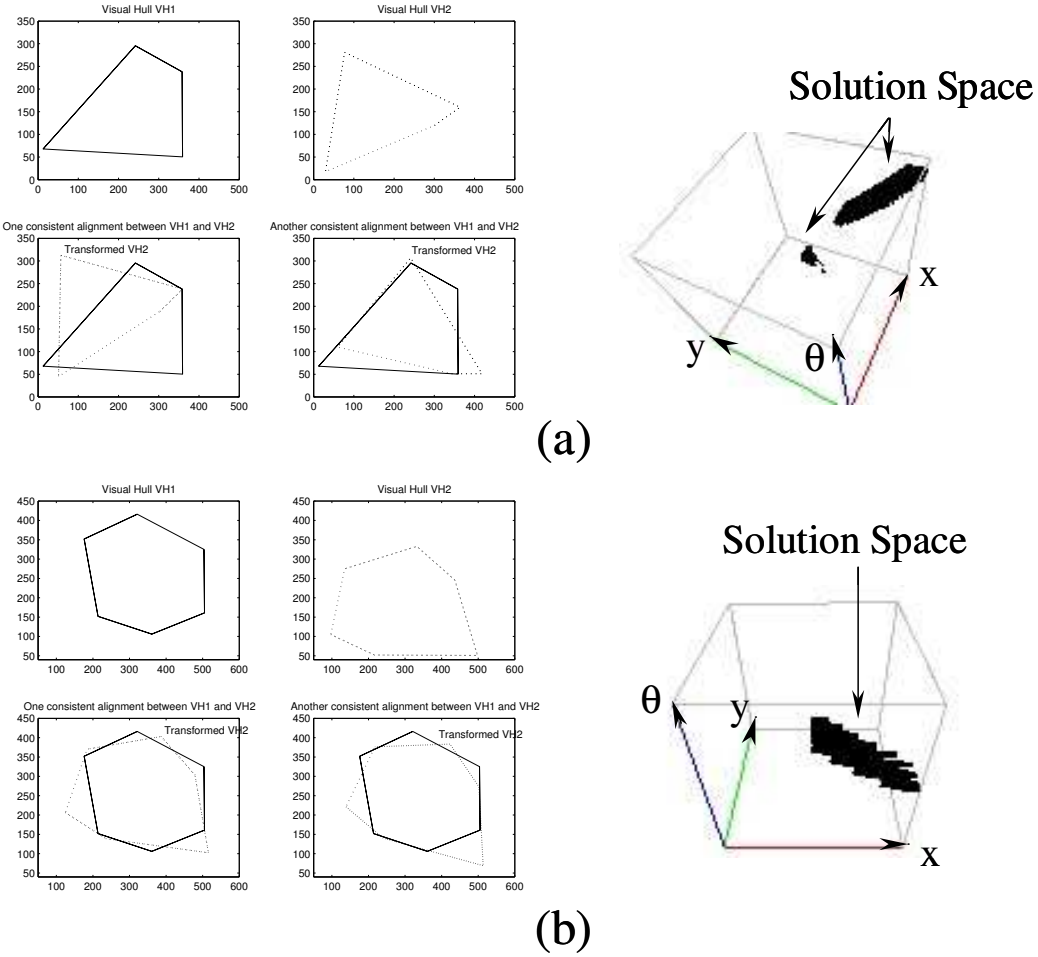


Figure 5.4: (a) An example of two synthetic 2D Visual Hulls (each with four edges) and the space of consistent alignments. (b) An example of two synthetic 2D Visual Hulls (each with six edges) and the solution space of consistent alignments.

To illustrate how these constraints can be used in practice, two synthetic 2D Visual Hulls (polygons) each with four edges were generated and Lemma 5.3 was used to search for the space of all consistent alignments. In 2D there are only three degrees of freedom (two in translation and one in rotation). The space of consistent alignments are shown in Figure 5.4(a). As can be seen, there are two unconnected subsets of the solution space, clustered around different rotation angles. Another set of synthetic 2D Visual Hulls with 6 edges are also shown in Figure 5.4(b). In this example, there is only one connected subset of solutions.

Now consider a variant of Lemma 5.3 which we called Lemma 5.4 as follows:

Lemma 5.4:

$(\boldsymbol{R}, \boldsymbol{t})$ is a consistent alignment of two 2D Visual Hulls H_1 and H_2, constructed from silhouette sets $\{S_j^k\}$; $j = 1, 2$ if and only if the following condition is satisfied : for each edge E_1^i of $T_{(\boldsymbol{R},\boldsymbol{t})}(H_1)$, there exists at least one point P on E_1^i such that the projection of P onto the k^{th} image lies inside or on the boundary of the silhouette S_2^k for all $k = 1, \cdots\cdots, K$.

Proof: See Appendix B.4 ■

Lemma 5.4 expresses the constraints in terms of the silhouette images rather than the Visual Hull. For 2D objects, there is no significant difference between using Lemma 5.3 or Lemma 5.4 to specify the alignment constraints because all 2D Visual Hulls can be represented easily by a polygon with finite number of edges. For 3D objects, however, the 3D version of Lemma 5.3 is not very practical because it is difficult to represent a 3D Visual Hull exactly and completely. By expressing the geometrical constraints in terms of the silhouette images (Lemma 5.4) instead of the Visual Hull itself (Lemma 5.3), the need for an exact and complete Visual Hull representation can be avoided. In the next section, we extend Lemma 5.4 to 3D and establish the constraints for aligning two 3D Visual Hulls using only the silhouette images.

5.2.2 Geometric Constraints for Aligning 3D Visual Hulls

The geometric constraints for aligning two 3D Visual Hulls are expressed in terms of Lemma 5.5 below:

Lemma 5.5:

For two *convex* 3D Visual Hulls H_1 and H_2 constructed from silhouette sets $\{S_j^k\}$; $j = 1, 2$, the necessary and sufficient condition for a transformation $(\boldsymbol{R}, \boldsymbol{t})$ to be a consistent alignment between H_1 and H_2 is as follows: for any Bounding Edge E_1^i (defined by (4.2) in Chapter 4) constructed from the silhouette image set $\{S_1^k\}$, there exists at least one point

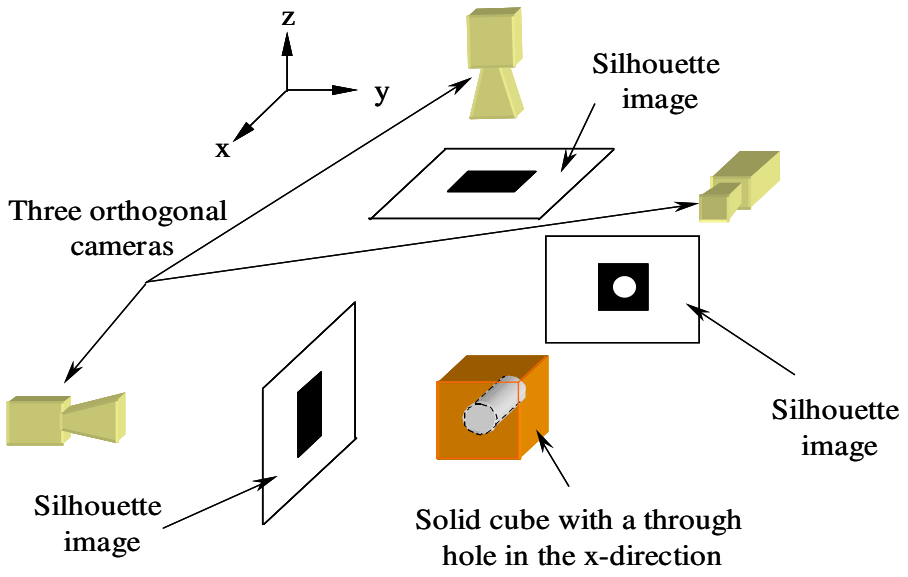


Figure 5.5: An example scenario of a solid cube with a through hole in the x-direction. The sufficient part of Lemma 5.5 is not valid in this example.

P on E_1^i such that the projection of the point $T_{(\boldsymbol{R},\boldsymbol{t})}(P)$ onto the k^{th} image lies inside or on the silhouette S_2^k for all $k = 1, \cdots\cdots, K$. Similarly, for any Bounding Edge E_2^i constructed from $\{S_2^k\}$, there exists at least one point P on E_2^i such that the projection of the point $T^{-1}_{(\boldsymbol{R},\boldsymbol{t})}(P)$ on the k^{th} image lies inside or on the silhouette S_1^k.

Proof: See Appendix B.5 ■

Note that the condition in Lemma 5.5 is still necessary but not sufficient if either one or both of the two Visual Hulls are non-convex. Figure 5.5 shows a counter example of a cube with a hole through the middle of the cube in the x-direction and three cameras placed in an orthogonal setup around the cube. Consider two sets of silhouettes which are captured at t_1 and t_2 with the cube at exactly the same position and orientation. The transformation $(\boldsymbol{I}, \underline{0})$ is obviously a consistent alignment between t_1 and t_2 because it is the true solution and hence the conditions in Lemma 5.5 are satisfied for the identity transformation (the necessary part of the Lemma). Now consider the transformation of rotating the cube around the z-axis for 90 degrees without any translation. This is not a consistent alignment because the

silhouettes at t_1 and t_2 from the y-direction camera do not match under this transformation. However, the conditions in Lemma 5.5 are still satisfied with this 90 degree rotation about z-axis transformation. Note that this transformation will be a consistent alignment if there is no hole in the cube.

For general 3D objects, Lemma 5.5 is useful to reject inconsistent alignments between two Visual Hulls but cannot be used to prove if an alignment is consistent. Theoretically we can prove if an alignment is consistent as follows. First transform the Visual Hulls using the alignment transformation and compute the intersection of the two Visual Hulls. The resultant Visual Hull is then rendered with respect to all the cameras at both times and compared with the two original sets of silhouette images. If the new Visual Hull exactly explains all the original silhouette images, then the alignment is consistent. In practice, however, this idea is computationally very expensive and is inappropriate as an algorithm to compute the correct alignment between two 3D Visual Hulls. In Section 5.3.3, we will show how the hard geometric constraints stated in Lemma 5.5 can be approximated by soft constraints and combined with photometric consistency to align 3D Visual Hulls.

5.3 Resolving the Alignment Ambiguity

Since aligning Visual Hulls using silhouette images alone is ambiguous, additional information is required in order to find the correct alignment. In this section we show how to resolve the alignment ambiguity using color information. First we combine the 2^{nd} FPVH (introduced in Chapter 4) with stereo to extract a set of 3D points (which we call Colored Surface Points) on the surface of the object at each time instant. Then the two sets of 3D Colored Surface Points are used to align the Visual Hulls through the 2D color images. The idea is discussed in details subsequently. We assume that besides the set of silhouette images $\{S_j^k\}$, the set of original color images (which the silhouette images were derived from) are also given and represented by $\{I_j^k\}$.

5.3.1 Colored Surface Points (CSPs)

Although the Second Fundamental Property of Visual Hull tells us that each Bounding Edge touches the object at at least one point, it does not provide a way to find this touching point. Here we propose a simple (one-dimensional) search based on the stereo principle to locate this touching point as follows.

If we assume the object is Lambertian and all the cameras are color balanced, then any point on the surface of the object should have the same projected color in all of the color images. In other words, for any point on the surface of the object, its projected color variance across the *visible* cameras should be zero. Hence on a Bounding Edge, the point which touches the object should have zero projected color variance. This property provides a good criterion for locating the touching points. Hereafter we call these touching points the *Colored Surface Points (CSP)* of the object.

To express the idea mathematically, consider a Bounding Edge E_j^i from the j^{th} Visual Hull. Since we denoted the Bounding Edge E_j^i by a set of *ordered* 3D vertex pairs $\left\{\left(SV_j^i(m), FV_j^i(m)\right)\right\}$ (Equation (4.2)), we can parameterize a point $W_j^i(m,w)$ on E_j^i by two parameters m and w, where $m \in \{1, \cdots\cdots, M_j^i\}$ and $0 \leq w \leq 1$ with

$$W_j^i(m,w) = SV_j^i(m) + w * \left(FV_j^i(m) - SV_j^i(m)\right) . \tag{5.1}$$

Let $\boldsymbol{c}_j^k(P)$ be the function which returns the projected color of a 3D point P on the k^{th} color image at time t_j. The projected color mean $\mu_j^i(m,w)$ and variance $\sigma_j^i(m,w)$ of the point $W_j^i(m,w)$ are given as

$$\begin{aligned} \mu_j^i(m,w) &= \frac{1}{n_j^i}\sum_k \boldsymbol{c}_j^k(W_j^i(m,w)); \\ \sigma_j^i(m,w) &= \frac{1}{n_j^i}\sum_k [\boldsymbol{c}_j^k(W_j^i(m,w)) - \mu_j^i(m,w)]^2 . \end{aligned} \tag{5.2}$$

The projected color $\boldsymbol{c}_j^k(W_j^i(m,w))$ from camera k is used in calculating the mean and variance *only if* $W_j^i(m,w)$ is *visible* in that camera and n_j^i denotes the number of the visible

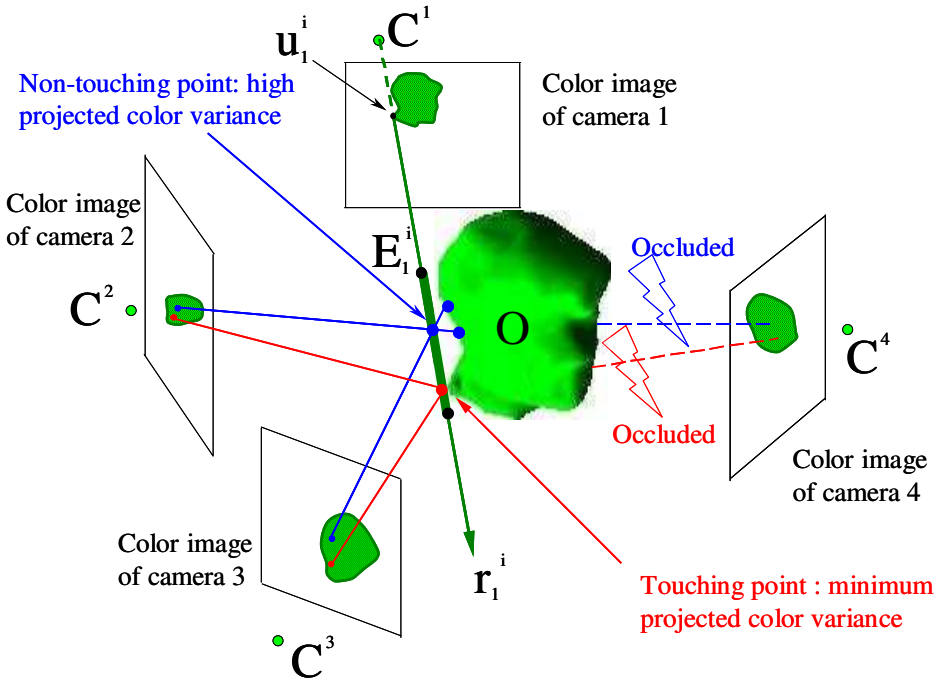


Figure 5.6: Locating the touching point (Colored Surface Point) by searching along the Bounding Edge for the point with the minimum projected color variance.

cameras for point W_j^i. The question of how to conservatively determine the visibility of a 3D point with respect to a camera using only the silhouette images will be addressed shortly. Figure 5.6 illustrates the idea of locating the touching point by searching along the Bounding Edge.

In practice, due to noise and inaccuracies in color balancing, instead of searching for the point which has zero projected color variance, we locate the point with the minimum variance. In other words, we set the Colored Surface Point of the object on E_j^i to be $W_j^i(\tilde{m}, \tilde{w})$ where $\tilde{m}$ and $\tilde{w}$ minimizes $\sigma_j^i(m, w)$ for $0 \leq w \leq 1; \quad m \in \{1, \cdots, \cdots, M_j^i\}$. Note that by choosing the point with the minimum variance, the problem of tweaking parameters or thresholds of any kind is avoided. The need to adjust parameters or thresholds is always a problem in other shape reconstruction methods such as space carving [KS00] or multi-baseline stereo [OK93]. Space carving relies heavily on a color variance threshold to

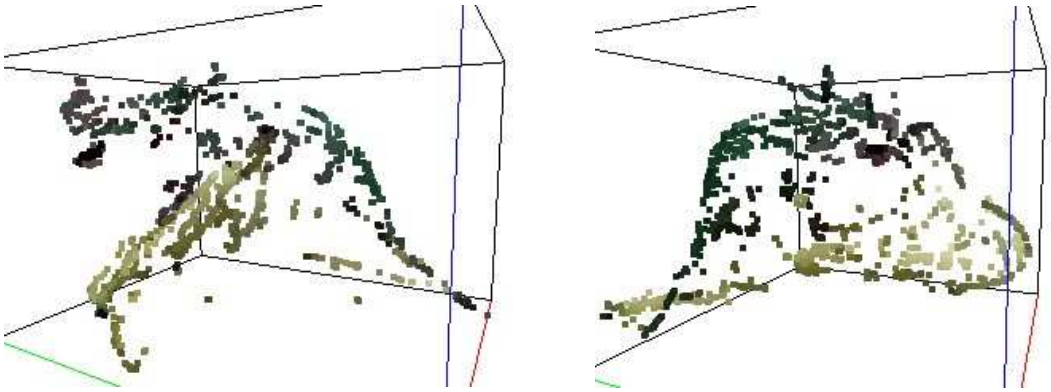


Figure 5.7: Two sets of CSPs of the dinosaur/bananas dataset (see Figure 5.1) obtained at two time instants with different positions and orientations (the points are drawn as small cubes for better display). Note that the CSPs are sparsely sampled and there is no point-to-point correspondence between the two sets of CSPs.

remove non-object voxels and stereo matching results are sensitive to the search window size. In our case, knowing that each Bounding Edge touches the object at at least one point (2^{nd} FPVH) is the key piece of information that allows us to avoid any thresholds. In fact locating CSPs is a special case of the problem of matching points on pairs of epipolar lines as discussed in [SG98, IHA02]. In [SG98] and [IHA02], points are matched on "general" epipolar lines on which there may or may not be a matching point so a threshold and an independent decision is needed for each point. To locate CSPs, points are matched on "special" epipolar lines which guarantee to have at least one matching point so no threshold is required. Hereafter, for simplicity we drop the notation dependence of m, w, $\tilde{}$ and denote (with a slight abuse of notation) the CSPs $W_j^i(\tilde{m}, \tilde{w})$ by W_j^i and its color $\mu_j^i(\tilde{m}, \tilde{w})$ by μ_j^i.

5.3.2 Alignment by Color Consistency

Suppose we have located two sets of Colored Surface Points at two different time instants t_1 and t_2. For example, Figure 5.7 shows two sets of CSPs for the dinosaur/bananas (see Figure 5.1) obtained at two time instants at two different positions and orientations. Since

the sets of CSPs lie on their corresponding (rigid) Visual Hulls H_1 and H_2, the problem of aligning H_1 and H_2 is equivalent to aligning the two sets of CSPs. The question now is how can we align the two sets of CSPs. Before answering this question, we have to point out two very important facts about CSPs. First of all, the CSPs at each time instant are points on the occluding contours. This means that CSPs are only sparsely sampled points on the surface of the object (as opposed to the 3D data points acquired from laser range devices [CYB, CTI, TTI] which produce densely sampled surface points on the object). The point sparsity prohibits us from using well established 3D point alignment methods such as the Iterative Closet Point (ICP) method [BM92, Zha94, RL01]. Secondly the only property common of the two sets of CSPs is that they all lie on the surface of the object. There is *no* point-to-point correspondence between any two sets of CSPs obtained at different time instant. Because of this, alignment methods which are used in the structure-from-motion literature [TK92, PK92, QK96] cannot be used to align the CSPs.

To solve the CSP alignment problem, we use an idea similar to that used to solve the 2D image registration problem in [Sze94]. In our case, instead of registering a 2D image with another 2D image, we align 2D images ($\{I_2^k\}$) at time t_2 with a "3D image" (the Colored Surface Points $\{W_1^i\}$) at time t_1 through the projection functions $\{\Pi^k\}$. The error measure used is the sum of color differences between the Colored Surface Points at time t_1 and their projected colors from the color images at time t_2 and vice versa. Mathematically, let $\{I_j^k, S_j^k, W_j^i, \mu_j^i; \quad i = 1, \cdots\cdots, L_j; \quad k = 1, \cdots\cdots, K; \quad j = 1, 2\}$ be the two sets of data. To find the most color consistent alignment $(\boldsymbol{R}, \boldsymbol{t})$, consider the color error function

$$e = \sum_{i=1}^{L_2} e_{1,2}^i + \sum_{i=1}^{L_1} e_{2,1}^i \,, \tag{5.3}$$

$$e_{1,2}^i = \sum_k e_{1,2}^{i,k} = \sum_k [\boldsymbol{c}_1^k(\boldsymbol{R}^T W_2^i - \boldsymbol{R}^T \boldsymbol{t}) - \mu_2^i]^2 \,, \tag{5.4}$$

$$e_{2,1}^i = \sum_k e_{2,1}^{i,k} = \sum_k [\boldsymbol{c}_2^k(\boldsymbol{R} W_1^i + \boldsymbol{t}) - \mu_1^i]^2 \,, \tag{5.5}$$

where $e_{2,1}^{i,k}$ represents the difference between the mean color μ_1^i of the Colored Surface Point W_1^i at time t_1 and its projected color $\boldsymbol{c}_2^k(\boldsymbol{R}W_1^i + \boldsymbol{t})$ in camera k at time t_2. Note that at time t_2, the new position of W_1^i due to the motion of the object is $\boldsymbol{R}W_1^i + \boldsymbol{t}$. Likewise, $e_{1,2}^{i,k}$ is the difference between the mean color μ_2^i of W_2^i and its projected color $\boldsymbol{c}_1^k(\boldsymbol{R}^T W_2^i - \boldsymbol{R}^T\boldsymbol{t})$ in camera k at time t_1. From now on, we refer to the error of aligning 3D points with the 2D images forward in time (e.g. 3D points at t_1 and 2D images at t_2) as the forward error. Similarly the error of aligning 3D points with the 2D images backward in time (e.g. 3D points at t_2 and 2D images at t_1) is referred to as the backward error. In the current example, $e_{2,1}^i$ is the forward error while $e_{1,2}^i$ is the backward error. Just as when locating the CSPs on the Bounding Edge in Equation (5.2), the summations in equations (5.4) and (5.5) include the projected color of camera k *only if* the point of interest is visible in that camera. The process of Visual Hull alignment by color consistency is illustrated in Figure 5.8.

If we parameterize $\boldsymbol{R}$ and $\boldsymbol{t}$ as

$$\boldsymbol{\Phi} = [\Phi_1, \Phi_2, \Phi_3, \Phi_4, \Phi_5, \Phi_6]^T \tag{5.6}$$

where Φ_1, Φ_2, Φ_3 are the Euler's angles of $\boldsymbol{R}$ and Φ_4, Φ_5, Φ_6 are the x, y, z components of $\boldsymbol{t}$, the minimization of (5.4) can be solved by an iterative method similar to the Levenberg-Marquardt algorithm [DS83, PTVF93]:

1. With an initial estimate $\hat{\boldsymbol{\Phi}}$, calculate the Hessian matrix $\boldsymbol{H} = \{h_{mn}\}$ and the difference vector $\boldsymbol{d} = \{d_m\}$ with $m, n = 1, \cdots\cdots, 6$ as

$$h_{mn} = \sum_{i=1}^{L_2}\sum_k \frac{\partial e_{1,2}^{i,k}}{\partial[\boldsymbol{\Phi}]_m}\frac{\partial e_{1,2}^{i,k}}{\partial[\boldsymbol{\Phi}]_n} + \sum_{i=1}^{L_1}\sum_k \frac{\partial e_{2,1}^{i,k}}{\partial[\boldsymbol{\Phi}]_m}\frac{\partial e_{2,1}^{i,k}}{\partial[\boldsymbol{\Phi}]_n}\,, \tag{5.7}$$

$$d_m = -2[\sum_{i=1}^{L_2}\sum_k e_{1,2}^{i,k}\frac{\partial e_{1,2}^{i,k}}{\partial[\boldsymbol{\Phi}]_m} + \sum_{i=1}^{L_1}\sum_k e_{2,1}^{i,k}\frac{\partial e_{2,1}^{i,k}}{\partial[\boldsymbol{\Phi}]_m}]\,. \tag{5.8}$$

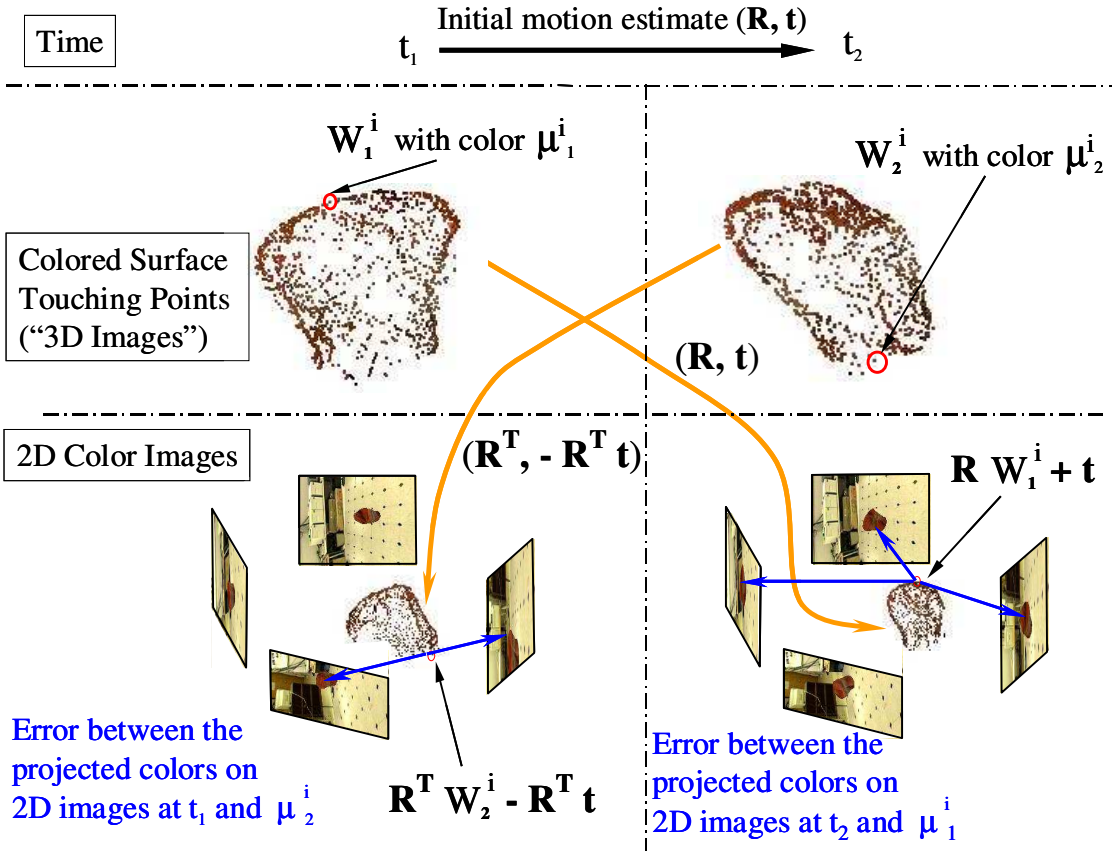


Figure 5.8: Visual Hull Alignment using color consistency. The error between the colors of the 3D surface points and their projected image colors is minimized.

2. Update the parameter $\hat{\mathbf{\Phi}}$ by an amount $\delta\mathbf{\Phi} = (\boldsymbol{H} + \lambda \boldsymbol{I})^{-1}\boldsymbol{d}$, where λ is a time-varying stabilization parameter.

3. Go back to 1. until the estimate of $\hat{\mathbf{\Phi}}$ converges.

To initialize the optimization parameters, we approximate the two sets of CSPs at t_1 and t_2 each by an ellipsoidal shell (using Equations (3.3), (3.4) and (3.5) in Section 3.3.2). The initial estimate of the translation vector $\boldsymbol{t}$ is then set as the relative positions of the centers of the two ellipsoids. Similarly the initial guess for the rotation matrix $\boldsymbol{R}$ is set as the relative orientation of the two ellipsoids. This simple initialization method works well

for most objects when the rotation of the object between the two time instants is less than 90 degrees.

5.3.3 Alignment by Color Consistency and Geometrical Constraints

Since the above formulation for aligning two sets of CSPs is inspired by the 2D image registration problem [Sze94], the error measure (Equation (5.3)) is based solely on color consistency (stereo). Though simple, this formulation does not take into account an important fact: the CSPs lie on the surface of Visual Hulls whose alignment is governed by the geometric constraints stated in Lemma 5.5. Here we show how the hard constraints of Lemma 5.5 can be converted into soft constraints and combined with color consistency to align the CSPs.

Recall Lemma 5.5 states that if $(\boldsymbol{R}, \boldsymbol{t})$ is a consistent alignment, then for any Bounding Edge E_1^i, there exists at least one point P on E_1^i such that the projection of the transformed point $\boldsymbol{R}P + \boldsymbol{t}$ lies inside or on the boundary of all the silhouette images $\{S_2^k\}$ at time t_2 and vice versa. In fact P is the point where the object touches the Bounding Edge, which we have extracted as a CSP. Hence the constraint is equivalent to saying that all transformed CSP points at time t_1 must lie inside or on the boundary of the silhouette images $\{S_2^k\}$ and vice versa. In practice, due to noises and calibration errors, instead of applying this hard constraint directly to the optimization procedures (Equations (5.6) to (5.8)), we incorporate it as a soft constraint by minimizing the distance between the projected CSP and the silhouettes as explained below.

Assume we have the same sets of data $\{I_j^k, S_j^k, W_j^i, \mu_j^i; \quad i = 1, \cdots\cdots, L_j; \quad k = 1, \cdots\cdots, K; \quad j = 1, 2\}$ as before. Let $(\boldsymbol{R}, \boldsymbol{t})$ be an estimate of the rigid transformation. Consider first the calculation of the forward error. For a CSP W_1^i (with color μ_1^i) at time t_1, its 3D position at time t_2 would be $\boldsymbol{R}W_1^i + \boldsymbol{t}$. Consider two different cases of the projection of $\boldsymbol{R}W_1^i + \boldsymbol{t}$ into the k^{th} camera:

1. The projection lies inside the silhouette S_2^k. In this case, we use the color difference $[\boldsymbol{c}_2^k(\boldsymbol{R}W_1^i + \boldsymbol{t}) - \mu_1^i]^2$ as the error measure, where as defined before, $\boldsymbol{c}_2^k(P)$ is the projected color of a 3D point P into the color image I_2^k. Here we set the color error to zero if the projection of P lies outside S_2^k. We call this error the forward photometric error.

2. The projection lies outside S_2^k. In this case, we use the distance of the projection from S_2^k, represented by $d_2^k(\boldsymbol{R}W_1^i + \boldsymbol{t})$ as an error measure. The distance is zero if the projection lies inside S_2^k. We call this error the forward geometric error.

Summing over all cameras in which W_1^i is *visible*, the forward error measure of W_1^i with respect to $(\boldsymbol{R}, \boldsymbol{t})$ is given by

$$e_{2,1}^i = \sum_k \{\tau * d_2^k(\boldsymbol{R}W_1^i + \boldsymbol{t}) + [\boldsymbol{c}_2^k(\boldsymbol{R}W_1^i + \boldsymbol{t}) - \mu_1^i]^2\}\,, \tag{5.9}$$

where τ is a weighing constant. Equation (5.9) combines the color consistency constraint (stereo) with the geometric constraint (Shape-From-Silhouette). Similarly, the backward error measure of a CSP W_2^i at time t_2 is written as the sum of the backward photometric and geometric errors:

$$e_{1,2}^i = \sum_k \{\tau * d_1^k(\boldsymbol{R}^T(W_2^i - \boldsymbol{t})) + [\boldsymbol{c}_1^k(\boldsymbol{R}^T(W_2^i - \boldsymbol{t})) - \mu_2^i]^2\}\,. \tag{5.10}$$

The problem of estimating $(\boldsymbol{R}, \boldsymbol{t})$ is now turned into the problem of minimizing the sum of the forward and backward error

$$\min_{\boldsymbol{R},\boldsymbol{t}} e = \min_{\boldsymbol{R},\boldsymbol{t}} \sum_{i=1}^{L_2} e_{1,2}^i + \sum_{i=1}^{L_1} e_{2,1}^i\,, \tag{5.11}$$

which can be solved using the same Iterative LM algorithm described in Section 5.3.2. Hereafter, we refer to this Visual Hull across time algorithm as the temporal SFS algorithm (for rigid objects) and summarize the steps as follows:

Temporal SFS Algorithm for Rigid Objects

1. Construct a list of Bounding Edges $\{E_j^i\}$ from the silhouette images $\{S_j^k\}$ at t_j where $j = 1, 2$.
2. Extract a set of Colored Surface Points $\{W_j^i, \mu_j^i\}$ at t_j from the list of Bounding Edges $\{E_j^i\}$ and the color images $\{I_j^i\}$.
3. Initialize the translation and rotation parameters by ellipsoid fitting.
4. Apply the Iterative LM algorithm (Section 5.3.2) to minimize the sum of the forward and backward errors in Equation (5.11) with respect to the (6D) motion parameters until convergence is attained or for a fixed number of iterations.

5.4 Visibility Issues

5.4.1 Determining Visibility for Locating CSPs

To locate the Colored Surface Points using Equation (5.2), the visibility of the 3D point $W_j^i(m, w)$ with respect to all K cameras is required. Here, we present a way to determine the visibilities *conservatively* using only the silhouette images. Suppose we are given a 3D point P and a set of silhouette images $\{S_j^k\}$ with camera centers $\{C^k\}$ and projection functions $\{\Pi^k()\}$. The following lemma then holds:

Lemma 5.6:
Let $\Pi^l(P)$ and $\Pi^l(C^k)$ be the projections of the point P and the k^{th} camera center C^k on the (infinite) image plane of camera l. If the 2D line segment joining $\Pi^l(P)$ and $\Pi^l(C^k)$ does not intersect the silhouette image S_j^l, *then* P is visible with respect to camera k at time t_j.

Proof: See Appendix C ■

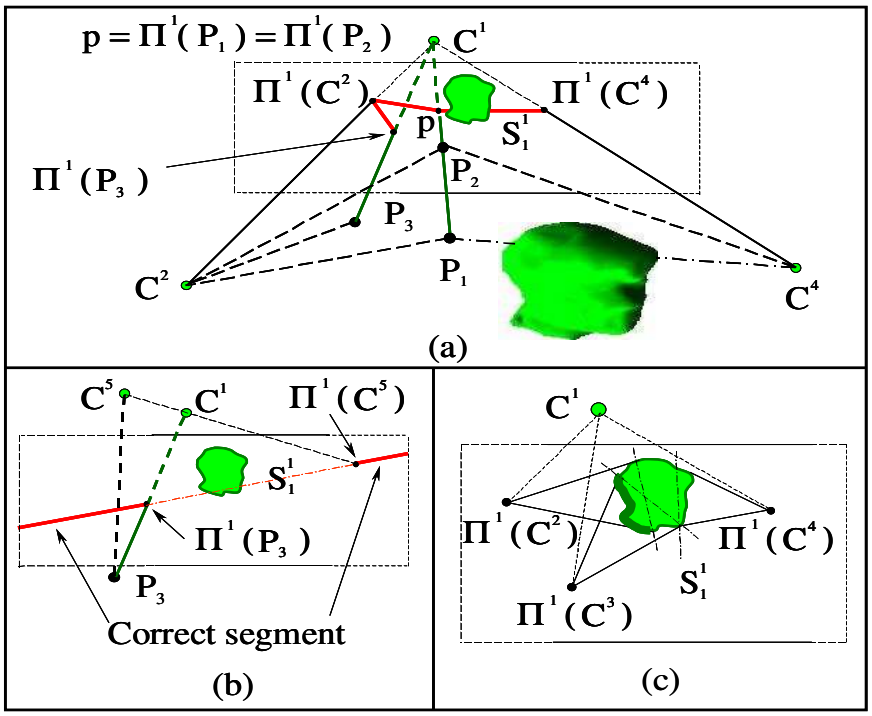


Figure 5.9: (a) Visibility of points with respect to cameras using Lemma 5.6. (b) An example where C^5 is behind C^1. The correct line to be used in Lemma 5.6 is the outer segment which passes through infinity instead of the direct segment. (c) Boundary points that can be used to construct Bounding Edges are marked by the thick boundary. These boundary points are the ones which the resulting Bounding Edges can be seen by at least two other cameras besides camera 1.

Figure 5.9(a) gives examples where the points P_1, P_2 and P_3 are visible with respect to camera 2. The converse of Lemma 5.6 is *not* necessarily true: the visibility *cannot* be determined if the segment joining $\Pi^l(P)$ and $\Pi^l(C^k)$ intersects the silhouette S^l_j. One counter example is shown in Figure 5.9(a). Both points P_1 and P_2 project to the same 2D point p on the image plane of camera 1 and the segment joining p and $\Pi^1(C^4)$ intersects with S^1_1. However, P_1 and P_2 have different visibilities with respect to camera 4 (P_2 is visible while P_1 is not). Note that special attention must be given to situations in which camera center C^k lies behind camera center C^l. In such cases, the correct line segment to be used in Lemma 5.6 is the outer line segment (passing through infinity) joining $\Pi^l(P)$ and $\Pi^l(C^k)$ rather than the direct segment. An example is given in Figure 5.9(b).

Though conservative, there are three advantages of using Lemma 5.6 to determine vis-

ibility for locating CSPs. First of all, Lemma 5.6 uses information directly from the silhouette images, avoiding the need to estimate the shape of the object for the visibility test. Secondly, recall that to construct a Bounding Edge E_j^i, we start with the boundary point u_j^i of the k^{th} silhouette. Hence all the points on E_j^i project to the same 2D point u_j^i on camera k which implies all points on the Bounding Edge E_j^i have the same set of conservative visible images. This property ensures that the color consistencies of points on the same Bounding Edge are calculated from the same set of images. Accuracy in searching the optimal point W_j^i is increased because comparisons are made fairly among points on the same Bounding Edge. Finally Lemma 5.6 also provides a guideline to sample the silhouette boundary points for constructing Bounding Edges. To have meaningful color consistencies, the number of color images used in Equation (5.2) has to be at least 2 (otherwise the projected color variance will be 0). By Lemma 5.6, boundary points u_j^i are chosen such that the resulting E_j^i is seen by at least 2 other images (excluding the image S_j^k from which the boundary point is chosen from). An example is shown in Figure 5.9(c). Only points on part of the boundary of S_1^1 (marked by thicker lines) are used to construct Bounding Edges because they are the points from which the resulting Bounding Edges can be seen by at least two other cameras, namely cameras 2 and 3.

5.4.2 Determining Visibility During Alignment

To perform the alignment using Equation (5.11), we have to determine the visibility of the transformed 3D point $\boldsymbol{R}W_1^i + \boldsymbol{t}$ w.r.t. the cameras at time t_2 (and vice versa the visibility for the transformed point $\boldsymbol{R}^T(W_1^i - \boldsymbol{t})$ w.r.t. the cameras at time t_1). Naively, we can just apply Lemma 5.6 to the transformed point $\boldsymbol{R}W_1^i + \boldsymbol{t}$ directly. In practice, however, this "direct approach" does not work for the following reason. Since the CSP W_1^i lies on the surface of the object, the projection of the transformed point $\boldsymbol{R}W_1^i + \boldsymbol{t}$ should lie *inside* the silhouettes at time t_2, unless it happens to be on the occluding contour of the object again at t_2 such that its projection lies on the boundary of some of the silhouette images. Either way, this means

that no matter where the camera centers are, the line joining the projection of $\boldsymbol{R}W_1^i + \boldsymbol{t}$ and the camera centers almost always intersects the silhouettes as shown in Figure 5.10(a). Hence, the visibility of the point W_1^i at t_2 will almost always be treated as indeterminable by Lemma 5.6 due to its over-conservative nature.

Here we suggest a "reverse approach" to deal with this problem. Instead of applying the transformation $(\boldsymbol{R}, \boldsymbol{t})$ to the point W_1^i, we apply the inverse transform $(\boldsymbol{R}^T, -\boldsymbol{R}^T\boldsymbol{t})$ to the camera centers and project the transformed camera centers into the one silhouette image (captured at t_1) where W_1^i is originated from as shown in Figure 5.10(b). Lemma 5.6 is then applied to the boundary point u_1^i (which generates the Bounding Edge E_1^i that W_1^i lies on) and the projections of the transformed camera centers to determine the visibility. Since the object is rigid, the reverse approach generates the correct visibility of $\boldsymbol{R}W_1^i + \boldsymbol{t}$ w.r.t. the cameras at t_2 at least in the limit that the alignment $(\boldsymbol{R}, \boldsymbol{t})$ tends to the correct values.

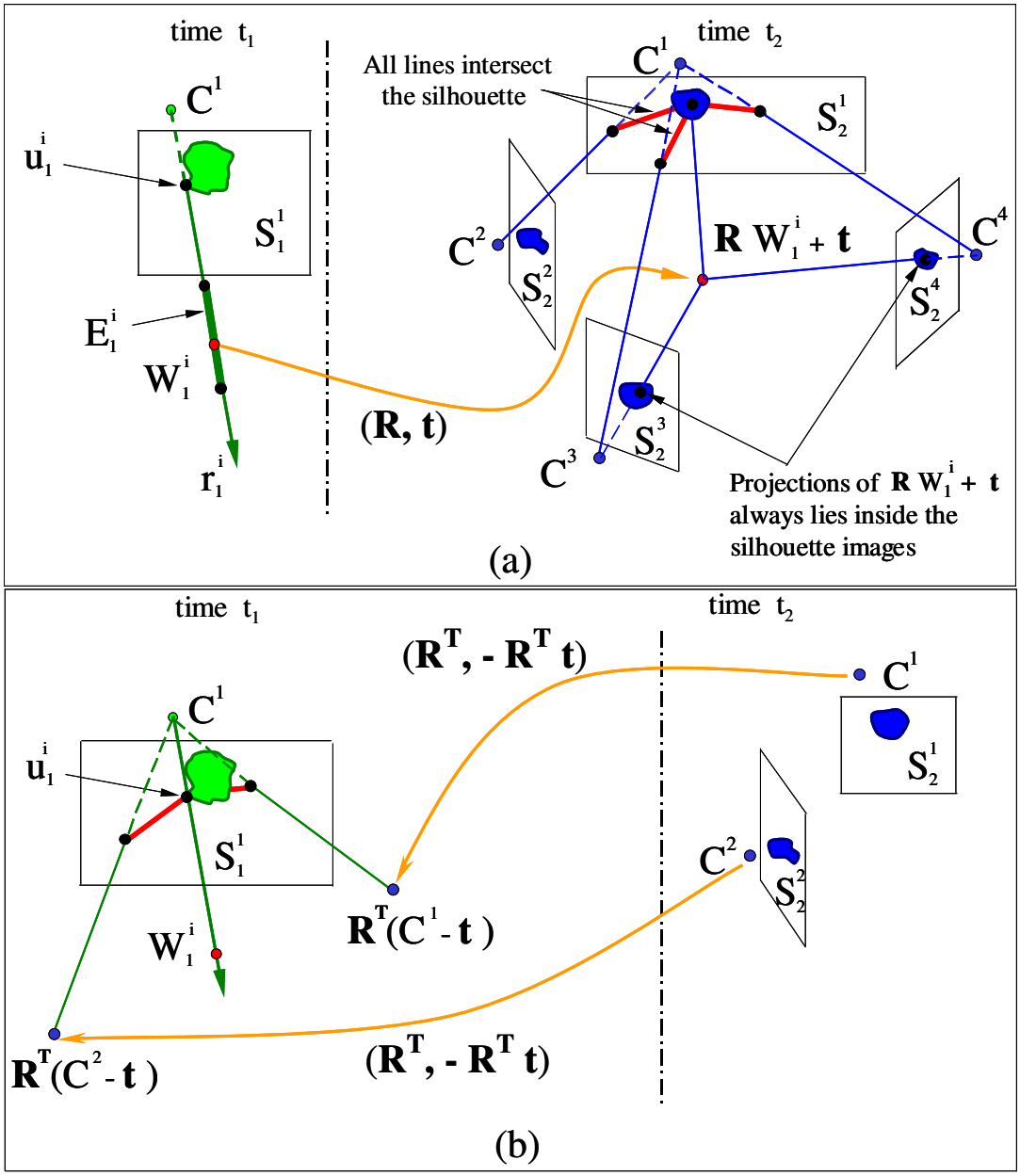


Figure 5.10: (a) The "Direct approach" of applying Lemma 5.6 to determine the visibility of $\boldsymbol{R}W_1^i + \boldsymbol{t}$ w.r.t. $\{S_2^k\}$. The projection of $\boldsymbol{R}W_1^i + \boldsymbol{t}$ almost always lies inside $\{S_2^k\}$. The over-conservative nature of Lemma 5.6 prohibits us for determining the visibility of $\boldsymbol{R}W_1^i + \boldsymbol{t}$. (b) The "Reverse approach" of applying Lemma 5.6 to determine visibility of $\boldsymbol{R}W_1^i + \boldsymbol{t}$ w.r.t. $\{S_2^k\}$. The camera centers are inversely transformed by $(\boldsymbol{R}^T, -\boldsymbol{R}^T\boldsymbol{t})$ and then projected onto $\{S_1^k\}$. The visibility can then be determined by checking if the lines joining u_1^i and the projections of the transformed camera centers intersect with S_1^1 exactly as in Lemma 5.6.

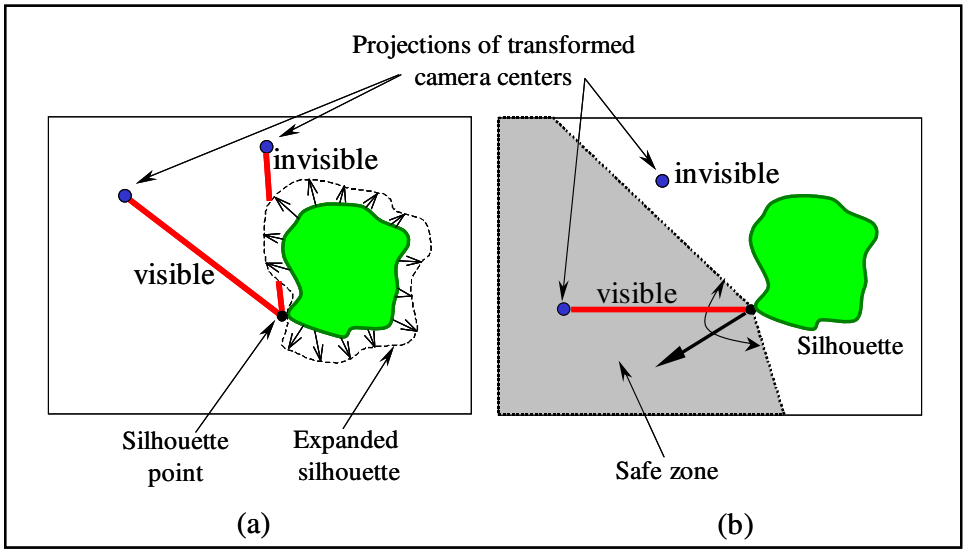


Figure 5.11: Two measures to increase the conservativeness of the visibility test at the beginning of the optimization process. (a) "Expand the silhouette away" from the point under consideration. (b) Create a "safe zone" around the local normal at the silhouette image point.

Although the over-conservative nature of Lemma 5.6 is avoided using the reverse approach, there is one downside. Some of the visibility tests may not be correct in the first few iterations of the alignment optimization when the estimated transformation $(\boldsymbol{R}, \boldsymbol{t})$ is far from the correct solution, thus causing optimization instability. To compensate for this initial jitter, we increase the conservativeness of the visibility test (at the beginning of the optimization process) using one or both of the two measures shown in Figure 5.11. The first measure (Figure 5.11(a)) works by "expanding the silhouette" away from the point under consideration to deal with self-occlusion by points far away. The second measure (Figure 5.11(b)) creates a "safe zone" by estimating the local normal of the silhouette image point and only considering the cameras whose centers are projected inside a pre-defined wedge (around the normal) as visible. When the estimated motion parameters converge toward the correct solutions, the above measures are relaxed as the visibility tests using the reverse approach become more and more accurate and the minimization framework is able to handle small errors in the visibility tests.

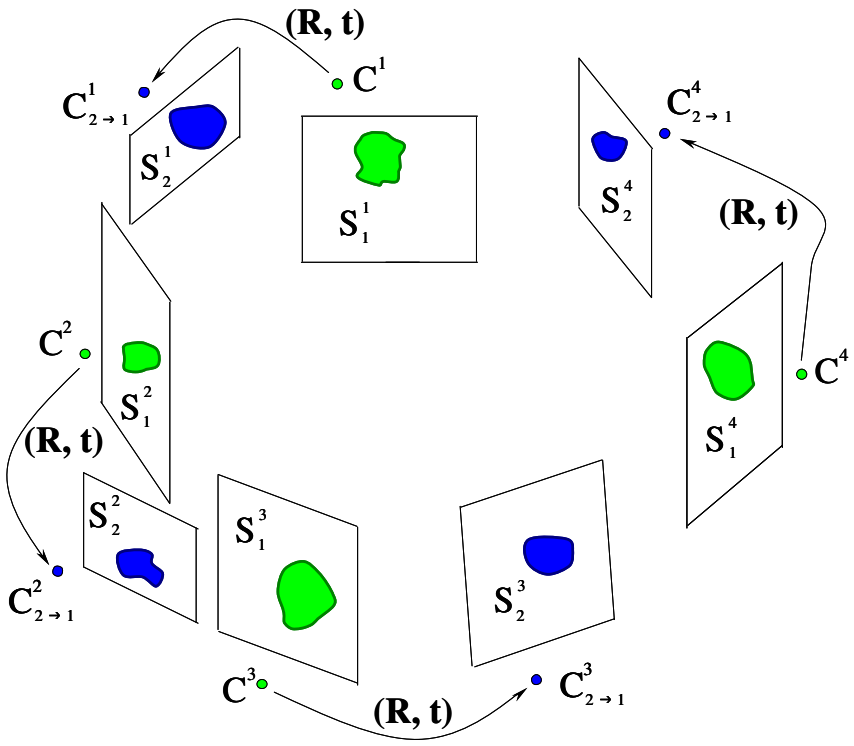


Figure 5.12: Visual Hull Refinement: the silhouette images at time t_2 are incorporated into time t_1 by transforming the camera centers according to the recovered rigid motion $(\boldsymbol{R}, \boldsymbol{t})$.

5.5 Visual Hull Refinement

After estimating the alignment across time, the rigid motion $\{(\boldsymbol{R}_j, \boldsymbol{t}_j)\}$ is used to combine the J sets of silhouette images $\{S_j^k;\ k = 1, \cdots\cdots, K;\ j = 1, \cdots\cdots, J\}$ to get a tighter upper bound of the shape of the object. By fixing t_1 as the reference time, we combine $\{S_j^k\};\ j = 2, \cdots\cdots J$ with $\{S_1^k\}$ by considering the former as "new" silhouette images captured by additional cameras placed at positions and orientations transformed by $(\boldsymbol{R}_j, \boldsymbol{t}_j)$. In other words, for the silhouette image S_j^k captured by camera k at time j, we use a new perspective projection function $\Pi_{j\to 1}^k$ derived from Π^k through the rigid transformation $(\boldsymbol{R}_j, \boldsymbol{t}_j)$. As a result, the effective number of cameras is increased from K to $K \times J$. The idea of warping the cameras from time t_2 to time t_1 is depicted in Figure 5.12.

5.6 Experimental Results

Two types of sequences are used to demonstrate the validity of our alignment and refinement algorithm. Firstly, a synthetic sequence is used to obtain a quantitative comparison of several aspects of the the algorithm. Two sets of experiments are run on the synthetic sequence. The first set of experiments (Algorithms I, II and III in Section 5.6.1.1) studies how the alignment accuracy is affected by each component (color consistency and geometrical constraints) of the error measure in Equations (5.9) and (5.10). The second set of experiments (Algorithms II, IV and V in Section 5.6.1.2) compares the effectiveness of using Colored Surface Points to align Visual Hulls with (1) voxel models created by Shape-From-Silhouette and (2) Space Carving [KS00]. After our alignment algorithm is tested on synthetic data, several sequences of real objects are used in Sections 5.6.2 for qualitative evaluation on data with real noise, calibration errors and imperfectly color balanced cameras.

5.6.1 Synthetic Data Set (Torso Sequence)

A synthetic data set is created using a textured wire-frame computer model resembling the human torso. The model was moved under a known trajectory for twenty two frames. At each time instant, images of six cameras ($K = 6$) with known camera parameters are rendered using OpenGL. A total of 22 sets of color and silhouette images are generated. Some input images for cameras 1 and 6 at a variety of frames are shown in Figure 5.13.

5.6.1.1 Experiment Set 1: Effect of Error Measure on Alignment Accuracy

A. Alignment

In the first set of experiments, Bounding Edges and Colored Surface Points are first extracted. Then three alignment algorithms were implemented to investigate the effect of each component (color consistency and geometric constraints) of the error measure on the alignment accuracy. In Algorithm I, only the error from the geometrical constraints is used

time t_1 time t_6 time t_{11} time t_{16} time t_{21}

Input images of the Torso sequence from Camera 1

time t_1 time t_6 time t_{11} time t_{16} time t_{21}

Input images of the Torso sequence from Camera 6

Figure 5.13: Some of the input images of cameras 1 and cameras 6 of the synthetic torso sequence.

(i.e. the first term $d_2^k(\boldsymbol{R}W_1^i + \boldsymbol{t})$ in Equation (5.9)). In Algorithm II, only the error caused by the color inconsistency is used (i.e. the second term $[\boldsymbol{c}_2^k(\boldsymbol{R}W_1^i + \boldsymbol{t}) - \mu_1^i]^2$ in Equation (5.9)). In Algorithm III, both errors are used. The results for the 6 motion parameters estimated over time from this experiment set are shown in Figures 5.14, 5.15 and 5.16. In the figures, the ground-truth values are drawn with solid black lines, the results obtained from using both geometric constraints and color consistency (Algorithm III) are drawn with magenta dotted lines with an inverted triangle, the results with only the geometric constraints (Algorithm I) are drawn with blue dashed-dotted lines with circle, and the results with only color consistency (Algorithm II) are drawn with red dashed lines with asterisks. As expected the results of using both error components at the same time are the best, followed by the results using only the color consistency. The results obtained using only the geometric constraints are the worst of the three. As discussed in Section 5.2, aligning Visual Hulls using only geometric (silhouette) information is inherently ambiguous. This means that if color consistency (the second term of Equation (5.9)) is not used, there exists more than

one global minimum to Equation (5.11) (see the 2D examples in Figure 5.4). Under such situations, optimizing Equation (5.11) may converge to any global minimum other than the actual motion of the object. This explains why the results of Algorithm I are not as good as Algorithms II and III.

B. Refinement

To study the effect of alignment on refinement, the estimated parameters in the alignment experiments are used to refine the shape of the torso model using the voxel-based SFS method [Sze93]. The Visual Hull at time t_j is constructed using the silhouettes at t_j and all those from the previous frames $\{t_1, \cdots\cdots, t_{j-1}\}$, transformed by the estimated motion parameters as described in Section 5.5. To quantify the refinement results, the ground-truth wire-frame model (see Figure 5.21(a)) used to render the input images is converted into a ground-truth voxel model (see Figure 5.21(b)) and compared to the refined voxel models. The results are plot in Figure 5.17 with graphs (a) and (b) showing the number of extra and missing voxels between the refined shapes and the ground-truth voxel models against the number of frames used. Figure 5.17(c) illustrates the ratio of total incorrect (missing plus extra) to total voxels.

In all of the results obtained from Algorithm I, II and III, the number of extra voxels decreases as the number of frames used increases because a tighter Visual Hull is obtained with an increase in the number of distinct silhouette images. However, the number of missing voxels also increases as the number of frames used increases. This is due to alignment errors which remove correct voxels during construction. The refinement results are the best with the motion parameters estimated using both the color consistency and the geometric constraints (the magenta dotted lines with inverted triangle) from Algorithm III. Again Algorithm II (just color consistency) is better than Algorithm I (just geometric constraints).

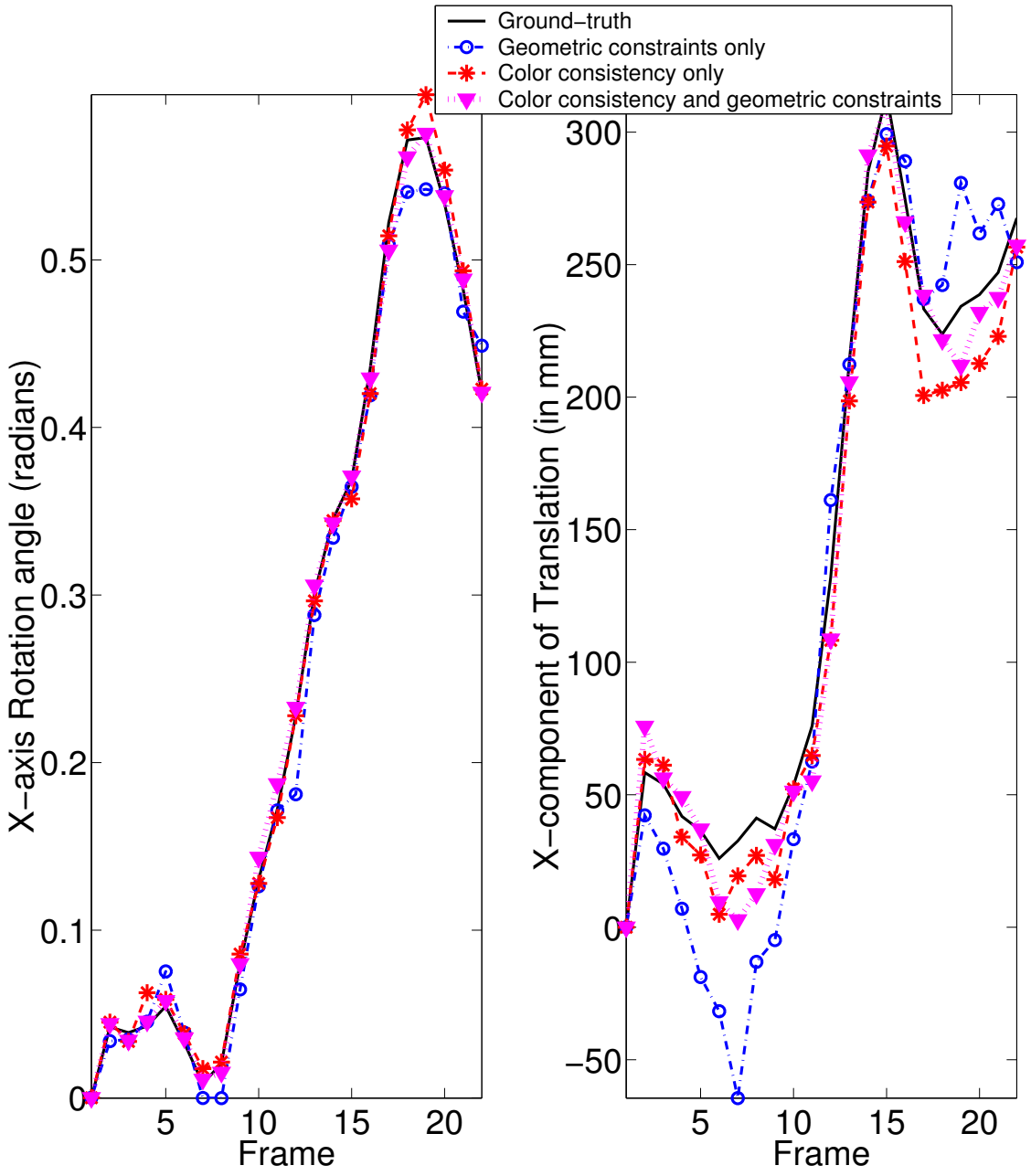


Figure 5.14: Results of X-axis rotation angle and X-component of translation estimated over time from Experiment Set 1 with different error measure: only geometric constraints is used (blue dashed-dotted lines with circle), only color consistency is used (red dashed lines with asterisks), both geometric constraints and color consistency are used (magenta dotted lines with inverted triangle). The solid black lines represents the ground-truth values. Results obtained using both error components are the best followed by results using only color consistency. Due to the alignment ambiguity, results using only geometrical constraints are the worst among the three.

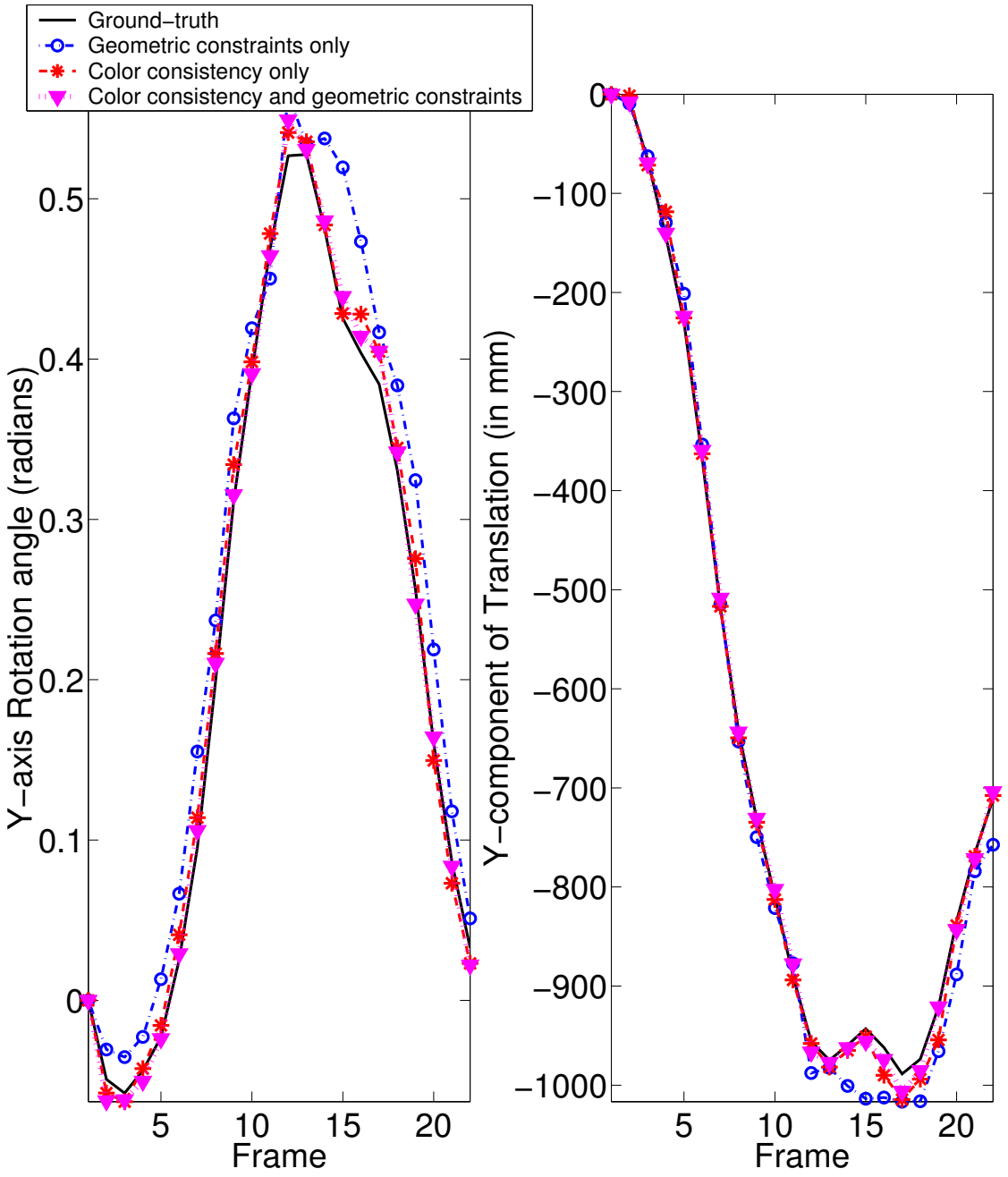


Figure 5.15: Results of Y-axis rotation angle and Y-component of translation estimated over time from Experiment Set 1 with different error measure. See caption of Figure 5.14 for further details.

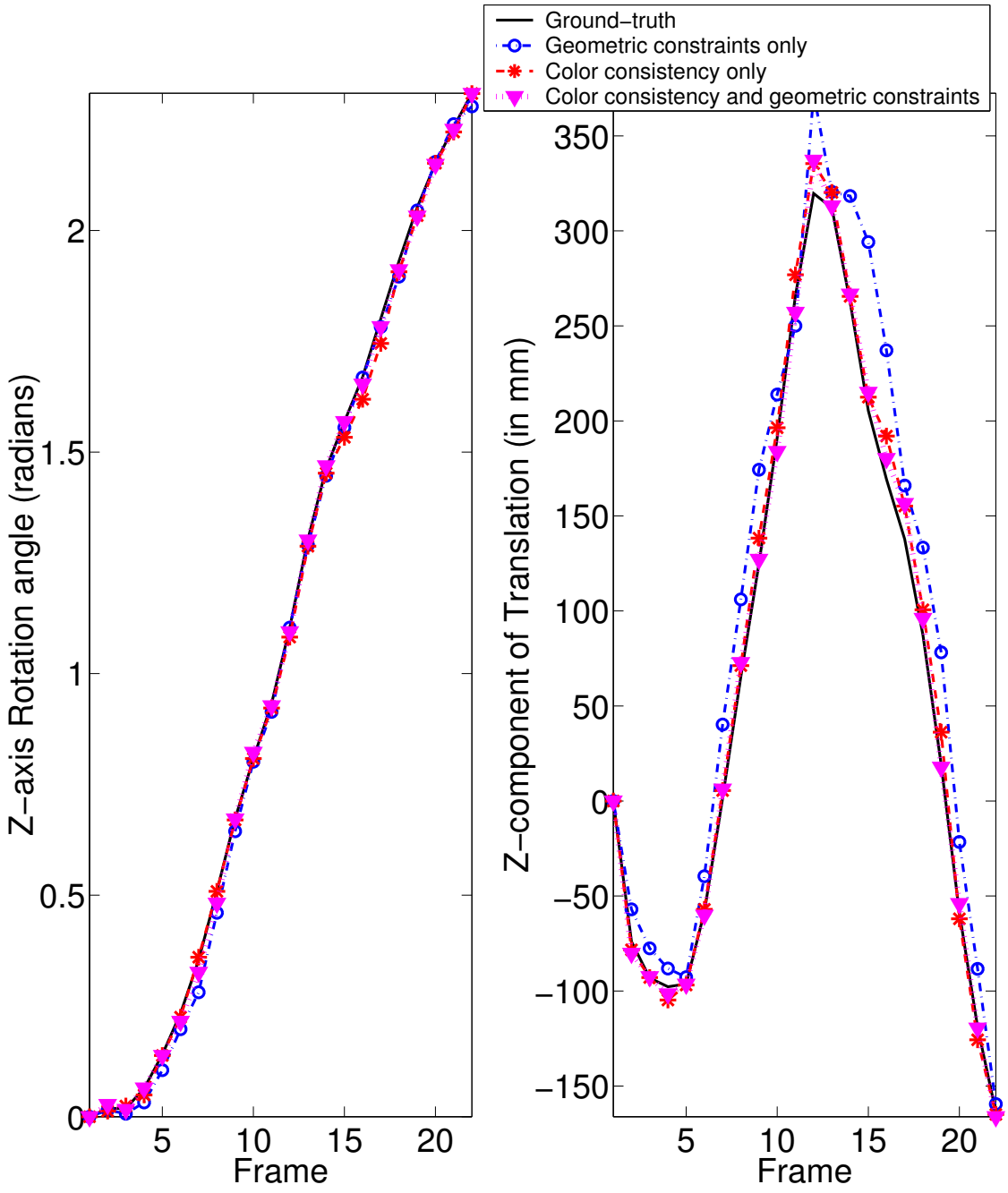


Figure 5.16: Results of Z-axis rotation angle and Z-component of translation estimated over time from Experiment Set 1 with different error measure. See caption of Figure 5.14 for further details.

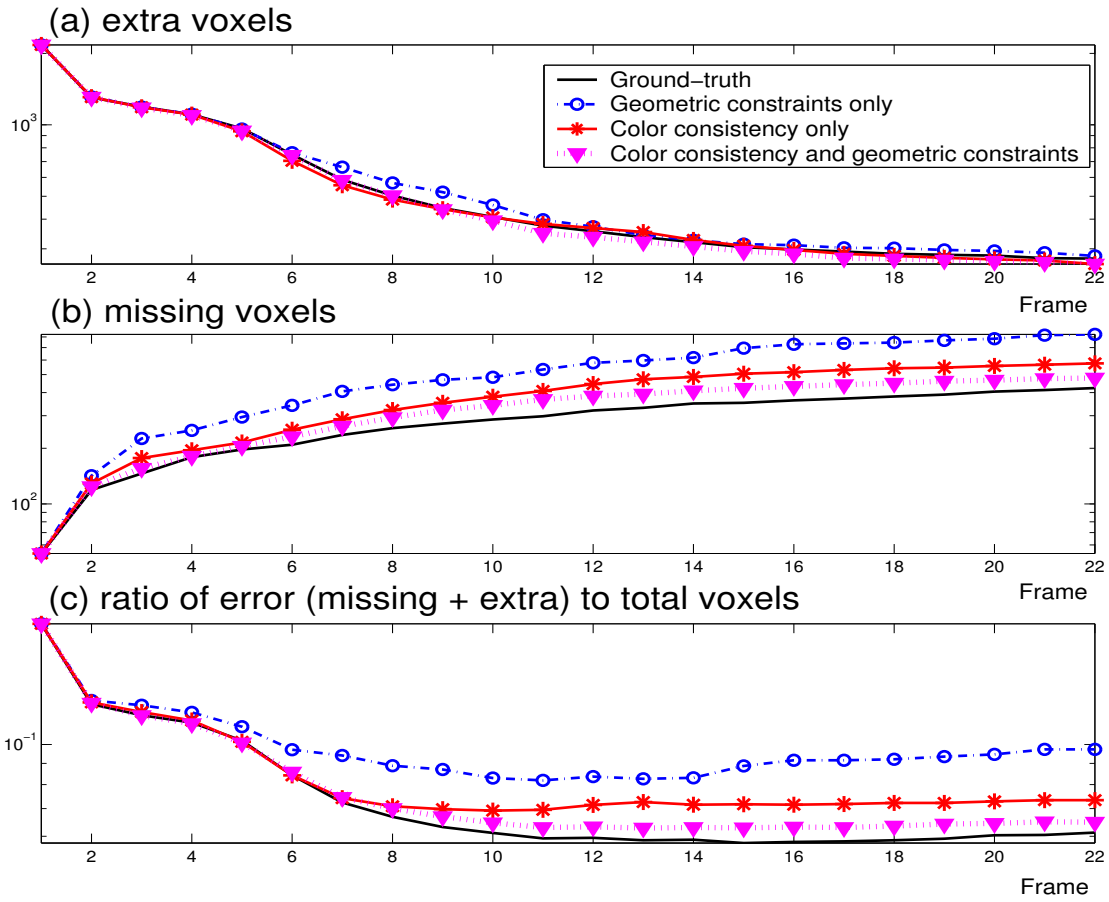


Figure 5.17: Graphs of refinement errors (missing and extra voxels) across time (frames). Using both color consistency and geometric constraints has lower error ratio than just using either one of them.

5.6.1.2 Experiment Set 2: BE/CSP versus SFS and SC

A. Alignment

In the second set of experiments, two more alignment algorithms (Algorithms IV and V) were implemented to show the effectiveness of using Bounding Edges/Colored Surface Points (Algorithm II) to align Visual Hulls compared to using voxel models created by Shape-From-Silhouette (SFS) and Space Carving (SC) [KS00]. In Algorithm IV, a voxel model is built from the silhouette images using voxel-based Shape-From-Silhouette (SFS). Surface voxels are extracted and colored by back-projecting onto the color images. The centers of the colored surface voxels are then treated as input data points to the same alignment algorithm used in Algorithm II (i.e. only color consistency but not geometric

constraints are used in the error measure). In Algorithm V, a voxel model is first built using SFS (as in Algorithm IV) and further refined by Space Carving (SC). The centers of the surface voxels (which are already colored by SC) are used for alignment using only the color consistency error measure. To investigate the effect of the space carving threshold on alignment, different values of the threshold are used and the estimated motion parameters are compared with the ground truth values. Graphs of the average RMS errors in the rotation and translation parameters against the threshold used are shown as blue dotted-dashed lines in Figure 5.18. When the threshold is too small, many correct voxels are carved away, resulting in a voxel model much smaller than the actual object. When the threshold is too big, extra incorrect voxels are not carved away, leaving a voxel model bigger than the actual object. In both cases, the wrong data points extracted from the incorrect voxel models cause errors in the alignment process. The optimal threshold value is found to be around 0.108 and the graph is amplified in the vicinity of this value in the bottom part of Figure 5.18. As a comparison, the average RMS errors for the rotation and translation parameters obtained from Algorithm II (using BE and CSPs) is drawn as the horizontal red dashed line. With the optimal SC threshold, the performance of using SFS+SC voxel models is comparable but less accurate than that of using Bounding Edges and Colored Surface Points.

The results of the Y-axis rotation angle and the X-component of translation by Algorithm V with the optimal threshold are plotted as thick blue dotted lines in Figure 5.19 while the results by Algorithm IV are plotted as magenta dotted-dashed lines. For comparison, the estimated parameters of Algorithm II (red dashed lines with asterisks) from Experiment Set 1 and the ground-truth values (solid black lines) are drawn again in the same figures. As can be seen, alignment using the SFS voxel model is much less accurate than using Bounding Edges and Colored Surface Points. Space Carving with the optimal threshold performs well but is also not quite as good as using Bounding Edges. Table 5.6.1 gives a rough comparison of the computational time needed for each step of all of the experiments. The timing is obtained on a 500MHz Pentium III CPU.

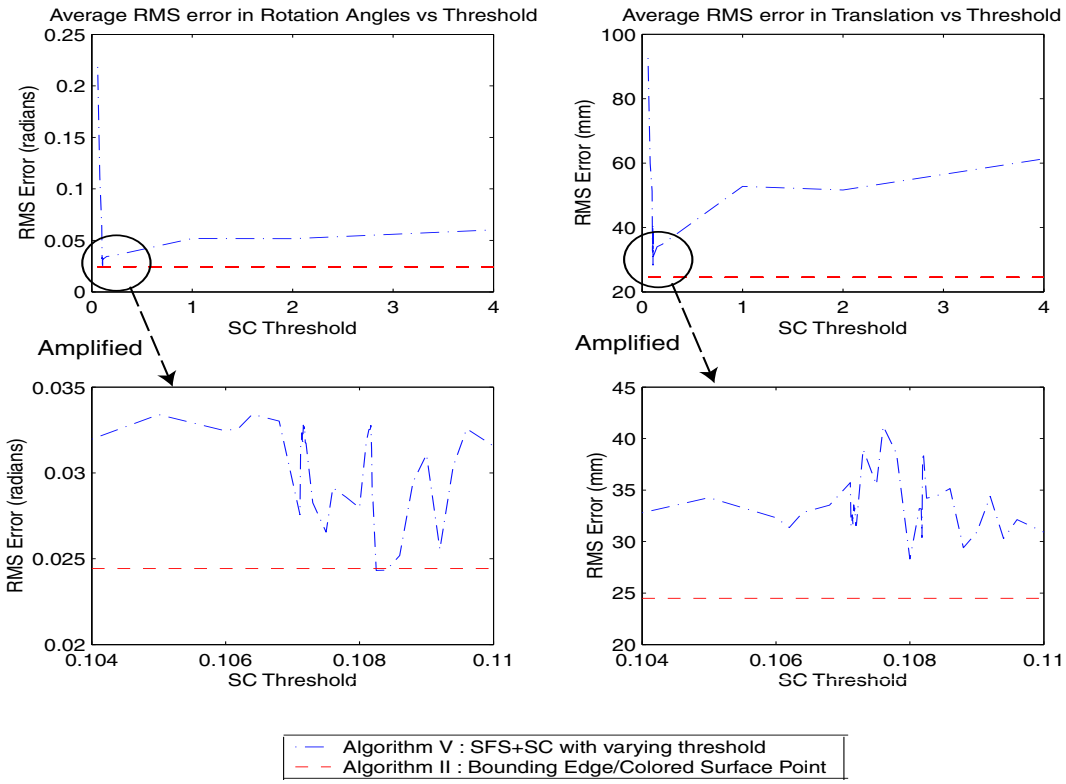


Figure 5.18: Graphs of the average RMS errors in rotation and translation against the threshold used in SC in algorithm V. The bottom half of the figure illustrates the amplified part of the graph near the optimal threshold value (0.108). Using Bounding Edges (the red dashed line) is always more accurate than using SC in alignment, even if the optimal threshold is used for SC.

Step	**Approximate Time required per frame**
Extracting Bounding Edge (BE)	0.92s
Locating Colored Surface Points (100 points searched on each BE)	0.16s
SFS with 128^3 voxels	1.08s
SFS + SC with 128^3 voxels and optimal threshold	4.74s
Alignment	16.2s

Table 5.1: The approximate time for each step in the alignment experiments. Bounding Edge is about the same as SFS and faster than SFS+SC.

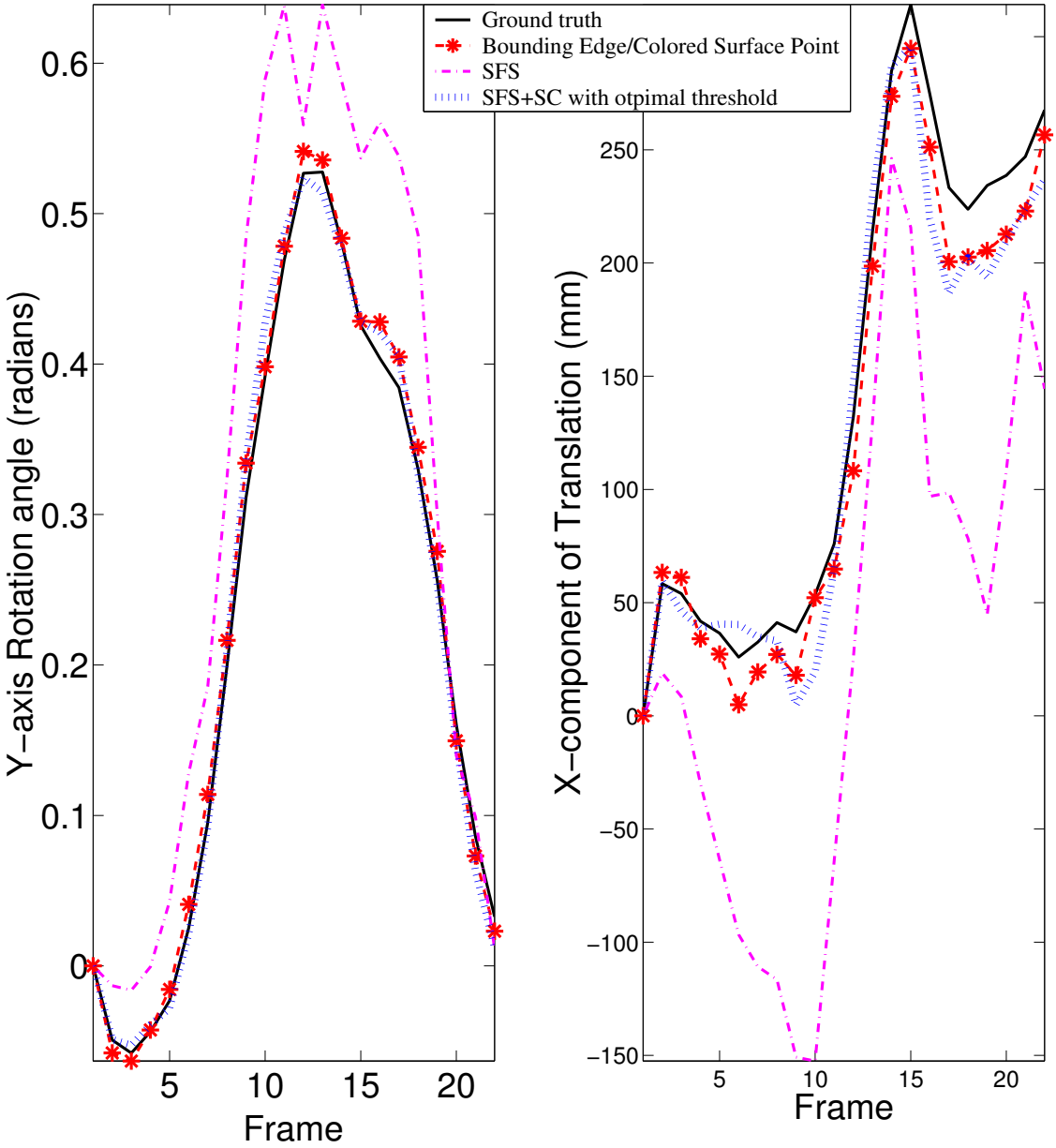


Figure 5.19: Results of Y-axis rotation angle and X-component of translation estimated over time from Experiment Set 2 with different input data: Bounding Edges/Colored Surface Points (red dashed lines with asterisks), SFS voxel models (magenta dotted-dashed lines), SFS+SC voxel models with optimal threshold (blue thick dotted lines) and ground-truth values (solid black lines). Using Bounding Edges/Colored Surface Points are better than using either SFS or SFS+SC.

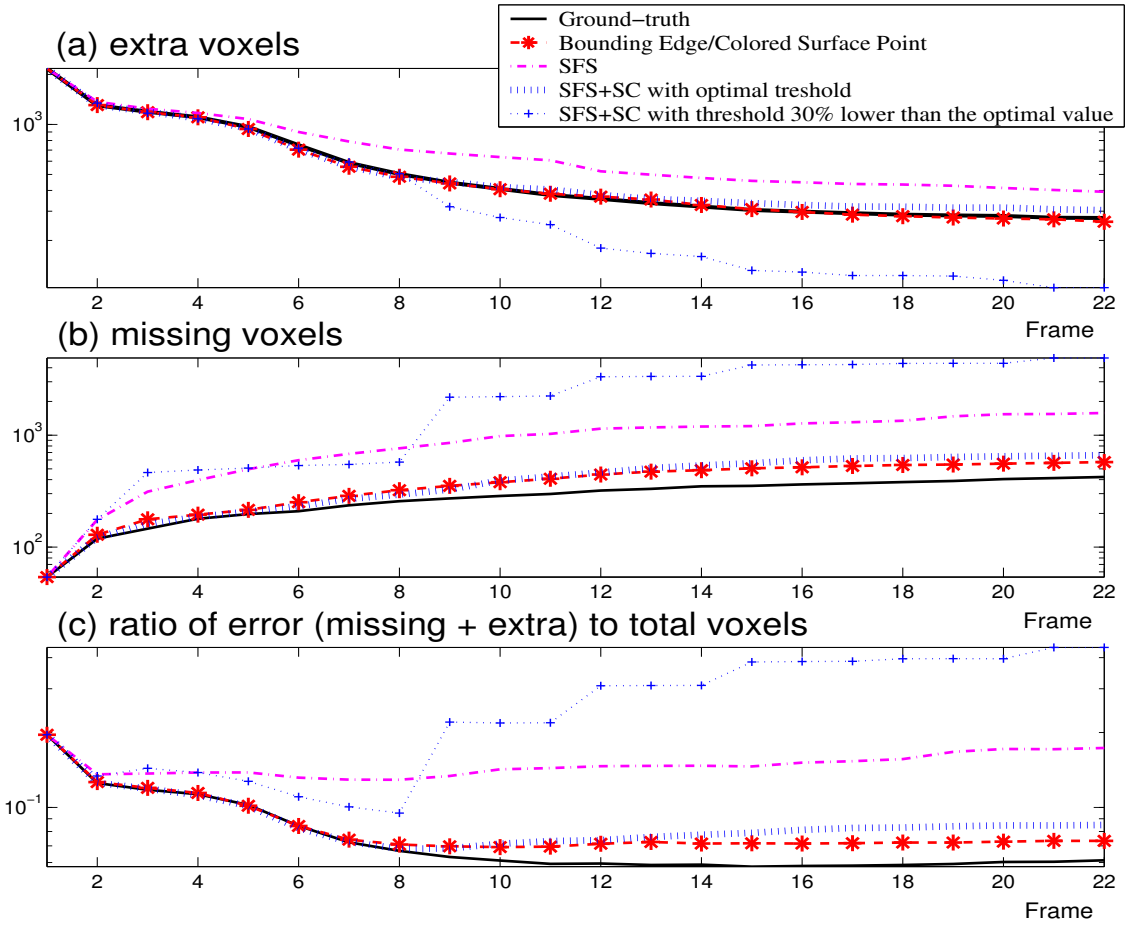


Figure 5.20: Graphs of refinement errors (missing and extra voxels) across time (frames). Using Bounding Edges has lower error ratio than using either SFS or SFS+SC.

B. Refinement

The estimated parameters in Experiment Set 2 are used to refine the shape of the torso model exactly as in Section 5.6.1.1. The results are plot in Figure 5.20 with graphs (a) and (b) showing respectively the number of extra and missing voxels between the refined shapes and the ground-truth voxel models against the number of frames used. Figure 5.20(c) illustrates the ratio of total incorrect (missing plus extra) to total voxels.

From the figure it can be seen that the number of missing voxels is very large if the alignments are way off (e.g. the magenta dotted-dashed curve of results from Algorithm IV or the blue dotted curves with '+' markers of results from Algorithm V with threshold 30%

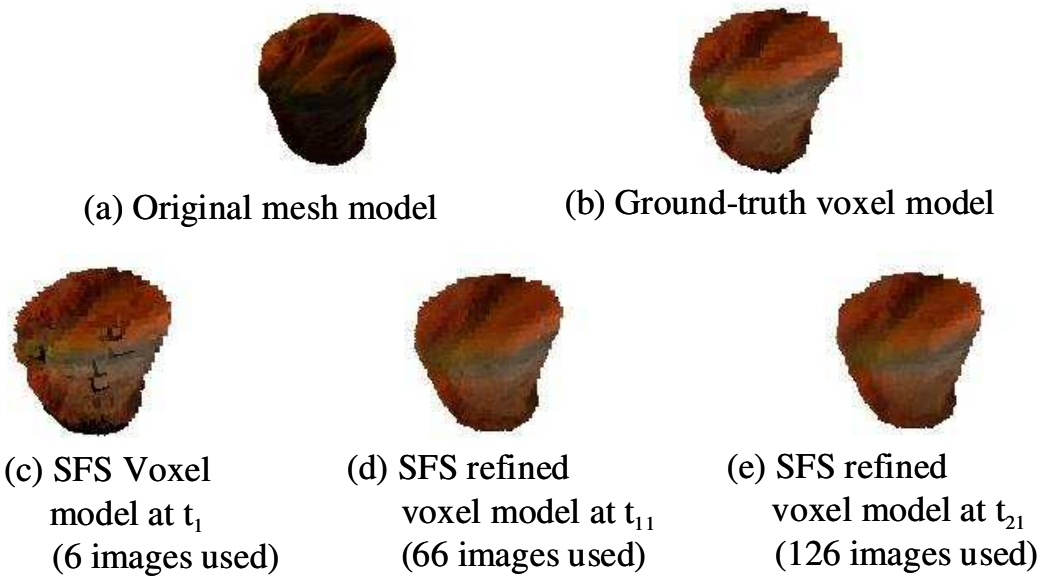


Figure 5.21: (a) Voxel model constructed at t_1 using only 6 silhouette images. (b) Refined SFS voxel model at t_{11} using 66 silhouette images. (c) Refined SFS voxel model at t_{21} using 126 silhouette images. There is significant improvement in shape from (a) to (c).

lower than the optimal value). The refinement result is the best using the motion parameters estimated using Bounding Edges (the red dashed lines with asterisks in Figure 5.20). The SFS refined models (using estimated motions from Algorithm III) at time t_1, t_{11} and t_{21} are shown in Figure 5.21(b)(c)(d) respectively. A video clip **Torso.mpg**[*] shows one of the six input image sequences (camera 4), the unaligned and aligned Colored Surface Points and the temporal refinement/alignment results using Algorithm III.

5.6.2 Real Data Sets: Toy Pooh and Toy Dinosaur

5.6.2.1 Pooh Sequence:

The first test object is a toy (Pooh). Six calibrated cameras ($K = 6$) are used. The toy is placed on a table and moved to new but unknown positions and orientations manually in each frame. A total of fifteen frames are captured. The input images of cameras 1 and

[*]All movie clips of this chapter can be found at http://www.cs.cmu.edu/~german/research/Thesis/Video/Chapter5/

time t_1 time t_4 time t_7 time t_{10} time t_{15}

Input images of the Pooh sequence from Camera 1

time t_1 time t_4 time t_7 time t_{10} time t_{15}

Input images of the Pooh sequence from Camera 4

Figure 5.22: Some of the input images of camera 1 and camera 4 of the Pooh sequence.

4 at several times are shown in Figure 5.22. The CSPs extracted at time t_1 are shown in Figure 5.23(a). Figures 5.23(b) and (c) show respectively the unaligned and aligned Colored Surface Points from all fifteen frames. Refinement is done using the voxel-based SFS method. Figures 5.23(d),(e) and (f) illustrate the refinement results at three time instants t_1 (6 images), t_6 (36 images) and t_{15} (90 images). The improvement in shape is very significant from t_1 when 6 silhouette images are used to t_{15} when 90 silhouette images are used. Note that for shape refinement, Space Carving (SC) can also be used. Figures 5.23(g), (h) and (i) show the refinement results using SFS + SC at t_1, t_6 and t_{15} respectively. Generally with a *good threshold*, refinement using SFS + SC is better than SFS for the same number of images. The video clip **Pooh.mpg** shows one of the six input image sequences (camera 4), the unaligned/aligned Colored Surface Points and the temporal refinement/alignment results.

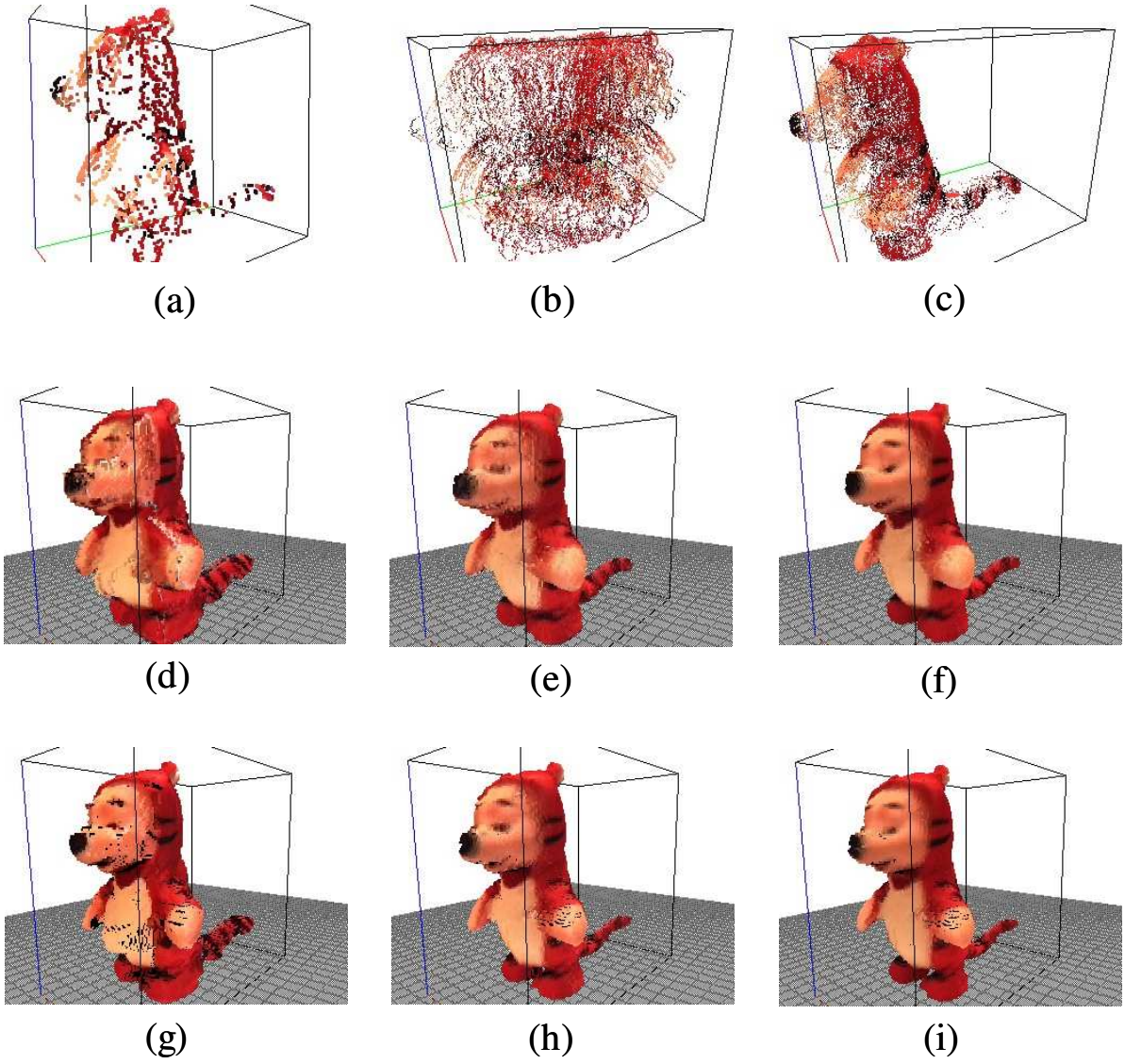


Figure 5.23: Pooh Data Set. (a) Colored surface points at t_1. (b) Unaligned Colored Surface Points from all frames. (c) Aligned Colored Surface Points of all frames. (d) SFS model at t_1 (6 images used). (e) SFS refined shape at t_6 (36 images used). (f) SFS refined shape at t_{15} (90 images used). (g) SFS + SC model at t_1. (h) SFS + SC refined model at t_6. (i) SFS + SC refined model at t_{15}. See **Pooh.mpg** for a movie illustrating these results.

5.6.2.2 Dinosaur-Banana Sequence:

The objects used in the second real data set are the toy dinosaur/bananas shown in Figure 4.3. Six cameras are used and the dinosaur/bananas are placed on a turn-table with unknown rotation axis and rotation speed. Fifteen frames are captured and the alignment and refinement results are shown in Figure 5.24. The video clip **Dinosaur-Banana.mpg** shows one of the six input image sequences (camera 4), the unaligned/aligned Colored Surface Points and the temporal refinement/alignment results.

Note that we also apply the temporal SFS algorithm to real sequences of a person rigidly standing on a turn-table. The results will be presented in Chapter 7 (Section 7.2) when we describe a system for building kinematic models of humans.

5.7 Related Work

Despite the popularity of SFS as a shape reconstruction method at single time instant, little work has been done in extending it across time. The work most related to ours is by Cipolla, Wong and Mendonca [MWC00, WC01b, WC01a] who study the problem of estimating structure and motion of smooth object undergoing *circular motion* from silhouette profiles (also see [JBJ01]). They assume a single camera which is weakly calibrated (i.e. with known intrinsic but unknown extrinsic parameters). Either the camera (on a robotic arm) or the object (on a turntable) performs unknown circular motion while the silhouette images are taken. In [MWC01] symmetric properties of the surface of revolution swept by the rotating object are used to recover the revolution axis, leading to the estimation of homographies and full epipolar geometries between images using one-dimensional search. In [WC01b], they identity and estimate the *frontier points* (see [JAP94] for detailed definition) on the silhouette boundary and use them to estimate the circular motion between images. Once the motion is estimated, the object shape can be reconstructed using the classic SFS method.

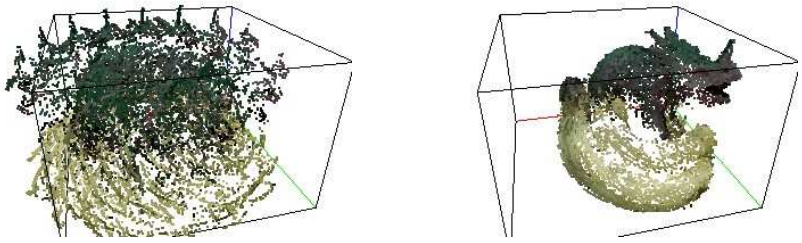
time t_1 time t_4 time t_7 time t_{10} time t_{15}

Input images of the Dinosaur-Bananas sequence from Camera 1

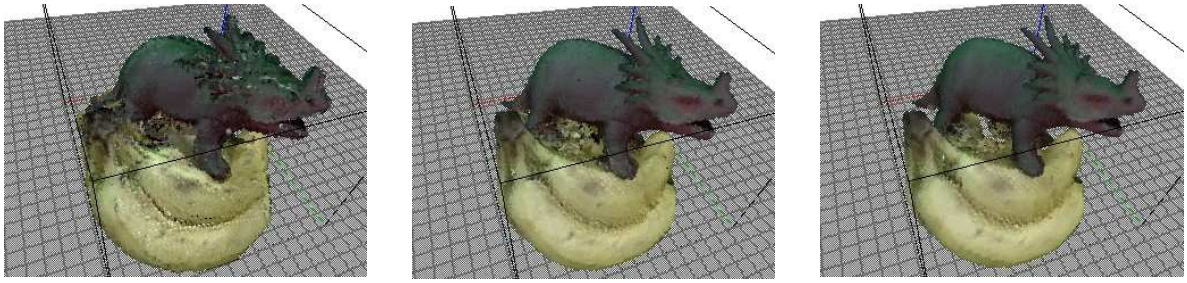
time t_1 time t_4 time t_7 time t_{10} time t_{15}

Input images of the Dinosaur-Bananas sequence from Camera 4

(a)



(b) (c)



(d) (e) (f)

Figure 5.24: Dinosaur-Banana Sequence. (a) Example input images. (b) Unaligned Colored Surface Points from all frames. (c) Aligned Colored Surface Points from all frames. (d) SFS model at t_1 (6 images used). (e) SFS refined shape at t_6 (36 images used). (f) SFS refined shape at t_{15} (90 images used). There is significant shape improvement from (d) to (f). See **Dinosaur-Banana.mpg** for a movie illustrating these results.

Another group of researchers, lead by Ponce [JAP94, JAP95, VKP96] have also studied the problem of recovering motion and shape of *smooth curved* object from silhouette images. They define a local parabolic structure on the surface of the object and use that, together with epipolar geometry, to locate corresponding frontier points on three silhouette images. Motion between the images is then estimated using a two-step nonlinear minimization. In contrast to these algorithms, our approach has two advantages: (1) no shape assumptions are made about the object and (2) no assumptions are made about the motion (in our algorithm the motion does not have to be infinitesimal).

5.8 Discussion

In this chapter we have investigated the problem of performing Shape-From-Silhouette across time for a rigid object undergoing arbitrary and unknown rigid motion. We studied the ambiguity issue of aligning Visual Hulls and proposed an algorithm using stereo to break the ambiguity. We first represent each Visual Hull using Bounding Edges. Colored Surface Points are then located on the Bounding Edges by comparing color consistencies. The Colored Surface Points are used to estimate the rigid motion of the object across time, using a 2D images/3D points alignment algorithm. Once the alignment is known, all of the images are considered as being captured at the same instant. The refined shape of the object can then be obtained by any reconstruction method such as SFS or Space Carving.

Our algorithm combines the advantages of both SFS and Stereo. A key principle behind SFS, expressed in the Second Fundamental Property of Visual Hulls, is naturally embedded in the definition of the Bounding Edges. The Bounding Edges give us, as a representation for the Visual Hull, a great deal of the accurate shape information that can be obtained from the set of silhouette images. To locate the touching surface points, multi-image stereo (color consistency among images) is used. Two major difficulties of doing stereo : visibility and search size are both handled naturally using the properties of the Bounding Edges.

The ability to combine the advantages of both SFS and Stereo is the main reason why using Bounding Edges/Colored Surface Points gives better results in motion alignment than using voxel models obtained from SFS or SC, as is evident in the results in Section 5.6.1.2. Another disadvantage of using voxel models and Space Carving is that each decision (voxel is carved away or not) is made *individually* for each voxel according to a criterion involving thresholds. On the contrary, in locating colored surface points on Bounding Edges, the decision (which point on the Bounding Edge touches the object) is made *cooperatively* (by finding the point with the highest color consistency) along all the points on the Bounding Edge, without the need of adjusting thresholds. In summary, the information contained in Bounding Edges/Colored Surface Points is more accurate than that contained in voxel models from SC/SFS. In parameter estimation, few but more accurate data is always preferred over abundant but less inaccurate data, especially in applications such as alignment.

Chapter 6

VH Across Time for Articulated Objects

In this chapter we extend the temporal SFS algorithm to articulated objects. An object is articulated if it consists of a set of rigidly moving parts connecting to each other at certain articulation points. A good example of an articulated object is the human body (if we approximate the body parts as rigid). Given CSPs of a moving articulated object, recovering the shape and motion requires two inter-related steps: (1) correctly segmenting the CSPs to each part of the object and (2) estimating the shape and motion of the individual parts. To solve this problem, we employ an idea similar to that used for multiple-layer motion estimation in [SA96]. The rigid parts of the articulated object are first modeled as separate and independent of each other. With this assumption, we iteratively (1) assign the extracted CSPs to different parts of the object based on their motions and (2) apply the rigid object temporal SFS algorithm to align each part across time. Once the motions of the parts are recovered, an articulation constraint is applied to estimate the joint positions. Note that this iterative approach can be categorized as belonging to the Expectation Maximization framework [DLR77]. The whole algorithm is explained below in detail using a two-part, one-joint articulated object. It can be generalized to objects with N parts.

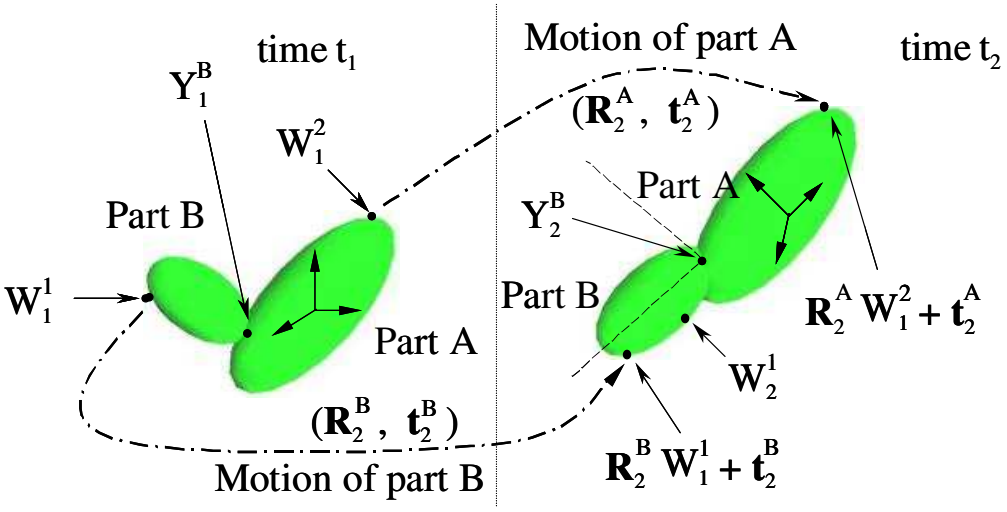


Figure 6.1: A two-part articulated object at two time instants t_1 and t_2.

6.1 Temporal SFS for Unknown Articulated Objects

6.1.1 Problem Scenario

Consider an unknown one-joint articulated object O which consists of two rigid parts A and B as shown in Figure 6.1 at two time instants t_1 and t_2. Assume CSPs of the whole object have been extracted from the color and silhouette images of K cameras, denoted by $\{I_j^k, S_j^k, W_j^i, \mu_j^i;\ \ j = 1, 2\}$. Furthermore, treating A and B as two independently moving rigid objects allows us to represent the relative motion of A between t_1 and t_2 as $(\boldsymbol{R}_2^A, \boldsymbol{t}_2^A)$ and that of B as $(\boldsymbol{R}_2^B, \boldsymbol{t}_2^B)$. Now consider the following two complementary cases.

6.1.2 Alignment with known Segmentation

Suppose we have segmented the CSPs at t_j into two groups belonging to part A and part B, represented by G_j^A and G_j^B respectively for both $j = 1, 2$. By applying the temporal SFS algorithm described in Section 5.3.3 (Equation (5.11)) to A and B separately, estimates of the relative motions $(\boldsymbol{R}_2^A, \boldsymbol{t}_2^A)$, $(\boldsymbol{R}_2^B, \boldsymbol{t}_2^B)$ can be obtained.

6.1.3 Segmentation with known Alignment

Assume we are given the relative motion $(\boldsymbol{R}_2^A, \boldsymbol{t}_2^A), (\boldsymbol{R}_2^B, \boldsymbol{t}_2^B)$ of A and B from t_1 to t_2. For a CSP W_1^i at time t_1, consider the following two error measures

$$e_{2,1}^{i,A} = \frac{1}{n_1^{i,A}} \sum_k \{\tau * d_2^k(\boldsymbol{R}_2^A W_1^i + \boldsymbol{t}_2^A) + [\boldsymbol{c}_2^k(\boldsymbol{R}_2^A W_1^i + \boldsymbol{t}_2^A) - \mu_1^i]^2\}\,, \tag{6.1}$$

$$e_{2,1}^{i,B} = \frac{1}{n_1^{i,B}} \sum_k \{\tau * d_2^k(\boldsymbol{R}_2^B W_1^i + \boldsymbol{t}_2^B) + [\boldsymbol{c}_2^k(\boldsymbol{R}_2^B W_1^i + \boldsymbol{t}_2^B) - \mu_1^i]^2\}\,. \tag{6.2}$$

Here $e_{2,1}^{i,A}$ is the error of W_1^i with respect to the color/silhouette images at t_2 if it belongs to part A. Similarly $e_{2,1}^{i,B}$ is the error if W_1^i lies on the surface of B. In these expressions the summations are over those cameras where the transformed point is *visible* and $n_1^{i,A}$ and $n_1^{i,B}$ represent the number of such visible cameras of the transformed points $\boldsymbol{R}_2^A W_1^i + \boldsymbol{t}_2^A$ and $\boldsymbol{R}_2^B W_1^i + \boldsymbol{t}_2^B$ respectively. By comparing the two errors in Equations (6.2) and (6.1), a simple strategy to classify the point W_1^i is devised as follows:

$$W_1^i \in \begin{cases} G_1^A & \text{if} \quad e_{2,1}^{i,A} < \kappa * e_{2,1}^{i,B} \\ G_1^B & \text{if} \quad e_{2,1}^{i,B} < \kappa * e_{2,1}^{i,A} \\ G_1^\emptyset & \text{otherwise} \end{cases}, \tag{6.3}$$

where $0 \leq \kappa \leq 1$ is a thresholding constant and $G_1^\emptyset$ contains all the CSPs which are classified as neither belonging to part A nor part B. Similarly, the CSPs at time t_2 can be classified using the errors $e_{1,2}^{i,A}$ and $e_{1,2}^{i,B}$.

In practice, the above decision rule does not work very well because of image/silhouette noise and camera calibration errors. Here we suggest using spatial coherency and temporal consistency to improve the segmentation. To use spatial coherency, the notion of a spatial neighborhood has to be defined. Since it is difficult to define a spatial neighborhood for the scattered CSPs in 3D space (see for example Figure 5.7), an alternate way is used. Recall (in Section 4.1) that each CSP W_1^i lies on a Bounding Edge which in turn corresponds

to a boundary point u_1^i of the silhouette image S_1^k. We define two CSPs W_1^i and W_1^{i+1} as "neighbors" if their corresponding 2D boundary points u_1^i and u_1^{i+1} are neighboring pixels (in 8-connectivity sense) in the same silhouette image. This neighborhood definition allows us to easily apply spatial coherency to the CSPs. From Figure 6.2 it can be seen that different parts of an articulated object *usually* project onto the silhouette image as continuous outlines. Inspired by this property, the following spatial coherency rule (SCR) is proposed:

Spatial Coherency Rule (SCR):
If W_1^i is classified as belonging to part A by Equation (6.3), it stays as belonging to part A if all of its m left and right immediate "neighbors" are also classified as belonging to part A by Equation (6.3), otherwise it is reclassified as belonging to $G_1^{\emptyset}$, the group of CSPs that belongs to neither part A nor part B. The same procedure applies to part B.

Figure 6.2 shows how the spatial coherency rule can be used to remove spurious segmentation errors. The second constraint we utilize to improve the segmentation results is temporal consistency as illustrated in Figure 6.3. Consider three successive frames captured at t_{j-1}, t_j and t_{j+1}. For a CSP W_j^i, it has two classifications due to the motion from t_{j-1} to t_j and the motion from t_j to t_{j+1}. Since W_j^i either belongs to part A or B, the temporal consistency rule (TCR) simply requires that the two classifications have to agree with each other:

Temporal Consistency Rule (TCR):
If W_j^i has the same classification by SCR from t_{j-1} to t_j and from t_j to t_{j+1}, the classification is maintained, otherwise, it is reclassified as belonging to $G_j^{\emptyset}$, the group of CSPs that belongs to neither part A nor part B.

Note that SCR and TCR not only remove wrongly segmented points, but they also remove some of the correctly classified CSPs. Overall though they are effective because less but more accurate data is preferred to abundant but less accurate data, especially in our case where the segmentation has a great effect on the motion estimation.

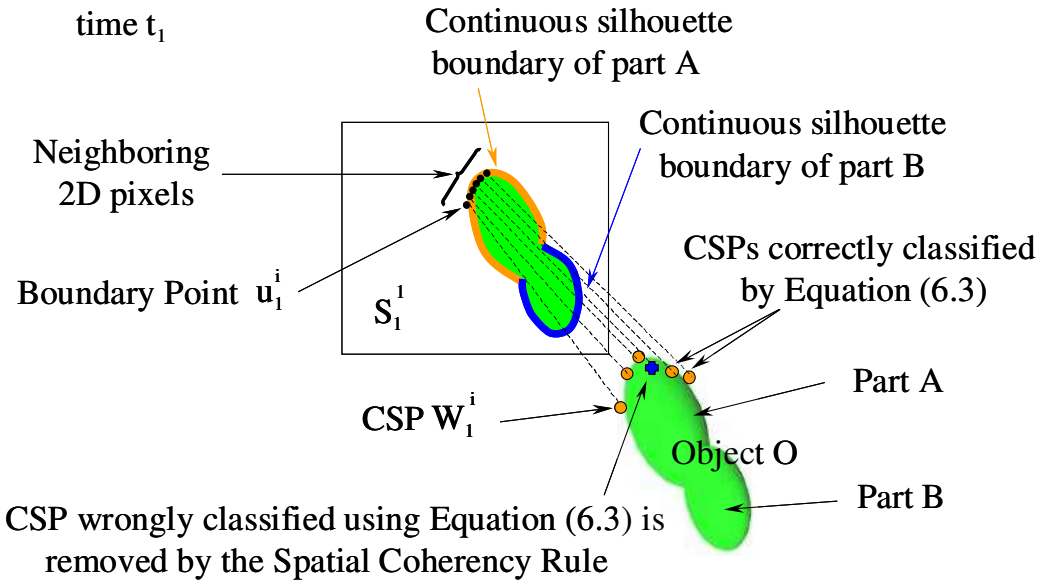


Figure 6.2: Spatial Coherency Rule removes spurious segmentation errors.

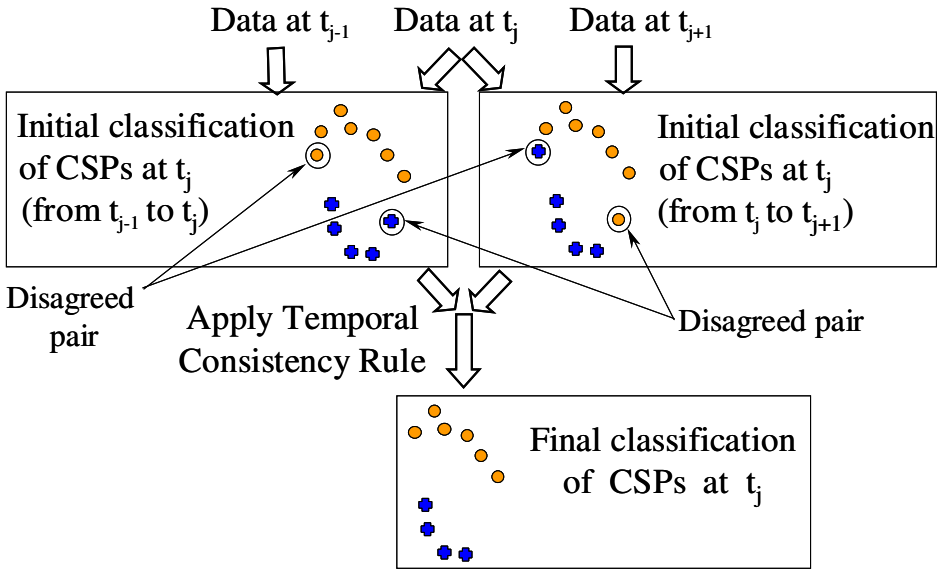


Figure 6.3: Temporal consistency ensures segmentation agrees between successive frames.

6.1.4 Initialization

As common to all iterative EM algorithms, initialization is always a problem [SA96]. Here we suggest two different approaches to start our algorithm. Both approaches are commonly used in the layer estimation literature [SA96, KK01]. The first approach uses the fact that the 6 DOF motion of each part of the articulated object represents a single point in a six dimensional space. In other words, if we have a large set of estimated motions of all the parts of the object, we can apply clustering algorithms on these estimates in the 6D space to separate the motion of each individual part. To get a set of estimated motions for all the parts, the following method can be used. The CSPs at each time instant are first divided into subgroups by cutting the corresponding silhouette boundaries into arbitrary segments. These subgroups of CSPs are then used to generate the motion estimates using the VH alignment algorithm, each time with a randomly chosen subgroup from each time instant. Since this approach requires the clustering of points in a 6D space, it performs best when the motions between different parts of the articulated object are relatively large so that the motion clusters are distinct from each other.

The second approach is applicable in situations where one part of the object is much larger than the other. Assume, say, part A is the dominant part. Since this assumption means that most of the CSPs of the object belong to A, the dominant motion $(\boldsymbol{R}^A, \boldsymbol{t}^A)$ of A can be approximated using all the CSPs. Once an approximation of $(\boldsymbol{R}^A, \boldsymbol{t}^A)$ is available, the CSPs are sorted in terms of their errors with respect to this dominant motion. An initial segmentation is then obtained by thresholding the sorted CSPs errors.

6.1.5 Summary: Iterative Algorithm

Summarizing the above discussion, we propose the following iterative segmentation/alignment process to estimate the shape and motion of parts A and B over J frames:

Iterative Temporal SFS Algorithm for Articulated Objects

1. Initialize the segmentation of the J sets of CSPs.
2. Iterate the following two steps until convergence (or for a fixed number of iterations):
   2a. Given the CSP segmentation $\{G_j^A, G_j^B\}$, recover the relative motions $(\boldsymbol{R}_j^A, \boldsymbol{t}_j^A)$ and $(\boldsymbol{R}_j^B, \boldsymbol{t}_j^B)$ of A and B over all frames $j = 2, \ldots J$ using the rigid object temporal SFS algorithm described in Section 5.3.3.
   2b. Repartition the CSPs according to the estimated motions by applying Equation (6.3), followed by the intra-frame SCR and inter-frame TCR.

Although we have described this algorithm for an articulated object with two rigid parts, it can easily be generalized to apply to objects with N parts.

6.1.6 Joint Location Estimation

After recovering the motions of parts A and B separately, the point of articulation between them is estimated. Suppose we represent the joint position at time t_1 as Y_1^B. Since Y_1^B lies on both A and B, it must satisfy the motion equation from t_1 to t_2 as follows

$$\boldsymbol{R}_2^A Y_1^B + \boldsymbol{t}_2^A = \boldsymbol{R}_2^B Y_1^B + \boldsymbol{t}_2^B \,. \tag{6.4}$$

Putting together similar equations for Y_1^B over J frames, we get

$$\boldsymbol{M} Y_1^B = \begin{bmatrix} \boldsymbol{R}_2^A - \boldsymbol{R}_2^B \\ \vdots \\ \boldsymbol{R}_J^A - \boldsymbol{R}_J^B \end{bmatrix} Y_1^B = \begin{bmatrix} \boldsymbol{t}_2^B - \boldsymbol{t}_2^A \\ \vdots \\ \boldsymbol{t}_J^B - \boldsymbol{t}_J^A \end{bmatrix} . \tag{6.5}$$

The least squares solution of Equation (6.5) can be computed using Singular Value Decomposition.

Note that the matrix $\boldsymbol{M}$ in Equation (6.5) is singular if the degree-of-freedom of the relative motion between A and B over the J frames is less than 3. This happens when the joint is a 1D revolute joint (or if A and B move with respect to each other as a 1D revolute joint during the J frames of motion). In this singular case the solution of Equation (6.5) is an arbitrary point on the axis of the revolute joint. To recover the actual joint position, we approximate the shape of parts A and B as ellipsoids and enforce the solution of Equation (6.5) to be closest to the tips of the approximated ellipsoids. It can be seen in Section 6.2.2 that this remedy works well in practice when we estimate the locations of the elbow and knee joints which are essentially 1D revolute joints.

6.1.7 Shape Refinement

The shape of the articulated object is refined in the same fashion as discussed in Section 5.5. The different parts of the articulated object are refined separately. For example for part A, the silhouette images captured at time t_j are considered as captured at time t_1 after the camera centers at t_j are transformed by the inverse motion of part A at t_j (w.r.t. t_1), similar to that as shown in Figure 5.12. Notice when refining the shape of a particular part, say part A, there is no need to segment out the part of silhouettes which are casted by part A (which is difficult to do due to occlusion) as long as the motions of that part is significantly different from each other over a period of the captured sequence. It is because voxels that do not belong to part A would be carved away by SFS over time as they do not follow the motion of part A.

6.2 Experimental Results

To validate our temporal SFS algorithm for articulated objects, both synthetic (for quantitative evaluation) and real data (for qualitative results) is used.

6.2.1 Synthetic Data Set

We use an articulated mesh model of a virtual computer human body as the synthetic test subject. To generate a set of test sequences, the computer human model is programmed to move one particular joint of the whole body and images of the movements are rendered using OpenGL. Since only one joint (and one body part) is moved each time, we can consider the virtual human body as an one-link two part articulated object. A total of eight sets of data sequences (each set with 8 cameras) are generated, corresponding to the eight joints: left/right shoulder/elbow/hip/knee joints of the virtual human model. For each of these synthetic sequences, we apply the articulated object temporal SFS algorithm to recover the shape, motion and the location of that joint of the virtual human. Since the size of the whole body is much larger than a single body part, the dominant motion initialization method is used. Figure 6.4 shows some frames of one of the input camera images and the segmentation/alignment/joint estimation results of the right elbow and the right hip joints of the synthetic sequences. As evident from the results, our iterative segmentation/alignment algorithm performs well and the joint positions are estimated accurately in both cases. Table 6.1 compares the ground-truth and the estimated joint positions of all the 8 synthetic sequences. The absolute distance errors between the ground-truth and the estimated joints locations are small (averaged about 26mm) as compared to the size of the human model ($\approx$ 500mm x 200mm x 1750mm). The input images, CSPs and the results for the left hip/knee joints of the synthetic data set can be seen in the movie **Synthetic-joints-leftleg.mpg**[*].

[*]All movie clips of this chapter can be found at http://www.cs.cmu.edu/~german/research/Thesis/Video/Chapter6/

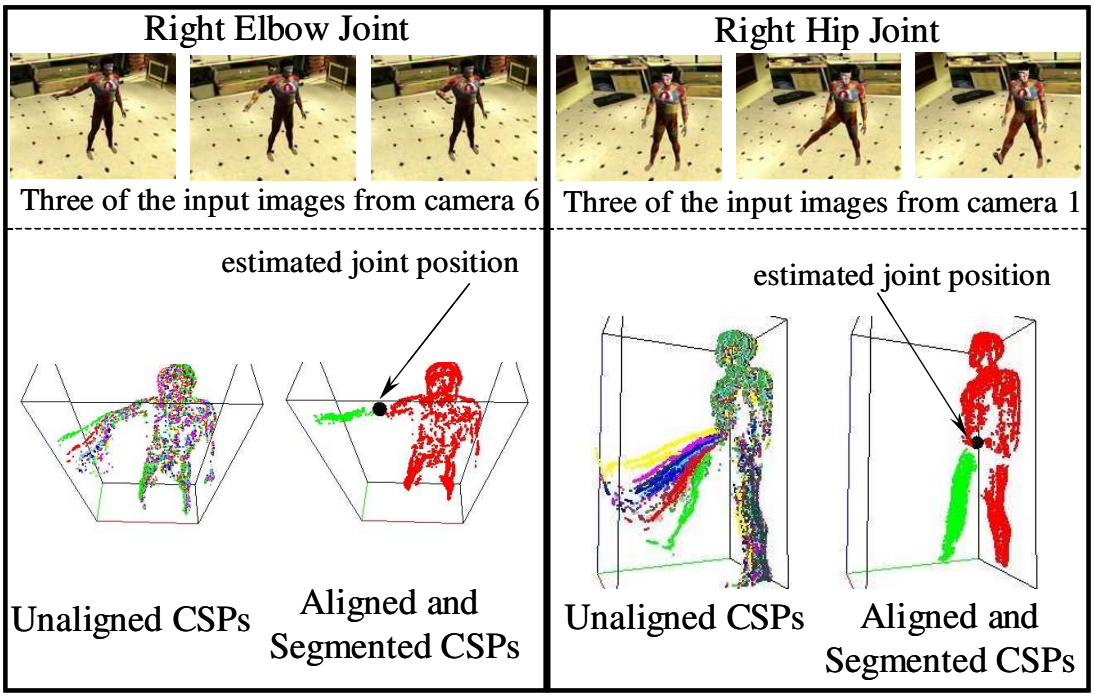


Figure 6.4: Input images and results for the right elbow and right hip joints of the synthetic virtual human. For each joint, the unaligned CSPs from different frames are drawn with different colors. The aligned and segmented CSPs are shown with two different colors to show the segmentation. The estimated articulation point (joint location) is indicated by the black sphere.

Joints	Ground-truth (x, y, z) positions (in mm)	Estimated (x, y, z) positions (in mm)	Distance error (in mm)
Left Shoulder	(199.61, 66.06, 1404.75)	(203.40, 54.06, 1403.80)	12.62
Right Shoulder	(-200.34, 66.06, 1404.75)	(-206.09, 73.87, 1398.53)	11.52
Left Elbow	(411.75, -116.60, 1333.54)	(412.98, -119.61, 1323.23)	10.81
Right Elbow	(-407.00, 146.01, 1258.53)	(-398.89, 178.54, 1288.19)	44.76
Left Hip	(87.02, 43.32, 974.75)	(92.16, 40.46, 976.77)	6.22
Right Hip	(-91.65, 42.37, 979.51)	(-85.20, -2.13, 965.11)	47.21
Left Knee	(251.57, -438.03, 853.29)	(285.14, -432.44, 857.50)	34.29
Right Knee	(-143.90, -399.59, 723.32)	(-102.92, -393.13, 741.42)	45.27

Table 6.1: The ground-truth and estimated positions of the eight body joints of the synthetic sequences. The absolute distance errors (averaged about 26mm) is small compared to the actual size of the human model ($\approx$ 500mm x 200mm x 1750mm).

Input images of the Pooh-Dinosaur sequence from Camera 3

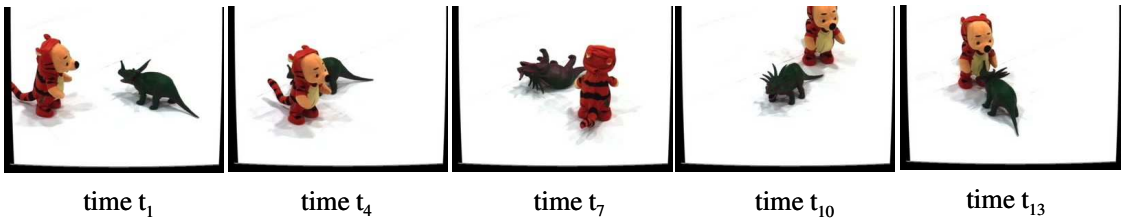


Figure 6.5: Some of the input images of camera 3 and camera 6 of the Pooh-Dinosaur sequence.

6.2.2 Real Data Sets

Two different data sets with real objects are captured. The first real data set applies the iterative segmentation/estimation procedure to two separate, independently moving rigid objects while the second real data set investigates the performance of our articulated object temporal SFS algorithm on joint estimations of two different humans.

6.2.2.1 Two Separate Moving Rigid Objects: Pooh-Dinosaur Sequence

The toy Pooh and toy dinosaur from Section 5.6.2 are used to test the performance of our iterative CSPs segmentation/motion estimation algorithm on two separate and independently moving rigid objects. Eight calibrated cameras ($K = 8$) are used in this Pooh-Dinosaur sequence. Both toys are placed on the floor and individually moved to new but unknown positions and orientations manually in each frame. Fourteen frames are captured and Figure 6.5 shows some of the input images from cameras 3 and 6. The segmentation/alignment results using our temporal SFS algorithm are illustrated in Figure 6.6. Figures 6.6(a) shows the unaligned CSPs of all the 14 frames (in the right part of the picture the CSPs are drawn with colors representing which frame they come from while in the left the CSPs are drawn with their own colors). Figures 6.6(b) shows the aligned and segmented CSPs. The fig-

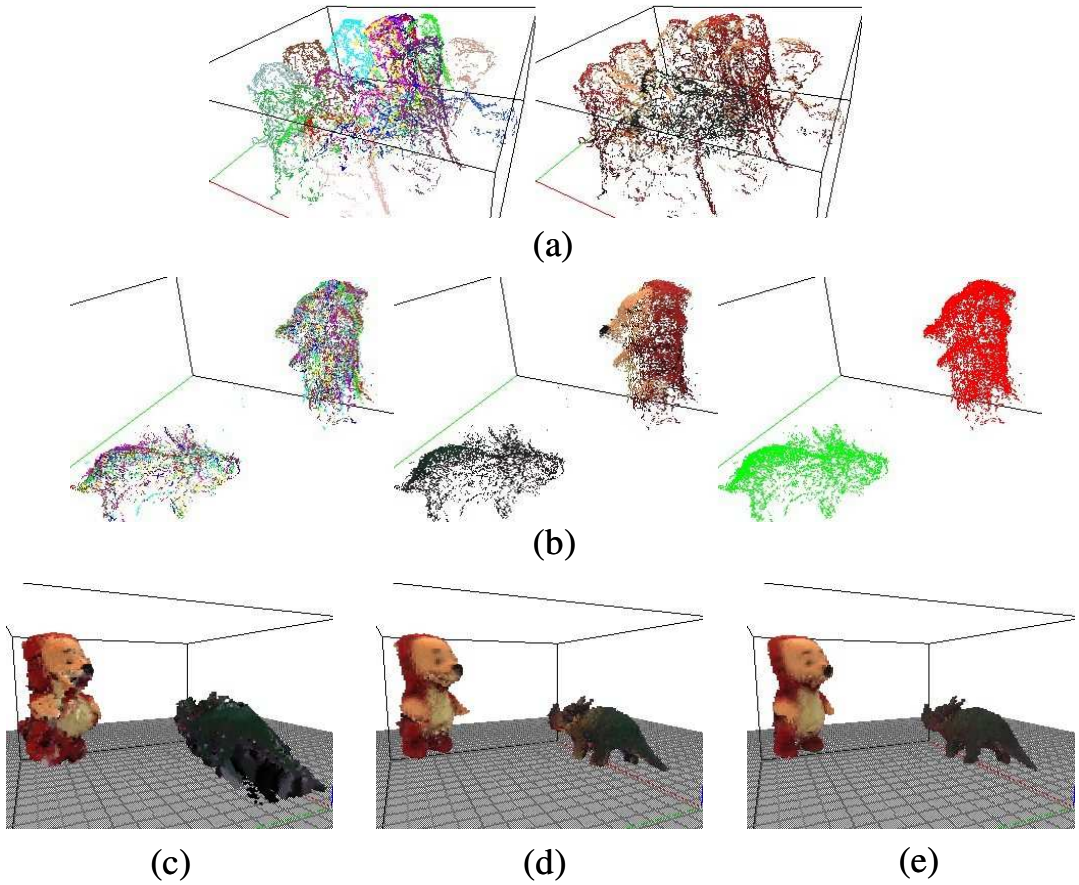


Figure 6.6: Segmentation/Alignment/Refinement results of the Pooh-Dinosaur sequence. (a) The unaligned CSPs from all frames. (b) The aligned and segmented CSPs. (c) SFS refined voxel models at t_1 (8 silhouette images are used). (d) SFS refined voxel models at t_5 (40 silhouette images are used). (e). SFS refined voxel models at t_{13} (104 silhouettes are used for the toy Pooh and 72 silhouette images are used for the dinosaur).

ures prove that our algorithm correctly segments the CSPs as belonging to each object. The alignments of both toys are also accurate except those of the dinosaur from frame 6 to frame 9 when the dinosaur rolled over for 360 degrees. In those frames, our alignment algorithm breaks down as the rotation angles between frames are too large (around 90 degrees). However, the alignment recovers after frame 9 when the dinosaur is upright again.

The shapes of the two toys are refined by SFS using the estimated motions in the same fashion as discussed in Section 5.5 (see Figure 5.12). Note that to refine the objects, there is no need to segment (which is difficult to do due to occlusion) the silhouettes as belonging

to which object as long as the motions of the objects are significantly different from each other for at least one frame. It is because voxels that do not belong to say the dinosaur would be carved away by SFS over time as they do not follow the motion of the dinosaur. Figures 6.6(c),(d) and (e) illustrate the SFS refined voxel models of both objects at t_1, t_5 and t_{13} respectively. Since the alignment data for the dinosaur from frame 6 to frame 9 are inaccurate, those frames are not used to refine the shape of the dinosaur. It can be seen that significant shape improvement is obtained from t_1 to t_{13}. The video clip **Pooh-Dinosaur.mpg** shows the input images from one of the eight cameras, the unaligned/aligned/segmented CSPs and the temporal refinement results.

6.2.2.2 Joints of Real Human

In the second set of real data, we use videos of two people (SubjectE and SubjectG) to qualitatively test the performance of our articulated object temporal SFS algorithm on joint location estimation. Eight sequences (each with 8 cameras) corresponding to the movement of the left/right shoulder/elbow/hip/knee joints of each person are captured. In each sequence, the person only moves one of their joints so that in that sequence their body can be considered as an one-joint, two parts articulated object, exactly as the synthetic data set. Again, the dominant motion initialization method is used. Some of the input images and the results of segmentation/alignment/position estimation of two selected joints (left elbow and left hip) of SubjectE are shown in Figure 6.7. It can be seen that the motion, the segmentation of the body parts and the joint locations are all estimated correctly in both sequences. Similarly the results of the left shoulder and left knee joints for SubjectG are shown in Figure 6.8. Some of the input images, the CSPs and the segmentation/estimation results of the right arm joints of SubjectE and right leg joints of SubjectG can be found in the movie clips **SubjectE-joints-rightarm.mpg** and **SubjectG-joints-rightleg.mpg**. Note that the joint estimation results of another person SubjectS can be found in the next chapter when we discuss human body kinematic modeling.

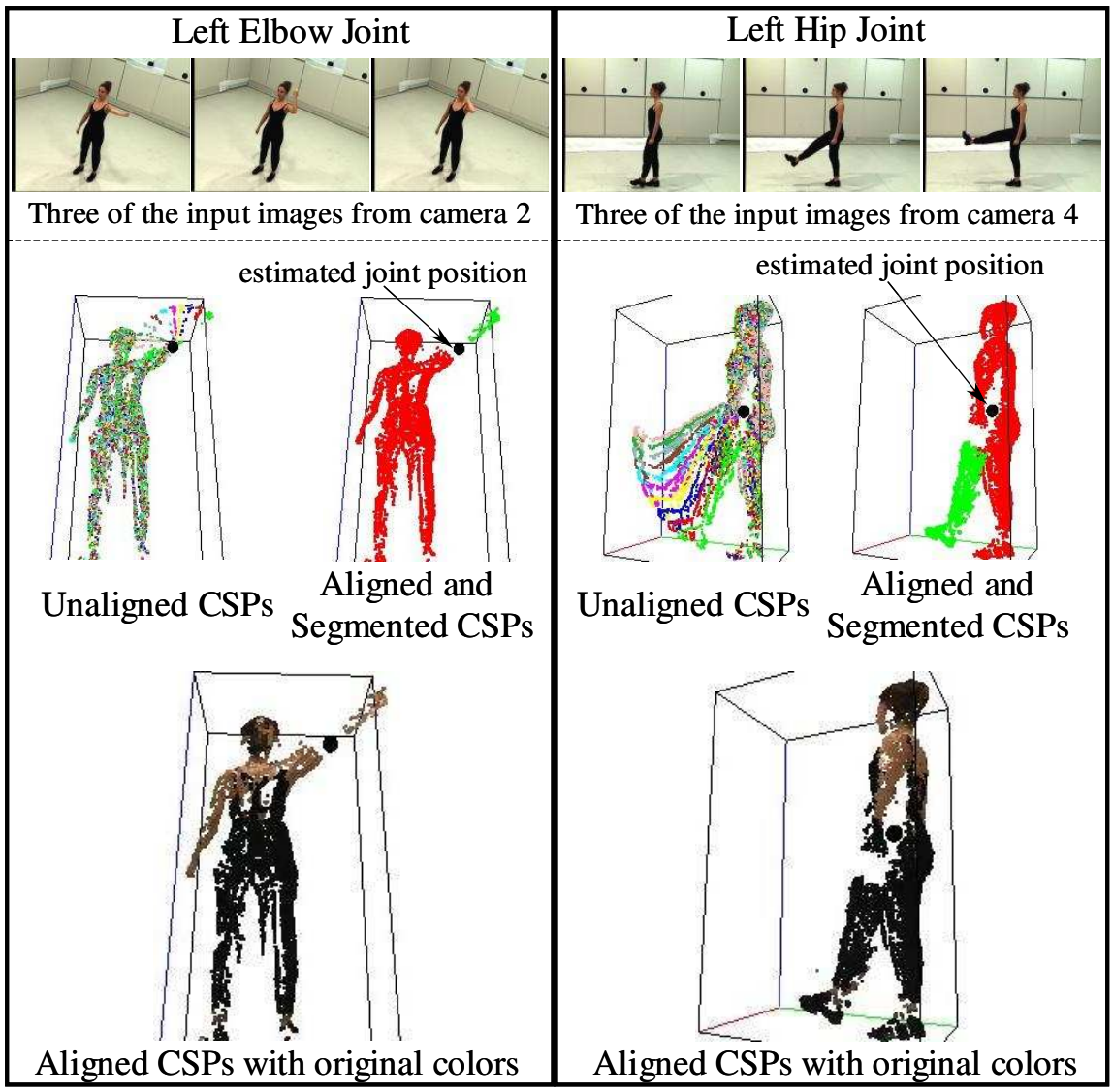


Figure 6.7: Input images and results for the left elbow and left hip joints of SubjectE. For each joint, the unaligned CSPs from different frames are drawn with different colors. The aligned and segmented CSPs are shown with two different colors to show the segmentation. The estimated articulation point (joint location) is indicated by the black sphere. The aligned CSPs with the original colors are also shown at the bottom of the figure.

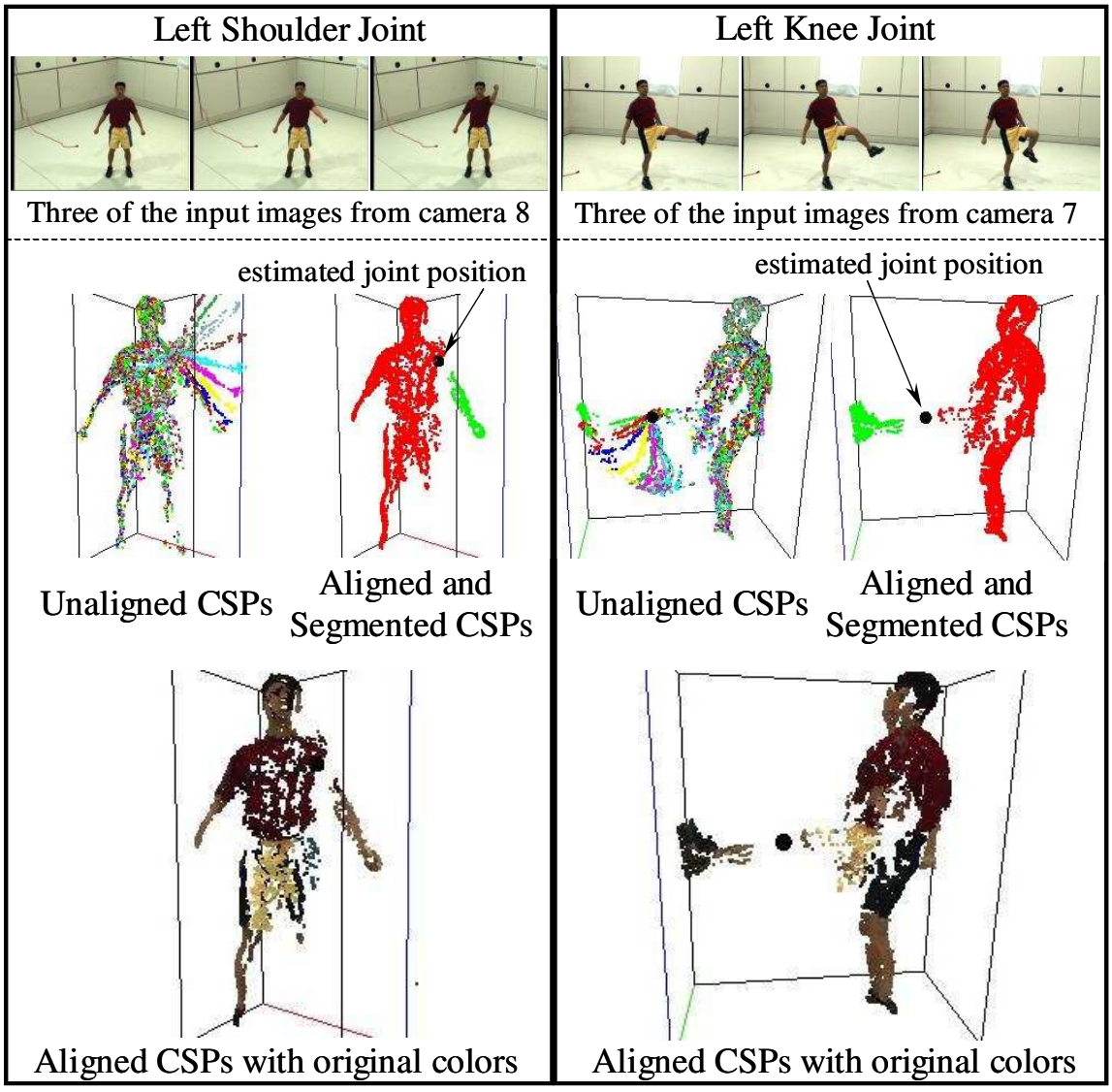


Figure 6.8: Input images and results for the left shoulder and left knee joints of SubjectG. For each joint, the unaligned CSPs from different frames are drawn with different colors. The aligned and segmented CSPs are shown with two different colors to show the segmentation. The estimated articulation point (joint location) is indicated by the black sphere. The aligned CSPs with the original colors are also shown at the bottom of the figure.

6.3 Related Work

Though the work by Krahnstoever in [KYS01, KYS03] uses only monocular images, their idea is very similar to ours in the sense that it is also based on the the layered motion segmentation/estimation formulation [SA96]. They first perform an EM-like segmentation/motion estimation of 2D regions on monocular images of the articulated object and then model the articulated parts by 2D cardboard models. As common to other monocular methods, their approach does not handle occlusion and has difficulties estimating the motion of objects which do not contain rotation around an axis perpendicular to the image plane.

6.4 Discussion

We have extended the SFS across time algorithm to (piecewise rigid) articulated objects and successfully applied it to solve the problem of human body joint position estimation. The advantage of our algorithm is that it solves the difficult problem of shape/motion/joint estimation of a moving articulated object by a two-step approach: first iteratively recover the shape (in terms of CSP) and the motion of the individual parts of the articulated object and then locate the joint through a simple motion constraint. The separation of the joint estimation and the motion estimation greatly reduces the complexity of the problem. Since our algorithm uses motion to segment the CSPs, it fails when the relative motion between the parts of the articulated objects is too small. Moreover, due to the EM formulation of the algorithm, the convergence of the algorithm depends on the initial estimates of the motion parameters. When the initial motion estimates are far from the correct values, the algorithm may fall into a local minimum.

Chapter 7

Human Kinematic Modeling

In this chapter we apply our temporal SFS algorithms (both rigid and articulated objects) to build a vision-based 3D human kinematic modeling system. Modeling human kinematic is an important application because precise 3D kinematic models are essential for solving a variety of difficult problems such as pose estimation, motion tracking/capture, gesture recognition, behavior understanding and motion rendering (see Chapter 9). Although there are a variety of complete systems [CYB, TTI] and algorithms [ACP03] for human body shape acquisition using laser-scanning devices, most of these systems are expensive and do not estimate the important joint information. Various vision-based human modeling systems have been proposed [LY95, KMB94, JBY96, KM98, PFD99, BK00, OBBH00, CKBH00, KYS01] in recent years. Among these systems, most of them use monocular images and reconstruct view-dependent 2D shape and joint models [KMB94, BK00, KYS01]. For those systems which uses multiple cameras, either imprecise shape [CKBH00] or incomplete joint information [KM98, PFD99] are recovered. Moreover, none of these systems have derived algorithms for building a complete 3D skeleton of humans. In view of this, the multi-camera modeling system described in this chapter aims at both acquiring precise 3D shapes of the body parts and constructing a full skeletal structure of the person.

There are three tasks to our vision-based human kinematic modeling: (1) constructing

a joint skeleton of the person, (2) acquiring detailed shape information and (3) merging the shape/joint information to build a kinematic model. Each task in our system is described in details below, together with the results of applying the system to three people: SubjectE, SubjectG and SubjectS.

7.1 Joint Skeleton Acquisition

The first task in our kinematic modeling system is to locate the joint positions of the person using the articulated object temporal SFS algorithm proposed in Chapter 6. Once the joint locations are recovered, they are aligned and registered with each other to form a complete joint skeleton of the person.

7.1.1 Estimating Individual Joint Positions

Although we can estimate all the joint positions of a person at the same time, in practice this approach suffers from the problem of falling into local minimum due to the high dimensionality. Instead we take a sequential approach and model the joints one at a time, just as what we have done in Sections 6.2.1 with the synthetic human and 6.2.2 with SubjectE and SubjectG. Again eight joint locations: left/right shoulder/elbow/hip/knee are recovered for each person. Some of the input images and the estimation results of the right shoulder and knee joints of SubjectS are shown in Figure 7.1 (the results for SubjectE and SubjectG have already been shown in Figures 6.7 and 6.8 in Chapter 6). The movie clip **SubjectS-joints-leftarm.mpg**[*] shows some of the input images, CSPs and the estimation of the left arm joints of SubjectS.

[*]All movie clips of this chapter can be found at http://www.cs.cmu.edu/~german/research/Thesis/Video/Chapter7/

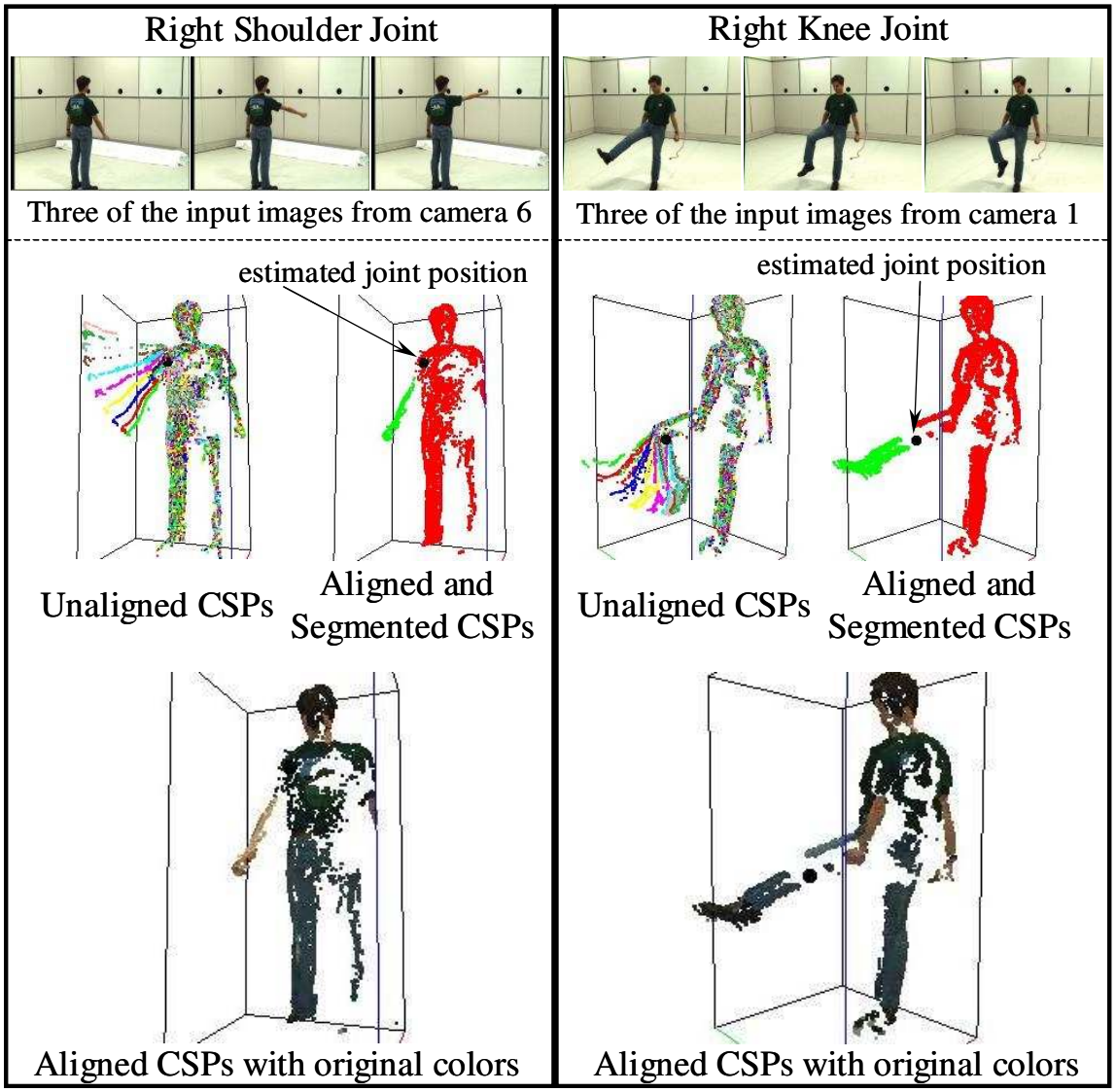


Figure 7.1: Input images and results for the right shoulder and right knee joints of SubjectS. For each joint, the unaligned CSPs from different frames are drawn with different colors. The aligned and segmented CSPs are shown with two different colors to show the segmentation. The estimated articulation point (joint location) is indicated by the black sphere. The aligned CSPs with the original colors are also shown at the bottom of the figure.

Generally, the estimation of the (shoulder and elbow) joints of the arms are more accurate than the (hip and knee) joints of the legs because it is more difficult to keep the rest of the body still when moving the leg than moving the arm. Moreover the shoulder and hip joints have better results than the elbow and knee joints as more CSPs are extracted from the whole arm (or leg) than just the lower arm (or lower leg).

7.1.2 Joint Registration

After the joints and the associated body parts (described by CSPs) are recovered individually, they are registered with respect to a reference frame (of images) to form an articulated model of the body. The registration process consists of two procedures. The first procedure involves aligning joints within each separate limb while the second procedure performs a global registration of all the joints and body parts with respect to the reference frame. Both procedures are explained below.

7.1.2.1 Limb Joints Alignment

Before registering all the joints to the reference frame, the two joints of each separate limb (i.e. the shoulder and elbow joints of the arm, the hip and knee joints of the leg) are first aligned with each other. The limb joints alignment procedure consists of four steps and is summarized below. The procedure is also illustrated graphically using the right arm of SubjectE as example in Figure 7.2. Though the procedure is described in terms of the arm, it applies to the legs by replacing the shoulder and elbow joints by the hip and knee joints.

The Limb Joints Alignment Procedure

1. The body CSPs from the shoulder joint sequence (Figure 7.2(a)) are aligned with respect to the first frame of the elbow sequence (Figure 7.2(b)) by the rigid body temporal SFS alignment algorithm.
2. Using the alignment result in Step 1, the shoulder joint location is transformed to the coordinate frame of the elbow joint sequence (Figure 7.2(c)(d)).

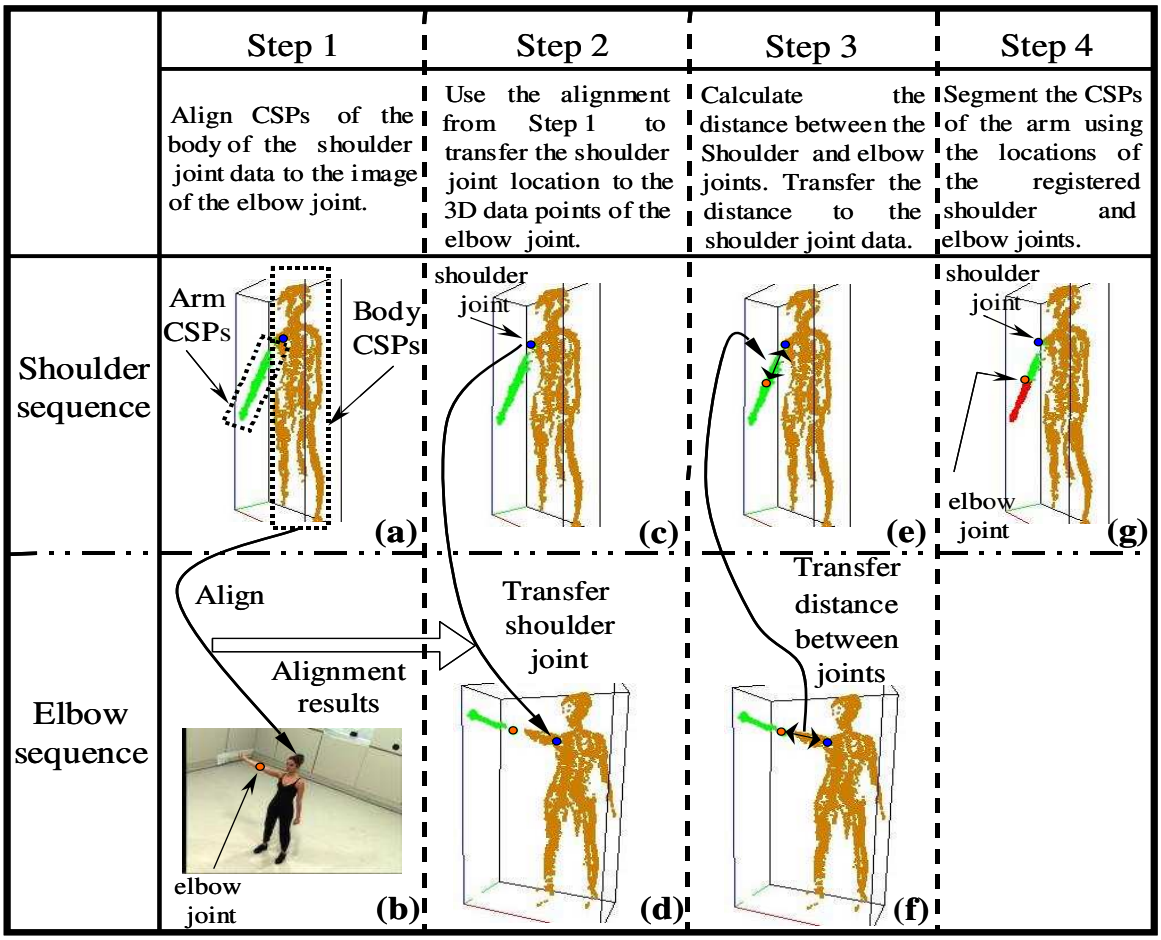


Figure 7.2: The four steps of the Limb Joints Alignment Procedure.

3. Calculate the distance between the transformed shoulder joint location and the elbow joint position. Mark the same distance from the shoulder joint along the arm as the elbow joint in the shoulder sequence (Figure 7.2(e)(f)). This step is valid as the distance between the shoulder and elbow joints are constant irrespective of their positions.

4. The arm CSPs (see Figure 7.2(a)) from the shoulder joint sequence are segmented according to the positions of the shoulder and elbow joints (the segmentation results are shown in Figure 7.2(g)).

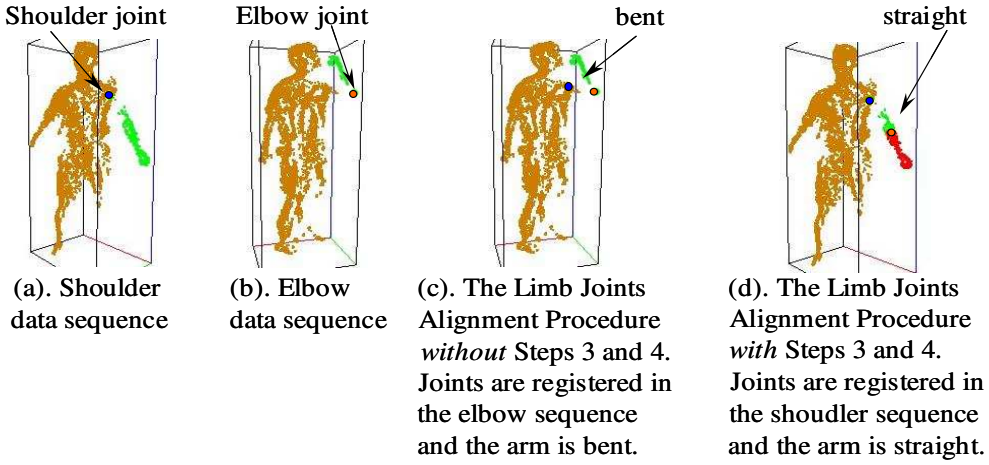


Figure 7.3: The left shoulder and elbow data sequences of SubjectG. In (c) the joints registered in the elbow sequence (without Steps 3 and 4) is bent while in (d) the joints registered w.r.t. the shoulder sequence with Steps 3 and 4 is straight.

The basic idea of the procedure is to align the shoulder and elbow joints with respect to the shoulder sequence with the arm being straight. As will be seen shortly, having the joints registered with the arm being straight greatly reduces the complexity of the subsequent global registration procedure. There are a couple of important things to be noted about the limb joints alignment procedure. Firstly when using Equation (5.11) of the rigid object temporal algorithm in Step 1, only the forward error term $e^i_{2,1}$ is present. This is because we are only using the error of projecting the body 3D CSPs of the shoulder sequence into the 2D color and silhouette images of (the first frame of) the elbow sequence. Secondly although judging from Figure 7.2(d) one may argue Steps 3 and 4 can be skipped, these two steps are necessary to ensure that the limb is straight after registration. Figure 7.3 shows an example using the left arm sequences of SubjectG. In Figure 7.3(c), the joints are registered w.r.t. the elbow sequence with the left arm in a bent state without performing Steps 3 and 4. Figure 7.3(d) shows that Steps 3 and 4 are needed to transform the registration back to the shoulder sequence with the arm in a straight state.

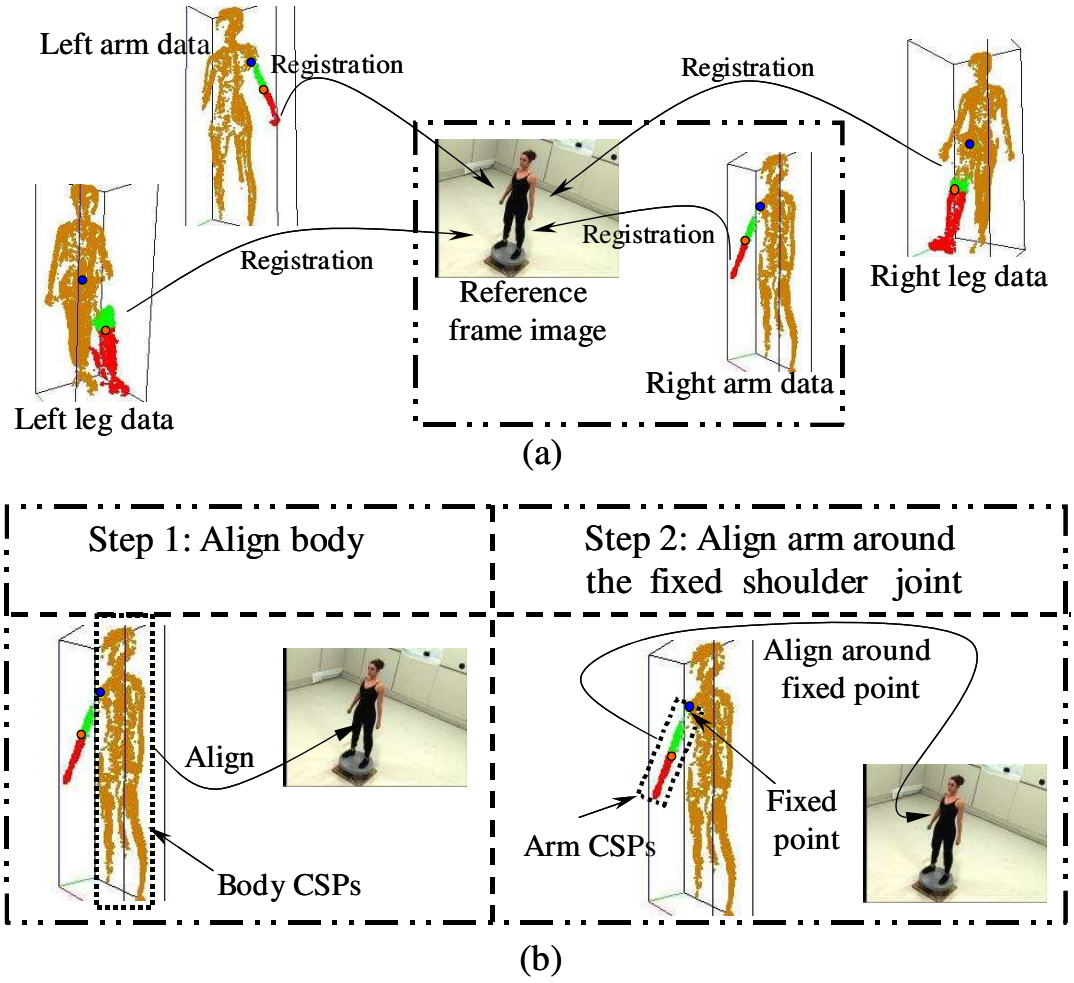


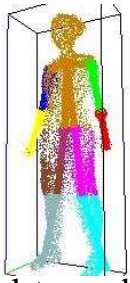
Figure 7.4: (a) Global joint registration for the four limbs. (b) For each limb, two steps are required to register the joints globally.

7.1.2.2 Global Registration

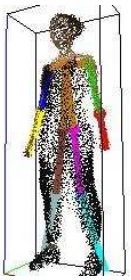
Once the joints within each limb are aligned, global registration is performed to build a joint skeleton model. The global registration for all four limbs is illustrated in Figure 7.4(a) and the procedure is explained using the right arm of SubjectE in Figure 7.4(b).

For each limb, the global registration procedure consists of two steps. The first step aligns the body CSPs against a reference frame. Once the 6D motion of the body is recovered, the position of the first limb joint (shoulder/hip) is calculated. The second step

Joint skeleton



Joint skeleton and CSPs displayed in colors of different body parts



Joint skeleton and CSPs in their original colors

Figure 7.5: Joint skeleton of SubjectE after the global registration procedure. For display clarity, the CSPs shown in the figures are down-sampled in a ratio of one in five.

involves the alignment of the limb itself. To simplify this step, we assume that the reference frame is chosen such that the images at the reference frame are captured with all of the person's limbs being straight (the choice of a good reference frame will become obvious in Section 7.2). Since the joints within each limb are already registered with the limb being straight (in the limb joint alignment procedure), the straight limb assumption of the reference frame images enables us to treat the whole limb as one rigid object rather than an articulated object with two parts. In other words, we can ignore the second limb joint (elbow/knee) and the problem becomes alignment of a rigid object around a fixed point with only 3 DOF (the rotation around the joint). For the sake of presentation, we defer discussion of the algorithm to solve this problem to Chapter 8 (see the last paragraph of Section 8.1.3). Meanwhile, Figure 7.5 illustrates the joint skeleton (formed by joining the joint locations together) of SubjectE and the registered CSPs obtained after the global registration procedure.

7.2 Body Shape Acquisition

The next task of the kinematic modeling system is to acquire the shape of the body. One direct choice is to use the CSPs extracted from the sequences used to estimate the individual

joints in 7.1.1 and registered in the global registration procedure 7.1.2 as shown in Figure 7.5. Though simple, there are two reasons why we do not use these CSPs to represent the body shape of the person. First of all, due to errors in all the alignment and registration procedures, some of these CSPs deviate considerably from the actual shape of the body parts. Secondly, these CSPs are not uniformly distributed over different body part of the person (most of these points come from the torso part of the body) which poses an disadvantage when applying the model to applications such as motion tracking and rendering.

Instead of using these erroneous and non-uniformly distributed CSPs obtained from joint estimation, an alternative approach is used. First a detailed voxel model of the person is built using the Visual Hull alignment and refinement algorithms proposed in Chapter 5. The centers of the *surface* voxels of the voxel model are then extracted and used to represent the shape of the person. There are two advantages of using this approach. Since the voxel model is reconstructed by SFS using more than 100 silhouettes, the model is very accurate and the surface voxel centers are close approximations to points on the surface of the actual person. Also since the voxel centers lie on a 3D grid, the distribution of the points is uniform.

To build voxel models of each person, video sequences of the person standing on a turn table were captured by eight cameras with thirty frames (roughly equal to a whole revolution of the turntable) per camera. Note that there is no need to calibrate the rotation axis and speed of the turn table beforehand as our rigid body temporal SFS algorithm is able to recover the motion automatically. The person is asked to remain still throughout the capture process to satisfy the rigidity assumption. Moreover, the person is also told to keep their limbs straight so that the first frame of the sequence can be chosen as the reference frame for the global body joints registration discussed in Section 7.1.2. After applying the rigid object temporal SFS algorithm to recover the motions, a refined voxel model of the person is built using the Visual Hull refinement technique as described in Section 5.5. The centers of the surface voxels of the model are extracted and colored by back-projecting

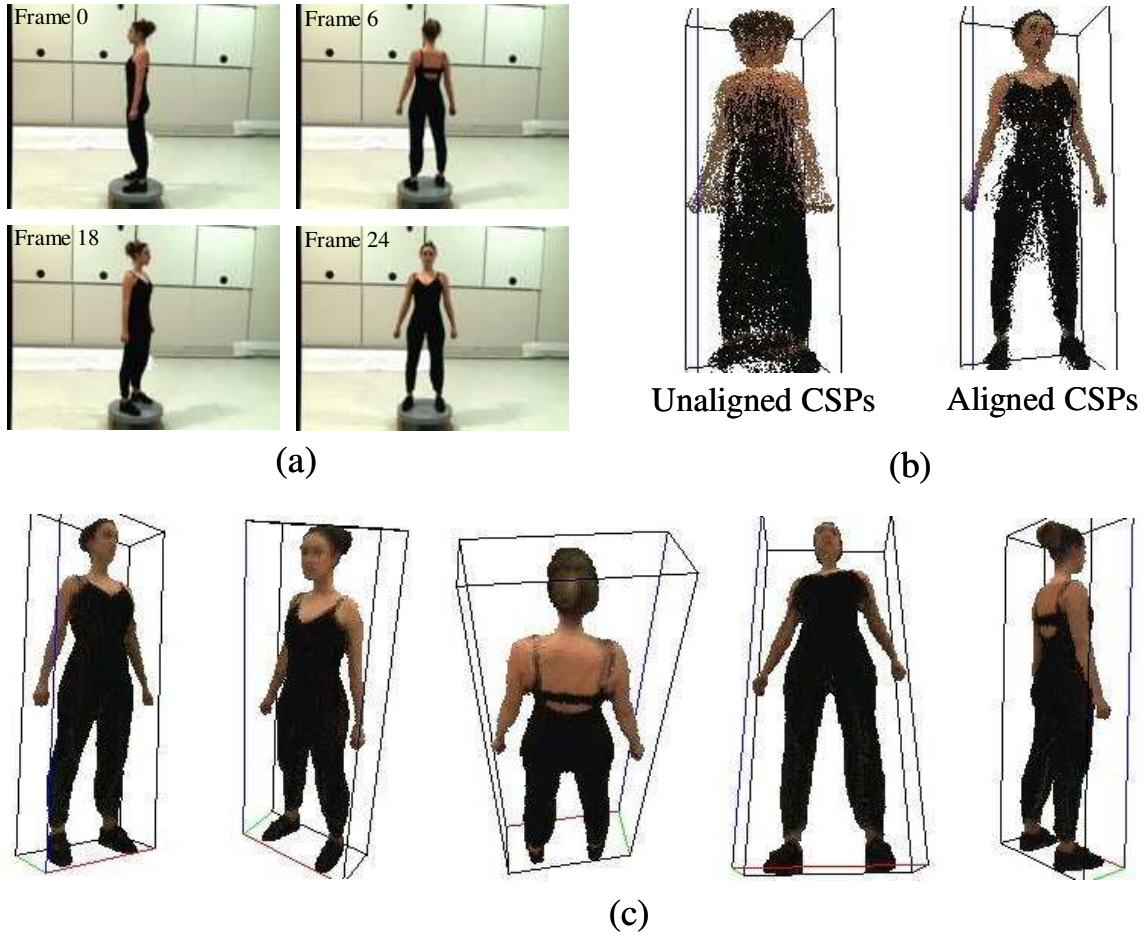


Figure 7.6: Results of body shape acquisition for SubjectE. (a) Four input images of camera 4, (b) unaligned and aligned colored surface points from all frames, (c) refined Visual Hull of the body displayed from several different view points.

them into the color images. The results on SubjectE, SubjectG and SubjectS are presented in Figures 7.6, 7.7 and 7.8 respectively. It can be seen that excellent shape estimates (see for example the Visual Hulls in Figures 7.7(c)) of the human bodies are obtained. Note that we name the input sequences in Figures 7.6 through 7.8 as the ESTILL, GSTILL and SSTILL sequences respectively. These STILL sequences will be used in Chapter 9 as source sequences to perform image-based motion rendering on SubjectE, SubjectG and SubjectS.

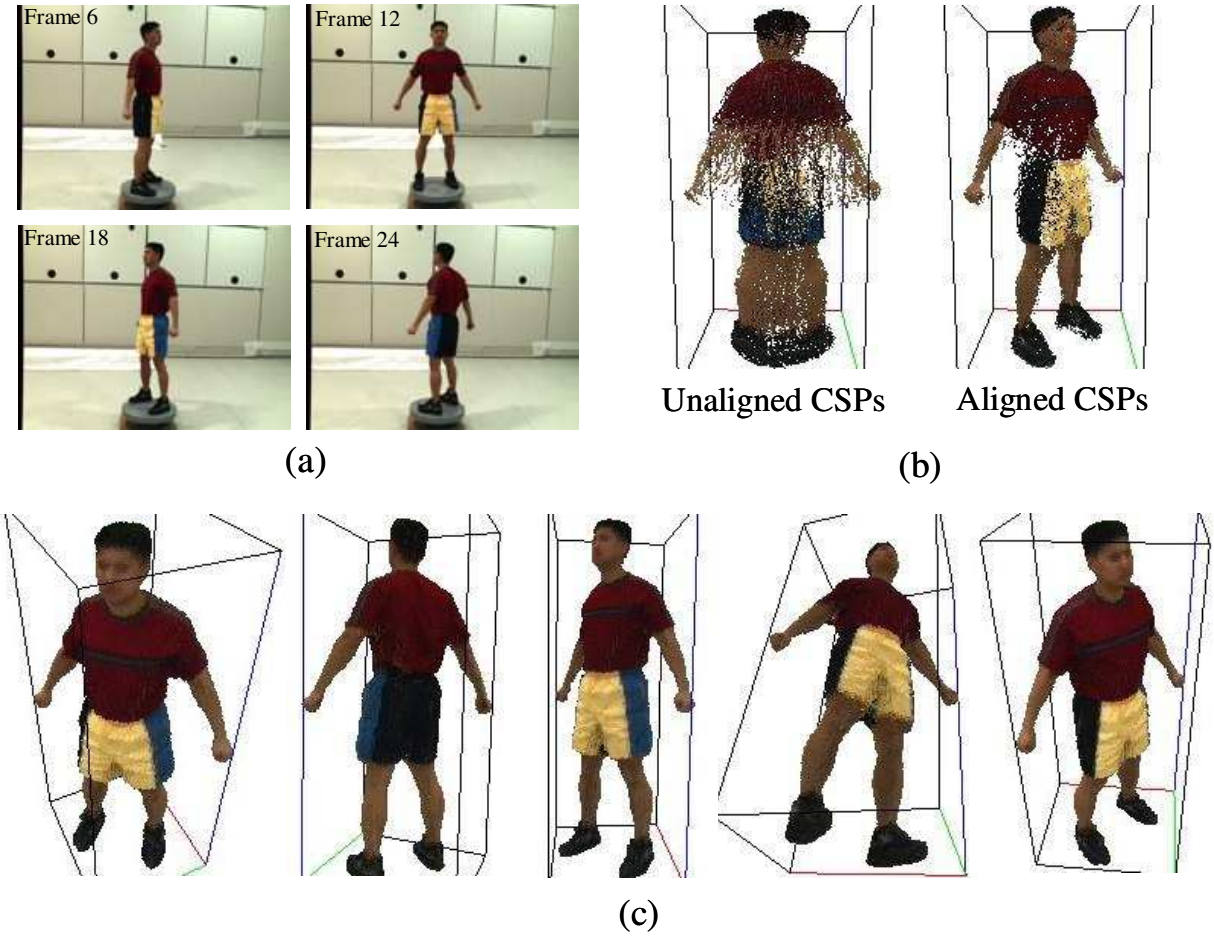


Figure 7.7: Results of body shape acquisition for SubjectG. (a) Four input images of camera 4, (b) unaligned and aligned colored surface points from all frames, (c) refined Visual Hull of the body displayed from several different view points.

7.3 Merging Shape and Joint Information

The last task of our modeling system is to merge the joint and shape information obtained from Sections 7.1 and 7.2 together. Before the merge, slight modifications are made to the joint positions to enforce left and right symmetry of the joint skeleton (the asymmetry is caused by errors in joint estimation and registration). Two rules are applied: (1) The left and right shoulder joints have the same height above the ground. The same applies to the two hip joints. (2) The distance between the shoulder and elbow joints on the left arm is equal to that on the right arm. The same applies to the distances between the hip and knee

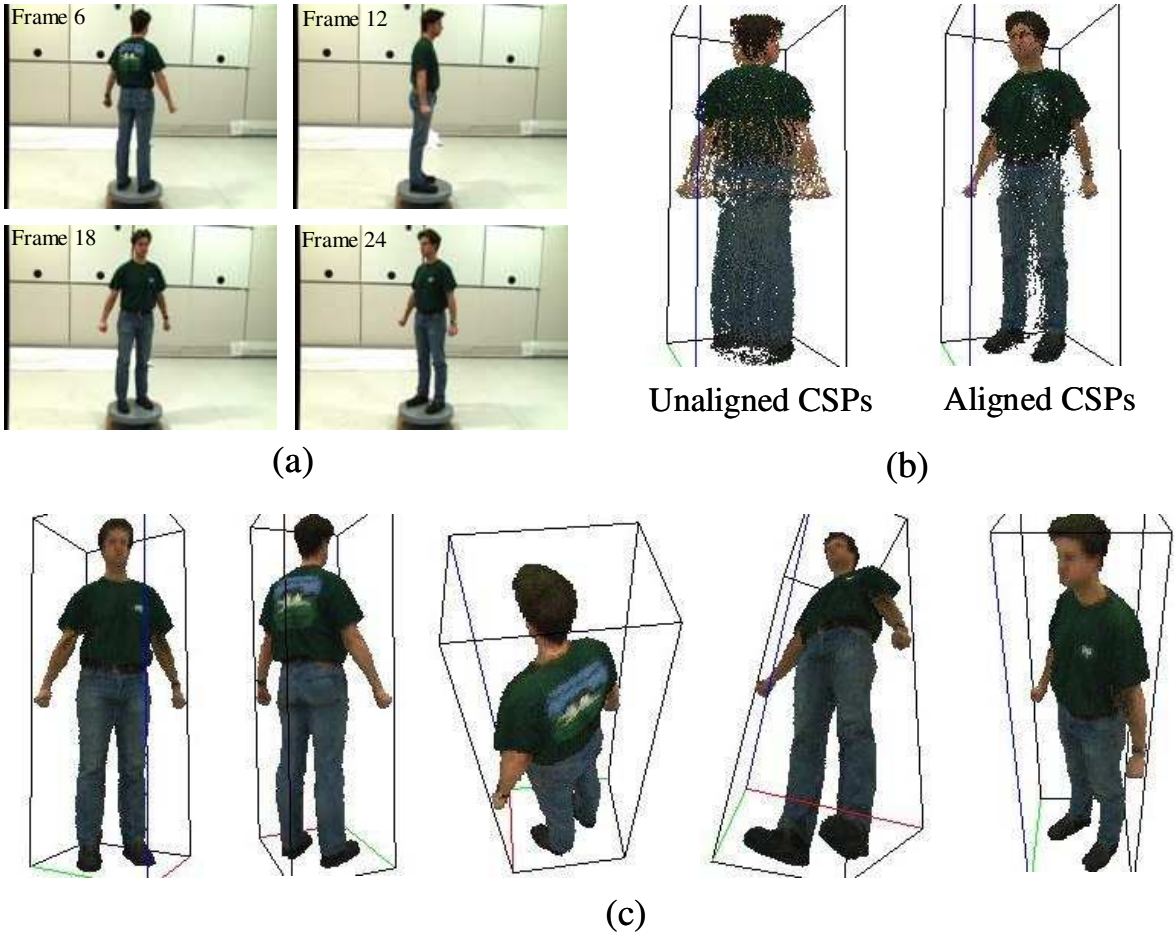


Figure 7.8: Results of body shape acquisition for SubjectS. (a) Four input images of camera 4, (b) unaligned and aligned colored surface points from all frames, (c) refined Visual Hull of the body displayed from several different view points.

joints on the legs. These two rules are reasonable because of the person's upright standing posture on the turntable when the reference frame is captured. The rules can be carried out by simply averaging the corresponding values from the left and right sides of the body. Once the joint positions are adjusted, they are transfered to the voxel model. Since the joints are registered w.r.t. the reference image used to create the voxel model, the transfer is straightforward.

The only problem left is to automatically segment the voxel centers to the corresponding body parts. Figure 7.9 illustrates an algorithm to segment the surface voxel centers based

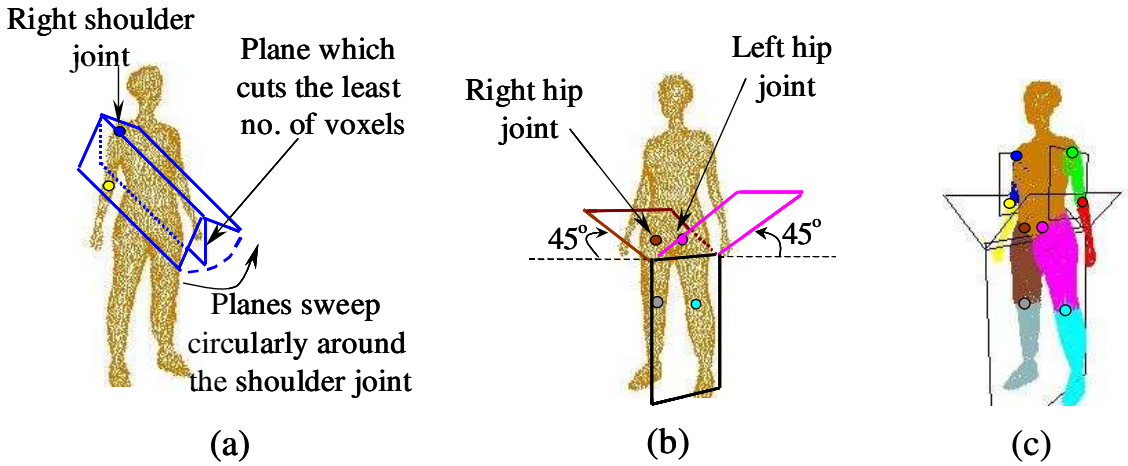


Figure 7.9: Segmenting all of the voxel centers to the appropriate body parts. (a) The arm cutting planes are found by sweeping a plane circularly around the shoulder joints. The plane which cuts the least number of voxels is chosen. (b) The leg cutting planes are formed by two planes passing through the hips joints at a 45 degree angle with the horizontal, and a vertical plane which separate the legs from each other. (c) The joints, the cutting planes and the segmented voxels of the model.

on the joint locations. First, five cutting planes are found to separate the four limbs away from the body (Figure 7.9(c)). Once the limb is segmented, it can be divided into the upper and lower parts easily using the elbow/knee joint location. The ideal cutting plane for the arm would be the one which passes through the shoulder joint and the arm pit. To find this ideal plane, planes with different orientations are sweep circularly around the shoulder joint across the body as shown in Figure 7.9(a). The plane which cuts the least number of voxels of the body model is then chosen to be the arm cutting plane. To separate the legs from the each other and from the body, three planes are used. The first plane passes through the right hip joint while the second plane passes through the left hip joint, and each of them making a 45 degree angle with the horizontal plane. The third plane is the vertical plane which make a "Y" with the first two planes as shown in Figure 7.9(b).

With a slight abuse of terminology, hereafter we treat the surface voxel centers as if they are CSPs and call the merged model an articulated CSP model of the person. The articulated CSP models of the synthetic virtual person, SubjectE, SubjectG, and SubjectS are shown in Figures 7.10(a)(b)(c) and (d) respectively. The video clip **Subject-EGS-kinematicmodels.mpg** shows some 3D fly-around views of the built models of SubjectE,

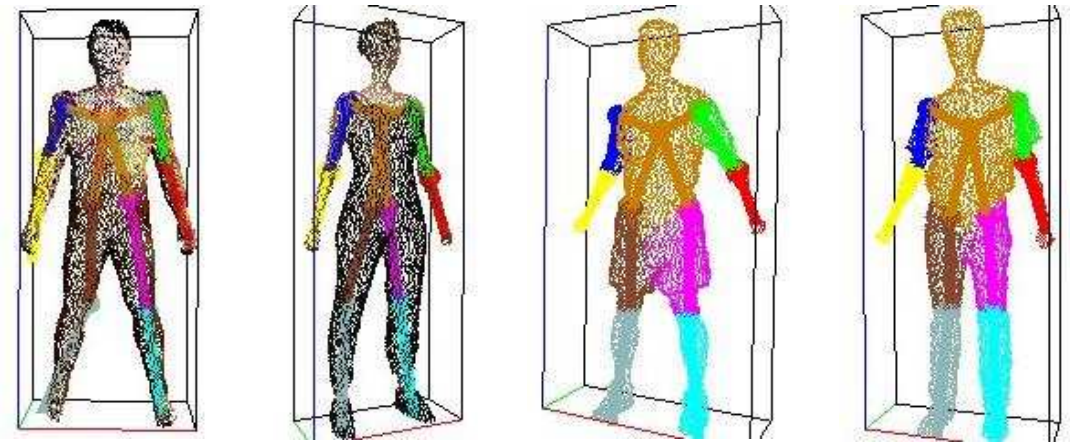


Figure 7.10: Articulated model of (a) synthetic virtual person, (b) SubjectE, (c) SubjectG and (d) SubjectS. In (a) and (b), the CSPs are shown with their original colors. In (c) and (d), the CSPs of different body parts are shown with different colors. For display clarity, the CSPs drawn are down-sampled at a ratio of one in two.

SubjectG and SubjectS. Note that the articulated CSP model can be turned into an articulated voxel model easily by substituting the center points by solid voxels (3D cubes). As will be seen in the next two chapters, an articulated voxel model is not only essential for image-based human motion rendering (Step 3 of Section 9.1.3.2), but is also useful for determining visibility (Section 8.2.3) in the problem of human motion tracking. As a summary, Figure 7.11 illustrates the three tasks of our vision-based human kinematic modeling system.

7.4 Related Work

The work most related to our vision-based human body kinematic information acquisition system is by Kakadiaris and Metaxas in [KM95]. They used deformable templates to segment the 2D body parts in a silhouette sequence. The segmented 2D shapes from three orthogonal view-points are then combined into a 3D shape by SFS. Although our idea of estimating the joint locations individually instead of all at once is partly inspired by

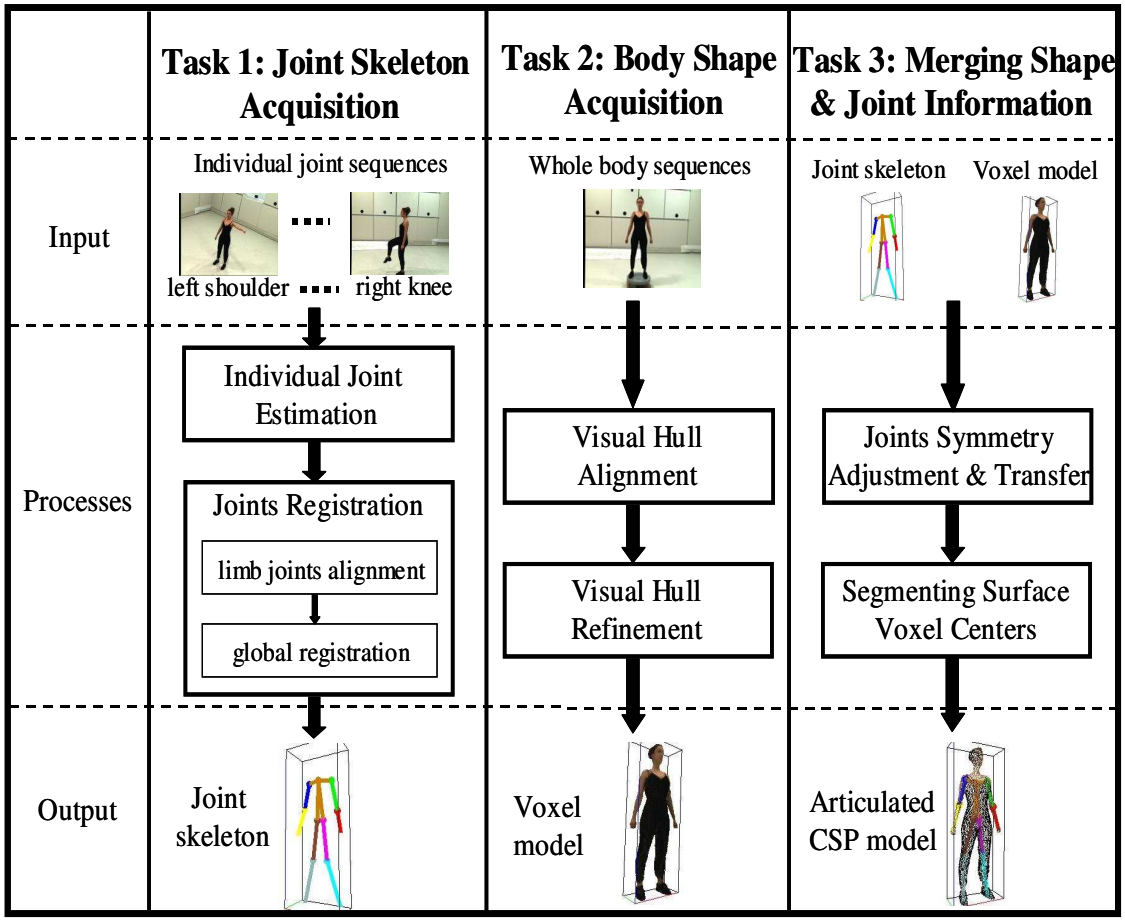


Figure 7.11: Flow chart illustrating the three tasks in our human kinematic modeling system.

their system, here we address the acquisition of motion, shape and articulation information, while [KM95] focuses mainly on shape estimation.

Besides the 2D work by Krahnstoever et al. in [KYS01, KYS03] (which we have already discussed in the last chapter), the research group led by Fua addressed the problem of 3D human body modeling using a three-camera system [PFD99, PF01, FGDP02]. They first extract dense feature points on the surface of the body parts by manual initialization and stereo matching. The feature points are then tracked across the video sequences using a template matching technique. A flexible but complex human model consisting of deformable metaballs [Bli82] as shape primitives is then used to fit the tracked feature points

through a least square framework. Though they have not demonstrated the modeling of a complete body, their approach is able to handle non-rigid deformation of the body parts. Sand et al. have also captured the non-rigid deformation of the human body skin using silhouette images [SMP03]. However, marker-based motion capture data is used in their system to estimate the joint skeleton and track the motion of the person.

7.5 Discussion

As opposed to other human modeling approaches which fit and modify generic human models composed of simple shape primitive to the input image data [LY95, KM98, PFD99, CKBH00], our vision-based kinematic modeling system constructs the body model from scratch using simple joint connection knowledge of the body. We acquire and register the skeletal structure using video sequences of the person moving their limbs and extract shape information (in terms of CSPs) of the body parts directly from the silhouette and color images. The joint and shape information is then merged to form a complete kinematic model consisting of voxels segmented into body parts using the joint locations. Compared to laser scanning human body modeling technology which usually only capture shape information, our system is simpler, cheaper, non-invasive and more importantly, provides the joint locations. However, since our system uses the motion of the body parts to recover the joint locations, it does not perform well with joints which have a restricted range of movement, such as the head, wrist and ankle joints.

Chapter 8

Human Motion Tracking

Human motion tracking and capture has long been an important research area in computer graphics for the entertainment industry of movies and games. Almost all of the state of the art motion capture systems [MET, MAC, VIC] attach optical or magnetic markers on the person whose motion is to be tracked and use triangulation on the positions of the markers to achieve tracking. Although these systems generally produce very good results, they are not very applicable to applications such as security/surveillance and human-computer interaction where placing markers on the person is either impossible or undesirable. For this reason, vision-based motion tracking has gained much attention in recent years (an extensive survey of the topic can be found in [MG01]) and researchers have proposed systems to track body parts from video sequences [RK95, GD96, BM97, BM98, HHD98, JTH99, DCR99, CR99a, CR99b, PRCM99, DF99, CKBH00, SDB00, SBF00, DBR00, DCR01, DC01, LC01, SC02, MTHC03, CTMS03] using a variety of model-based approaches. In almost all of these systems, generic shapes (e.g. rectangles/ellipses in 2D, cylinders/ellipsoids in 3D) are used to model the body parts of the person. Though generic models/shapes are simple to use and can be generalized to different persons, they suffer from two disadvantages. Firstly their coarse approximation to the actual body shape of the person limits the accuracy of motion tracking. Secondly generic shapes/models also lack

accurate joint information of the person. In vision-based motion tracking systems, precise kinematic (shape and joint) information is essential to obtain accurate motion data. In this chapter, we show how the kinematic model of a person obtained in Chapter 7 can be used to track the motion of the person in new video sequences. The formulation of our motion tracking algorithm is similar to the 3D CSPs/2D image alignment principle of the temporal SFS alignment algorithm proposed in Chapter 5, with the incorporation of joint constraints into the motion equations as described below.

8.1 Image-Based Articulated Object Tracking

In this section, we consider the problem of tracking an articulated object in (color and silhouette) video sequences of the object using a known articulated model of the object. We assume the articulated model is constructed using the human kinematic modeling system described in Chapter 7. The model consists of rigid parts with known shape described in terms of CSPs and are connected with each other at known joint locations.

8.1.1 Problem Scenario

Figure 8.1(a) depicts an articulated CSP model of an object consists of three rigid parts A, B and C with part A being the base of the object. Without loss of generality, we assume the model is at its reference configuration which means the rotation angles of the joints and the translation of the base part A are all zero. We assume the shape information of the model is given as sets of CSPs represented by $\{W_0^{i,A}, \mu_0^{i,A};\ i = 1, \cdots\cdots, L_0^A\}$, $\{W_0^{i,B}, \mu_0^{i,B};\ i = 1, \cdots\cdots, L_0^B\}$, $\{W_0^{i,C}, \mu_0^{i,C};\quad i = 1, \cdots\cdots, L_0^C\}$ for the parts A, B and C respectively and the joint locations of the model are known and denoted by Y_0^B and Y_0^C. Furthermore, we assume the model color and silhouette images $\{I_0^k, S_0^k;\ k = 1, \cdots\cdots, K\}$ that were used to construct the model are available.

Suppose we have imaged the articulated object by K cameras at each of J time in-

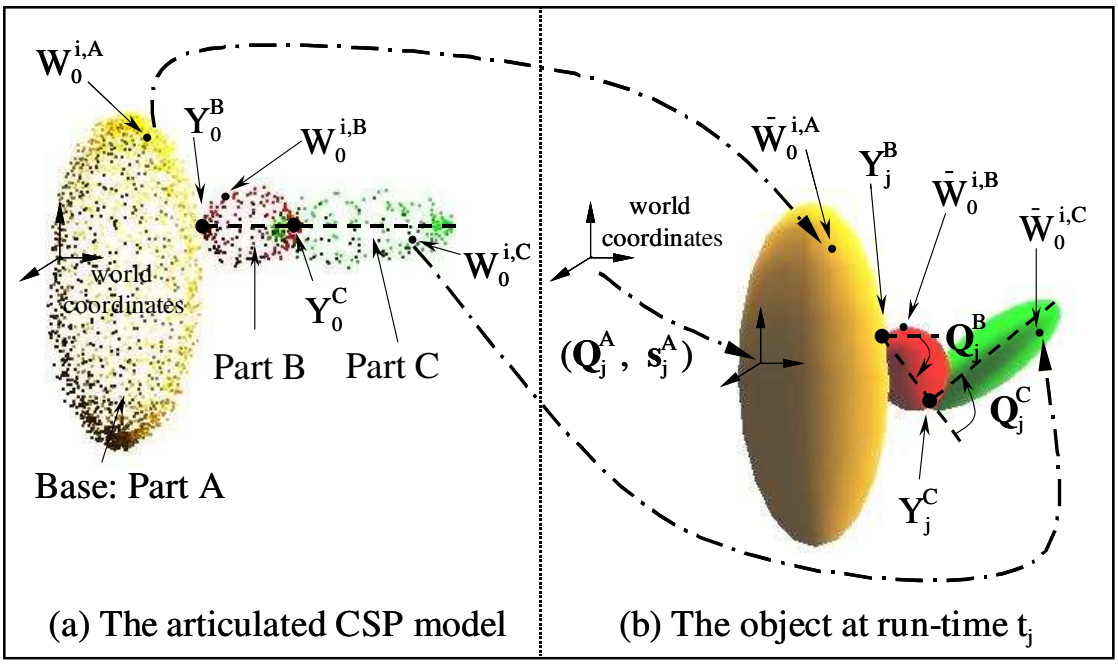


Figure 8.1: (a) The articulated CSP model of an articulated object with three rigid parts A, B and C. (b) The object itself at run-time t_j. The articulated CSP model in (a) is used to estimate the motion parameters of the object at t_j.

stants with the color and silhouette images represented by $\{I_j^k, S_j^k; \quad k = 1, \cdots\cdots, K; \quad j = 1, \cdots\cdots, J\}$. Also assume we have extracted from these images J sets of (unsegmented) CSPs $\{W_j^i, \mu_j^i\}$ of the object. Now the problem of image-based articulated object tracking can be stated as

The Image-Based Articulated Object Tracking Problem

Given the above input information, estimate the positions and orientations $(\boldsymbol{Q}_j^A, \boldsymbol{s}_j^A)$ of the base part A and the rotation matrices $\boldsymbol{Q}_j^B, \boldsymbol{Q}_j^C$ of the articulated joints at time t_j for all $j = 1, \cdots\cdots, J$.

8.1.2 Tracking Principle

Here we explain our tracking principle using the j^{th} frame data (captured at run-time t_j) of the sequence (see Figure 8.1(b)). We assume the articulated object is already tracked at

t_{j-1}, i.e. we have estimates of the parameters $\boldsymbol{Q}^A_{j-1}, \boldsymbol{s}^A_{j-1}, \boldsymbol{Q}^B_{j-1}$ and $\boldsymbol{Q}^C_{j-1}$. As a recap, we have the following information as the input data:

1. Model data:
   1a. segmented model CSPs $\{W^{i,A}_0, \mu^{i,A}_0, W^{i,B}_0, \mu^{i,B}_0, W^{i,C}_0, \mu^{i,C}_0\}$,
   1b. known model joint positions Y^B_0 and Y^C_0,
   1c. model color and silhouette images $\{I^k_0, S^k_0\}$ used to construct the model.

2. Data at t_j:
   2a. run-time unsegmented CSPs $\{W^i_j, \mu^i_j\}$,
   2b. run-time color and silhouette images $\{I^k_j, S^k_j\}$,
   2c. estimated parameters $\boldsymbol{Q}^A_{j-1}, \boldsymbol{s}^A_{j-1}, \boldsymbol{Q}^B_{j-1}$ and $\boldsymbol{Q}^C_{j-1}$ from previous frame.

Using the idea similar to that used in aligning two Visual Hulls in Section 5.3.3, we pose the problem of estimating $\boldsymbol{Q}^A_j, \boldsymbol{s}^A_j, \boldsymbol{Q}^B_j$ and $\boldsymbol{Q}^C_j$ as the problem of minimizing the geometric and color errors caused by projecting the 3D CSPs into the 2D images. To be more specific, there are two types of temporal errors we can use:

1. the forward geometric and photometric errors of projecting (respectively) the segmented model CSPs into the run-time silhouette and color images,
2. the backward geometric and photometric error of projecting (respectively) the run-time CSPs into the model silhouette and color images.

Provided with estimates of $\boldsymbol{Q}^A_j, \boldsymbol{s}^A_j, \boldsymbol{Q}^B_j$ and $\boldsymbol{Q}^C_j$, the forward errors are obtained easily by applying the appropriate motions to the already segmented model CSPs and projecting them into the run-time images. To calculate the backward errors, however, an extra step is required. In order to apply the correct motion transformations (due to part A, B or C) to the run-time CSPs, we have to decide for each run-time CSP W^i_j, which part of the articulated object it belongs to. In other words, we have to segment the set of CSPs $\{W^i_j, \mu^i_j\}$ to parts A, B and C. Generally segmenting a set of 3D points is an non-trivial problem, and different approaches are used under different situations. Two approaches for segmenting

the run-time CSPs based on the known shape information of the model and the estimated motion parameters from the previous frame will be proposed in Section 8.2.4 when we apply the tracking algorithm to human body. Once the run-time CSPs are segmented, the backward error can be calculated easily and added to the forward errors.

Theoretically, to estimate the motion parameters, it is sufficient to just include the forward errors in the optimization equations. However, the advantage of including the backward errors is that the motion parameters are then highly constrained. This means that with the addition of backward errors, the tracking is less likely to fall into local minimum, especially when any two jointed parts of the articulated object are very close to each other (see Section 8.2.5 for details). The disadvantage of including the backward errors is the extra step that is required to segment the run-time CSPs. Note that the backward errors should not be used if the segmentation of the run-time CSPs is not reliable.

8.1.3 Incorporating Joint Constraints into Optimization Equations

In this section we give the mathematical equations to incorporate the joint constraints into the calculation of the forward and backward errors. For the forward errors, let $\bar{W}_0^{i,A}, \bar{W}_0^{i,B}$ and $\bar{W}_0^{i,C}$ be the positions of $W_0^{i,A}, W_0^{i,B}$ and $W_0^{i,C}$ at run-time t_j (see Figure 8.1(b)). By the joint constraints between the parts, we have the following equations relating the transformed model CSPs and the joint positions (Y_j^B and Y_j^C) at t_j with the motion parameters:

$$\textit{Part } A: \qquad \bar{W}_0^{i,A} = \boldsymbol{Q}_j^A W_0^{i,A} + \boldsymbol{s}_j^A \,, \tag{8.1}$$

$$\textit{Part } B: \qquad Y_j^B = \boldsymbol{Q}_j^A Y_0^B + \boldsymbol{s}_j^A \,,$$

$$\bar{W}_0^{i,B} = \boldsymbol{Q}_j^A \boldsymbol{Q}_j^B (W_0^{i,B} - Y_0^B) + Y_j^B \,, \tag{8.2}$$

$$\textit{Part } C: \qquad Y_j^C = \boldsymbol{Q}_j^A \boldsymbol{Q}_j^B (Y_0^C - Y_0^B) + Y_j^B \,,$$

$$\bar{W}_0^{i,C} = \boldsymbol{Q}_j^A \boldsymbol{Q}_j^B \boldsymbol{Q}_j^C (W_0^{i,C} - Y_0^C) + Y_2^C \,. \tag{8.3}$$

Substituting the above equations into Equation (5.9), the forward errors are written as

$$
\begin{aligned}
e_{2,1} &= \sum_{i=1}^{L_0^A} \sum_k \left\{ \tau * d_j^k(\bar{W}_0^{i,A}) + [\boldsymbol{c}_j^k(\bar{W}_0^{i,A}) - \mu_0^{i,A}]^2 \right\} \\
&+ \sum_{i=1}^{L_0^B} \sum_k \left\{ \tau * d_j^k(\bar{W}_0^{i,B}) + [\boldsymbol{c}_j^k(\bar{W}_0^{i,B}) - \mu_0^{i,B}]^2 \right\} \\
&+ \sum_{i=1}^{L_0^C} \sum_k \left\{ \tau * d_j^k(\bar{W}_0^{i,C}) + [\boldsymbol{c}_j^k(\bar{W}_0^{i,C}) - \mu_0^{i,C}]^2 \right\} . \qquad (8.4)
\end{aligned}
$$

As in Equations (5.2) and (5.9) in Chapter 5, the error of a model CSP w.r.t the k^{th} run-time color and silhouette image is calculated only if the CSP is visible in that camera. Since in this case, the object consists of articulated rigid parts, the "reverse approach" described in Section 5.4.2 for testing visibility is not applicable. An alternative method for determining visibility for articulated object tracking will be discussed in Section 8.2.3.

To calculate the backward errors $e_{1,2}$, we first express the positions of the (now assumed segmented) run-time CSPs w.r.t. the model images in terms of the motion parameters $\boldsymbol{Q}_j^A, \boldsymbol{s}_j^A, \boldsymbol{Q}_j^B$ and $\boldsymbol{Q}_j^C$ by inverse transforming the set of motion relations in Equations (8.1) to (8.3). Then the transformed run-time CSPs are projected into the model silhouette and color images to get the geometric and photometric errors, again using Equation (5.9). Combining the backward and forward error terms (Equation (8.4)), the optimization equation becomes

$$
\min_{\boldsymbol{s}_j^A, \boldsymbol{Q}_j^A, \boldsymbol{Q}_j^B, \boldsymbol{Q}_j^C} [e_{2,1} + e_{1,2}] , \qquad (8.5)
$$

which can be solved by using the Levenberg-Marquardt algorithm [DS83, PTVF93].

Although we have described the tracking algorithm using an example articulated object consists of three parts, it can be easily extended to articulated objects with N parts. In the special case where the motion (rotation and translation) of the base (part A in our example) is known, or if it is static, the problem degenerates to tracking a multi-link object

around a fixed point. An example would be the situation we discussed in Section 7.1.2.2 for globally registering the joints of the limbs. Note that in such cases our algorithm still applies with the difference that $(\boldsymbol{Q}_j^A, \boldsymbol{s}_j^A)$ are known constants instead of parameters to be optimized in Equation (8.5). To conclude this section, we summarize the Image-Based Articulated Object Tracking Algorithm below:

The Image-Based Articulated Object Tracking Algorithm

1. Initialize the motion parameters in the first frame t_1.
2. For $j = 1, \cdots\cdots, J$, estimate the motion parameters at t_j by the following procedures:
   (a) Initialize the motion parameters at t_j with those estimated at t_{j-1}.
   (b) Segment the run-time CSPs at t_j.
   (c) Apply the Iterative LM algorithm (described in Section 5.3.2) to Equation (8.5) to minimize the sum of forward errors and backward errors with respect to the motion parameters $\boldsymbol{Q}_j^A, \boldsymbol{s}_j^A, \boldsymbol{Q}_j^B$ and $\boldsymbol{Q}_j^C$ until convergence is attained or for a fixed number of iterations.

8.2 Tracking Full Body Human Motion

8.2.1 The Articulated Human Model

The articulated CSP models used to track human motion are the same as those built in Chapter 7 (see for example Figure 7.10). Each model consists of nine body parts: torso, right/left lower/upper arms, right/left lower/upper legs, connected by eight joints: right/left shoulder/elbow joints, right/left hip/knee. Each body part is assumed to be rigid with the torso being the base. The shoulder and hip joints have 3 degree-of-freedom (DOF) each while there is 1 DOF for each of the elbow and knee joints. Including translation and rotation of the torso base, there are a total of 22 DOF in the model.

8.2.2 Hierarchical Tracking

The most straightforward way to use the Image-Based Articulated Object Tracking Algorithm for human motion tracking is to apply it directly to all the body parts (with a total of 22 DOF) of the articulated CSP model at the same time. In practice, however, this all-at-once approach is prone to the problem of too many local minima because of the high dimensionality. To reduce the chance of falling into local minimum, we instead use a two-step hierarchical approach: first fit the torso base and then each limb independently. This approach makes use of the fact that the motion of the body is largely independent of the motion of the limbs which are, under most of the cases, largely independent of each other. The first step of our hierarchical approach involves recovering the global translation and orientation of the torso base. This can be done using the 6 DOF temporal SFS algorithm for rigid objects in Section 5.3.3. Once the global motion of the torso has been estimated, the four joint positions: left/right shoulders and left/right hips are calculated. In the second step, the four limbs of the body are aligned separately around these fixed joint positions just as in the special case mentioned at the end of Section 8.1.3. Using such a hierarchical approach not only reduces the chance of falling into local minimum, but also reduces the processing time as there are only four unknowns to be optimized for each limb.

8.2.3 Determining Visibility

Since the human model consists of articulated body parts, the conservative silhouette-based visibility test used in Section 5.4 is inapplicable for alignment in motion tracking. Here we propose another method for testing visibility. The basic idea is to first turn our articulated CSPs model into an articulated voxel model by replacing the CSPs with solid voxels (as discussed at the end of Section 7.3). Then at time t_j, the voxels in the model are transformed using the *estimated* motion parameters at t_j to simulate a 3D visibility space as it would be formed by the actual human body. The visibility of a 3D point w.r.t. a camera is now

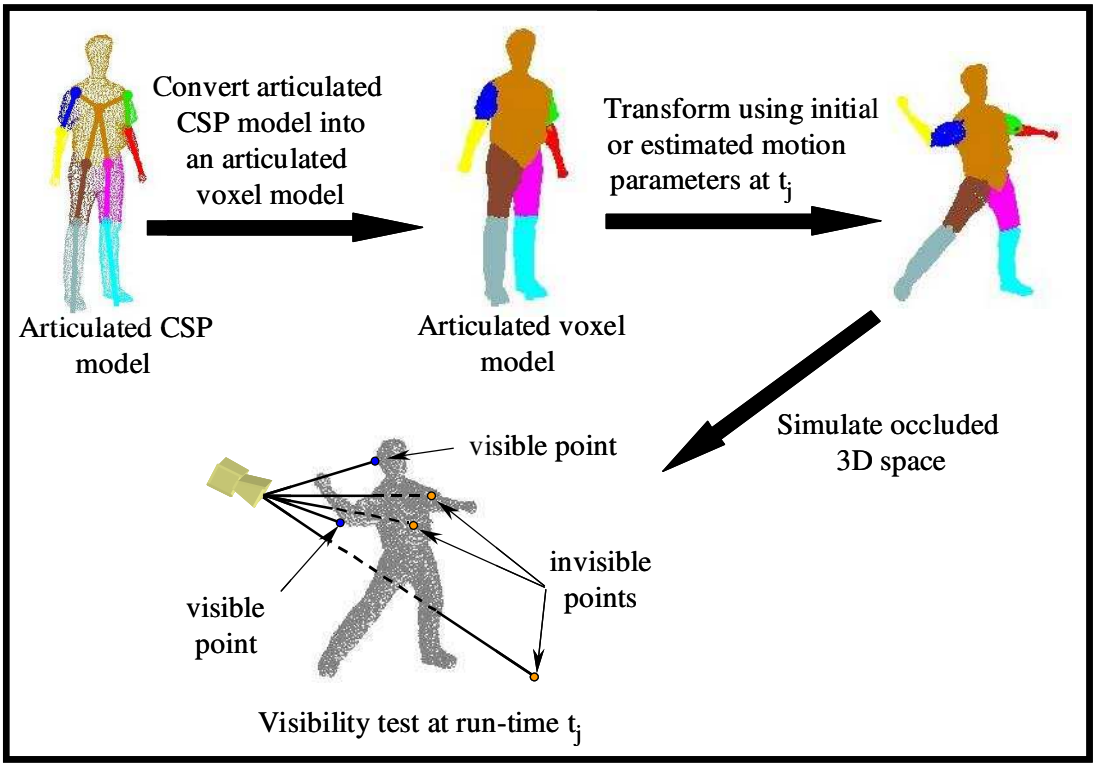


Figure 8.2: Determining visibility at time t_j using an articulated voxel model and the estimated motion parameters at t_j.

determined by intersecting the transformed 3D voxel model with the line joining the camera and the 3D point. If the line hits any part of any voxels of the model, the 3D point is invisible, otherwise it is visible. This visibility test is illustrated in Figure 8.2.

Since the voxel model is essentially the Visual Hull (and therefore an overestimated shape) of the person, there is a definite advantage in using the voxel model as it provides a conservative margin for the visibility test. It has to be pointed out that since the voxels are transformed using the estimated or approximated motion parameters instead of the true values, some of the visibility test results may not be correct (despite the conservative margin offered by the voxel model). As the estimated motion parameters converges toward the true solution, however, the visibility tests become more and more accurate.

8.2.4 Run-time CSPs Segmentation

Here we suggest two approaches to segment the run-time CSPs. The first approach approximates the body parts using simple geometrical primitives and uses the distance between the CSPs and the primitives for segmentation. The second approach segments the CSPs by segmenting the boundary of the run-time silhouette images which originate the CSPs.

8.2.4.1 Segmenting 3D CSPs using approximated ellipsoidal shells

Recall that in the real-time SFS system in Chapter 3, we fit and segment the reconstructed surface voxels using ellipsoidal shells (Section 3.3.2). The first approach of segmenting run-time CSPs uses a similar idea. The whole procedure (at time t_j) is illustrated in Figure 8.3. It consists of three steps listed follow:

1. The body parts of the CSP articulated model are approximated by ellipsoidal shells using Equations (3.3) to (3.5) on the given model CSPs.
2. The ellipsoidal shells are transformed using the estimated motion parameters at t_{j-1}.
3. Each run-time CSP at t_j is segmented as belonging to the body part whose ellipsoidal shell is closest in distance to that CSP.

8.2.4.2 Segmenting 3D CSPs by segmenting the 2D silhouette boundary

The idea of the second approach comes from the fact that all CSPs originate from points on the boundary of the silhouette. This means that segmenting the 3D run-time CSPs can be done by segmenting the boundaries of the 2D run-time silhouettes. Figure 8.4 illustrates the steps of segmenting the 2D run-time silhouette boundary at t_j:

1. The model CSPs of the articulated model are transformed using the estimated motion parameters at t_{j-1}.
2. To segment the boundary silhouette of run-time image S_j^k, the transformed model CSPs are projected (along with their body part identities) onto S_j^k.
3. For each boundary point of S_j^k, it is segmented to belong to the body part which has the highest number of projected CSPs within a fixed radius of the boundary point.

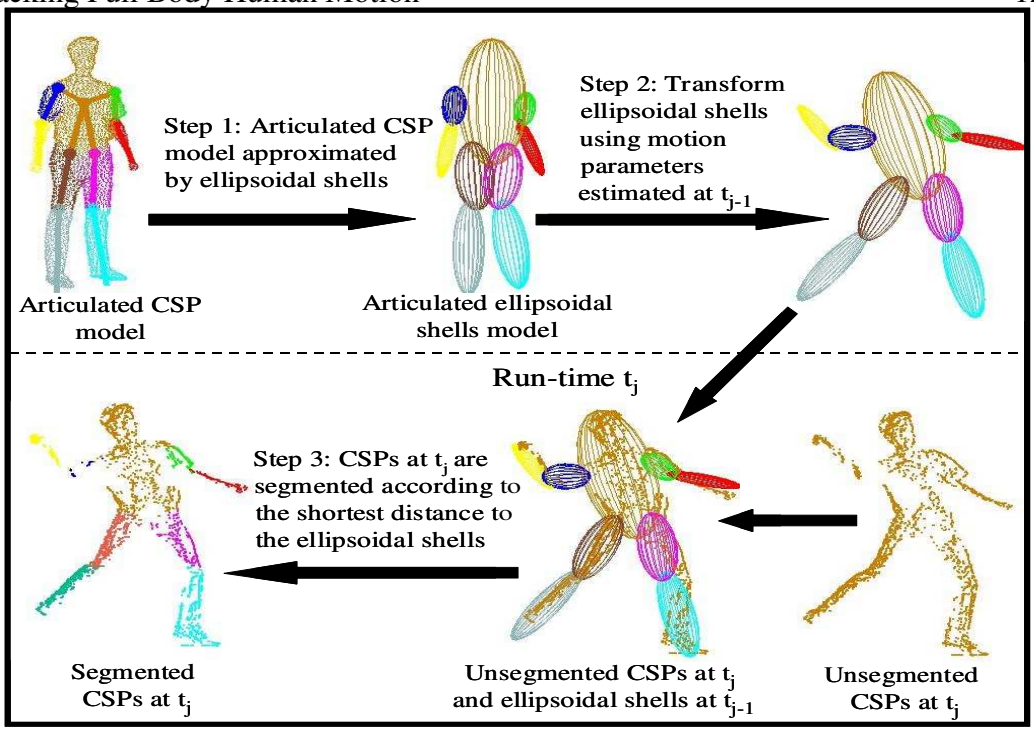


Figure 8.3: Segmenting the 3D CSPs at t_j using approximated ellipsoidal shells at t_{j-1}.

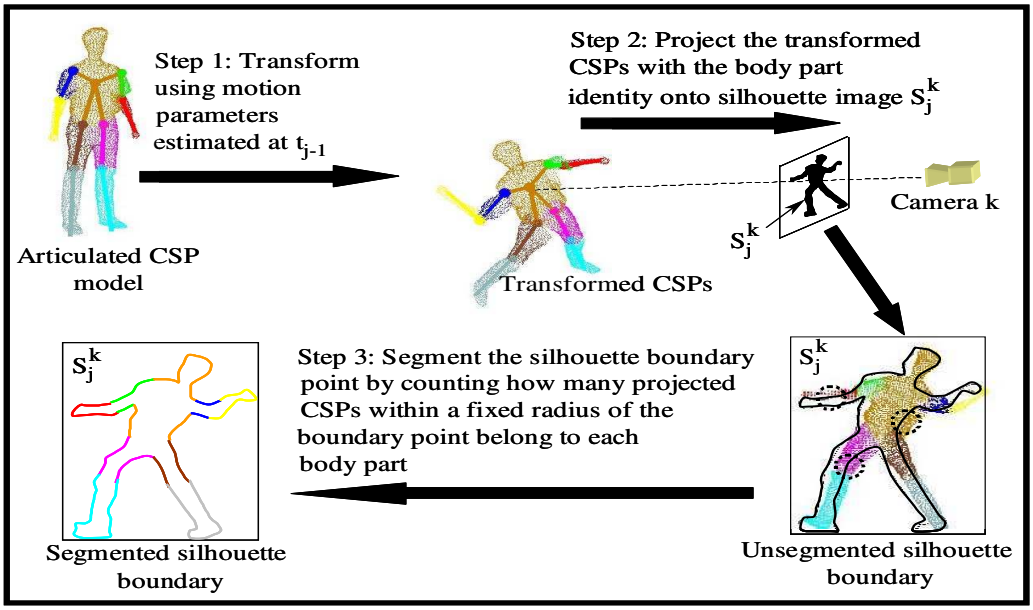


Figure 8.4: Segmenting the 3D CSPs by segmenting the 2D boundary of the silhouette image S_j^k at t_j.

Although the above two approaches are described using human body tracking as an example, they can be applied to the tracking of other articulated objects. The advantage of using the first approach is that it is fast and easy to implement. However, it is not applicable to articulated objects whose parts cannot be closely approximated by simple geometric primitives. In such situations, the second approach is preferred though it is slower.

8.2.5 Dealing with Local Minimum

As common to all methods which use an error minimization formulation, our human motion tracking algorithm is prone to the problem of local minima, especially since the human articulated body has very large number of DOF. Though we have used the hierarchical approach (discussed in Section 8.2.2) to reduce the high dimensionality of the tracking into multiple smaller sub-optimizations to lower the chance of falling into local minimum, the problem cannot be completely avoided.

There are three situations where our tracking algorithm is particularly vulnerable to local minima. The first situation occurs when one of the the arms is very close to the torso. In this situation, there is a big chance that the optimization will get trapped in a local minimum where the arm stays in a position inside the body of the person (see Figure 8.5(a) for an example). The second situation occurs when the legs of the person are crossing each other and the tracking algorithm is not able to distinguish between the left and right lower legs as shown in Figure 8.5(b). The third situation happens when the arm is straight and there is not enough color (or texture) information on the arm to differentiate the rotation angles of the shoulder joint about the axis along the length of the arm. An example is illustrated in Figure 8.5(c) where the palm of the left arm of SubjectE is facing upward (see the associated image) but the recovered joint angles have the palm of the arm facing downward (i.e. the joint angles of the left shoulder joint is rotated around the axis along the arm by 180 degrees).

To cope with the first two situations, collision detection and reinitialization is added to

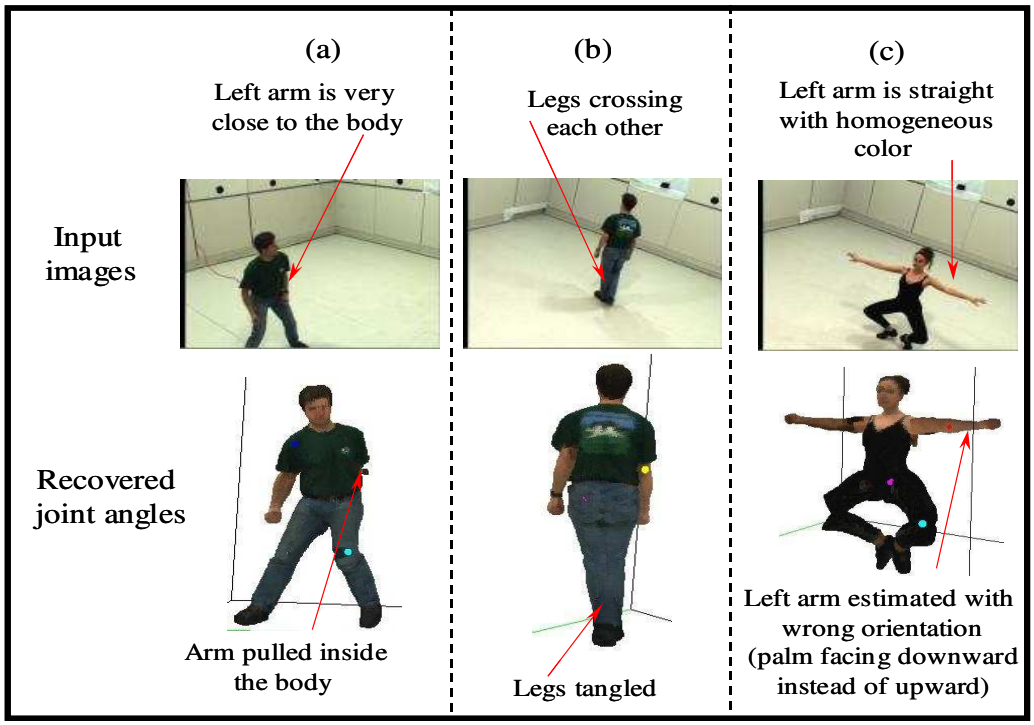


Figure 8.5: Three situations where our tracking algorithm is particularly vulnerable to local minima. (a) The arm is very close to the body. (b) The legs are crossing each other. (c) The arm is straight and of homogeneous color.

our algorithm. In each frame, after all the joint angles have been estimated, the body parts are checked for collision. If a collision is detected between a limb and the body, within each limb (e.g. collision of upper and lower arm) or between limbs, the joint angles of the limbs involved in the collision are reinitialized and re-aligned. To reinitialize, instead of using only the joint angles estimated from the previous one frame, those from the previous three frames are used to predict the initial guess. To increase the chance of climbing out of the local minimum, a small random perturbations is also added to the initial guess.

Note that although the above heuristic is sufficient to avoid some of the local minima, it still fails occasionally. For a failed frame, to avoid propagating the wrong estimates to the next frame, we set the joint angles to be those estimated from the previous frame. By doing so, it is hoped that the local minimum problem will be resolved in the next frame. For

cases where a limb is totally lost in the tracking, simple manual reinitialization is required to restart the tracking of that limb.

The third situation is difficult to deal with because the geometric constraints are not able to resolve the ambiguity due to the symmetry of the arm. In cases when there is no texture on the arm (as in the case of SubjectE), the constraints of photometric consistency are also unable to correct the mis-alignment. Although currently we have no solution to this situation, the tracking generally recovers by itself once the arm is bent (when the ambiguity can be resolved by the geometric constraints).

8.3 Experimental Results

To test our tracking algorithm, two types of data are used: (1) synthetic sequences with ground-truth are generated using OpenGL to obtain quantitative comparison and (2) sequences of real people with different motions are captured for qualitative results.

8.3.1 Synthetic Sequences

Two synthetic motion video sequences: KICK (60 frames) and PUNCH (72 frames) were generated using the synthetic human model in Section 6.2.1 with a total of eight cameras per sequence. The recovered articulated model shown in Figure 7.10(a) is used to track the motion in these sequences. Figure 8.6 compares the ground-truth and estimated joint angles of the left arm and right leg of the body in the KICK sequence. It can be seen that our tracking algorithm performs very well.

The movie **Synthetic-track.mpg**[*] illustrates the tracking results on both sequences. In the movie, the upper left corner shows one of the input camera sequences, the upper right corner shows the tracked body parts and joint skeleton (rendered in color) overlaid

[*]All movie clips of this chapter can be found at http://www.cs.cmu.edu/~german/research/Thesis/Video/Chapter8/

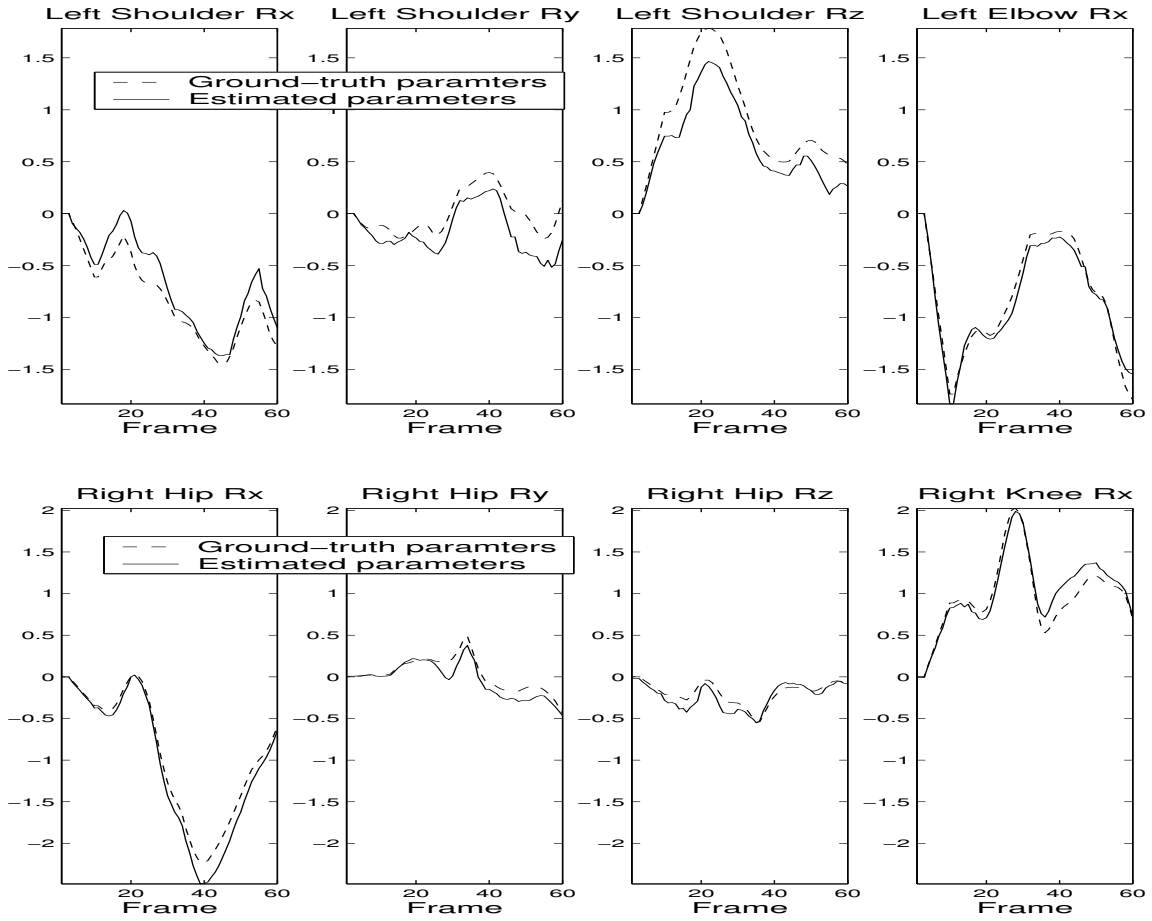


Figure 8.6: Graphs comparing ground-truth and estimated joint angles of the left arm and right leg of the synthetic sequence KICK. The estimated joint angles closely follow the ground-truth values throughout the whole sequence. The tracking results of the KICK sequence can be seen in the movie **Synthetic-track.mpg**.

on one of the input images (which are converted from color to gray-scale). The lower left corner depicts the ground-truth motion rendered using an avatar and the lower right corner represents the tracked motions with the same avatar. The avatar renderings show that the ground-truth and tracked motions are almost indistinguishable from each order.

8.3.2 Real Sequences

Our tracking algorithm is tested on a variety of real human subjects performing a wide range of motions. For SubjectG, three video sequences: STILLMARCH (158 frames) , AEROBICS (110 frames) and KUNGFU (200 frames) were captured to test the tracking algorithm with eight cameras used in each sequence. Figures 8.7 and 8.8 show the tracking results on the AEROBICS and KUNGFU sequences respectively. Each figure shows selected frames of the sequence with the (color) tracked body parts and the joint skeleton overlaid on one of the eight camera input images (which are converted to gray-scale for display). The movie **SubjectG-track.mpg** contains results on all three sequences. In the movie, the upper left corner represents one of the input camera images and the upper right corner illustrates the tracked body parts with joint skeleton overlaid on a gray-scale version of the input images. The lower left corner illustrates the results of applying the estimated motion data to a 3D articulated voxel model (obtained from the articulated CSP model as discussed at the end of Section 7.3) of the person while the lower right corner shows the results of applying the estimated motion data to an avatar. The video demonstrates that our tracking algorithm tracks well on both simple motions (STILLMARCH, AEROBICS) and complicated motions (KUNGFU). Note that in the above three sequences, the remedy discussed in Section 8.2.5 is not used for dealing with the problem of local minimum. Since the motions in the STILLMARCH and AEROBICS are simple, no local minimum problems are encountered in these two sequences. However, for the KUNGFU sequence, the tracking of the right arm is lost in frame 91 for 10 frames due to local minimum but recovers automatically at frame 101.

A motion sequence THROW (155 frames) of SubjectS is captured. The sequence is first tracked by our algorithm without using the local minimum remedy. Since body parts are not checked for collision, when the left arm is very close to the body at frame 70, local minimum pulls the left arm inside the body (see Figure 8.5(a)). Moreover, the tracking of both legs is also lost around frame 43 (which is shown in Figure 8.5(b)) when the legs

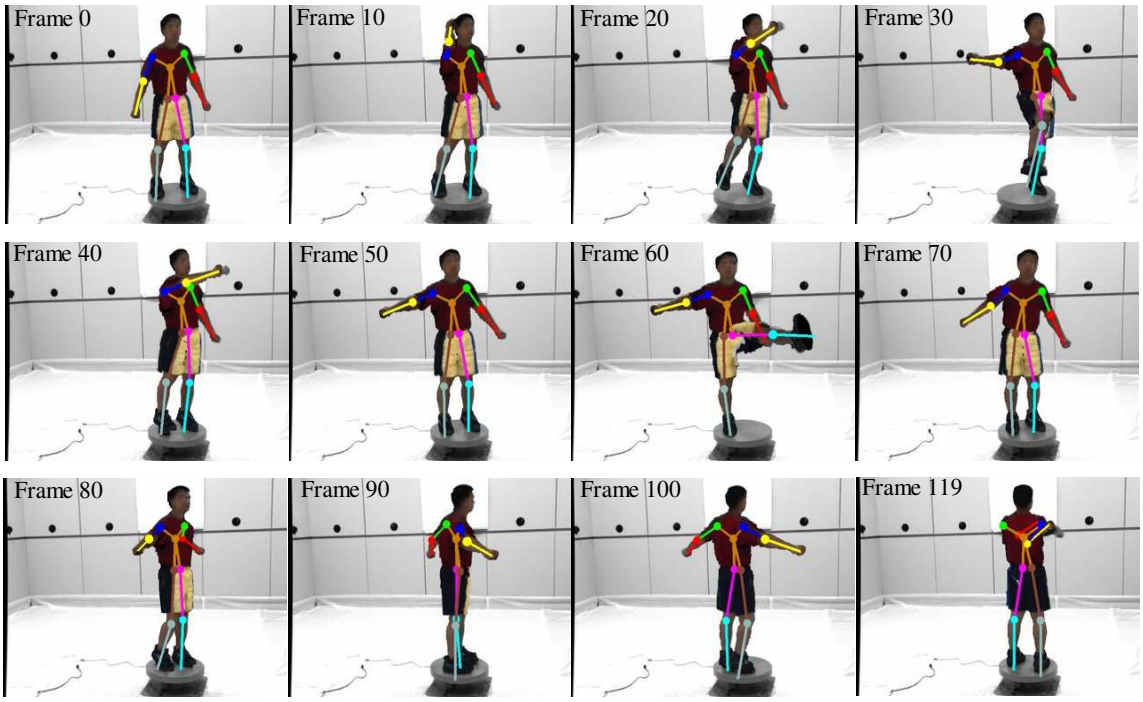


Figure 8.7: Tracking results of the AEROBICS sequence with 12 selected frames. The tracked body parts and joint skeleton (rendered color) are overlaid on one of the input camera images (which are converted from color to gray-scale for clarity). The whole sequence can be seen in the movie **SubjectG-track.mpg**.

started to cross each other. To resolve these problems, the sequence is re-tracked with the local minimum remedy turned on. The results are shown in Figures 8.9 which shows 24 selected frames of the sequence with the (color) tracked body parts and the joint skeleton overlaid on one of the eight camera input images (which are converted to gray-scale for display). The local minima problems of the legs and the left arm are successfully resolved by checking for body part collision and reinitialization. The whole THROW sequence can be seen in the movie **SubjectS-track.mpg**.

Two sequences: SLOWDANCE (270 frames) and STEP-FLEX (90 frames) of SubjectE are also captured and tracked. Some of the tracked frames are shown in Figure 8.10 for the SLOWDANCE sequence and Figure 8.11 for the STEP-FLEX sequence (the tracking results of both sequences are included in the movie clip **SubjectE-track.mpg**). The shoulder joint ambiguity problem (Figure 8.5(c)) happens in the SLOWDANCE sequence on the left

Step	Approximate time required per frame
Segmenting run-time CSPs	0.26s
Tracking the torso	61.6s
Tracking the right arm	16.3s
Tracking the left arm	13.2s
Tracking the right leg	49.7s
Tracking the left leg	56.0s
Collision detection	0.18s

Table 8.1: The approximate time required for each step in our tracking algorithm. It takes longer time to align the torso base and the legs than the arms because the former have much more CSPs than the latter. The time needed to segment the run-time CSPs and detect body parts collision is negligible compare to that required for alignment.

arm around frame 28 and on the right arm around frame 85 though the tracking recovers in later frames of the sequence (see movie clip **SubjectE-track.mpg** for better views of the problem). In the STEP-FLEX sequence, although the waist joint is not modeled, our tracking algorithm is able to approximate the bending of the body (around the waist) using the hip joints.

Table 8.1 gives the approximate time required for each step of our tracking algorithm. It takes longer time to align the torso base and the legs than the arms because the former have much more CSPs than the latter. The time needed to segment the run-time CSPs and detect body parts collision is negligible compare to that required for alignment. These timings are averaged from tracking the first 100 frames of the SLOWDANCE sequence on a machine with a 750MHz Pentium CPU.

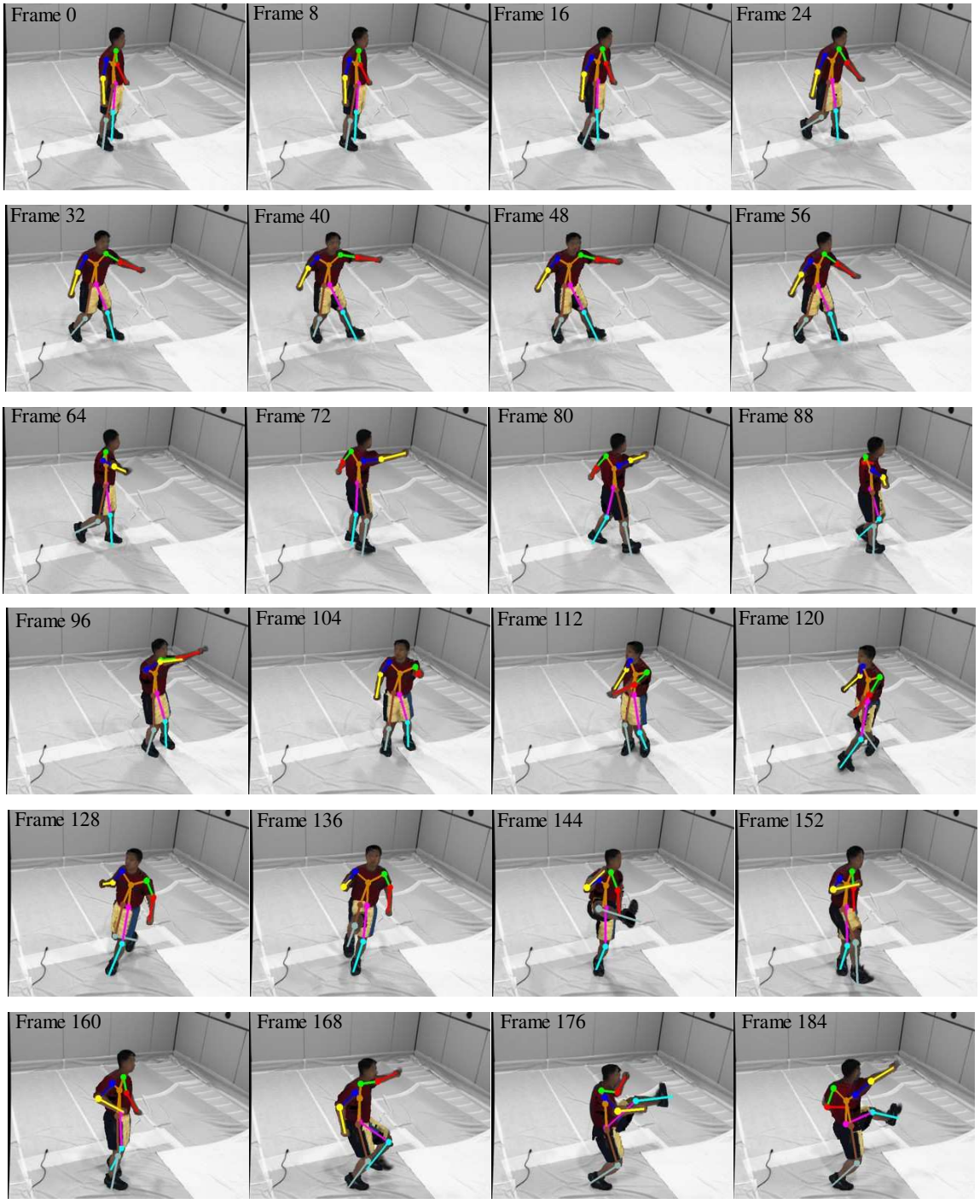


Figure 8.8: Tracking results of the KUNGFU sequence with 24 selected frames. The whole sequence can be seen in the movie **SubjectG-track.mpg**.

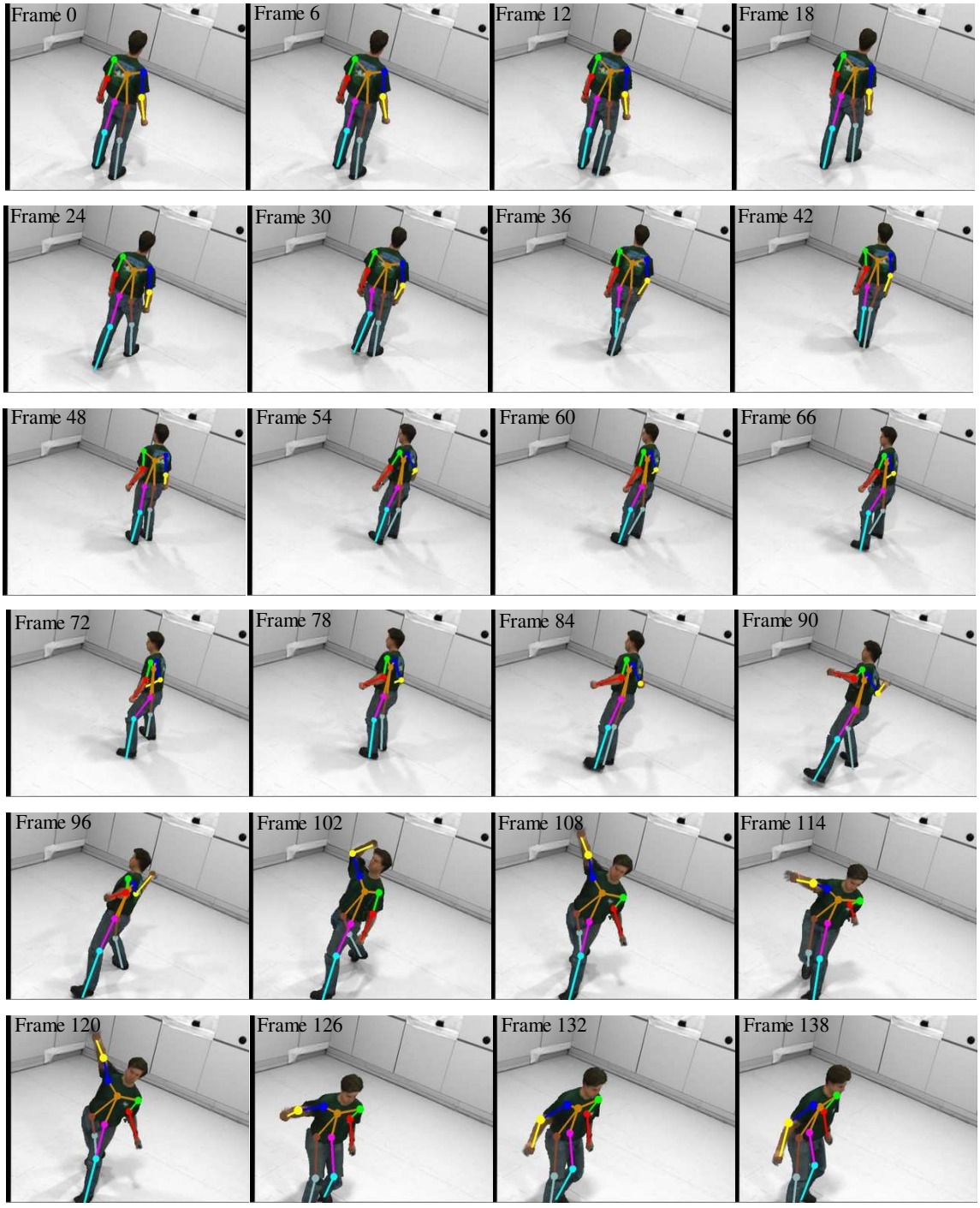


Figure 8.9: Tracking results of the THROW sequence with 24 selected frames. The whole sequence can also be seen in the movie **SubjectS-track.mpg**.

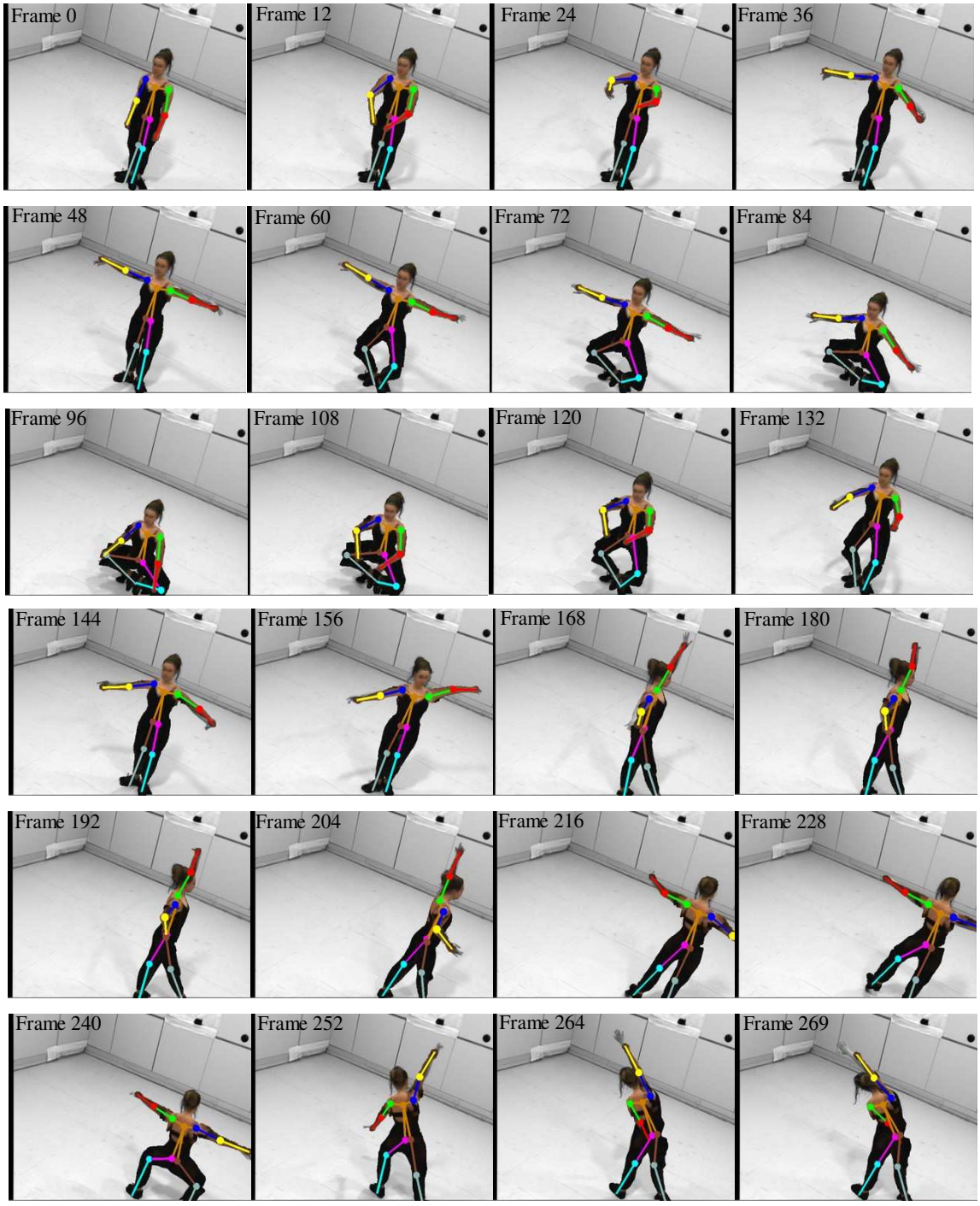


Figure 8.10: Tracking results for the SLOWDANCE sequence with 24 selected frames. The whole sequence can also be seen in the movie **SubjectE-track.mpg**.

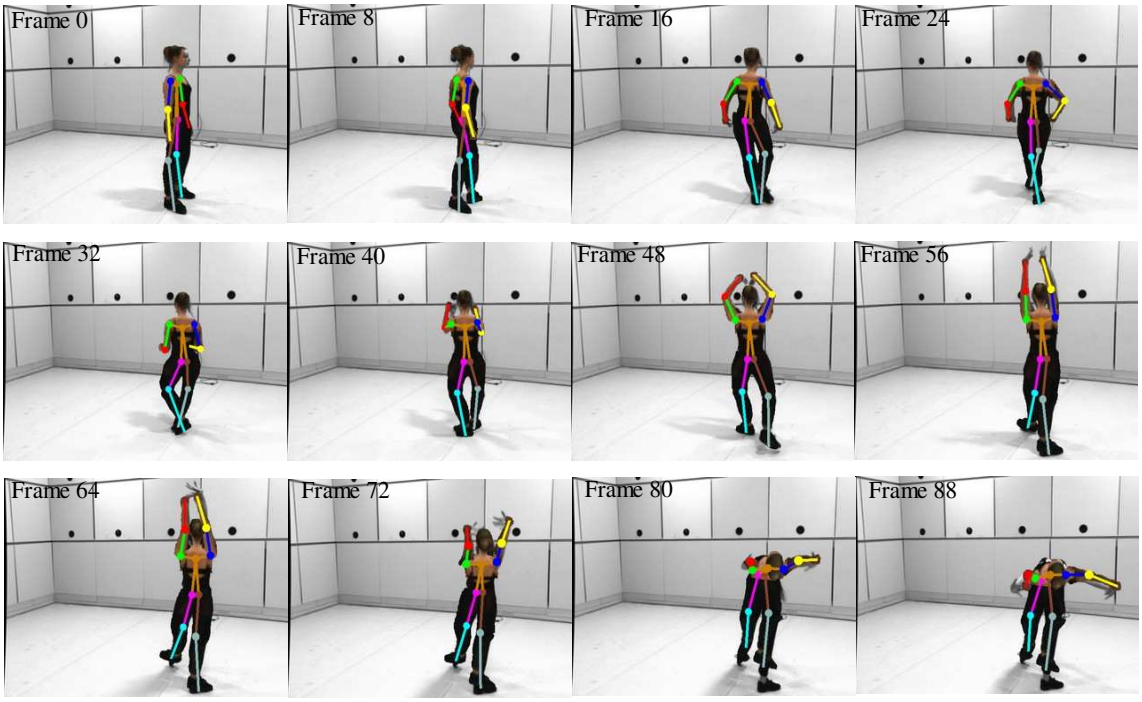


Figure 8.11: Tracking results for the STEP-FLEX sequence with 12 selected frames. The whole sequence can also be seen in the movie **SubjectE-track.mpg**.

8.4 Related Work

Among all of the model based approaches to track human motion, the work by Sidenbladh et al. in [SDB00, SBF00], that by Delamarre and Faugeras in [DF99], that by Carranza et al. in [CTMS03] and that by Mikic et al. in [MTHC03] are the most related to our tracking algorithm.

Sidenbladh et al. [SBF00] perform human motion tracking by first modeling the person using articulated cylinders as body parts. Each body part is projected into a reference image to create an appearance model [SDB00]. Using a particle filtering framework [DBR00], the articulated 3D appearance model is then used to track the motion [SBF00]. As pointed out by the authors themselves, their model works well for tracking a single body part but is too weak for constraining the motion of the entire body without using specific motion models.

Hence their approach is restricted to tracking simple motions such as walking or running for which motion model can be created by collecting examples [SBF00].

In [DF99], silhouette contours from multiple cameras are used to constraint the articulated model (which consists of geometric primitives such as cylinders or truncated cones) of a person. The way of generating "forces" to align 2D contours of the projected model with the silhouette boundary is similar to the geometric constraints we use in our tracking algorithm. In [CTMS03], Carranza et al. first render a human model using graphics hardware and then compare the rendered images (using pixel-wise XOR) with the silhouette images extracted from video sequences to track human motion. Although it is unclear exactly how their XOR errors are formulated as driving forces for optimizing the motion parameters, their grid-search initialization procedure provides a good way to reduce the problem of local minima. Mikic et al. also use multiple-view silhouettes in [MTHC03] for motion tracking, although their body part fitting is done in 3D space and is closely related to our previous work in [CKBH00]. None of the above work uses color information, unlike in our algorithm.

8.5 Discussion

Due to the high number of degree of freedom of the human body, motion tracking is a difficult problem. The problem is even more challenging for vision-based (no markers) approaches because of self occlusion, unknown kinematic information, perspective distortion and cluttered environment. In this chapter, we have shown how to use the Visual Hull Alignment idea to perform human motion tracking. Our tracking algorithm has two major advantages compared to other model-based methods. First, our person specific models consist of CSPs which closely approximate the actual shape of the body parts, with joint information estimated directly from the motion of the person. The accurate kinematic model gives better shape and joint constraints than methods which uses simple approximating ge-

ometric primitives. Secondly the (color) appearance model provided by the CSPs help to combine seamlessly the geometric constraints and the color consistency in one single optimization formulation. Most other vision-based motion tracking methods lack this feature of using both color and shape information simultaneously and effectively.

For relatively simple motions, such as the STILLMARCH and AEROBICS sequences, our tracking algorithm works very well. However, for complex motions such as those in the KUNGFU and THROW sequences, our algorithm suffers from the problem of local minima. This problem is unavoidable because of the error minimization formulation of the algorithm. Although the remedy we suggested in Section 8.2.5 is able to resolve some of these local minima problems, there are un-resolvable situations such as the one in Figure 8.5(c). Another way to deal with the local minima problem is to apply joint angles limits (or the constraints of reachable workspace as defined in [MLS94]) to the tracking error measure. More details about this will be discussed in Section 10.2 as future work.

Chapter 9

Human Motion Rendering

In the previous chapters, we have derived algorithms to build detailed kinematic model of human body and to perform motion tracking in video sequences using the acquired model. In this chapter, we propose an algorithm to photo-realistically render the articulated human body and demonstrate how the algorithm can be used for interesting computer graphics applications such as generating pictures of "faked" motion of a person or exchanging motions between two people. The algorithm, which we called Image-Based Articulated Model Rendering Algorithm uses the image-based rendering technique [CW93, WHH95, GGSC96, LH96, SD96, McM97, SK98, Deb98, BSV+00, VBK02] to render, from any viewpoint, the articulated human model performing new motion from a set of "source images". The details of the algorithm, including the required input data, the various rendering steps and the implementation issues are described below.

9.1 Image-Based Articulated Model Rendering Algorithm

9.1.1 Input Data

The following data and information of a person is assumed to be available:

1. An articulated voxel model of the person built using the kinematic modeling system described in Chapter 7.

2. Video sequence of the person performing some motion. It is assumed that there are K calibrated cameras and the sequence contains J frames of color images. We call these JK images as the source images and the first frame of this source sequence is set as the reference frame.

3. The motion data (which we will refer to as the source motion data) of the person corresponding to his motion in the video sequence in (2) above. The motion information can be obtained using our motion tracking algorithm in Chapter 7 or marker-based motion capture system. The data is assumed to be registered with respect to the reference frame.

4. Target motion data to be applied to the model for rendering. The target motion can be obtained from tracking the motion of another person or from a motion database.

9.1.2 Algorithm Outline

Our rendering algorithm consists of three parts: (I) pre-rendering processing, (II) pixels rendering and (III) post-rendering processing. In the pre-rendering process, the articulated voxel model is first converted to a mesh-based model. Then the target motion data is applied to the converted mesh model to transform the body parts to their target positions. The voxel-to-mesh conversion is done to ensure continuity between body parts when the target data is applied to the model. In a voxel model, since the voxels are not properly

connected around the joints, discontinuity artifacts happen when the person moves. The problem is avoided when mesh-based model is used because meshes are connected with each other at their vertices. In Section 9.1.3.1, we will suggest a simple way to convert the articulated voxel model built in Section 7.3 to a mesh model and describe how the mesh vertices around the articulation joints are moved flexibly using motion weights.

After the mesh model is transformed by the target motion data, it can be rendered from any virtual camera viewpoint. Hereafter, we call the rendered image of the person performing the target motion as the target image and its pixels as the target pixels. To produce photo-realistic pictures, the second part of our IBAMRA render the target image using an image-based rendering technique by which the target pixel colors are taken from the source images. This rendering process is illustrated in Figure 9.1 using SubjectE as an example.

For a target pixel, its color is determined using the following four rendering steps. First a viewing ray is casted from the (virtual) camera center through the target pixel into the 3D space. The ray is intersected with the transformed mesh model of the person. If the ray does not intersect the transformed model, the target pixel is a background pixel and is assigned the appropriate background color. Otherwise, the body part Z where the ray first intersects the mesh model, together with the point of intersection (hereafter denoted by P and referred to as the target model point) are found (Step 1 in Figure 9.1). Using the inverse target motion transformation equations of part Z, the position of P at the source reference frame, denoted by P_1 is computed. Once P_1 is known, its positions at all the other source frames, represented by $\{P_j; \quad j = 2, \cdots\cdots, J\}$ are calculated using the given source motion data (Step 2 in Figure 9.1). We name P_j as the source model points of the target pixel. Once computed, each source model point P_j is projected onto the K source color images of the j^{th} frame. If the projected point is visible in the k^{th} image, the color of the projected pixel is *valid*, otherwise the color is *invalid* (Step 3 in Figure 9.1). Finally the color of the target pixel is assigned to be the weighted average of the *valid* source pixel colors using the

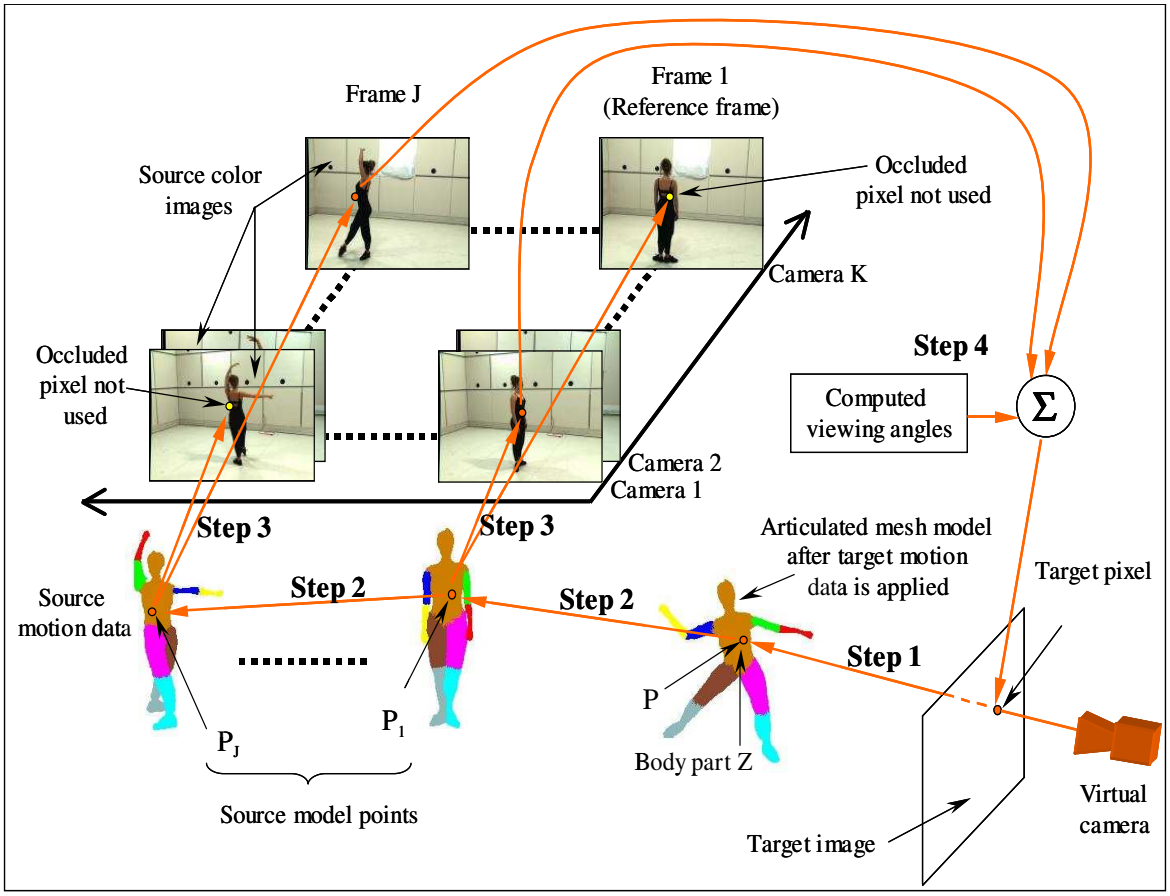


Figure 9.1: The pixel rendering part of our Image-Based Articulated Model Rendering Algorithm. Four steps are used to determine the color of a target pixel.

viewing angles between the virtual camera, the source camera and the target model point P as weights (Step 4 in Figure 9.1). The viewing angle between two cameras and a 3D point is defined as the angle between the two lines joining the point and the cameras.

The final part of our algorithm involves a post-rendering process to fill in those target pixels whose colors cannot be determined because their source model points are invisible in all of the source images. Background and shadows are also added to enhance the photo-realism of the final image. As a summary, our rendering algorithm is summarized below:

Image-Based Articulated Model Rendering Algorithm (IBAMRA)

(I) **Pre-rendering Processing**

Convert the articulated voxel model into a mesh model and apply the target motion data to the converted mesh model.

(II) **Pixels Rendering**

Determine the color of each target pixel of the target image by:

1. Intersect the viewing ray (originated from the virtual camera center) of the target pixel with the transformed target mesh model to locate the target model point P.
2. Compute the source model points $\{P_j;\ \ , j = 1, \cdots\cdots , J\}$ from the target model point P, using both the target and source motion data.
3. For each source model point P_j and camera k, if P_j is visible to camera k at frame j, project P_j into the k^{th} color images of frame j and collect the color of the projected point as a source pixel color.
4. Average the source pixel colors collected in Step 3 with weights according to the viewing angles between the source camera, the virtual camera and the target model point P. Set the averaged color as the target pixel color.

(III) **Post-rendering Processing**

Fill in colors of the target pixels which are not visible in any of the source images. Add background and shadows to the final target image.

9.1.3 Implementation Details

In the last section, we briefly explain the steps of our rendering algorithm. Here we discuss some of the implementation issues of the algorithm. Note that the idea provided in this section focus on simplicity. Naturally there are other ways to implement our algorithm.

9.1.3.1 Pre-rendering Processing

There are two tasks to the pre-rendering process: convert the voxel model to a mesh model and apply the target motion data to the converted model. To convert the voxel model to a mesh model, Marching Cubes Algorithm [LC87] is used. Since the voxel data is discrete, the mesh model generated by the Marching Cubes Algorithm is blocky. To remove the blockiness, the mesh is smoothed (but without being shrunk) by applying a low-pass filter to the positions of the vertices of the mesh [JH97]. Once the mesh is smoothed, the same segmentation algorithm proposed in Section 7.3 (see Figure 7.9) is used to segment the vertices to belong to different body parts. Each triangular face of the mesh is also segmented according to the segmentation of its three vertices (i.e. a face is segmented as belonging to body part Z if 2 or more of its vertices belong to Z).

One way to apply the target motion data to the mesh model is to move each vertex of the mesh according to the 6D transformation of the body part the vertex belongs to. Though simple, this method causes abrupt changes of the mesh around the joints where two body parts meet. To create smoother transition of the mesh between body parts, each vertex is transformed by a weighted sum of the motions of the body part around that vertex. This idea, which is also known as skinning, is commonly used in the computer animation of skin. Here we suggest a very simple way to compute the weights. Figure 9.2 shows a vertex V and all the vertices which share an edge with V. The segmentation of the vertices are also indicated in the figure by black and white dots which represent two different body parts. The idea is to set the motion weight of V w.r.t. a body part as the fraction of vertices around V (including V itself) which are classified as belonging to that body part. For the example in Figure 9.2, the motion weight for V w.r.t the body part represented by the black dot is $\frac{3}{7}$ while that represented by the white dot would be $\frac{4}{7}$.

9.1.3.2 Pixels Rendering

Step 1: Intersect casting rays with the articulated model

The simplest way to find the intersection point P is to intersect the viewing ray directly

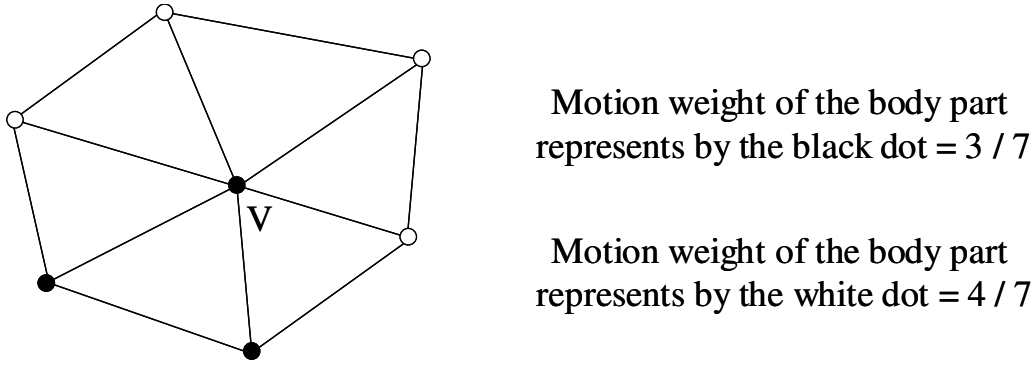


Figure 9.2: Pre-rendering processing: motion weights for a vertex V are calculated using the segmentations of the vertices around V.

with all the faces of the articulated mesh model and choose the one which is closest to the camera. However, in practice this approach is too slow as the model usually composes of thousands of faces. Instead, we employ the graphic hardware acceleration idea (the *item buffer*) that is used in [WHG84, VBK02]. Each mesh face of the model is assigned a distinct RGB color as its identity (ID) number (note that with color of 24 bits, up to 16M faces can be assigned with distinct ID numbers). After the model is transformed by the target motion data, the mesh faces are rendered with their ID colors using OpenGL and graphic hardware. The triangular face that intersects the casting ray of a target pixel can easily be found by reading the ID color of the same pixel of the rendered ID picture. Once the intersecting mesh face is found, it is intersected with the viewing ray to locate the target model point P.

Step 2: Compute the source model points P_j

Because of the motion weighing strategy (Section 9.1.3.1) used to smooth the motion of the model near the joints, some of the mesh faces are stretched after the target motion data is applied (see Figure 9.3). The following procedure is used to compensate for this stretching when calculating P_1 (the source model point at the reference frame) from P. Let V^1, V^2, V^3 be the vertices of the intersecting mesh face after applying the target motion data, and V_1^1, V_1^2, V_1^3 be the corresponding vertices of the same face at the reference frame.

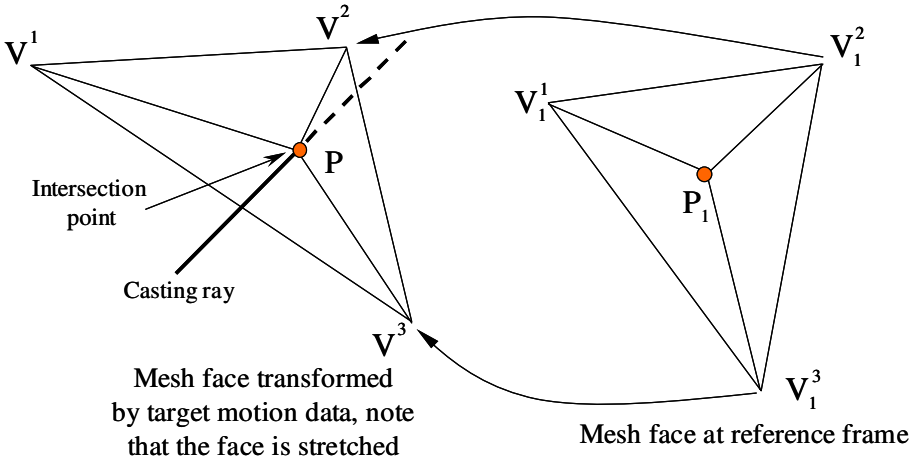


Figure 9.3: Step 2 of the pixel rendering process: mesh face is stretched because of the different motion weights of the vertices. This stretching has to be compensated when calculating P_1 from P.

Note that V_1^1, V_1^2, V_1^3 are known and V^1, V^2, V^3 can be calculated using the motion weights. Now since P lies on and inside the triangular patch formed by V^1, V^2, V^3, we have

$$P = a^1 V^1 + a^2 V^2 + a^3 V^3 \,, \tag{9.1}$$

where a^1, a^2, a^3 are constants lies between 0 and 1. Now we apply the same constants to the corresponding vertices V_1^1, V_1^2, V_2^3 at the reference frame as

$$P_1 = a^1 V_1^1 + a^2 V_1^2 + a^3 V_1^3 \,. \tag{9.2}$$

By substituting Equation (9.1) into Equation (9.2), P_1 is calculated by

$$P_1 = \begin{bmatrix} V_1^1 & V_1^2 & V_1^3 \end{bmatrix} \begin{bmatrix} V^1 & V^2 & V^3 \end{bmatrix}^{-1} P \,. \tag{9.3}$$

Step 3: Project P_j onto visible source images to get valid source pixel colors

After computing the source model points P_j, they are projected into the source color images to get a total of JK source pixel colors that *can* be used as the color of the target pixel. However, among these JK pixel colors, only those which come from frame j and camera k such that P_j is visible in camera k are valid. Here we suggest two different approaches to

test the visibility of P_j against the k^{th} camera at frame j: the 3D approach as proposed in Section 8.2.3, and the 2D z-buffer approach used in [FVFH92, VBK02]. The first approach checks the visibility directly by intersecting the line joining the point and the camera with the visibility space formed by the human articulated voxel model (see Section 8.2.3). Since this approach involves intersecting a line with a voxel model, it is relatively slow. The second approach first generates a 2D depth image of the voxel model at frame j of camera k using the z-buffer graphic hardware. The visibility of any point is then determined by first projecting the point into the depth image to get a depth value and then comparing this depth value with the distance of the point from the camera (details of the approach can be found in [FVFH92, VBK02]). This 2D z-buffer approach of testing visibility is faster than the 3D approach because the depth images can be generated in advance and the test only requires one 3D to 2D projection and one scalar (depth) comparison.

Step 4: Average the visible source pixel colors according to the viewing angles

The naive way for calculating the target pixel color is to simply average the set of all visible source pixel colors. There are, however, two problems associated with this naive approach. First of all, among the visible source pixels, some of them are at oblique directions between the source and the virtual cameras. In general these pixels are not reliable and should not be used in calculating the target pixel color. Secondly, due to variations in camera color responses, averaging *all* the visible pixel colors would lead to a "blur" effect and reduce the sharpness of the resulting target pictures.

To distinguish reliable source pixels from the unreliable ones, we use the viewing angles between the virtual camera, the the source cameras and target model points as weights to average the source pixel colors. For simplicity, the viewing angles are all calculated w.r.t. the reference frame. Figure 9.4 shows an example of computing the viewing angle for the source pixel color from the k^{th} camera at the j^{th} frame. The basic idea is to transform both the virtual camera $C^{virtual}$ and the k^{th} source camera C^k to the reference frame. Let the 6D transformations between P (the target frame) and P_1 (the reference frame) be $\boldsymbol{T}$ and that

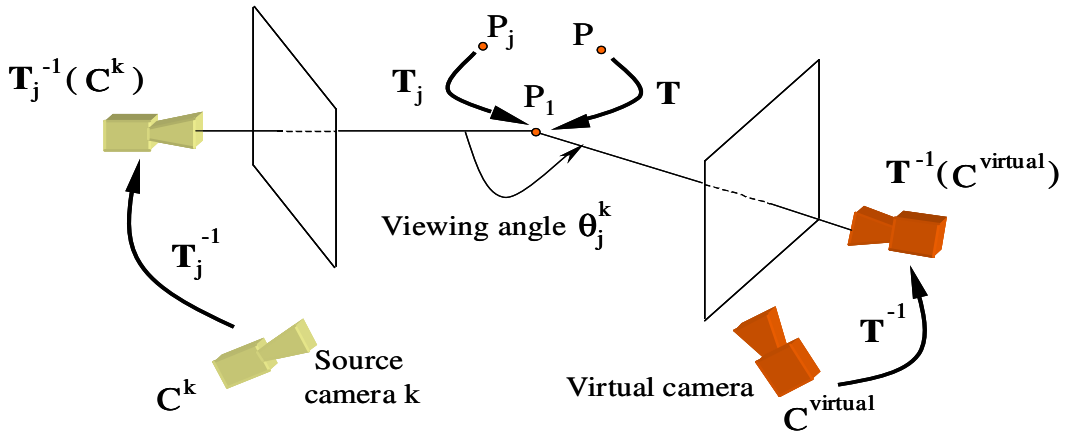


Figure 9.4: Step 4 of the pixel rendering process: computing the viewing angle between the virtual camera, the k^{th} source camera at the j^{th} frame and the target model point P.

between P_j (the j^{th} source frame) and P_1 be $\boldsymbol{T}_j$, i.e. $P_1 = \boldsymbol{T}(P)$ and $P_1 = \boldsymbol{T}_j(P_j)$ (Note that both $\boldsymbol{T}_j$ and $\boldsymbol{T}$ can be found from the target and the source motion data). Now the position of $C^{virtual}$ and C^k at the reference frame are given as $\boldsymbol{T}^{-1}(C^{virtual})$ and $\boldsymbol{T}_j^{-1}(C^k)$ respectively. The viewing angle θ_j^k is then calculated using the following equation:

$$\cos\theta_j^k = \frac{(\boldsymbol{T}_j^{-1}(C^k) - P_1)\cdot(\boldsymbol{T}^{-1}(C^{virtual}) - P_1)}{\|\boldsymbol{T}_j^{-1}(C^k) - P_1\|\|(\boldsymbol{T}^{-1}(C^{virtual}) - P_1\|} . \tag{9.4}$$

Note that since the reliability of the source pixel is inversely proportional to the viewing angle (i.e. the smaller the viewing angle, the more reliable the pixel is), we generate weights (see [DTM96, VBK02]) for each color pixel as $\frac{1}{1-\cos\theta_j^k}$. Once all the viewing angles are calculated, the *valid* source pixel colors are sorted according to their associated weights. The source pixel colors correspond to the n largest weights are then averaged using their weights to generate the target pixel color. The value of n is chosen to be small in order to get a sharp target picture.

9.1.3.3 Post-Processing of Rendered Image

Depends on the source images used, there may exist some target pixels whose pixel color cannot be determined. This happens when the 3D source model points P_j of a target pixel are invisible in all of the cameras. To fill in the color of a missing target pixel (or hole),

the average color of its neighboring pixels is used. Note that in the averaging process, only colors of those neighbors from the same body part as the hole pixel are used, so as to prevent the artifacts of color diffusion between body parts. Besides holes filling, shadows and background are added to the rendered image to increase photo-realism. For the sake of simplicity we use an easy planar surface shadowing technique [Bli88], though other more sophisticated soft shadow generation methods [WPF90] can also be used.

9.1.4 Experimental Results

Two sets of real data experiments are performed to study the performance of our rendering algorithm. Experiment I compares different ways of rendering the motion: (1) render the transformed voxel model directly, (2) render the transformed mesh model directly and (3) render using our Image-Based Articulated Model Rendering Algorithm. The effect of the number of averaging source pixels on the quality of the rendered target pictures is also studied in Experiment I. Experiment II investigates how well our rendering algorithm performs by comparing the rendered images with real images of the same person performing the same motion.

9.1.4.1 Experiment I

In the first rendering experiment, the kinematic model of SubjectE (see Figure 7.10(b)) is rendered with the PUNCH motion (which was used to test the tracking algorithm in Section 8.3.1). We use the ESTILL sequence (which was captured in Section 7.2 to build the voxel model of SubjectE) as the source sequence with a total of 240 source images (8 cameras with 30 frames per camera). Figure 9.5 shows rendered results of the 38^{th} frame of the target PUNCH motion. In the top row of the figure, the virtual camera is set to coincide with camera 3 of the source sequence. In the middle row the virtual camera is placed at a new position. The bottom row of the figure redisplays the portion of the top row images (where the face is) at a higher resolution for better visual comparison. Figures 9.5(a) and (b) respectively show results of direct rendering of the colored voxel and texture-mapped mesh

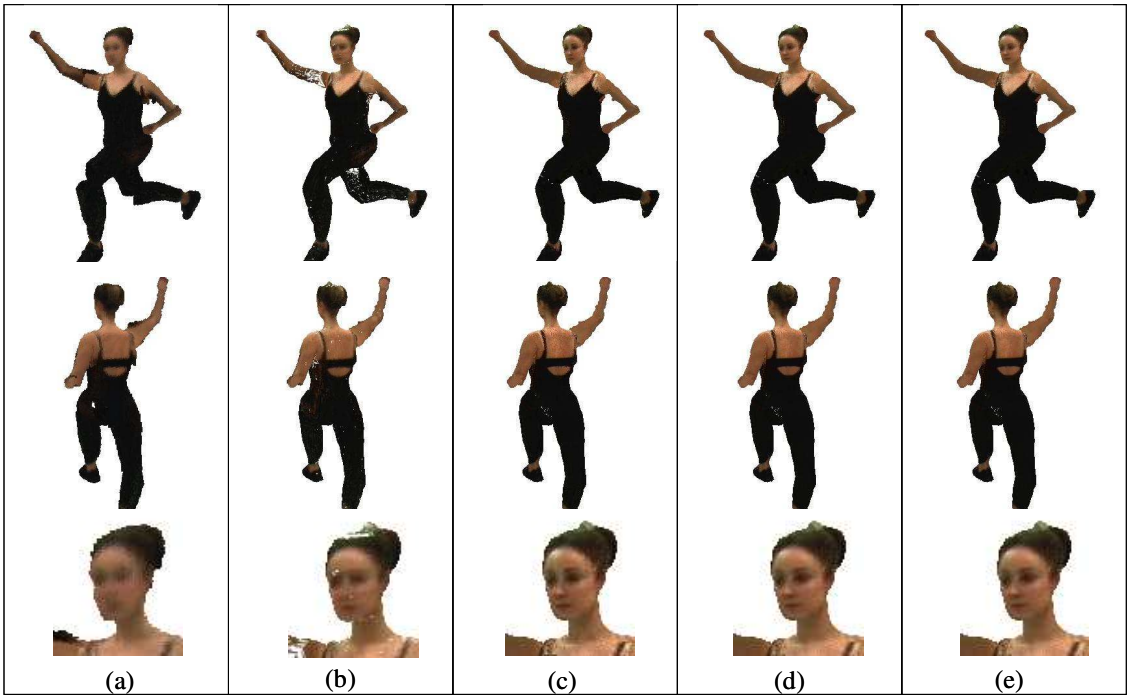


Figure 9.5: Images obtained by (a) direct rendering of the colored voxel model, (b) direct rendering of the texture-mapped mesh model, (c) using the Image-Based Articulated Model Rendering Algorithm with each target pixel color averaged from 1 source pixel, (d) 5 source pixels and (e) 9 source pixels. The top row shows results with the virtual camera set to coincide with camera 3 of the source sequence. The middle row shows results with the virtual camera placed at a new position. The bottom row redisplays the portion (the face of the person) of the images in the top row at a higher resolution for better visual comparison.

models using OpenGL. Figures 9.5(c)(d) and (e) show rendering results using the Image-Based Articulated Model Rendering Algorithm with each target pixel color averaged using 1, 5 and 9 source pixels (of highest weights) respectively.

Comparing Figures 9.5(a), (b) and (c), the images obtained using our rendering algorithm are much shaper than those obtained from direct rendering of either the voxel or mesh models, especially at detailed area such as the face. Moreover, the voxel model also suffers from joint discontinuity problem (see both knee joints in Figure 9.5(a)). Note that the white patches in Figure 9.5(b) represents the part of the mesh model where no texture can be obtained from the source sequence. From Figure 9.5(c), (d) and (e), it can be seen that the rendered images using different number of averaged source pixels are visually very similar

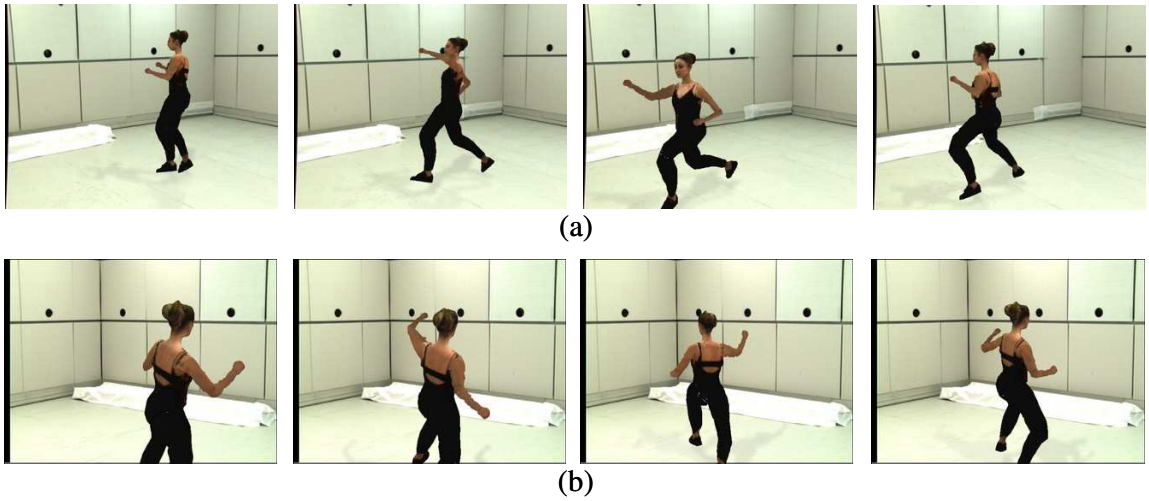
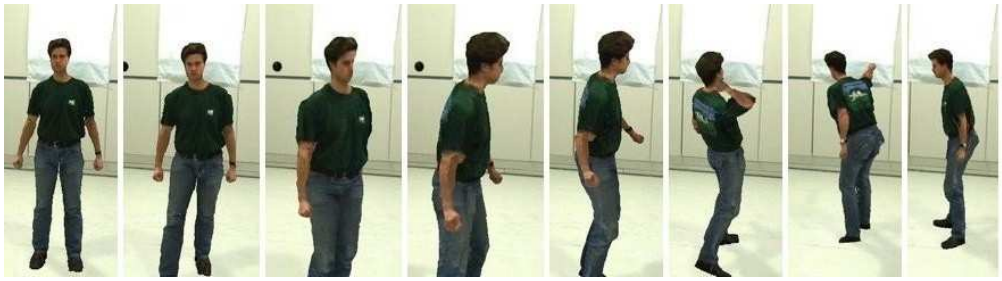


Figure 9.6: Selected frames of the SubjectE performing the PUNCH motion rendered using IBAMRA with the ESTILL sequence as the source sequence. One averaging pixel is used to generate these pictures. Background with soft shadows are added to increase the photo-realism of the images. In (a) the viewpoint is set as the same as camera 3 of the source sequence while a completely new viewpoint is used to generate pictures in (b). The rendered sequence from both viewpoints can be seen in the video clip **SubjectE-rendered-PUNCH.mpg**.

to each other, indicating that if the cameras are well color-balanced (which is the case in the ESTILL source sequence), our rendering algorithm is not sensitive to the number of averaged source pixels used . Note that due to occlusion, some of the target pixels do not have enough visible source pixels for averaging as set by the algorithm. In such cases, the color of the target pixel is obtained by averaging all the visible source pixel colors.

Figure 9.6 shows some selected frames of the rendered sequence (using IBAMRA with 1 averaging source pixel color) of SubjectE performing the PUNCH motion from two different camera viewpoints. The rendered sequences from both viewpoints, together with four of the eight source images sequences (ESTILL) and sequence of a stick figure illustrating the PUNCH motion are included in the video clip **SubjectE-rendered-PUNCH.mpg**[*].

[*]All movie clips of this chapter can be found at http://www.cs.cmu.edu/~german/research/Thesis/Video/Chapter9/



(a)

(b)

Figure 9.7: Comparison between rendered images and real images of SubjectS performing the THROW motion: (a) rendered images of the THROW motion using the SSTILL sequence as the source sequence, (b) corresponding images from the THROW sequence. It can be seen that the quality of the rendered images are comparable to the real images.

9.1.4.2 Experiment II

Our second experiment compares rendered images with real images of a person performing the same motion. We rendered SubjectS with the motion data tracked from the THROW sequence in Section 8.3.2 using the SSTILL sequence (captured in Section 7.2) as the source images. Some rendered frames with the virtual camera set to source camera 7 are shown in Figure 9.7(a). The corresponding real images of the THROW sequence from the same camera are shown in Figure 9.7(b) for comparison. It can be seen that the quality of the rendered images are comparable to the real images. The rendered images from two different camera viewpoints, together with the corresponding real images from the THROW sequence can be found in the video clip **SubjectS-rendered-THROW.mpg**.

Table 9.1 lists the averaged time required for each step of our rendering algorithm to generate the images in Experiment II. Note that in our implementation, the depth infor-

Processing Step	Approximate Time required
I. Pre-rendering Processing:	
Convert the voxel model to mesh model	4.2s
Smooth the mesh model	78.3s
II. Pixels Rendering	
Step 1: Intersect viewing ray	1.5s per image (640 x 480 pixels)
Step 2: Compute source model points	8.7s per image (640 x 480 pixels)
Step 3: Project model points and test visibility	258s per image (640 x 480 pixels)
Step 4: Get and average source pixels	244s per image (640 x 480 pixels)
III. Post-rendering processing:	
Add background and render shadows	1.7s per image (640 x 480 pixels)
Fill holes	0.26s per image (640 x 480 pixels)

Table 9.1: The approximate time required for each processing step of the Image-Based Articulated Object Rendering Algorithm.

mation used to test visibility is stored in the computer memory while the source images are accessed from the hard-disks whenever needed. This explains why Step 4 of the pixel rendering process requires much longer time than just averaging the source pixel colors. The timing results are obtained on a machine with a 750MHz Pentium CPU.

9.1.5 Applications

The most direct application of our rendering algorithm is to generate pictures of a person performing some ”faked” motion. Good examples are the results shown in Experiment I and II above. Our algorithm can also be applied to alter or edit video of human motions, similar to Seitz and Kutulakos’s idea of video editing in [SK98]. Another interesting application of our algorithm is to exchange motions between people.

Consider the scenario shown in Figure 9.8 where two people perform (separately) two different motions. Assume the motions are recorded by multiple cameras. The vision-based motion exchange idea is to render new videos of each person performing the motion of the other person, after building their articulated models and tracking their motions from the

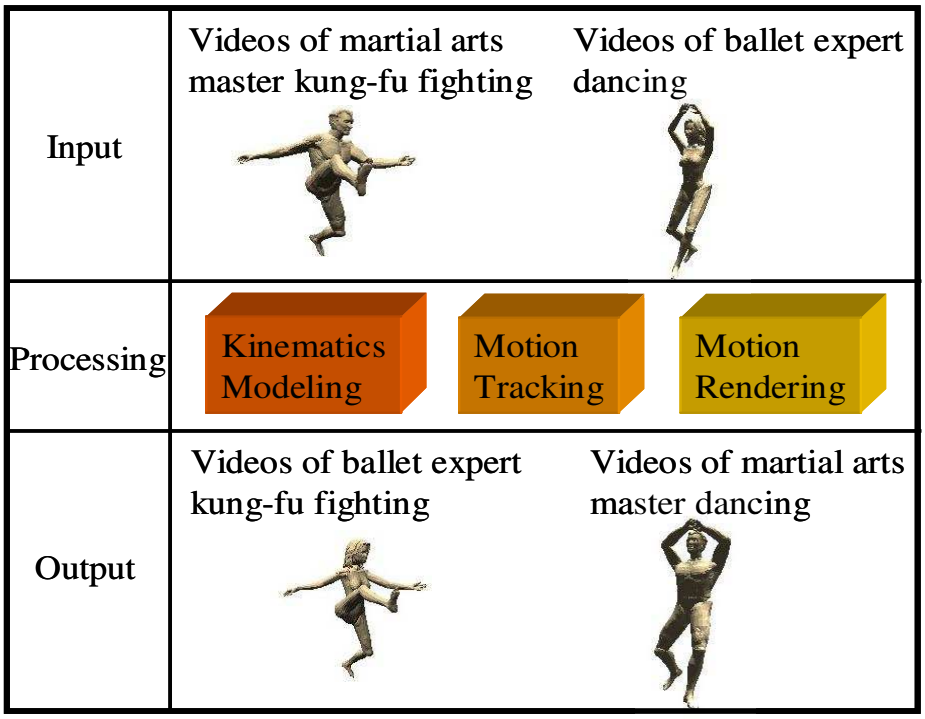


Figure 9.8: Motion Transfer between two people.

recorded video.

To illustrate this idea we exchange the motions KUNGFU, STEP-FLEX and THROW (which were all tracked in Section 8.3.2) of SubjectG , SubjectE and SubjectS. More specifically, the motion KUNGFU (originally performed by SubjectG) is transferred to SubjectS, the motion THROW (originally performed by SubjectS) is transferred to SubjectE and the motion STEP-FLEX (originally performed by SubjectE) is transferred to SubjectG. The sequences ESTILL, GSTILL and SSTILL are used as the source sequences and the target motions are smoothed before applying to the mesh models. Figure 9.9(a) shows some of the images from the rendered sequence of SubjectE performing the THROW motion while Figure 9.9(b) shows the images from the original THROW sequence of SubjectS. Despite some visual artifacts, it can be seen that we successfully transfer the motions from SubjectS to SubjectE. The video clips **SubjectE-transfer-THROW.mpg** shows some of the source images, the THROW motion and rendering results from different fixed and moving virtual camera view-points.

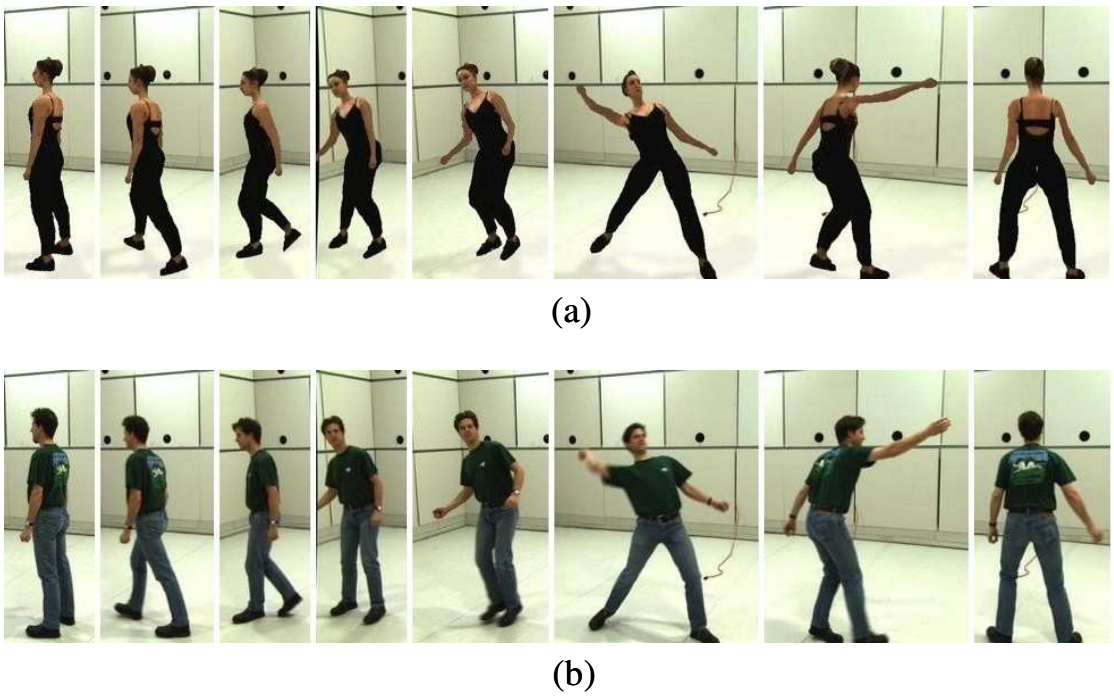


Figure 9.9: The THROW motion is transferred from (b) SubjectS to (a) SubjectE. The whole sequence can be found in the video clip **SubjectE-transfer-THROW.mpg**.

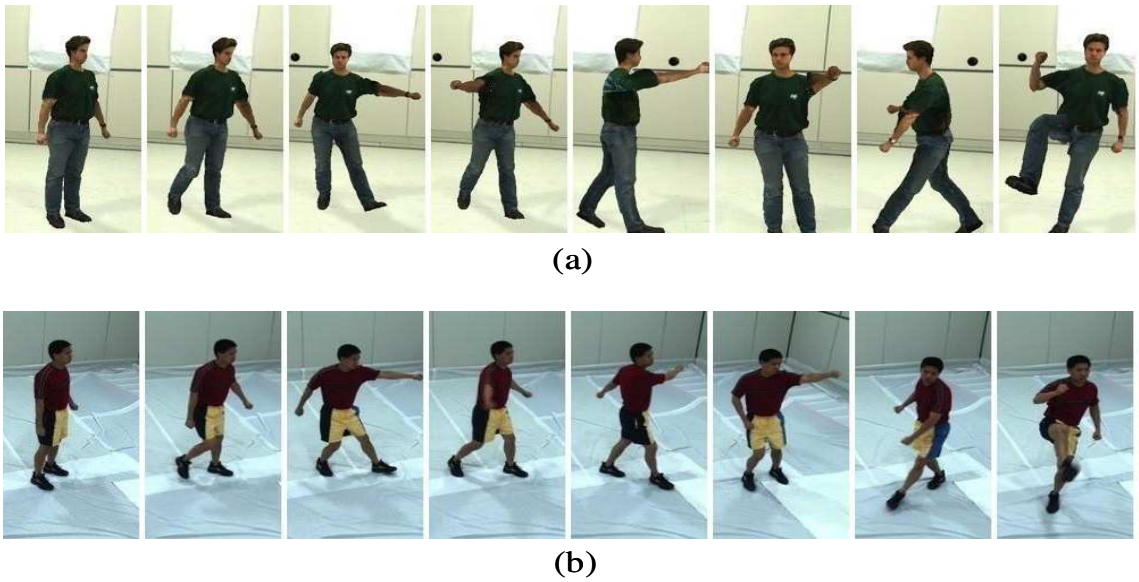


Figure 9.10: The KUNGFU motion is transferred from (b) SubjectG to (a) SubjectS. The whole sequence can be found in the video clip **SubjectS-transfer-KUNGFU.mpg**.

(a)

(b)

Figure 9.11: The STEP-FLEX motion is transferred from (b) SubjectE to (a) SubjectG. The whole sequence can be found in the video clip **SubjectG-transfer-STEP-FLEX.mpg**.

Similarly the rendered and original images of transferring the KUNGFU motion from SubjectG to SubjectS and that of STEP-FLEX motion from SubjectE to SubjectG are shown in Figures 9.10 and 9.11 and respectively in the video clips **SubjectS-transfer-KUNGFU.mpg** and **SubjectG-transfer-STEP-FLEX.mpg**.

9.2 Related Work

In work concurrent with this paper, Carranza et al. rendered tracked human motion using a view-dependent texture mapping algorithm which is similar to our pixel rendering algorithm [CTMS03]. However, they do not render the person with any new motion or perform motion transfer. In [VBK02], Vedula et al. proposed an image-based spatial and temporal view interpolation algorithm for non-rigid dynamic events using scene flow [VBR+99, VBSK00]. There are two major differences between our algorithm and the one

in [VBK02]. The first difference lies in the type of models used. In our rendering algorithm, we assume the human model consists of articulated rigid body parts, each with rigid motions while in [VBK02], the authors assume the human is totally non-rigid with motions represented by the scene flow. Moreover, Vedula et al. aim at "interpolating" the motion spatially and temporally while we focus on "extrapolating" the human body model with new motions. Note that some of the speed optimization techniques in [VBK02] and the original view-dependent texture mapping idea by Debevec et al. in [DTM96] are adopted in our algorithm.

9.3 Discussion

The recording and replaying of static and dynamic events have been a very active research topic in computer vision and graphics over the last ten years [DTM96, KRN97, RNK97, Sei97, KSV98, Deb98, NRK98, SBK+99, BSV+00]. In the case of dynamic events, all of the above systems replayed the events according to what have actually happened. Although there is research on removing/adding objects/persons [IRP94, SK98, TSA01, JFB02] from an video sequence, and recently work on increasing the spatial and/or temporal resolution of rendering a dynamic event [Ved01, VBK02, SCI02], little attempts has been made to replay an "altered" dynamic event. Here "altered" means the course or motions of the persons or objects in the replay are changed and different from what have actually happened (and recorded) in the video sequences. The ability to replay an altered dynamic sequence is useful in a lot of entertainment and visual effects applications. For example, in choreographing a dancing performance, the choreographer can see the effect of adjusting the positions of the body/arms/legs of the dancers easily without asking them to re-perform the dance. Another extreme example is in movie making when the director can change a scene of the movie without re-shooting it.

While replaying an event truthfully and photo-realistically from a new virtual viewpoint

is already a challenging task, replaying an altered event is even more difficult as it involves solving a lot of difficult computer vision and graphics problems: how to accurately track people/objects in the video, how to remove/add objects/people to the scene [IRP97, SK98, JF01, TSA01], how to estimate and simulate the lighting conditions [SSI99, SSI03] and how to render changed motion photo-realistically. The purpose of this chapter is to provide a feasible solution to the last problem. We have shown that with an accurate articulated model and video sequences (with tracked motion) of a person, videos of the person performing new or altered motion can be created using our image-based rendering algorithm. Compared to direct rendering of a texture-mapped model, our algorithm is able to generate much more photo-realistic looking pictures.

There are two factors which cause the visual artifacts in the images rendered using our algorithm. First, although we have adopted the weighted sum approach for transforming the mesh vertices between body parts, the transition are not smooth enough when the motion is large. This problem can be solved if more sophisticated skinning methods such as those in [SK00, WP02] are used. Secondly, since the position of the 3D source model point (which in turn determine the source pixel color) in each source frame is calculated using the motion data of the source image sequence, any tracking errors in the source motion data would cause the algorithm to pick the wrong source pixel color. To reduce artifacts caused by this problem, it is important to make sure the motion of the person in the source sequence is tracked correctly. For this reason, source sequences with simple and easily tracked motions (such as the STILL sequences) are better than those with complex motions.

Chapter 10

Conclusion

The ultimate goal of this thesis is to study how traditional Shape-From-Silhouette methods can be improved to apply better to the problems of human articulated body modeling, motion tracking and rendering. Two problems, non-realtime reconstruction speed and coarse shape approximation (caused by an inadequate number of cameras), which prohibited the effective use of traditional SFS algorithms in human related applications were considered. We addressed the first problem by proposing a fast testing/projection algorithm (SPOT) for voxel-based SFS algorithms. To deal with the second problem, we combined silhouette information over time to effectively increase the number of cameras without physically adding new cameras. The resulting temporal SFS algorithm was first developed for rigid objects and then extended to multiple and articulated objects. These temporal SFS algorithms were applied to acquire kinematic (shape and joint positions) models of humans. Once the articulated human models were built, they were used for motion tracking (analysis) and motion rendering (synthesis).

10.1 Thesis Contributions

The contributions of this thesis can be categorized into two areas: (1) theoretical analysis, improvement of and extensions to Shape-From-Silhouette and (2) practical applications of the improved algorithms to human related problems.

Theoretical Contributions to Shape-From-Silhouette:

- **Analysis of silhouette image noise.** The effect of noisy silhouette images on the accuracy of the traditional voxel-based SFS method was analyzed. Based on the analysis, a fast testing algorithm SPOT for voxel-based SFS methods was derived. The main contribution of SPOT is that it provides a speed-optimized strategy for reconstruction given a required accuracy ratio and knowledge of the silhouette noise statistics.

- **Visual Hull representation by Bounding Edges.** We defined the notion of a Bounding Edge and proposed using it as a representation of a Visual Hull. We also showed that the Second Fundamental Property of Visual Hulls (2^{nd} FPVH) is closely related to the definition of a Bounding Edge. This thesis established and emphasized the importance of the 2^{nd} FPVH (which previously has often been overlooked by SFS researchers) in shape reconstruction as it expresses one important aspect of how the shape information of the object is stored in the silhouette images.

- **Theoretical study of Visual Hull alignment ambiguity.** We showed that the problem of aligning two Visual Hulls using silhouette images only is inherently ambiguous and is governed by a set of geometric (shape) constraints. Also, by expressing the constraints in terms of Bounding Edges and silhouette images, an inconsistent alignment between two Visual Hulls can be easily rejected using simple tests.

- **Temporal Shape-From-Silhouette algorithm for rigid objects.** One of the major contributions of this thesis is the development of the temporal SFS algorithm for rigid objects. The algorithm extracts Colored Surface Points (CSPs) on the surface of the object using stereo and applies both geometric (shape) constraints and color consistency of the CSPs to resolve the alignment ambiguity. Once the motions of the object are recovered, a refined shape of the object is reconstructed using all of the silhouette images (after the motion is compensated for) captured over time. Our algorithm not only extends traditional SFS over time, but it also combines the key advantages of the two important and *complementary* [BSK01] shape reconstruction principles: Shape-From-Silhouette and Stereo.

- **Temporal Shape-From-Silhouette algorithm for articulated objects.** Using the Expectation-Maximization paradigm, the temporal SFS algorithm was extended to articulated objects. It first iteratively segments and aligns the extracted CSPs to estimate the shape and motion of individual parts of the articulated object. Once the motions of all parts are recovered, the joint positions are estimated using articulation constraints. The extension to include articulated objects is a critical contribution as it allows us to apply the temporal SFS algorithms to the human body (which approximately consists of articulated parts) for kinematic information acquisition.

Practical Contributions to Human Related Applications:

- **Real-time 3D voxel motion reconstruction system.** Based on SPOT, we have built a real-time system which reconstructs rough 3D voxel models of a person and fits ellipsoidal shells as body parts to the model. This system has inspired the construction of other real-time people tracking/posture estimation systems [MHTC01, LSS02].

- **Articulated human body modeling.** This thesis proposed a step-by-step procedure to acquire articulated models (with accurate shape information and joint location) of human bodies using only cameras and an uncalibrated turn-table. Our vision-based procedure provides an inexpensive alternative for human body modeling to expensive laser-scanning based commercial systems (most at which acquire body shape without the joint information).

- **Vision-based human motion tracking from video sequences.** Another contribution of this thesis is that we successfully applied the temporal SFS algorithms (of articulated objects) to the difficult problem of non-invasive human motion tracking. We built a practical vision-based system which emphasized two important aspects: (1) using detailed person-specific body shape models and (2) combining geometric and photometric constraints during tracking. The system accurately tracks complicated human motions from video sequences without using markers of any kind.

- **Image-based articulated model rendering for motion editing and transfer.** This thesis has demonstrated a practical motion transfer system using recorded video sequences of a person to generate photo-realistic images of the same person performing completely different motion. The idea is realized in practice through the Image-Based Articulated Model Rendering Algorithm. We showed that convincing images of a person performing complicated motions can be synthesized from video sequences of the person performing simple motions.

10.2 Future Work

In Chapter Three we studied the effect of noisy silhouettes on the accuracy of voxel-based Shape-From-Silhouette. Besides silhouette noise, the shape estimation accuracy of SFS is also greatly affected by errors in the camera projection parameters. For SFS at a single time instant, the projection errors come from inaccurate camera calibration. For temporal

SFS with silhouettes captured at different time instants, there are additional errors caused by mis-alignment. An in-depth theoretical and quantitative analysis of how SFS is affected by camera projection errors (similar to what we have done with silhouette noise) would be very useful in deriving an optimal or robust SFS algorithm for both calibration errors at a single time, and alignment errors across time. Note that using Tsai's camera model [Tsa87], Niem has done a preliminary but limited analysis of camera calibration errors on SFS in his paper [Nie97]. This paper can be used as a starting point for such a study.

While our temporal SFS algorithm can be used to recover the motion and shape of moving rigid and articulated rigid objects, a lot of naturally occurring objects are non-rigid and/or are deformable. A logical future direction is to extend our temporal SFS algorithms to deformable objects such as a piece of cloth or a crawling caterpillar. There are two major difficulties in extending temporal SFS to non-rigid objects. The first difficulty, which is common to other surface-point-based 3D shape/motion estimation methods [ACLS94], is to assume suitable deformable shape and motion models for the object. The choice of deformable model is critical and depends on the application it may not be an easy task. The second difficulty is caused by the fact that since our temporal SFS algorithm is not feature-based, the CSPs are not tracked over time and there is no point-to-point correspondence between two sets of CSPs extracted at different instants. Hence it is unclear how the chosen deformable model can be applied to the CSPs across time. Despite these difficulties, however, the possibility of extending temporal SFS to non-rigid objects is worth studying as it would help to solve important non-rigid tracking problems in computer vision.

Although our human motion tracking algorithm works well, it suffers from the problem of local minimum which is common in methods that use the error optimization formulation. In Chapter 8 we suggested to include joint angles limits to minimize the problem of local minimum. This line of future work is briefly described below. Prior to tracking, the allowable range of motions (of all the limbs due to the joint angles limits) of the person is recorded or estimated. The space of all joint parameters is then divided into the valid and

invalid workspaces. This a priori workspace information can then be incorporated into the tracking optimization equations by adding very high errors to the error criterion when the body joint angles are in the invalid workspace, while no extra error is added when the joint angles are in the valid zones. This approach would effectively remove the local minimum vulnerability of our tracking algorithm to the first and second situations discussed in 8.2.5.

The final proposed future work is to improve the implementation of the Image-Based Articulated Model Rendering Algorithm. Possible areas of improvement include better mesh smoothing methods (after the voxel model is converted to the surface model), better skinning techniques to compute the motion around the joints, motion data re-targeting and better hole filling methods in the post-rendering processing.

Bibliography

[ACLS94] J. Aggarwal, Q. Cai, W. Liao, and B. Sabata. Articulated and elastic non-rigid motion: A review. In *Proceedings of IEEE Workshop on Motion of Non-rigid and Articulated Objects'94*, pages 16–22, 1994.

[ACP03] B. Allen, B. Curless, and Z. Popovic. The space of human body shapes: Reconstruction and parameterization from range scans. In *Computer Graphics Annual Conference Series (SIGGRAPH'03)*, pages 587–594, San Diego, CA, July 2003.

[AV89] N. Ahuja and J. Veenstra. Generating octrees from object silhouettes in orthographic views. *IEEE Transactions Pattern Analysis and Machine Intelligence*, 11(2):137–149, February 1989.

[Bau74] B.G. Baumgart. *Geometric Modeling for Computer Vision*. PhD thesis, Stanford University, 1974.

[BDC01] A. Broadhurst, T. Drummond, and R. Cipolla. A probabilistic framework for space carving. In *Proceedings of International Conference on Computer Vision (ICCV'01)*, Vancouver, Canada, June 2001.

[BK99] D. Beymer and K. Konolige. Real-time tracking of multiple people using stereo. In *Proceedings of International Conference on Computer Vision (ICCV'99)*, Corfu, Greece, September 1999.

[BK00] C. Barron and I. Kakadiaris. Estimating anthropometry and pose from a single image. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR'00)*, Hilton Head Island SC, June 2000.

[BL00] A. Bottino and A. Laurentini. Non-intrusive silhouette based motion capture. In *Proceedings of the Fourth World Multiconference on Systemics, Cybernetics and Informatics SCI 2001*, pages 23–26, July 2000.

[BL01] A. Bottino and A. Laurentini. Interactive reconstruction of 3d objects from silhouette. In *Proceedings of the 9th International Conference on Computer Graphics, Visualization and Computer Vision (WSCG'2001)*, pages 5–9, February 2001.

[Bli82] J. Blinn. A generalization of algebraic surface drawing. *ACM Transactions on Graphics*, 1(3):235–256, 1982.

[Bli88] J. Blinn. Me and my (fake) shadow. *IEEE Computer Graphics and Applications*, 8(1):82–86, January 1988.

[BM92] P. Besl and N. McKay. A method of registration of 3D shapes. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 14(2):239–256, February 1992.

[BM97] C. Bregler and J. Malik. Video motion capture. Technical Report CSD-97-973, University of California Berkeley, 1997.

[BM98] C. Bregler and J. Malik. Tracking people with twists and exponential map. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR'98)*, volume 1, pages 8–15, Santa Barbara, CA, June 1998.

[BMM01] C. Buehler, W. Matusik, and L. McMillan. Polyhedral visual hulls for real-time rendering. In *Proceedings of the 12th Eurographics Workshop on Rendering*, 2001.

[BMMG99] C. Buehler, W. Matusik, L. McMillan, and S. Gortler. Creating and rendering image-based visual hulls. Technical Report MIT-LCS-TR-780, MIT, 1999.

[BSK01] S. Baker, T. Sim, and T. Kanade. A characterization of inherent stereo ambiguities. In *Proceedings of International Conference on Computer Vision (ICCV'01)*, Vancouver, Canada, June 2001.

[BSV+00] S. Baba, H. Saito, S. Vedula, K. Cheung, and T. Kanade. Appearance-based virtual-view generation for fly through in a real dynamic scene. In *Proceedings of IEEE TCVG Symposium on Visualization (VisSym'00)*, May 2000.

[CA96] Q. Cai and J. Aggarwal. Tracking human motion using multiple cameras. In *Proceedings of International Conference on Pattern Recognition (ICPR'96)*, volume 3, pages 68–72, August 1996.

[CA98] Q. Cai and J. Aggarwal. Automatic tracking of human motion in indoor scenes across multiple synchronized video streams. In *Proceedings of the Sixth International Conference on Computer Vision (ICCV'98)*, Bombay, India, January 1998.

[CBK03] G. Cheung, S. Baker, and T. Kanade. Visual hull alignment and refinement across time:a 3D reconstruction algorithm combining shape-frame-silhouette with stereo. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR'03)*, Madison, MI, June 2003.

[CKBH00] G. Cheung, T. Kanade, J. Bouquet, and M. Holler. A real time system for robust 3D voxel reconstruction of human motions. In *Proceedings of IEEE*

Conference on Computer Vision and Pattern Recognition (CVPR'00), Hilton Head Island, SC, June 2000.

[Coe98] M. Coen. Design principals for intelligent environments. In *Proceedings of AAAI Spring Symposium on Intelligent Environments*, Stanford, CA, 1998.

[CR99a] T. Cham and J. Rehg. A multiple hypothesis approach to figure tracking. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR'99)*, Ft. Collins, CO, June 1999.

[CR99b] T. Cham and J. Rehg. Dynamic feature ordering for efficient registration. In *Proceedings of International Conference on Computer Vision (ICCV'99)*, Corfu, Greece, September 1999.

[CTI] Cyra technologies inc. http://www.cyra.com.

[CTMS03] J. Carranza, C. Theobalt, M. Magnor, and H. Seidel. Free-viewpoint video of human actors. In *Computer Graphics Annual Conference Series (SIGGRAPH'03)*, pages 569–577, San Diego, CA, July 2003.

[CW93] S. Chen and L. Williams. View interpolation for image synthesis. In *Computer Graphics Annual Conference Series (SIGGRAPH'93)*, pages 279–285, 1993.

[CYB] Cybearware. http://www.cyberware.com.

[DBR00] J. Deutscher, A. Blake, and I. Reid. Articulated body motion capture by annealed particle filtering. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR'00)*, Hilton Head Island, SC, June 2000.

[DC01] T. Drummond and R. Cipolla. Real-time tracking of highly articulated structures in the presence of noisy measurements. In *Proceedings of International Conference on Computer Vision (ICCV'01)*, pages 315–320, Vancouver, Canada, June 2001.

[DCR99] D. DiFranco, T. Cham, and J. Rehg. Recovering of 3D articulated motion from 2d correspondences. Technical Report CRL 99/7, Compaq Cambridge Research Laboratory, 1999.

[DCR01] D. Difranco, T. Cham, and J. Rehg. Reconstruction of 3D figure motion from 2D correspondences. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR'01)*, Kauai, HI, December 2001.

[Deb98] P. Debevec. Combining dynamic simulation, high dynamic range photography and global illumination. In *Computer Graphics Annual Conference Series (SIGGRAPH'98)*, 1998.

[DF99] Q. Delamarre and O. Faugeras. 3D articulated models and multi-view tracking with silhouettes. In *Proceedings of International Conference on Computer Vision (ICCV'99)*, Corfu, Greece, September 1999.

[DLR77] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of Statistical Society*, B 39:1–38, 1977.

[DS83] J. Dennis and R. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Prentice Hall, Englewood Cliffs, NJ, 1983.

[DTM96] P. Debevec, C. Taylor, and J. Malik. Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach. In *Computer Graphics Annual Conference Series (SIGGRAPH'96)*, pages 11–20, 1996.

[DV99] J. DeBonet and P. Viola. Roxels: Responsibility weighted 3D volume reconstruction. In *Proceedings of International Conference on Computer Vision (ICCV'99)*, Corfu, Greece, September 1999.

[EHD99] A. Elgammal, D. Harwood, and L. Davis. Non-parametric model for background subtraction. In *Proceedings of International Conference on Computer Vision (ICCV'99), Frame-rate Workshop*, September 1999.

[FGDP02] P. Fua, A. Gruen, N. D'Apuzzo, and R. Plänkers. Markerless full body shape and motion capture from video sequences. *International Archives of Photogrammetry and Remote Sensing*, 34(5):256–261, 2002.

[FHPB00] P. Fua, L. Herda, R. Plänkers, and R. Boulic. Human shape and motion recovery using animation models. In *XIX ISPRS Congress*, July 2000.

[FVFH92] J. Foley, A. VanDam, S. Feiner, and J. Hughes. *Computer Graphics: Principle and Practice*. Addison Wesley, second edition, 1992.

[GD96] G. Gavrila and L. Davis. Tracking of humans in action : 3D model-based approach. In *ARPA Image Understanding Workshop 1996*, February 1996.

[GGSC96] S. Gortler, R. Grzeszczuk, R. Szeliski, and M. Cohen. The lumigraph. In *Computer Graphics Annual Conference Series (SIGGRAPH'96)*, pages 43–54, 1996.

[HHD98] I. Haritaoglu, D. Harwood, and L. S. Davis. W4 : Who? when? where? what? a real time system for detecting and tracking people. In *Proceedings of IEEE International Conference on Automatic Face and Gesture Recognition (ICAFGR'98)*, Nara, Japan, April 1998.

[HHD99] T. Horprasert, D. Harwood, and L. Davis. A statistical approach for real-time robust background subtraction and shadow detection. In *Proceedings of Inter-*

national Conference on Computer Vision (ICCV'99), Frame-rate Workshop, September 1999.

[IBL00] Y. Ivanov, A. Bobick, and J. Liu. Fast lighting independent background subtraction. *International Journal on Computer Vision*, 37(2):199–207, 2000.

[IHA02] M. Irani, T. Hassner, and P. Anandan. What does the scene look like from a scene point? In *Proceedings of European Conference on Computer Vision (ECCV'02)*, pages 883–897, Copenhagen, Denmark, May 2002.

[IRP94] M. Irani, B. Rousso, and S. Peleg. Computing occluding and transparent motions. *International Journal on Computer Vision*, 12(1):5–16, January 1994.

[IRP97] M. Irani, B. Rousso, and S. Peleg. Recovery of ego-motion using region alignment. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(3):268–272, 1997.

[Jai89] A. Jain. *Fundamentals of Digital Image Processing*. Prentice Hall, 1989.

[JAP94] T. Joshi, N. Ahuja, and J. Ponce. Towards structure and motion estimation from dynamic silhouettes. In *Proceedings of IEEE Workshop on Motion of Non-rigid and Articulated Objects*, pages 166–171, November 1994.

[JAP95] T. Joshi, N. Ahuja, and J. Ponce. Structure and motion estimation from dynamic silhouettes under perspective projection. Technical Report UIUC-BI-AI-RCV-95-02, University of Illinois Urbana Champaign, 1995.

[JBJ01] D Jacobs, P. Belhumeur, and I. Jermyn. Judging whether multiple silhouettes can come from the same object. In *Proceedings of the 4th International Workshop on Visual Form*, Capri, Italy, 2001.

[JBY96] S. Ju, M. Black, and Y. Yacoob. Cardboard people: A parameterized model of articulated image motion. In *Proceedings of IEEE International Conference*

on Automatic Face and Gesture Recognition (ICAFGR'96), Vermont, USA, October 1996.

[JF01] N. Jojic and B. Frey. Learning flexible sprites in video layers. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR'01)*, Kauai, HI, December 2001.

[JFB02] A. Jepson, D. Fleet, and M. Black. A layered motion representation with occlusion and compact spatial support. In *Proceedings of European Conference on Computer Vision (ECCV'02)*, volume 1, pages 692–706, Copenhagen, Denmark, May 2002.

[JH97] A. Johnson and M. Hebert. Control of polygonal mesh resolution for 3D computer vision. Technical Report CMU-RI-TR-96-20, Carnegie Mellon University, Pittsburgh, PA, April 1997.

[JTH99] N. Jojic, M. Turk, and T. Huang. Tracking self-occluding articulated objects in dense disparity maps. In *Proceedings of International Conference on Computer Vision (ICCV'99)*, Corfu, Greece, September 1999.

[KA86] Y. Kim and J. Aggarwal. Rectangular parallelepiped coding: A volumetric representation of three dimensional objects. *IEEE Journal of Robotics and Automation*, RA-2:127–134, 1986.

[KK01] Q. Ke and T. Kanade. A subspace approach to layer extraction. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR'01)*, Kauai, HI, December 2001.

[KM95] I. Kakadiaris and D. Metaxas. 3D human body model acquisition from multiple views. In *Proceedings of International Conference on Computer Vision (ICCV'95)*, pages 618–623, Cambridge MA, June 1995.

[KM98] I. Kakadiaris and D. Metaxas. 3D human body model acquisition from multiple views. *International Journal on Computer Vision*, 30(3):191–218, 1998.

[KMB94] I. Kakadiaris, D. Metaxas, and R. Bajcsy. Active part-decomposition, shape and motion estimation of articulated objects: A physics-based approach. Technical Report IRCS Report 94-18, University of Pennsylvania, 1994.

[KRN97] T. Kanade, P. Rander, and P. Narayanan. Virtualized reality: Constructing virtual worlds from real scenes. *IEEE Computer Society Multimedia*, 4(1):34–47, March 1997.

[KS00] K. Kutulakos and S. Seitz. A theory of shape by space carving. *International Journal of Computer Vision*, 38(3):199–218, 2000.

[KSV98] T. Kanade, H. Saito, and S. Vedula. The 3D room: Digitizing time-varying 3D events by synchronized multiple video streams. Technical Report CMU-RI-TR-98-34, Carnegie Mellon University, 1998.

[KYS01] N. Krahnstoever, M. Yeasin, and R. Sharma. Automatic acquisition and initialization of kinematic models. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR'01), Technical Sketches*, Kauai, HI, December 2001.

[KYS03] N. Krahnstoever, M. Yeasin, and R. Sharma. Automatic acquisition and initialization of articulated models. In *To be appeared in Machine Vision and Applications*, 2003.

[Lau91] A. Laurentini. The visual hull : A new tool for contour-based image understanding. In *Proceedings of the Seventh Scandinavian Conference on Image Analysis*, pages 993–1002, 1991.

[Lau94] A. Laurentini. The visual hull concept for silhouette-based image understanding. *IEEE Transactions Pattern Analysis and Machine Intelligence*, 16(2):150–162, February 1994.

[Lau95] A. Laurentini. How far 3D shapes can be understood from 2D silhouettes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(2):188–195, February 1995.

[Lau97] A. Laurentini. How many 2d silhouettes does it take to reconstruct a 3D object? *Computer Vision and Image Understanding*, 67(1):81–87, 1997.

[Lau99] A. Laurentini. The visual hull of curved objects. In *Proceedings of International Conference on Computer Vision (ICCV'99)*, Corfu, Greece, September 1999.

[LBP01] S. Lazebnik, E. Boyer, and J. Ponce. On computing exact visual hulls of solids bounded by smooth surfaces. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR'01)*, Kauai HI, December 2001.

[LC87] W. Lorensen and H. Cline. A high resolution 3D surface reconstruction algorithm. *Computer Graphics*, 21(4):163–169, 1987.

[LC01] D. Liebowitz and S. Carlsson. Uncalibrated motion capture exploiting articulated structure constraints. In *Proceedings of International Conference on Computer Vision (ICCV'01)*, Vancouver, Canada, June 2001.

[LH96] M. Levoy and M. Hanrahan. Light field rendering. In *Computer Graphics Annual Conference Series (SIGGRAPH'96)*, 1996.

[LSS02] M. Li, H. Schirmacher, and H. Seidel. Combining stereo and visual hull for on-line reconstruction of dynamic scenes. In *Proceedings of IEEE 2002*

Workshop on Multimedia and Signal Processing, St. Thomas, Virgin Islands, December 2002.

[LY95] M. Leung and Y. Yang. First sight : A human body outline labeling system. *IEEE Transactions Pattern Analysis and Machine Intelligence*, 17(4):359–377, April 1995.

[LZG98] M. Lucente, G. Zwart, and A. George. Visualization space: A testbed for deviceless multimodal user interface. In *Proceedings of AAAI Spring Symposium on Intelligent Environments*, Stanford, CA, 1998.

[MA83] W. Martin and J. Aggarwal. Volumetric descriptions of objects from multiple views. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5(2):150–174, March 1983.

[MAC] Motion analysis corporation. http://www.motionanalysis.com.

[Mat01] W. Matusik. Image-based visual hulls. Master's thesis, Massachusetts Institute of Technology, 2001.

[MBR+00] W. Matusik, C. Buehler, R. Raskar, S. Gortler, and L. McMillan. Image-based visual hulls. In *Computer Graphics Annual Conference Series (SIGGRAPH'00)*, New Orleans, LA, July 2000.

[McM97] L. McMillan. *An Image-Based Approach to Three-Dimensional Computer Graphics*. PhD thesis, Department of Computer Science, University of North Carolina at Chapel Hill, Chapel Hill, North Carolina, 1997.

[MET] Meta motion. http://www.metamotion.com.

[MG01] T. Moeslund and E. Granum. A survey of computer vision-based human motion capture. *Computer Vision and Image Understanding: CVIU*, 81(3):231–268, 2001.

[MHTC01] I. Mikic, E. Hunter, M. Trivedi, and P. Cosman. Articulated body posture estimation from multi-camera voxel data. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR'01)*, Kauai, HI, December 2001.

[MLS94] R. Murray, Z. Li, and S. Sastry. *A Mathematical Introduction to Robotic Manipulation*. CRC Press, 1994.

[MTG97] S. Moezzi, L. Tai, and P. Gerard. Virtual view generation for 3D digital video. *IEEE Computer Society Multimedia*, 4(1), January-March 1997.

[MTHC03] I. Mikic, M. Trivedi, E. Hunter, and P. Cosman. Human body model acquisition and tracking using voxel data. *International Journal on Computer Vision*, 53(3):199–223, July 2003.

[MWC00] P. Mendonca, K. Wong, and R. Cipolla. Camera pose estimation and reconstruction from image profiles under circular motion. In *Proceedings of European Conference on Computer Vision (ECCV'00)*, pages 864–877, Dublin, Ireland, June 2000.

[MWC01] P. Mendonca, K. Wong, and R. Cipolla. Epipolar geometry from profiles under circular motion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(6):604–616, June 2001.

[NFA88] H. Noborio, S. Fukuda, and S. Arimoto. Construction of the octree approximating three-dimensional objects by using multiple views. *IEEE Transactions Pattern Analysis and Machine Intelligence*, 10(6):769–782, November 1988.

[Nie97] W. Niem. Error analysis for silhouette-based 3d shape estimation. In *Proceedings of International Workshop on Synthetic-Natural Hybrid Coding and Three Dimensional Imaging (IWSNHC3DI'97)*, pages 5–9, September 1997.

[NRK98] P. Narayanan, P. Rander, and T. Kanade. Constructing virtual worlds using dense stereo. In *Proceedings of the Sixth International Conference on Computer Vision (ICCV'98)*, pages 3–10, Bombay, India, January 1998.

[OBBH00] J. O'Brien, R. Bodenheimer, G. Brostow, and J. Hodgins. Automatic joint parameter estimation from magnetic motion capture data. In *Proceedings of Graphics Interface'00*, pages 53–60, May 2000.

[OK93] M. Okutomi and T. Kanade. A multiple-baseline stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(4):353–363, 1993.

[PD98] A. Prock and C. Dyer. Towards real-time voxel coloring. In *In Proceedings of Image Understanding Workshop, 1998.*, 1998.

[PF01] R. Plänkers and P. Fua. Articulated soft objects for video-based body modeling. In *Proceedings of International Conference on Computer Vision (ICCV'01)*, pages 394–401, Vancouver, Canada, June 2001.

[PFD99] R. Plänkers, P. Fua, and N. D'Apuzzo. Automated body modeling from video sequences. In *Proceedings of the 1999 International Workshop on Modeling People (MPEOPLE'99)*, Corfu, Greece, September 1999.

[PK92] C. Poelman and T. Kanade. A paraperspective factorization method for shape and motion recovery. Technical Report CMU-CS-TR-92-208, Carnegie Mellon University, Pittsburgh, PA, October 1992.

[Pot87] M. Potmesil. Generating octree models of 3D objects from their silhouettes in a sequence of images. *Computer Vision, Graphics and Image Processing*, 40:1–20, 1987.

[PRCM99] V. Pavlovic, J. Rehg, T. Cham, and K. Murphy. A dynamic bayesian network approach to figure tracking using learned dynamic models. In *Proceedings*

of International Conference on Computer Vision (ICCV'99), Corfu, Greece, September 1999.

[PSB99] S. Penny, J. Smith, and A. Bernhardt. Traces: Wireless full body tracking in the cave. In *Proceedings of International Conference on Artificial Reality and Telexistence (ICAT'99)*, December 1999.

[PTVF93] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, 1993.

[QK96] L. Quan and T. Kanade. A factorization method for affine structure from line correspondences. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR'96)*, pages 803–808, San Francisco, CA, 1996.

[RK95] J. Rehg and T. Kanade. Model-based tracking of self-occluding articulated objects. In *Proceedings of International Conference on Computer Vision (ICCV'95)*, pages 612–617, Cambridge MA, June 1995.

[RL01] S. Rusinkiewicz and M. Levoy. Efficient variants of the icp algorithm. In *Third International Conference on 3D Digital Imaging and Modeling*, pages 145–52, May 2001.

[RNK97] P. Rander, P. Narayanan, and T. Kanade. Virtualized reality : Constructing time-varying virtual worlds from real world events. In *Proceedings of IEEE Conference on Visualization (ICV'97)*, pages 277–283, October 1997.

[RT00] M. Ruzon and C. Tomasi. Alpha estimation in natural images. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR'00)*, volume 1, pages 18–25, Hilton Head Island, SC, June 2000.

[SA96] H. Sawhney and S. Ayer. Compact representations of videos through dominant and multiple motion estimation. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 18(8):814–830, 1996.

[SB96] A. Smith and J. Blinn. Blue screen matting. In *Computer Graphics Annual Conference Series (SIGGRAPH'96)*, pages 259–268, 1996.

[SBF00] H. Sidenbladh, M. Black, and D. Fleet. Stochastic tracking of 3D human figures using 2D image motion. In *Proceedings of European Conference on Computer Vision (ECCV'00)*, Dublin, Ireland, June 2000.

[SBK+99] H. Saito, S. Baba, M. Kimura, S. Vedula, and T. Kanade. Appearance-based virtual view generation of temporally-varying events from multi-camera images in the 3D room. Technical Report CMU-CS-99-127, Carnegie Mellon University, 1999.

[SC02] J. Sullivan and S. Carlsson. Recognizing and tracking human action. In *Proceedings of European Conference on Computer Vision (ECCV'02)*, Copenhagen, Denmark, May 2002.

[SCI02] E. Shechtman, Y. Caspi, and M. Irani. Increasing space-time resolution in video. In *Proceedings of European Conference on Computer Vision (ECCV'02)*, Copenhagen, Denmark, May 2002.

[SD96] S. Seitz and C. Dyer. View morphing. In *Computer Graphics Annual Conference Series (SIGGRAPH'96)*, pages 21–30, 1996.

[SDB00] H. Sidenbladh, F. DeLaTorre, and M. Black. A framework for modeling the appearance of 3D articulated figures. In *Proceedings of IEEE International Conference on Automatic Face and Gesture Recognition (ICAFGR'00)*, Grenoble, France, March 2000.

[Sei97] S. Seitz. *Image-Based Transformation of Viewpoint and Scene Appearance*. PhD thesis, University of Wisconsin Madison, 1997.

[SG98] R. Szeliski and P. Golland. Stereo matching with transparency and matting. In *Proceedings of the Sixth International Conference on Computer Vision (ICCV'98)*, pages 517–524, Bombay, India, January 1998.

[SK98] S. Seitz and K. Kutulakos. Plenoptic image editing. In *Proceedings of the Sixth International Conference on Computer Vision (ICCV'98)*, Bombay, India, January 1998.

[SK00] K. Singh and E. Kokkevis. Skinning characters using surface oriented free-form deformations. In *Proceedings of Graphics Interface'00*, pages 35–42, May 2000.

[SKB+98] S. Shafer, J. Krumm, B. Brumitt, B. Meyers, M. Czerwinski, and D. Robbins. The new easyliving project at microsoft research. In *Proceedings of Joint DARPA/NIST Smart Spaces Workshop*, Gaithersburgh, MD, July 1998.

[SMP03] P. Sand, L. McMillan, and J. Popovic. Continuous capture of skin deformation. In *Computer Graphics Annual Conference Series (SIGGRAPH'03)*, pages 578–586, San Diego, CA, July 2003.

[SP91] K. Shanmukh and A. Pujari. Volume intersection with optimal set of directions. *Pattern Recognition Letter*, 12:165–170, 1991.

[SSI99] I. Sato, Y. Sato, and K. Ikeuchi. Acquiring a radiance distribution to superimpose virtual objects onto a real scene. *IEEE Transactions on Visualization and Computer Graphics*, 5(1):1–12, January-March 1999.

[SSI03] I. Sato, Y. Sato, and K. Ikeuchi. Illumination from shadows. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(3):290–300, March 2003.

[SVZ00] D. Snow, P. Viola, and R. Zabih. Exact voxel occupancy with graph cuts. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR'00)*, volume 1, pages 345–352, Hilton Head Island, SC, June 2000.

[Sze93] R. Szeliski. Rapid octree construction from image sequences. *Computer Vision, Graphics and Image Processing: Image Understanding*, 58(1):23–32, July 1993.

[Sze94] R. Szeliski. Image mosaicing for tele-reality applications. Technical Report CRL 94/2, Compaq Cambridge Research Laboratory, 1994.

[TK92] C. Tomasi and T. Kanade. Shape and motion from image streams under orthography: A factorization method. *International Journal of Computer Vision*, 9(2):137–154, November 1992.

[Tsa87] R. Tsai. A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf tv cameras and lenses. *IEEE Journal of Robotics and Automation*, RA-3(4):323–344, August 1987.

[TSA01] P. Torr, R. Szeliski, and P. Anandan. An integrated bayesian approach to layer extraction from image sequences. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(3):297–303, 2001.

[TTI] Thirdtech inc. http://www.3rdtech.com.

[VBK02] S. Vedula, S. Baker, and T. Kanade. Spatio-temporal view interpolation. In *Proceedings of the 13th Eurographics Workshop on Rendering*, June 2002.

[VBR+99] S. Vedula, S. Baker, P. Rander, R. Collins, and T. Kanade. Three-dimensional scene flow. In *Proceedings of International Conference on Computer Vision (ICCV'99)*, Corfu, Greece, September 1999.

[VBSK00] S. Vedula, S. Baker, S. Seitz, and T. Kanade. Shape and motion carving in 6D. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR'00)*, Hilton Head Island, SC, June 2000.

[Ved01] S. Vedula. *Image Based Spatio-Temporal Modeling and View Interpolation of Dynamic Events*. PhD thesis, Carnegie Mellon University, 2001.

[VIC] Vicon motion systems. http://www.vicon.com.

[VKP96] B. Vijayakumar, D. Kriegman, and J. Ponce. Structure and motion of curved 3D objects from monocular silhouettes. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR'96)*, pages 327–334, San Francisco, CA, June 1996.

[WADP97] C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland. Pfinder: Real-time tracking of the human body. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):780–785, July 1997.

[WC01a] K. Wong and R. Cipolla. Head model acquisition and silhouettes. In *Proceedings of International Workshop on Visual Form (IWVF-4)*, May 2001.

[WC01b] K. Wong and R. Cipolla. Structure and motion from silhouettes. In *Proceedings of International Conference on Computer Vision (ICCV'01)*, Vancouver, Canada, June 2001.

[WHG84] H. Weghorst, G. Hooper, and D. Greenberg. Improved computational methods for ray tracking. *ACM Transactions on Graphics*, 3(1):52–69, 1984.

[WHH95] T. Werner, R. D. Hersch, and V. Hlavac. Rendering real-world objects using view interpolation. In *Proceedings of the Fifth International Conference on Computer Vision (ICCV'95)*, pages 957–962, Cambridge MA, June 1995.

[WP02] X. Wang and C. Phillips. Multi-weight enveloping: Least-squares approximation techniques for skin animation. In *Computer Graphics Annual Conference Series (SIGGRAPH'02)*, San Antonio, TX, July 2002.

[WPF90] A. Woo, P. Poulin, and A. Fournier. A survey of shadow algorithms. *IEEE Computer Graphics and Applications*, 10(6):13–32, November 1990.

[Zha94] Z. Zhang. Iterative point matching for registration of free-form curves and surfaces. *International Journal of Computer Vision*, 13(2):119–152, October 1994.

[Zha99] Z. Zhang. Flexible camera calibration by viewing a plane from unknown orientations. In *Proceedings of International Conference on Computer Vision (ICCV'99)*, pages 666–673, September 1999.

Appendix A

Proof of Equivalence of Visual Hull Definitions

To prove that the following two definitions of Visual Hull are equivalent.

Definition I (Intersecting Visual Cones): *The Visual Hull H_j with respect to a set of consistent silhouette images $\{S_j^k\}$ is defined to be the intersection of the K visual cones, each formed by projecting the silhouette image S_j^k into the 3D space through the camera center C^k.*

Definition II (Maximally Exactly Explains): *The Visual Hull H_j with respect to a set of consistent silhouette images $\{S_j^k\}$ is defined to be the largest possible volume which* exactly explains $\{S_j^k\}$ *for all* $k = 1, \cdots\cdots, K$.

Proof:

Let's denote the Visual Hull defined by the first definition (visual cones intersection) by H^I and that defined by the second definition (maximally exactly explains) by H^{II}. To prove the two definitions are equivalent to each other, we are going to prove $H^{II} \subseteq H^I$ and $H^I \subseteq H^{II}$ as below.

- Proof of $H^{II} \subseteq H^{I}$:

  1. It is suffice to show that for any 3D point P, $P \in H^{II} \implies P \in H^{I}$.
  2. To prove (1) above, let P be a 3D point inside or on the boundary of H^{II}. Since H^{II} exactly explains all the silhouettes, the projections of P must lie inside or on the boundary of all of the silhouettes (or otherwise H^{II} will not be exactly explains the silhouettes).
  3. From (2) we know that P has to be inside or on the boundary of the intersection of all the visual cones formed by the camera centers and silhouettes (or otherwise the projection of P lies outside one of the silhouettes).
  4. The statement in (3) in turn implies P lies inside or on the boundary of H^{I}. Hence $H^{II} \subseteq H^{I}$.

- Proof of $H^{II} \subseteq H^{I}$:

  1. Since H^{I} is formed by intersecting the visual cones formed by the silhouettes, the projection of H^{I} into the k^{th} camera must lies inside all the silhouette S^{k} for all cameras k, i.e. $\Pi^{k}(H^{I}) \subseteq S^{k} \forall k$.
  2. As we have assumed that the silhouettes are consistent, there exists a non-empty object O which exactly explains all the silhouettes.
  3. Combining (1) and (2) above, the volume $H^{I} \cup O$ formed by the union of H^{I} and 0 exactly explain all the silhouettes.
  4. The statement in (3) means that $H^{I} \cup O \subseteq H^{II}$ since H^{II} is the maximal volume that exactly explains the silhouettes.
  5. Using (4) we have $H^{I} \subseteq H^{I} \cup O \subseteq H^{II}$. This means $H^{I} \subseteq H^{II}$.

Combining both proofs we get $H^{I} \equiv H^{II}$. ■

Appendix B

Proofs of Alignment Ambiguity Lemmas

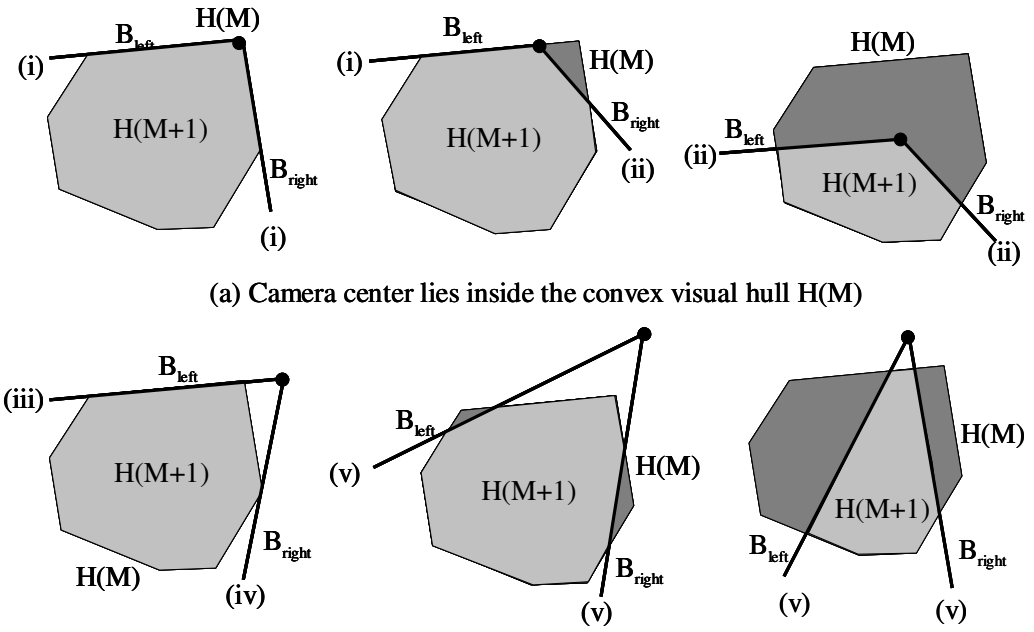
B.1 Proof of Lemma 5.1

Lemma 5.1:

For a *closed* and *connected* 2D object, its Visual Hull from K silhouette images is a *convex* polygon with at most $2K$ Bounding Edges. Conversely, any convex polygon with $M \geq 4$ edges can be thought of as a Visual Hull formed from K silhouettes of some closed and connected 2D object where $K = \lceil \frac{M}{2} \rceil$.

Proof:

Since the 2D object is connected, its silhouette on each camera is a continuous one dimensional line. This means that the bounding area formed by each camera and its silhouette is a convex wedge. The Lemma is obviously true for two-camera case as the intersection of two convex wedge is a convex polygon. Now assume the Lemma is true for M cameras. This means we can represent the Visual Hull constructed from M silhouette images by a convex polygon $H(M)$ with L edges where $L \leq 2M$. Suppose we add one more camera. The new Visual Hull $H(M+1)$ can be obtained by intersecting $M(M)$ with the convex wedge from the new camera. Assume the convex wedge is bound by a left bound B_{left} and a right bound



(a) Camera center lies inside the convex visual hull H(M)

(b) Camera center lies outside the convex visual hull H(M)

Figure B.1: Cases of intersecting an existing Visual Hull with a bounding wedge formed by a new camera and its silhouette. The number (i, ii, iii) at the end of each edges indicate the cases presented in the proof.

B_{right}). Depends on the position of the new camera, we divide the situation into two cases as follows:

(a) The new camera center is inside or on the boundary of $H(M)$ which is shown in Figure B.1(a). In this case, the left bound B_{left} of the wedge intersect $H(M)$ at one of the following two cases : (i) B_{left} coincides with one of the edges of $H(M)$, (ii) B_{left} intersect $H(M)$ at one point. For (i), the intersection does not change $H(M)$. For (ii), the intersection will results in another convex polygon with at most one more edge than $H(M)$ (or less edges than $H(M)$). The same applies to B_{right}. Hence $H(M+1)$ is also a convex polygon with at most $L+2 \leq 2(M+1)$ edges.

(b) The new camera center is outside of $H(M)$ as shown in Figure B.2(b). Here B_{left} intersect $H(M)$ at one of the following three cases: (iii) B_{left} coincides with one of the edges of $H(M)$, (iv) B_{left} intersects $H(M)$ at a corner, (v) B_{left} intersects $H(M)$

at two points. For (iii) and (iv), the intersection does not change $H(M)$. For (v), the intersection will results in another convex polygon with at most one more edge than $H(M)$ (or less edges than $H(M)$ in some cases). The same applies to B_{right}. Hence $H(M+1)$ will have at most $L+2 \leq 2(M+1)$ edges.

Combining both cases, $H(M+1)$ is a convex polygon with at most $2(M+1)$ edges. Hence by Mathematical Induction, the Lemma is true for any number of cameras.

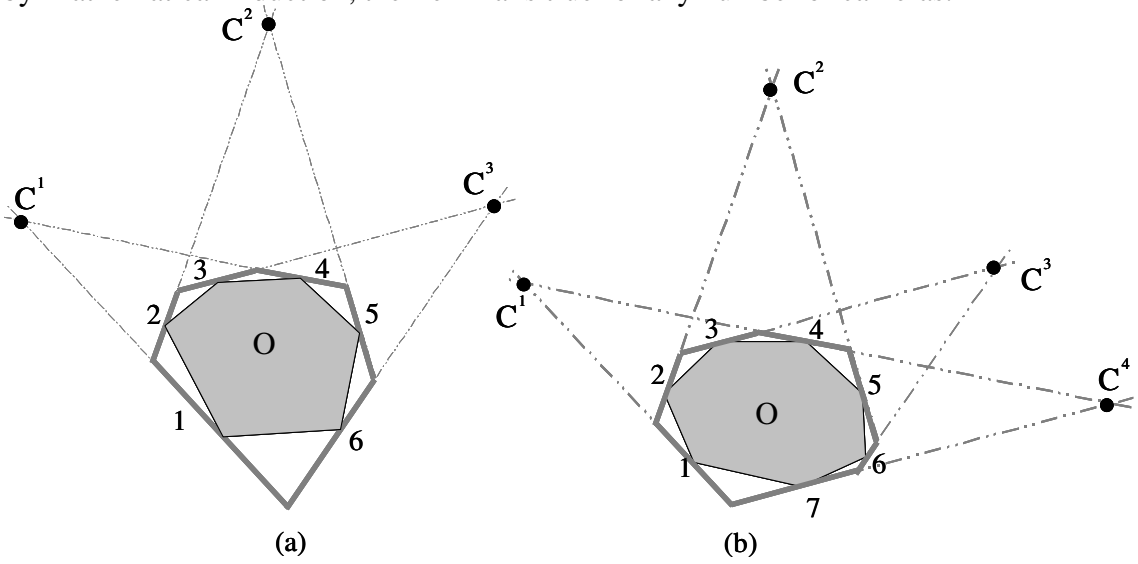


Figure B.2: Examples of reconstructing object O and locations of $\lceil \frac{L}{2} \rceil$ cameras to form the convex polygonal Visual Hull H. (a) L is even, (b) L is even.

-Converse part:

Assume we are given a convex polygon H with $L \geq 4$ edges. We have to show that it is the Visual Hull formed by $\lceil \frac{L}{2} \rceil$ cameras of a closed, connected object. Now since H is convex, any two *non-adjacent* edges will intersect at a point outside H (If the two edges are parallel, then they will intersect at a point at infinity). Let's label the edges of H in clockwise direction and let $L' = \lceil \frac{L}{2} \rceil$. The edges are then paired up every L' edges. For example, edge 1 is paired with edge $L'+1$, edge 2 is paired up with edge $L'+2$, etc. Moreover, let O be an object formed by the convex hull of the mid-points of the L edges of H. Note that O is closed and connected because it is the convex hull of L points. Now we

can divide the situation into two cases: L is even and L is odd.

(a) If L is even, then there are exactly L' pairs of edges, by which the edges in each pair will not be adjacent to each other (as $L \geq 4 \implies L' \geq 2$). Hence each pair of edges will intersect at a point outside H. Let the L' intersecting points be the positions of L' cameras. In this case H will then be the Visual Hull of the object O formed from these L' cameras. An example is shown Figure B.2(a).

(b) If L is odd, then there are exactly $L' - 1$ pairs of edges and an unpaired edge (edge L). Randomly choose from edge 2 to $L - 2$ to pair with edge L to form a total L' pairs of edges. Again we choose O as the convex hull of the mid-points of all the L edges and the L' intersecting points of the edges as the cameras. An example is shown Figure B.2(b).

From the above construction, we find an object O and locations of $\lceil \frac{L}{2} \rceil$ cameras by which the L-sided polygon H is the Visual Hull of O from the cameras. ■

B.2 Proof of Lemma 5.2

Lemma 5.2:

Each edge of the 2D polygonal Visual Hull H of an object O has to touch the object O at at least one point. Conversely any closed and connected 2D object O which satisfies the two conditions: (1) $O \subseteq H$, and (2) O touches each edge of H at at least one point, is an object which forms the silhouettes of H.

Proof:

Firstly by the definition of Visual Hull, the object has to touch both the left bound B_{left} and right bound B_{right} of all the bounding wedges formed by the cameras and their silhouettes. Secondly from Lemma 5.1 and its construction proof, all the edges of the Visual Hull are part of the left or right bound of a bounding wedge. Moreover since the Visual Hull is

convex, each left or right bound can only contribute to at most one edge. This means no two edges of the Visual Hull can come from the same left bound or right bound. Combining this with the fact that the object has to touch all the left and right bounds, it means that each bounding edge has to be touched by the object at at least one point. ■

-Converse part:

Consider a closed and connected 2D object O and a Visual Hull H such that $O \subseteq H$ and O touches each edge of H at at least one point. Let $\{S^k\}$ be the set of silhouettes which form H. We have to prove that $\Pi^k(O) \equiv S^k \quad \forall k$. Since $O \subseteq H$, we have $\Pi^k(O) \subseteq S^k$. Moreover, as O is closed and connected, $\Pi^k(O)$ is a continuous line. Suppose there exists a silhouette image S^k such that O does not exactly explain S^K, i.e. $\Pi^k(O) \subset S^k$. This means one of the endpoints of S^k (which is itself a continuous line), say the left endpoint lie outside of $\Pi^k(O)$. Hence the left bound B_{left} of the 2D bounding wedge formed by S^k lies outside of (and do not touch) O. At the same time, since B_{left} forms one of the edge of H (from the construction proof in Lemma 5.1). This implies one of the edge of H does not touch O. This leads to contradiction that O touches all edges of H at at least one point. Therefore $\Pi^k(O) \equiv S^k \quad \forall k$. ■

B.3 Proof of Lemma 5.3

Lemma 5.3:

Given two 2D Visual Hulls H_1 and H_2, the necessary and sufficient condition for them to be aligned consistently with transformation $(\boldsymbol{R}, \boldsymbol{t})$ is given as follows : No edge of $T_{(\boldsymbol{R},\boldsymbol{t})}(H_1)$ lies completely outside H_2 and no edge of H_2 lies completely outside $T_{(\boldsymbol{R},\boldsymbol{t})}(H_1)$.

Proof:

-Necessary part:

Assume the two Visual Hulls are aligned consistently with transformation $(\boldsymbol{R}, \boldsymbol{t})$. By the definition of consistent alignment, there exists an object O such that H_1 is the Visual Hull of

the object at reference position and orientation $(\boldsymbol{I}, \underline{0})$ and H^2 is the Visual Hull of the object at $(\boldsymbol{R}, \boldsymbol{t})$. That means the object is bounded by both regions $T_{(\boldsymbol{R},\boldsymbol{t})}(H_1)$ and H_2. Now Suppose one edge E_1^1 of $T_{(\boldsymbol{R},\boldsymbol{t})}(H_1)$ lies completely outside H_2. Since the object O is bounded by H_2, it means that the edge E_1^1 is completely outside O. This leads to a contradiction to Lemma 5.2 which states that edge E_1^1 has to touch the object O at at least one point. Hence all the edges of $T_{(\boldsymbol{R},\boldsymbol{t})}(H_1)$ have to either (1) lie completely inside H_2 or (2) intersecting or touching at least one edge of H_2. In other words, at least one point of each edge of $T_{(\boldsymbol{R},\boldsymbol{t})}(H_1)$ has to lie on or inside H_2. The same applies vice versa.

-Sufficient part:

Assume we are given two Visual Hulls H_1, H_2 and a transformation $(\boldsymbol{R}, \boldsymbol{t})$. Consider the object O constructed by intersecting $T_{(\boldsymbol{R},\boldsymbol{t})}(H_1)$ and H_2. Suppose the transformation $(\boldsymbol{R}, \boldsymbol{t})$ is such that no edge of $T_{(\boldsymbol{R},\boldsymbol{t})}(H_1)$ is completely outside of H_2 and vice versa. This means that each edge of O comes from (the whole or part of) one edge from H_1 or H_2. This implies O will touch every edge of H_1 and H_2 at at least one point. Since O is the intersection of H_1 and H_2, it is smaller than either of them. Hence by the converse of Lemma 5.2, O is an object which forms the silhouettes of H_1 and H_2. As a conclusion, both H_1 and H_2 are Visual Hulls of O at two different orientations and positions related by the transformation $(\boldsymbol{R}, \boldsymbol{t})$. ■

B.4 Proof of Lemma 5.4

Lemma 5.4:

$(\boldsymbol{R}, \boldsymbol{t})$ is a consistent alignment of two 2D Visual Hulls H_1 and H_2, constructed from silhouette sets $\{S_j^k\}$; $j = 1, 2$ if and only if the following condition is satisfied : for each edge E_1^i of $T_{(\boldsymbol{R},\boldsymbol{t})}(H_1)$, there exists at least one point P on E_1^i such that the projection of P onto the k^{th} image lies inside or on the boundary of the silhouette S_2^k for all $k = 1, \cdots\cdots, K$.

Proof:

The condition of Lemma 5.4 is equivalent to the condition that at least one point of the bounding edge E_1^i have to lie inside H^2 which is exactly the condition of Lemma 5.3. ■

B.5 Proof of Lemma 5.5

Lemma 5.5:

For two *convex* 3D Visual Hulls H_1 and H_2 constructed from silhouette sets $\{S_j^k\}$; $j = 1, 2$, the necessary and sufficient condition for a transformation $(\boldsymbol{R}, \boldsymbol{t})$ to be a consistent alignment between H_1 and H_2 is as follows: for any Bounding Edge E_1^i (defined by (4.2) in Chapter 4) constructed from the silhouette image set $\{S_1^k\}$, there exists at least one point P on E_1^i such that the projection of the point $T_{(\boldsymbol{R},\boldsymbol{t})}(P)$ onto the k^{th} image lies inside or on the silhouette S_2^k for all $k = 1, \cdots\cdots, K$. Similarly, for any Bounding Edge E_2^i constructed from $\{S_2^k\}$, there exists at least one point P on E_2^i such that the projection of the point $T_{(\boldsymbol{R},\boldsymbol{t})}^{-1}(P)$ on the k^{th} image lies inside or on the silhouette S_1^k.

Proof:

-Necessary part:

Assume the two Visual Hulls are aligned consistently with transformation $(\boldsymbol{R}, \boldsymbol{t})$. By the definition of consistent alignment, there exists an object O such that H_1 is the Visual Hull of the object at reference position and orientation $(\boldsymbol{I}, \underline{0})$ and H^2 is the Visual Hull of the object at $(\boldsymbol{R}, \boldsymbol{t})$. That means the object is bounded by both volumes $T_{(\boldsymbol{R},\boldsymbol{t})}(H_1)$ and H_2. Now Suppose one edge E_1^1 of $T_{(\boldsymbol{R},\boldsymbol{t})}(H_1)$ lies completely outside H_2. Since the object O is bounded by H_2, it means that the edge E_1^1 is completely outside O. This leads to a contradiction to the Second Fundamental Property of Visual Hull which states that edge E_1^1 has to touch the object O at at least one point. Hence all the edges of $T_{(\boldsymbol{R},\boldsymbol{t})}(H_1)$ have to either (1) lie completely inside H_2 or (2) intersecting or touching at least one edge of H_2. In other words, at least one point of each edge of $T_{(\boldsymbol{R},\boldsymbol{t})}(H_1)$ has to lie on or inside H_2

and this is equivalent to saying that there exists at least one point P on E_1^i such that the projection of the point $T_{(\boldsymbol{R},\boldsymbol{t})}(P)$ onto the k^{th} image lies inside or on the silhouette S_2^k for all $k = 1, \cdots\cdots, K$. The same applies vice versa.

-Sufficient part:

Assume we are given two Visual Hulls H_1, H_2 and a transformation $(\boldsymbol{R}, \boldsymbol{t})$. Consider the object O constructed by intersecting $T_{(\boldsymbol{R},\boldsymbol{t})}(H_1)$ and H_2. Since O is the intersection of H_1 and H_2, it is smaller than either of them. Moreover, since H_1 and H_2 are convex, O is also convex. Now suppose the transformation $(\boldsymbol{R}, \boldsymbol{t})$ is such that for any Bounding Edge E_1^i, there exists at least one point P on E_1^i such that the projection of the point $T_{(\boldsymbol{R},\boldsymbol{t})}(P)$ onto the k^{th} image lies inside or on the silhouette S_2^k for all $k = 1, \cdots\cdots, K$ and vice versa. This is equivalent to the condition that all possible Bounding Edges of $T_{(\boldsymbol{R},\boldsymbol{t})}(H_1)$ has to touch H_2 at at least one point and vice versa. This means that the (infinite) ray projected through any camera from any point on the boundary of the corresponding silhouettes at t_1 has to touch the object O. The same applies to t_2. Since O is convex, the above implies that the projection of O on camera k have to be exactly the same as the silhouettes S_1^k and S_2^k for all k. Hence O exactly explains the silhouettes which form H_1 and H_2. As a conclusion, both H_1 and H_2 are Visual Hulls of O at two different orientations and positions related by the transformation $(\boldsymbol{R}, \boldsymbol{t})$. ■

Appendix C

Proofs of Visibility Lemma

Lemma 5.6:

Let $\Pi^l(P)$ and $\Pi^l(C^k)$ be the projections of the point P and the k^{th} camera center C^k on the (infinite) image plane of camera l. If the 2D line segment joining $\Pi^l(P)$ and $\Pi^l(C^k)$ does not intersect the silhouette image S^l_j, *then* P is visible with respect to camera k at time t_j.

Proof:

The Lemma can be proved by contradiction. The 2D segment joining $\Pi^l(P)$ and $\Pi^l(C^k)$ is the projection of the 3D plane passing through P, C^k and C^l. Suppose the 2D segment does not intersect with the silhouette image S^l_j, this means the corresponding 3D plane is totally free of any object. Now if P is *not* visible with respect to camera k, then there must exist some object on the 3D line (which lies on the above 3D plane) joining P and C^k, blocking P from C^k. This leads to contradiction that the 3D plane is totally free of object. Hence P is visible to camera k. ■