Abstract

Title of dissertation	Towards Markerless Motion Capture: Model Estimation, Initialization and Tracking
	Aravind Sundaresan, Doctor of Philosophy, 2007
Directed by	Professor Ramalingam Chellappa Department of Electrical and Computer Engineering

Motion capture is an important application in diverse areas such as bio-mechanics, computer animation, and human-computer interaction. Current motion capture methods use markers that are attached to the body of the subject and are therefore intrusive. In applications such as pathological human movement analysis, these markers may introduce unknown artifacts in the motion and are, in general, cumbersome. We present a computer vision based system for *markerless* human motion capture that uses images obtained from multiple synchronized and calibrated cameras. We model the human body as a set of rigid segments connected in articulated chains. We use a volumetric representation (voxels) of the subject using images obtained from the cameras in our work. We propose a novel, bottom-up approach to segment the voxels into different articulated chains based on their mutual connectivity, by mapping the voxels into Laplacian Eigenspace. We prove properties of the mapping that show that it is ideal for mapping voxels on non-rigid chains in normal space to nodes that lie on smooth 1D curves in Laplacian Eigenspace. We then use a 1D spline fitting procedure to segment the nodes according to which 1D curve they belong to. The segmentation is followed by a top-down approach that uses our knowledge of the structure of the human body to register the segmented voxels to different articulated chains such as the head, trunk and limbs. We propose a hierarchical algorithm to simultaneously initialize and estimate the pose and body model parameters for the subject. Finally,

we propose a tracking algorithm that uses the estimated human body model and the initialized pose for a single frame of a given sequence to track the pose for the remainder of the frames. The tracker uses an iterative algorithm to estimate the pose, that combines both motion and shape cues in a predictor-corrector framework. The motion and shape cues complement each other and overcome drift and local minima problems. We provide results on 3D laser scans, synthetic data, and real video sequences with different subjects for our segmentation, model and pose estimation algorithms.

Towards Markerless Motion Capture: Model Estimation, Initialization and Tracking

by

Aravind Sundaresan

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park, in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2007

Advisory Committee:

Professor Ramalingam Chellappa, Chair
Professor William S. Levine
Professor Ankur Srivastava
Professor Larry S. Davis
Professor John Jeka

To my parents,
For their love, support and sacrifice
over the years.

Acknowledgments

I am indebted to my advisor, Prof. Rama Chellappa, for offering me the opportunity to work at the Center for Automation Research at the University of Maryland and his generous support and advice throughout the duration of my graduate studies. I am grateful to Prof. William Levine, Prof. Larry Davis, Prof. John Jeka and Prof. Ankur Srivastava for serving on my dissertation committee. I am particularly grateful to Prof. Levine for carefully proofreading the entire dissertation and providing valuable feedback both at the time of the proposal as well as the final oral defense. I would also like to thank Prof. Min Wu who served on my dissertation committee during the dissertation proposal but was unable to serve on the final dissertation committee. I would also like to acknowledge the National Science Foundation, who supported much of the research in this dissertation.[1]

I am thankful to Prof. Thomas Andriacchi, Dr. Eugene Alexander, Dr. Lars Mündermann, Dr. Ajit Chaudhuri and Dr. Stefano Corazza at Stanford University for helpful discussions and providing access to the excellent facilities at the Stanford Biomotion Laboratory. Gene, Lars, Ajit and Stefano were quite generous with their time and provided the biomechanical perspective of the markerless motion capture problem. A special thanks goes to James Sherman Jr. along with whom I was able to design and set up *Hydra*, the portable multi-camera facility for motion analysis. James also assisted me in several data capture sessions both at the Keck Laboratory at the University of Maryland and at the Biomotion Laboratory at Stanford University. James has been an integral part of the markerless motion capture project and his enthusiasm both inside and outside the laboratory was much appreciated. Fritz McCall, Mark Goh, Brad Erdman, Wesley Madoo and other members of the UMIACS staff were instrumental in renovating the aging Keck laboratory multi-camera system and were a huge help in designing *Hydra*. Fritz, in particular, was always on hand and extremely generous with his time

[1]The research was supported in part through NSF ITR 03-25715.

to offer technical support and trouble-shoot the system during its many hiccups. I would also like to thank Fritz, Janet He, and the UM Institute for Advanced Computer Studies staff for the excellent and flexible computing facilities which eased the huge data storage, back-up and retrieval as well as data processing requirements of the multi-camera image acquisition and motion capture algorithms. I would also like to thank James Davis and Héctor González-Banõs, then at the Honda Research Institute, with whom I worked on a marker-based motion capture system. The experience was useful as it provided a contrast to the markerless system presented in this thesis. I also gained useful experience with multi-camera systems during the time I spent at HRI.

Individually acknowledging all my friends and colleagues during my life as a graduate student is impossible. I am thankful to the faculty, student, and staff members at the Center for Automation Research, the University of Maryland Institute for Advanced Computer Studies and the Department of Electrical and Computer Engineering. In particular, I would like to thank Amit Roy-Chowdhury, Volker Krüger, Amit Kale, Francesc Romà i Frigolé and Naresh Cuntoor with whom I worked on the Human Identification at a Distance project, which was the precursor to the Markerless motion capture project. Francesc was my Linux guru and I am indebted to him for introducing me to Linux and the concept of Free Software. I would also like to thank other colleagues and friends at CfAR: Ashok Veeraraghavan, Gaurav Agarwal, Aswin Sankaranarayan, Narayanan Ramanathan, Mahesh Ramachandran, Gaurav Aggarwal, Himaanshu Gupta and Amit Agrawal for providing a collegial and enjoyable environment at work.

Deepak Iyengar and Deepak Malghan were my apartment-mates and fellow graduate students for the first four years of my life in the United States and were a huge influence and source of support. I would like to thank my brother, Srikanth and my cousins, Amaresh, Swarupa and Karthik, all of whom were also fellow graduate students. Srikanth has been a constant source of support especially in the last two years of my dissertation.

Finally, I am greatly indebted to my parents for their constant encouragement and unstinting love and sacrifice in all my endeavors, academic and non-academic, over the years. To them, I dedicate this dissertation.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Motion capture for humans describes the activity of analyzing and expressing human motion in mathematical terms. Motion capture was pioneered by Eadweard Muybridge (1830-1904) in his famous experiments entitled *Animal Locomotion*, a study into the way in which animals and birds moved. The study included recording photographs of the subjects at discrete time intervals, using multiple cameras, in order to visualize motion. Muybridge also captured multi-camera sequences of human subjects. Photographs of human models engaged in over one hundred and sixty different activities were published in *The Human Figure in Motion* [47]. Muybridge had his human models walk in an open shed which had three batteries of twelve cameras each, positioned at angles of approximately 30°, 90°, and 150° with respect to one wall of the shed. Three photographs were taken simultaneously, one from each battery. Figure 1.1 illustrates one such activity of a woman throwing a ball. Ètienne-Jules Marey (1830-1904) was another pioneer in motion capture who used a single camera to record images from which scientific measurements could be taken. He published the book, *Le Mouvement* [35], on human locomotion in 1894. While Muybridge used multiple cameras, Marey devised a revolutionary method to capture multiple exposures on a single photographic surface in order to study motion.

Advances in technology, primarily silicon and other sensors, and the continually lowering cost of video capture and processing, have allowed easier access to the equipment required for motion capture and broadened the range of applications. Today, motion capture has applications in diverse fields ranging from human motion analysis in clinical studies and sports medicine, to animation in

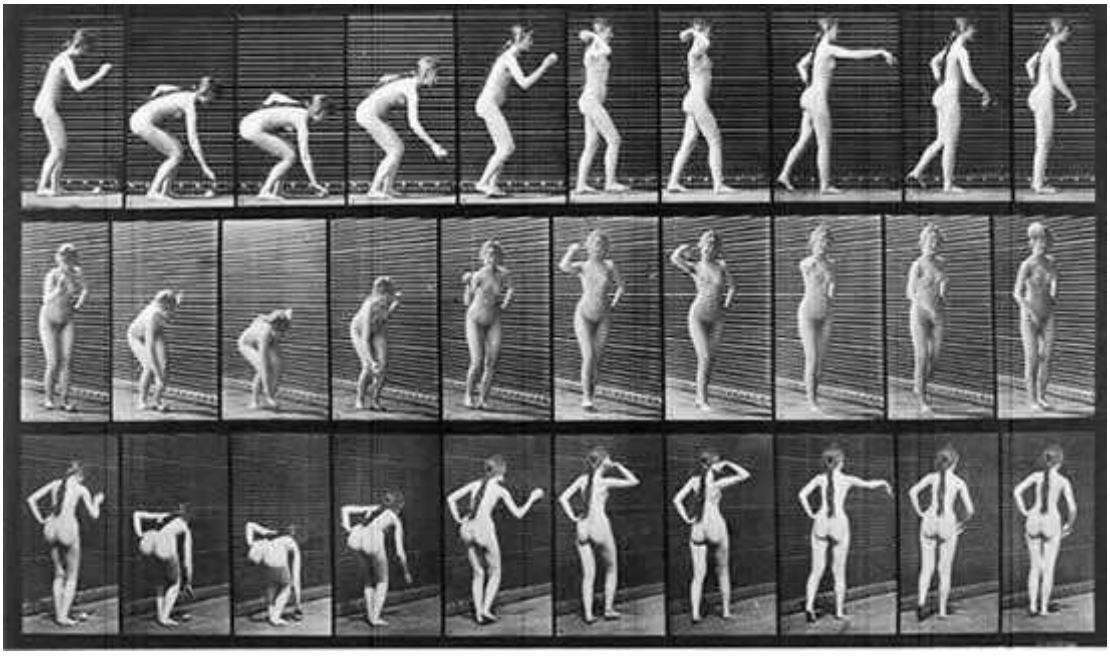


Figure 1.1: "Woman throwing a ball", Eadweard Muybridge, *Human Locomotion*, 1890

the motion picture and video game industries, and human-computer interaction. The specifications of each of these applications vary in terms of environmental conditions, accuracy requirements, speed requirements, and complexity of models used, but they all require the measurement of the pose of the subject in terms of the various joint angles of the model used.

1.1 Motivation

The state-of-the-art motion capture techniques used today are typically accomplished by one of four technologies: optical, magnetic, electro-mechanical or inertial motion trackers. All of them entail the use of some kind of markers in some form that are worn by the subject. These technologies have a severe disadvantage due to the fact that they use external markers that are attached to the subject. The very presence of these markers can be cumbersome and hinders the free movement of the subject thereby introducing artifacts in the motion that

is to be measured. Besides, the placement of these markers requires expertise, especially in the case of clinical applications, and is time consuming. Markerless motion capture is a method for motion capture that eschews the use of markers, instead relying directly on images obtained from multiple cameras placed around the subject to estimate the pose of the subject. With the current imaging technology, it is now possible to capture color images using multiple synchronized cameras at speeds of more than 100 frames per second in a laboratory setting at a reasonable cost.

1.2 Markerless motion capture system

A motion capture procedure typically consists of the following steps: model estimation, pose initialization, and pose tracking as illustrated in Figure 1.2. The figure describes the relation between the input data, the different algorithms and their role in the three steps of a motion capture system, and the output of the system in terms of the model and pose parameters. In this dissertation, we describe the components of a complete automatic markerless motion capture system. The components of the system are further described in the following subsections.

1.2.1 Input to the system

In our markerless motion capture system we use images obtained from multiple cameras that are placed around the volume of interest in which the subject moves. We use both 2D data in the form of the original images from the cameras and the corresponding foreground silhouettes, as well as 3D data in the form of voxels. The different types of input data are illustrated in Figure 1.3. The foreground silhouettes are obtained from the original images by performing background subtraction. Voxels are points on a 3D grid that lie inside the body and are analogous to 3D pixels. The voxel data for a frame can be computed by projecting points on a regular 3D grid onto each of the images and determining whether they lie

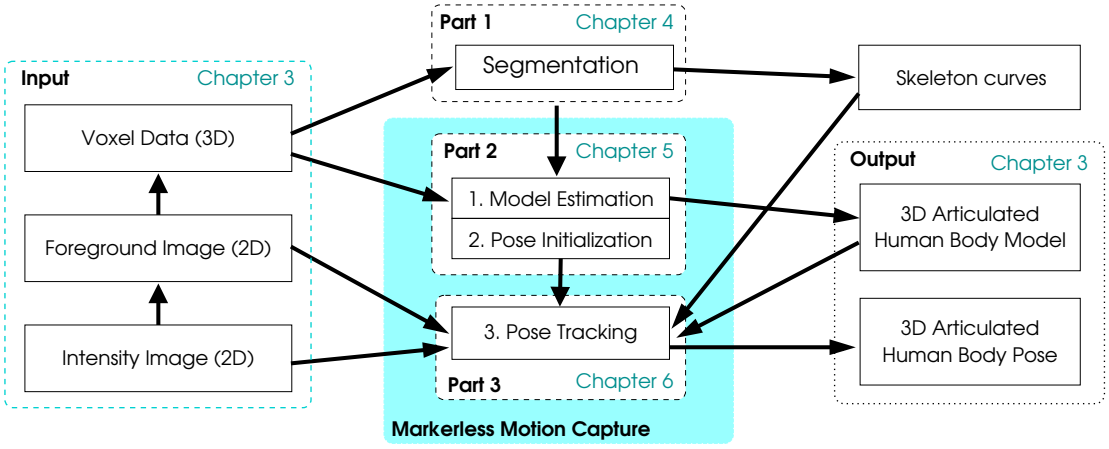


Figure 1.2: Motion capture system schematic: The relationship between the input data, algorithms, and the output in our motion capture system. The role of the three steps in a motion capture system in the whole process is illustrated as are the three main parts of our work.

inside the silhouette.

1.2.2 Output of the system

Our objective is to estimate both the human body model parameters as well as the pose of the subject. The human body can be visualized as rigid body segments attached to each other in articulated chains. Our human body model consists of six articulated chains; the trunk, the head, the two arms and the two limbs as illustrated in Figure 1.4. The pose is described in terms of the position of the base body in the chain and the relative pose of each segment in each chain with respect to its parent. Given a video sequence, we wish to estimate the parameters of the articulated human body model, as well as the articulated pose in each frame of the sequence.

1.2.3 Data processing and algorithms

The estimation of the human body model and pose from the input data involves a number of steps, including pre-processing images, segmentation of 3D data,

Figure 1.3: The algorithms use plain images obtained from the cameras, the foreground images, and the voxel reconstruction that is obtained using space carving.

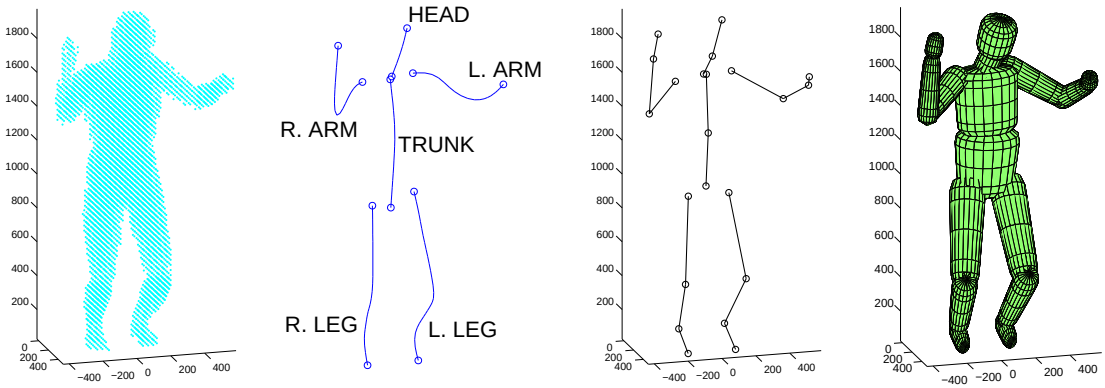
registration, model estimation, pose initialization and pose tracking. The role of these algorithms in the motion capture system is illustrated in Figure 1.2. We divide the algorithms into three parts based on the approaches and their functionality.

Part 1 One of the key steps is bottom-up segmentation of the voxel data of the human body into its component articulated chains.

Part 2 The second step is the initialization step where we use top-down techniques to register the segmented articulated chains, and estimate the body model parameters as well as initialize the pose.

Part 3 The final step is the fusion of the motion cues and the 2D and 3D shape cues to perform pose tracking using the estimated human body model and initialized pose.

We note that *Part 1* describes a bottom-up approach for segmentation and *Part 2* describes a top-down approach to use the primitives obtained in *Part 1* to obtain model and initialize pose. *Part 3* tackles the problem of tracking using different primitives including the segmentation results of *Part 1*. Marr [36] describes the



(a) Voxel (b) Skeleton curves (c) Skeleton model (d) SQ model

Figure 1.4: Human body models: Illustration of two kinds of models corresponding to the voxel data. (a) denotes the voxel data of the subject in a frame. (b) denotes the skeleton curve for each of the articulated chains comprising the human body. (c) is the skeleton model and (d) is the super-quadric (SQ) model. The skeleton model uses a subset of the parameters of the complete SQ model.

typical representational framework for deriving shape information from images which can be extended to 3D data as well.

1.2.3.1 Part 1: Segmentation in Laplacian Eigenspace

One of the key steps in pose and model estimation is the segmentation of the 3D voxels belonging to different articulated chains of the human body. We consider the voxels as nodes in a graph and present a novel algorithm for performing the segmentation by mapping the nodes into the Laplacian Eigenspace (LE) of the graph they form. The transformation maps the nodes on different articulated chains to different smooth 1D curves in LE. This allows us to fit a different 1D spline to nodes (voxels) on different 1D curves (articulated chains) and thus perform the segmentation. We prove certain properties of the LE transformation that show why it is suitable for segmenting objects that consist of non-rigid or articulated chains. We also compare the LE transformation to other manifold methods such as Isomap, that are typically used for dimensionality reduction and

pose invariant transforms. We note that an important by-product of the segmentation is the implicit computation of the position of each node (voxel) along the 1D curve (articulated chain). This enables us to compute a skeleton curve for that articulated chain, which is a useful feature that is used in all three steps of the motion capture algorithm; model estimation, pose initialization and pose tracking.

1.2.3.2 Part 2: Model and pose initialization

Having presented a bottom-up algorithm for segmenting the input voxel data into the component articulated chains, we use top-down methods to perform pose and model estimation. While human body dimensional variability is fairly large across different demographics and sexes, it is not arbitrary. We can use our knowledge of the structure of the human body to guide pose estimation, towards which end we use a suitably complex human body model. Following segmentation of voxels using 1D splines in LE, we register each 1D spline to the known articulated chains in the human body such as the limbs, trunk and head. We use a probabilistic registration algorithm based on the properties of each articulated chain as well as their mutual connectivity. A probabilistic approach allows us to easily deal with difficult poses where there is self-contact. The computed probability of a registration for a given frame allows us to determine the success of the registration. We use a set of frames where registration is successful for the model and pose initialization process. The skeleton curves that were computed for each articulated chain as part of the segmentation algorithm are used in a hierarchical approach to estimate the pose and model parameters. We begin with a single parameter, the stature of the subject and optimize for the stature value that best fits the skeleton curves. We then use a skeleton model illustrated in Figure 1.4 (c) and optimize for pose and skeleton model parameters. Finally, we augment our skeleton model using superquadric shapes for each rigid segment and the segmented voxel data to compute the complete SQ-based model illustrated in Figure 1.4 (d).

1.2.3.3 Part 3: Pose tracking

The final step in our motion capture system is the tracking of the full body pose of the subject through the entire sequence. We performed segmentation of the voxels and computed the skeleton curves in the first part. We obtained the human body model parameters of the subject and initialized the pose for a set of frames in the second part. We note that the proposed registration works on a single frame and typically succeeds only in some of the frames in the sequence. We propose a temporal registration algorithm that registers the computed skeleton curves to their corresponding articulated chains for the remainder of the frames. We then perform tracking using motion cues that include 2D pixel displacement in the image plane and shape cues that include skeleton curves, foreground silhouettes, as well as motion residues. It is not always possible to use shape cues to track pose due to unavailability of reliable cues for the whole or part of the body. These missing or unreliable cues are caused by faulty segmentation or registration, which often occur, *e.g.*, when the limbs are close to the body. Purely silhouette-based and voxel-based methods typically experience difficulties in such cases, but we are able to handle such errors as we use motion cues in our tracking algorithm. Shape based methods, unlike pixel-motion based methods, also have the weakness that they are often unable to deal with rotation about the axis of the body segment. On the other hand, pixel motion based tracking methods have their share of problems, of which, the primary one is that they suffer from drift, as they do not use absolute features to perform the tracking. Motion and shape cues, therefore, complement each other and combining both types of cues in the tracking enables us to overcome the weakness associated with either of the two.

1.3 Contributions of the dissertation

In this dissertation, we have proposed and described algorithms for performing the key tasks in a markerless motion capture system. Our contribution has been

threefold. Firstly, we have proposed a novel segmentation algorithm that can be used to segment an object consisting of non-rigid chains into its component chains [67]. The segmentation is performed in Laplacian Eigenspace of the graph of the object, using a 1D spline fitting algorithm as briefly described in Section 1.2.3.1. We have proved properties of the Laplacian Eigespace transform [68], that show that it is ideal for performing the segmentation of non-rigid chains such as the voxel data of a human body. We show that it can be applied to voxel data of different human subjects in a variety of poses, including those where there is self contact. Secondly, we show that it is possible using a top down approach to register the segmented voxel data of the subject to the model, resolving ambiguities using a probabilistic formulation. We also estimate the human body model parameters [65,68] and initialize the pose in a set of frames that have been fully segmented and registered using a hierarchical approach as outlined in Section 1.2.3.2. We provide results of the registration, model estimation and pose estimation algorithms on an array of subjects, whose voxel data have been obtained from different sources such as images, 3D scans and synthetic sequences. Thirdly, we show that we can use the estimated human body models and initialized pose in order to track the pose of the subject in an entire sequence using complementary cues such as motion and shape [69]. We formulate the pixel velocity, under perspective projection camera models, as a linear function of the pose velocity [70], and hence propose an iterative algorithm to estimate the change in pose from pixel displacement [66]. The estimated pose is corrected using both 2D and 3D shape cues. The use of motion and shape cues, which are complementary, in a predictor-corrector framework enables us to overcome common problems in tracking such as drift and local minima.

1.4 Organization of the dissertation

The dissertation is organized as follows. We present a brief survey of pose estimation and tracking algorithms in computer vision and compare our algorithm with existing and related algorithms in Chapter 2. We briefly describe the processing of the input image data to compute foreground silhouettes, and 3D voxels that we use in our work in Chapter 3. We also describe some of the common human body models in the literature as well as describe the details of our articulated human body model and the associated pose vector. The first part of our system, in which we motivate and describe the use of the Laplacian Eigenspace mapping to perform the segmentation of voxels is presented in Chapter 4. The second part which includes single frame registration of the segmented voxels, as well as model and pose initialization is presented in Chapter 5. The pose tracking algorithm using 2D and 3D shape and motion cues is described in Chapter 6. We conclude with a summary of our contributions, suggest extensions to our motion capture system and outline future research directions in Chapter 7. An outline of *Hydra*, a portable multi-camera capture system that has been designed for motion analysis is presented in Appendix A. Appendix B describes the eigenvectors of simple graphs and Appendix C explores the possible solutions to the eigenvalues of the Laplacian of the extended tree graph.

Chapter 2

Related work

Pose estimation from images and video sequences has been steadily gaining in importance in the last decade. The last few years especially have seen rapid progress in the development of pose estimation algorithms targeting a wide range of applications. These algorithms can be quite different depending on the scope of the application as well as the kind of input data available. Most computer vision-based pose estimation algorithms target applications where it is required to obtain an approximate estimate of the pose from either a single image or a monocular video stream. Segmentation of the image into different, possibly self-occluding, body parts and tracking them is an inherently difficult problem especially due to the complex articulated structure of human beings as well as the ambiguity introduced by the kinematic singularity problem. Recently, there has been more of a focus on performing pose estimation using images from multiple cameras, with a view to replacing marker-based motion capture techniques with markerless techniques. These methods target applications requiring complete articulated 3D pose of the subject in controlled environments and using multiple camera images and typically use a human body model to guide pose estimation. It is therefore necessary to estimate the human body model parameters as well.

Gavrila and Davis [22], Aggarwal and Cai [1], Moeslund and Granum [40], and, more recently, Wang *et al.* [76] and Sigal and Black [61] provide surveys of human motion pose tracking and analysis methods. We describe in this chapter some specific works in human body model and pose estimation and pose tracking related to our own. We list some bottom-up segmentation techniques based on manifold and graph methods in Section 2.1. We then describe some methods

that use volumetric data for both human body model and pose estimation in Section 2.2. Finally, we cover a broad range of tracking algorithms that primarily use silhouettes and image based methods for tracking and pose estimation in Section 2.3.

2.1 Segmentation of human body volume data

Most techniques for segmentation of human body volume data for pose estimation use voxel data. Voxel data naturally lends itself to graph-based methods for analysis, and in particular, segmentation. The human body can be visualized as consisting of 1D chains embedded in 3D space. There are a number of dimensionality reduction techniques to analyze such structures. Belkin and Niyogi [5] describe the construction of a representation for data lying in a low dimensional manifold embedded in a high dimensional space and use Laplacian Eigenmaps for dimensionality reduction. Lafon and Lee [33] present Diffusion Maps, a variant on Laplacian Eigenmaps which uses a Gaussian kernel of width σ to construct a weighted graph and normalizes the Laplacian operator. Two other popular dimensionality reduction techniques are Isomaps proposed by Tenenbaum *et al.* [73] and Locally Linear Embedding (LLE) proposed by Roweis and Saul [56]. There also exist other methods such as charting a manifold proposed by Brand [6] and Kernel Eigenvalue analysis [57]. Elad and Kimmel propose an algorithm for reducing articulated objects to pose-invariant structure [20]. Weiss [77] presents a unifying view on segmentation using eigenvectors. Belkin and Niyogi analyze the relation between Laplacian Eigenmap embedding and Locally Linear Embedding [56]. The Laplacian Eigenmap also has similarities to Normalized Cuts proposed by Shi and Malik [58]. Laplacian Eigenmap and other manifold methods have been applied to dimensionality reduction problems such as classification and face retrieval, *e.g.*, Laplacianfaces [26]. However, we actually map the voxels to a *higher* dimensional space in order to extract the 1D manifold that the articulated chain segments lie

on. The dimension of the Laplacian Eigenspace depends on the number of chains we wish to segment. Xiao *et al.* [79] and Werghi *et al.* [78] propose a Reeb graph approach based on Geodesic distance in order to segment 3D scans of human bodies in various postures.

2.2 Human pose estimation from 3D data

A popular class of algorithms [13, 38, 44, 67] uses voxels in order to perform pose estimation. We use a voxel based algorithm [67] in order to perform pose initialization and model estimation. However, typically, voxel-based algorithms can be used for pose initialization in only a limited number of frames in the sequence. There usually are frames in a sequence where errors in the voxel reconstruction due to noise in the background silhouettes, or segmentation, result in missing body segments. The stand-alone registration, therefore, fails in these frames.

We also look at some existing methods that use either motion-based methods or silhouette or edge based methods to perform tracking. A large number of pose estimation algorithms uses a single image or single image stream to estimate the pose of the subject or use simplified models. Several pose tracking algorithms also assume that the initial pose is known. While we list pose estimation algorithms that use single cameras, we concentrate on works that estimate 3D pose using images obtained from multiple cameras. The accuracy and the robustness of these algorithms vary as does the suitability of the algorithms for different applications. There are several methods to estimate pose from a single view [55, 52, 54, 42] or images from multiple cameras [28, 38, 13, 11, 9]. Specifically the algorithms in [38, 13, 11] estimate the pose from voxel representations. Carranza *et al.* [9] describe a system that uses multi-view synchronized video footage of an actor's performance to estimate the motion parameters and to interactively re-render the actor's appearance from any viewpoint. Chu *et al.* [13] describe a method for pose estimation using Isomaps [73] to transform the voxel representation of the

human body to a pose-invariant intrinsic space representation and thus compute the skeleton. Cheung *et al.* [11] extend shape-from-silhouette methods to articulated objects. Given silhouettes of a moving articulated object, they propose an iterative algorithm to solve the simultaneous assignment of silhouette points to a body part and alignment of the body part. These methods work well with poses such as those in Figure 5.2 (a), but they are usually unable to handle poses (Figure 5.2 (c)) where there is self-contact, *i.e.*, one or more of the limbs touches the others. Anguelov *et al.* [2] describe an algorithm that automatically decomposes an object into approximately rigid parts, their location, and the underlying articulated structure given a set of meshes describing the object in different poses. They use an unsupervised non-rigid technique to register the meshes and perform segmentation using the EM algorithm. Krahnstoever [32] addresses the issue of acquiring articulated models directly from a monocular video. Structure, shape and appearance of articulated models are estimated, but this method is limited in its application as well as accuracy in extracting complete 3D human body models as it uses a single camera. Algorithms that estimate the complete human body model from multiple views are presented in Mikic *et al.* [38] and Kakadiaris *et al.* [28]. Mikic *et al.* [38] propose a model acquisition algorithm using voxels, which starts with a simple body part localization procedure based on fitting and growing templates computed using the shapes and dimensions of average body parts. Kakadiaris and Metaxas [28] present a Human Body Part Identification Strategy (HBPIS) that recovers all the body parts of a moving human based on the spatio-temporal analysis of its deforming silhouette using input from three mutually orthogonal views. However, they specify a protocol of movements that the subject is required to go through.

Our segmentation algorithm can also be viewed as a skeletonization algorithm, that obtains the skeletons of the individual articulated chains. Brostow *et al.* [8] present a skeletonization method that uses voxel data to estimate a novel skeleton representation using *spines*. We model the human body as a set of

rigid body segments that are connected to each other at specific joints forming kinematic chains originating from the trunk. Badler *et al.* [3] suggest several methods to represent human subjects in terms of their shape as well as their articulated structure. We find that using modified super-quadrics to represent shapes [23] is reasonably accurate for our purposes, although our approach can accommodate more sophisticated mesh-models if the data is accurate enough and if the application demands it.

2.3 Human pose tracking

Pixel flow is used by a number of tracking algorithms to track articulated pose. Barron *et al.* [4] present a survey of optical flow methods. Yamamoto and Koshikawa [80] analyze human motion based on a robot model and Yamamoto *et al.* [81] track human motion using multiple cameras. Gavrila and Davis [23] discuss a multi-view approach for 3D model-based tracking of humans in action. They use a generate-and-test algorithm in which they search for poses in a parameter space and match them using a variant of Chamfer matching. Ju *et al.* [27] use planar patches to model body segments. The motion of each patch is defined by eight parameters. For each frame the eight parameters are estimated by applying the optical flow constraint on all pixels in the predicted patches. Bregler and Malik [7] use an orthographic camera model and use optical flow to track pose using twists and exponential maps. Morris and Rehg [43] and Rehg *et al.* [53] describe ambiguities and singularities in the tracking of articulated objects and Cham and Rehg [10] propose a 2D scaled prismatic model. Sidenbladh *et al.* [59] provide a framework to track 3D human figures using 2D image motion and particle filters with a constrained motion model that restricts the kinds of motions that can be tracked. Kakadiaris and Metaxas [29] use silhouettes from multiple cameras to estimate 3D motion. Plänkers and Fua [51] use articulated soft objects with an underlying articulated skeleton as a model and use stereo and silhouette data for

shape and motion recovery. Theobalt *et al.* [74] project the texture of the model obtained from silhouette-based methods and refine the pose using the flow field. Delamarre and Faugeras [18] use 3D articulated models for tracking with silhouettes. They use silhouette contours and apply forces to the contours obtained from the projection of the 3D model so that they move towards the silhouette contours obtained from multiple images. Cheung *et al.* [11] use shapes from silhouette to estimate human body kinematics. Chu *et al.* [13] use volume data to acquire and track a human body model. Wachter and Nagel [75] track persons in monocular image sequences. They use an IEKF with a constant motion model and use edges to region information in the pose update step in their work. Moeslund and Granum [39] use multiple cues for model-based human motion capture and use kinematic constraints to estimate the pose of a human arm. The multiple cues are depth (obtained from a stereo rig) and the extracted silhouette, whereas the kinematic constraints are applied in order to restrict the parameter space in terms of impossible poses. Sigal *et al.* [62, 60] use non-parametric belief propagation to track in a multi view set up. Lan and Huttenlocher [34] use hidden Markov temporal models. DeMirdjian *et al.* [19] constrain pose vectors based on kinematic models using SVMs. Rohr [55] performs automated initialization of the pose for single camera motion. Krahnstoever [32] addresses the issue of model acquisition and initialization. Mikic *et al.* [38] automatically extract the model and pose using voxel data. Ramanan and Forsyth [52] also suggest an algorithm that performs rough pose estimation and can be used as an initialization step. Sminchisescu and Triggs present a method for monocular video sequences using robust image matching, joint limits and non-self-intersection constraints [64]. They also try to remove kinematic ambiguities in monocular pose estimation efficiently [63].

We present a complete initialization and tracking algorithm that uses both shape as well as motion cues to estimate and track the pose. Spatial cues are absolute and prevent drift in the tracking, but it is not possible to extract reliable spatial cues in each frame. We therefore base our tracker on motion cues, which

can be computed in every frame, using spatial cues to correct the drift. We also present a novel method to use spatial cues such as silhouettes and motion residues. It is also possible to incorporate other spatial cues such as edges in our method. We note that we do not constrain the motion or the pose parameters for specific types of motion and hence our method can be used to track any kind of motion and is quite general.

Chapter 3

Input data and output variables

We describe in this chapter the input data used in our human motion capture system as well as the details of our human body model and the associated pose parameter. The input data consists of images from several synchronized cameras, the corresponding foreground silhouettes, and voxel data. The voxel data serves as an input layer abstraction for some of the proposed algorithms. Pre-processing steps for computing silhouettes and voxels are described in Section 3.1, and the human body model that we use in our work along with the corresponding pose vector is described in Section 3.2.

3.1 Input data

The basic input data we use in our system are synchronized images from multiple cameras. Each image from each camera is processed to obtain the foreground silhouettes and the voxels that are described in the subsections that follow. An example of the various steps in the processing of the images is illustrated in Figure 3.1. The original images and silhouettes also serve as inputs to the tracking algorithm.

3.1.1 Image acquisition

The images are obtained from multiple synchronized cameras. The images can be gray-scale or color. Background subtraction algorithms perform better on color images, but optical flow or pixel displacement algorithms typically operate on gray-scale images. The cameras are completely calibrated, *i.e.*, their internal and

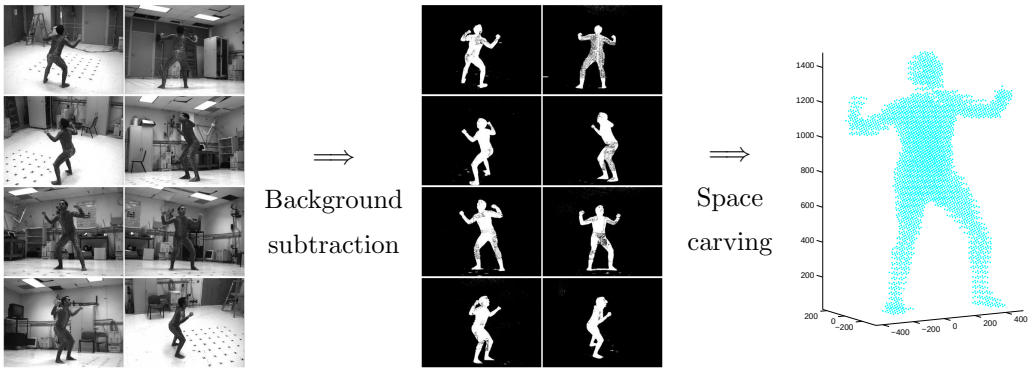


Figure 3.1: The images obtained from the cameras are processed to obtain the foreground images and voxel data. Foreground silhouettes are obtained using background subtraction, and the voxels are obtained using space carving.

external calibrations are known. The camera calibration parameters are obtained in three steps. In the first step, the internal parameters are obtained using the *OpenCV*[1] library functions. In the second step, the external calibration parameters are computed using the algorithm [72] proposed by Svoboda.[2] The method uses an LED pointer which is moved around in the volume of interest. The algorithm uses the previously computed internal calibration parameters and computes the external parameters up to a scale. In the third step, the scale and a reference world frame are determined using a separate calibration device. We thus compute the complete calibration parameters of the cameras. In our system the radial distortion parameters are very small and are ignored.

3.1.2 Computing silhouettes

The foreground silhouettes are computed using simple background subtraction techniques. An average background image is obtained for each camera from a video sequence of the static background. The threshold image is computed using the standard deviation of the background sequence as well as the intensity of the background image. Given a new image, the average background image is

[1]The library can be downloaded from http://sourceforge.net/projects/opencvlibrary/
[2]The software can be downloaded from http://cmp.felk.cvut.cz/∼svoboda/SelfCal/.

subtracted from the image and a given pixel in the difference image is set to be 1 if it is greater than the threshold image. The resulting binary image is passed through a morphological filter to remove isolated noisy patches. It is to be noted that as the number of cameras used in the system increases, the accuracy of the background subtraction algorithm becomes less important as far as the computation of the voxel is concerned. We also note that the background subtraction algorithm can be made completely parallel.

3.1.3 Computing voxels

A voxel image of a subject is a set of points on a regular 3D grid that lies inside the body of the subject. If a given voxel (3D point) is inside the body of the subject, then it lies inside the silhouette in all the images. We can therefore project all points on the 3D grid in the volume of interest onto all the foreground silhouettes and declare a 3D point to be a voxel if it lies inside the silhouette in all the images. In order to deal with errors in the background subtraction, we use a slightly modified algorithm to compute the voxels. Let us assume that we have images from N cameras. A voxel is considered to be part of the subject if it falls inside the silhouette in at least $N - M$ images, where $0 \leq M \leq N$. We can set $M = 0$ if the background subtraction is very good. If we set $M > 0$, we gain robustness in voxel construction at the expense of accuracy. For instance, if the background subtraction in one of the images labels part of the body as background, that part of the body goes missing in voxel construction if $M = 0$, but is present if $M > 0$. In a practical scenario, we can set the value of M depending on the number of cameras in the system and the performance of the background subtraction algorithm. In our experiments, we set M as

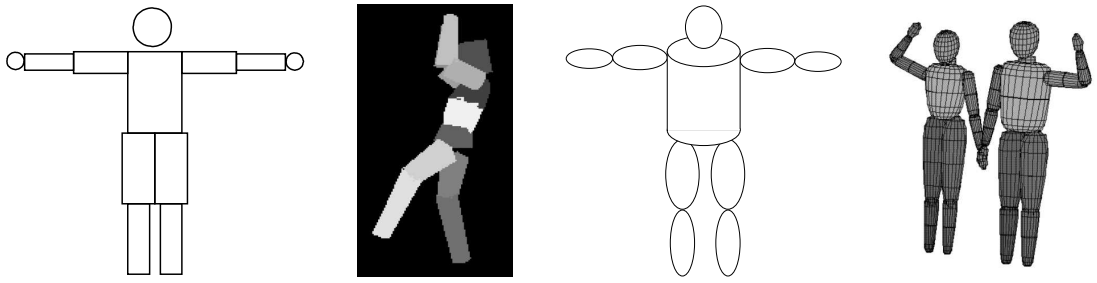
$$M = \begin{cases} 0 & \text{if } N \leq 8, \\ 1 & \text{if } 8 \leq N < 12, \\ 2 & \text{if } N > 12. \end{cases} \tag{3.1}$$

3.2 Human body model and pose

A human body model is used to guide the pose estimation in a large number of algorithms and it is advantageous to use a flexible, scalable human body model whose parameters can be easily estimated. We describe the class of human body models that we use in our system in Section 3.2.1 and compare it to popular human body models in the literature. Each body segment is described by a modified super-quadric described in Section 3.2.2. The pose of a subject is described in terms of the human body model and we describe the pose parameter with respect to our human body model in Section 3.2.3.

3.2.1 Human body model

Badler *et al.* [3] provide a detailed analysis of the different kinds of human body models that can be used. Figure 3.2 illustrates some of the models that have been used in the literature including super-quadrics [23], polyhedrons [81], ellipsoids [38], and cylinders [59]. The most flexible among the parametric models is the super-quadric based model proposed by Gavrila and Davis [23]. We build upon their super-quadric model incorporating flexibility in the motion as well as adding detail in terms of the number of body segments.



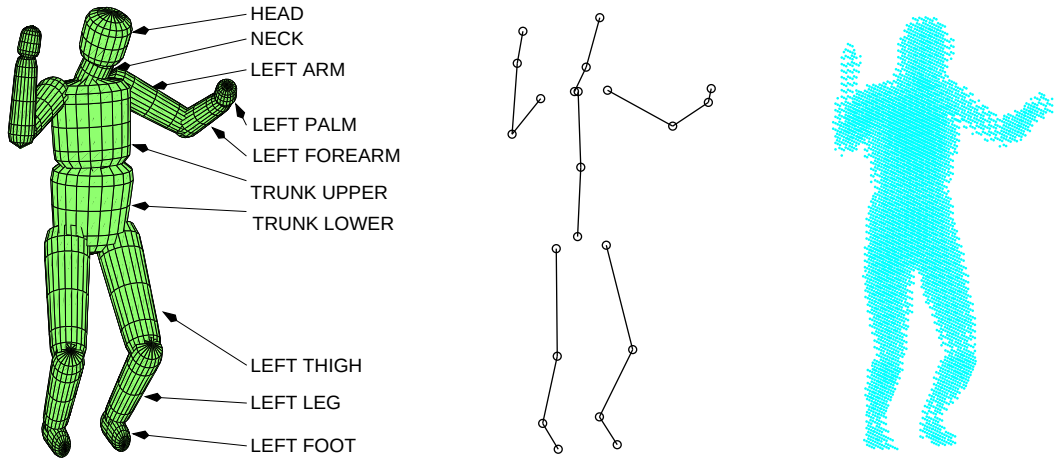
(a) Cylinders (b) Polyhedrons (c) Ellipsoid (d) Super-quadric

Figure 3.2: The different shape models to represent the human body: (a) Cylinder (b) Polyhedrons and (c) Ellipsoid (d) Super-quadric based models

The human body model that we use is illustrated in Figure 3.3 (a) with the

different body segments labeled. We model the human body as consisting of six articulated chains, namely the trunk (lower trunk, upper trunk), head (neck, head), two arms (upper arm, forearm, palm) and two legs (thigh, leg, foot) connected at joints as illustrated in the figure. The articulated chains are composed of rigid segments. Each rigid body segment is represented in its own coordinate reference frame that is attached to it and can be described in general by an arbitrary convex 3D mesh-model in terms of its frame coordinates, and in our case is modeled using a modified super-quadric. The modified super-quadric [65] is described in Section 3.2.2. The joint location of a body segment is described as a vector in the coordinate frame of its parent segment. We model the body segments as having a single joint location; joints such as the shoulder joint that are compound joints are modeled as joints with translation. The human body model consists of the joint locations and parameters of the modified super-quadrics describing each rigid segment. Our model takes into account the underlying skeleton structure and flexibility of the human body model. The trunk is represented using two segments in order to model the flexibility of the spine. The model can be simplified to a skeleton model using just the axes of the super-quadrics as illustrated in Figure 3.3 (b). Each body segment can, in general, move freely with respect to its parent segment. However, we impose constraints on the translational motion of most of the joints. The details of these constraints are elaborated upon when we discuss the pose vector in Section 3.2.3.

While there exists considerable human dimensional variability across different demographics and sexes, it is not arbitrary. The stature (height) of the subject is a parameter that is strongly related to a number of human body model parameters, such as the lengths of long bones in the body [50]. Anthropometric studies have been performed on certain demographic groups to study the relationship between stature and the long bones in the body [12,37]. These studies indicate that we can construct the skeleton for an average subject given the stature alone. The super-quadric parameters of the subject can also be estimated for an average subject.



(a) Super-quadric model (b) Skeleton model (c) Voxel data

Figure 3.3: Human body model: (a) Super-quadric human body model, (b) Skeleton model which uses a subset of the body model parameters, (c) Voxel data of the subject whose body model is illustrated.

Thus we see that we can construct a complete human body model for an *average* subject given just one parameter, the stature of the subject. We can also use a *build* parameter that is a scalar in $[0.9, 1.1]$, where a low value indicates a slim build and a high value indicates a broad build.

3.2.2 The modified super-quadric segment

The modified super-quadric illustrated in Figure 3.4 and described in (3.2), is characterized by five parameters x_0, y_0, z_0, d, and s. If sliced in a plane parallel to the xy plane, the cross section is an ellipse with parameters $x_0 r(z)$ and $y_0 r(z)$, *i.e.*,

$$\left(\frac{x}{x_0}\right)^2 + \left(\frac{y}{y_0}\right)^2 = r^2(z). \tag{3.2}$$

The scale parameter, s, denotes the amount of taper of the radial profile $r(z)$, and the degree parameter, d, denotes the curvature of the radial profile, $r(z)\sqrt{x_0 y_0}$, along the z-axis. The radial profile, $r(z)$, is a function of the z-coordinate z, z_0,

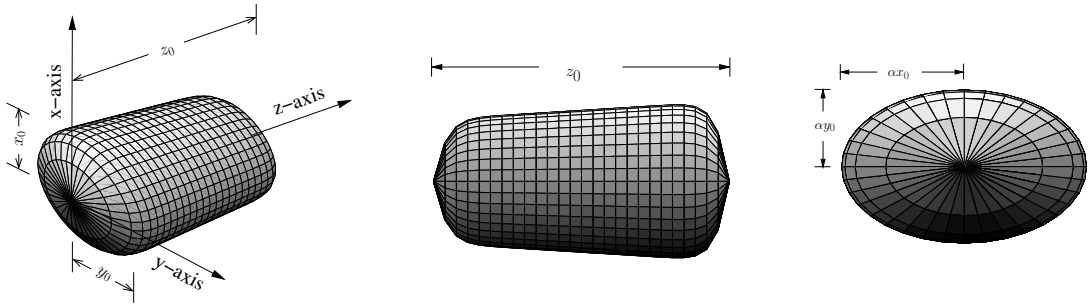


Figure 3.4: Super-quadric segment

s and d and is given by

$$r^2(z) = \left(1 + s\frac{z}{z_0}\right)\left(1 - \left(1 - 2\frac{z}{z_0}\right)^d\right), \qquad 0 \leq z \leq z_0. \tag{3.3}$$

The family of modified super-quadrics as defined above includes common geometric shapes such as cylinders, cones and ellipsoids. For instance, an ellipsoid has $d = 2, s = 0$, a right elliptical cylinder has $d = \infty, s = 0$, and a right-elliptical cone has $d = \infty, s = -1$.

3.2.3 The pose vector

The position of a body segment is given in terms of the position of its attached coordinate frame with respect to its parent segment and is represented by a transformation matrix, G. G is represented in homogeneous 3D coordinates and is a function of both the pose vector and the body model. In general, any transformation matrix has six degrees of freedom and can be expressed as a function of a rotational component, $\boldsymbol{\omega}$, and a translational component, $\boldsymbol{p}$. The pose vector for a body segment comprises both components and is given by $\boldsymbol{\varphi} = \binom{\boldsymbol{p}}{\boldsymbol{\omega}}$. G is expressed as

$$G(\boldsymbol{p}, \boldsymbol{\omega}) = \begin{pmatrix} R & \boldsymbol{p} \\ \mathbf{0}' & 1 \end{pmatrix} = \begin{pmatrix} e^{\hat{\boldsymbol{\omega}}} & \boldsymbol{p} \\ \mathbf{0}' & 1 \end{pmatrix}, \tag{3.4}$$

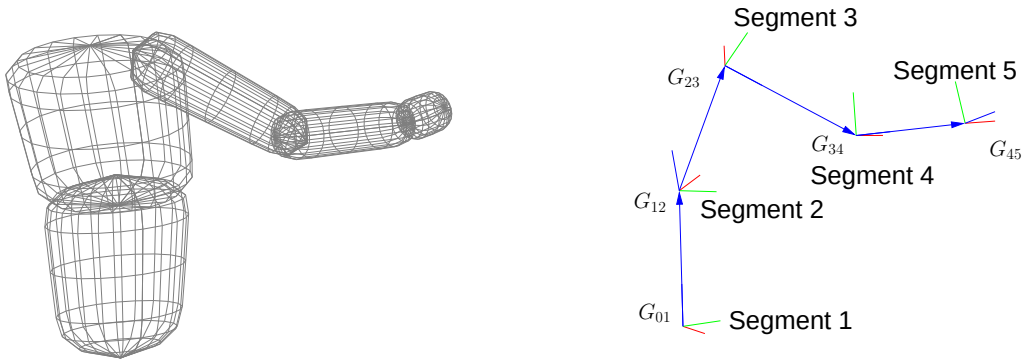
where

$$\hat{\boldsymbol{\omega}} = \begin{pmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{pmatrix}. \tag{3.5}$$

We drop the dependence on the rotational and translational vectors in certain instances for the sake of simplicity. The relation between body segments in a kinematic chain is illustrated in Figure 3.5. The figure illustrates five segments attached in an articulated chain. The root of the chain is referred to as the base body and is labeled 1. All body segments are attached to the base body in a kinematic chain and have, in general, six degrees of freedom with respect to their parent segments. If the translation component of all segments is constrained to be zero, as it is in the the case of most joints, it results in a pure articulated structure. G_{ij} represents a transformation matrix of a point from the coordinate frame of segment j to the coordinate frame of segment i. Note that index 0 refers to the world reference frame and G_{0i} is the transformation between the world reference frame and segment i. G_{01} represents the transformation matrix of the base body to the world reference frame. The position of the i^{th} segment with respect to the world reference frame is therefore given by

$$G_{0i} = G_{01}G_{12}\cdots G_{(i-1)i}. \tag{3.6}$$

The position of each body segment with respect to its parent is described by a combination of body model and pose parameters. We use the superscript S to denote a structure parameter of the body and P to denote a pose parameter. For instance, $\boldsymbol{p}^S$ is a joint location and is part of the body model, while $\boldsymbol{p}^P$ is the translational pose at the joint and is part of the pose vector. We consider the position of segment i with respect to its parent. $i = 3$ in the example in Figure 3.6 which illustrates the articulated structure as a function of pose. In general, segment i is connected to its parent at joint i, whose location is given



(a) Super-quadric segments

(b) Segment coordinate frames

Figure 3.5: Articulated structure and the relative positions of body segments in a chain: (a) The actual super-quadric body segments and (b) The coordinate frame attached to each of the rigid segments. The red, green, and blue axes together describe the pose of the coordinate frame for each segment.

by $\boldsymbol{p}^{(i)S}$ in the coordinate frame of the parent. The position of segment i in the coordinate frame of segment $i-1$ is given by $G_{(i-1)i}$ which is a function of the joint location and the pose parameter and is given by

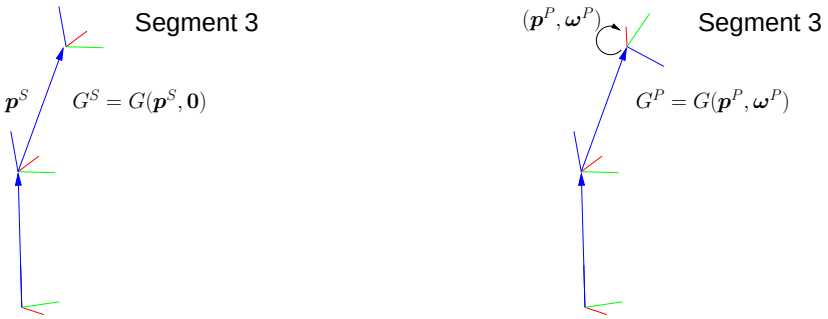
$$G_{(i-1)i} = \overbrace{G(\boldsymbol{p}^{(i)S}, \mathbf{0})}^{\text{Structure}} \overbrace{G(\boldsymbol{p}^{(i)P}, \boldsymbol{\omega}^{(i)P})}^{\text{Pose}}. \tag{3.7}$$

For a strictly articulated body, $\boldsymbol{p}^{(i)P} = \mathbf{0}\ \forall\ i > 1$. However, not all joints can be modeled accurately using only rotational motion. The shoulder joint, for instance, is actually a complex joint and is better modeled as multiple rotational joints [21, 25]. In order to model complex joints such as the shoulder joint, we allow limited translation at those joints such that

$$\|\boldsymbol{p}^{(i)S}\| < p_{\text{MAX}}, \tag{3.8}$$

where i denotes complex joints.

It is desirable to constrain the translation component to be zero at most joints in order to minimize the number of parameters while realistically modeling the



(a) Body model ($\boldsymbol{p}^S$) (b) Pose ($\boldsymbol{p}^P$, $\boldsymbol{\omega}^P$)

Figure 3.6: Articulated structure and the relative positions of a body segment as a function of the body model and pose. The position of the body segment as a function of (a) The body model and (b) the pose vector.

human body. Our human body model consists of sixteen rigid segments (as is illustrated in Figure 3.3). The pose of segment i is given by $\boldsymbol{\varphi}^{(i)} = \begin{pmatrix} \boldsymbol{p}^{(i)P} \\ \boldsymbol{\omega}^{(i)P} \end{pmatrix}$ and the complete pose vector is given by

$$\boldsymbol{\Phi} = \begin{pmatrix} \boldsymbol{\varphi}^{(1)} \\ \vdots \\ \boldsymbol{\varphi}^{(16)} \end{pmatrix}. \tag{3.9}$$

Chapter 4

Segmentation in Laplacian Eigenspace

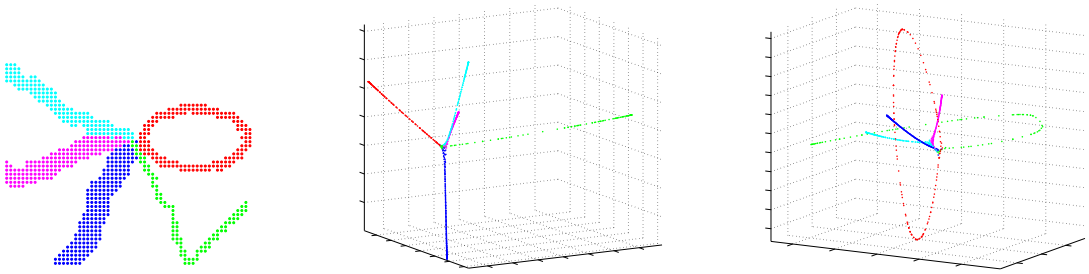
One of our first objectives in pose estimation from 3D data is to segment the voxel (volumetric) data into different body segments. As a first step, we segment the volumetric data into its component *articulated chains*. In other words, we aim to segment the voxel structure of the subject at joints such as shoulders, hips, and neck as these are the joints where *three or more* body segments meet. We describe, in this chapter, a novel technique for segmenting non-rigid (including articulated) chains of any object using the Laplacian Eigenspace (LE) transform. There are several advantages to performing segmentation in LE rather than the 3D world the voxels reside in. The transform is based on the graph on the voxels (nodes) where the edges between the nodes of the graph are between neighboring voxels. The eigenvectors of the Laplacian of the adjacency matrix of the graph are used to effect the transform to Laplacian Eigenspace. We particularly focus on the human body that is comprised of articulated (non-rigid) chains connected at joints. We show that mapping to LE is a natural choice for segmenting the human body into its component articulated chains. The mapping of voxels to LE achieves two important objectives. Firstly, the effect of pose is minimized because the LE transform depends on the connectivity of voxels, which is minimally affected by the articulation at joints. Secondly, the transform maps voxels belonging to different chains (such as the limbs in the human body) to points on *separate, smooth, 1D curves* in LE according to their position along the articulated chain. It is this important property that allows us to fit a 1D spline to the voxels belonging to an articulated chain and differentiates the LE transform from other manifold techniques such as Isomaps [73] or LLE [56]. We prove these properties of LE using

simple representative graphs and show that the LE transform is optimal from the point of view of mapping points to 1D curves in the case of graph structures composed of grid graphs. We finally describe how to exploit the structure of the voxels in LE by fitting 1D splines to the voxels in order to perform the actual segmentation. We present results of the segmentation on subjects with different body structures and different poses illustrating the efficacy of the LE transform on real world data.

The segmentation in LE using 1D spline plays a key role in our motion capture system as it provides an effective bottom-up segmentation of the voxels that can be exploited by higher level algorithms for tasks such as model and pose initialization as well as pose tracking. In some of the sections that follow, we describe the segmentation in LE in the context of segmenting the voxel data of a human subject. We note that the method is not limited to human body segmentation although that is our primary objective in this work. We begin with the mapping of the voxels to Laplacian Eigenspace in Section 4.1. The properties of the Laplacian Eigenvectors of two special types of graphs are explored in Section 4.2, and motivate our segmentation algorithm based on these properties. We compare the LE transform to other manifold techniques for dimensionality reduction, and in particular to the Isomap, in Section 4.3 and show why LE transform is the most suitable for our purpose. We describe the segmentation algorithm in LE with examples in Section 4.4. Finally, we describe how to compute the *skeleton curve* of the object using the output of the segmentation algorithm in Section 4.5 and present the results of the segmentation on data from different subjects in Section 4.6.

4.1 Mapping to Laplacian Eigenspace

The mapping to Laplacian Eigenspace is described in Table 4.1 and illustrated using the example in Figure 4.1. The example highlights certain features of the



(a) Nodes in image (b) Eigenvectors 1-3 (c) Eigenvectors 4-6

Figure 4.1: Example 2D object with multiple chains of varying thickness and self contact mapped to LE. (a) The nodes in the graph. (b) and (c) denote the nodes in 6D LE.

transform. The 2D object in the figure consists of several non-rigid chains of varying widths and lengths connected at a single joint. The different chains are color-coded for the purpose of illustration and no distinction based on color is used in the mapping. One of the chains (colored in red) loops around, *i.e.*, has self-contact and one of the chains (green) has a sharp "bend".

The object is sampled on a regular grid and the graph $G(V, E)$ that describes the connectivity between neighboring nodes in Figure 4.1 (a) is computed. Although the nodes lie on a 2D plane in this example, they could lie in any high dimensional space as long as we are able to compute $G(V, E)$. We assume that the graph $G(V, E)$ is completely connected, otherwise we choose the biggest connected component. The eigenvalues of L (Table 4.1, step 3) are positive and real, as L is positive semi-definite and symmetric. Chung [14] shows that the smallest eigenvalue of L, $\lambda_0 = 0$ and the corresponding eigenvector $\boldsymbol{x}_0 = \mathbf{1}$. If G is fully connected, then $\lambda_1 > 0$. The i^{th} row of Y (Table 4.1, step 5) provides the embedding, $\boldsymbol{y}_i^T$, for the i^{th} node.

Belkin and Niyogi [5] show that the Laplacian Eigenspace embedding described in Table 4.1 is optimal when we wish to obtain $\boldsymbol{y}_i$ such that the distance between neighbors,

$$\sum_{i,j} \|\boldsymbol{y}_i - \boldsymbol{y}_j\| W_{ij} = \operatorname{tr}\left(Y^{\mathsf{T}} L Y\right), \tag{4.1}$$

1. Compute the W matrix, such that

$$W_{ij} = \begin{cases} 1, & \text{if } i \text{ is a neighbor of } j, \\ 0, & \text{otherwise.} \end{cases}$$

2. Compute the D matrix, so that $D_{ii} = \sum_{k=1}^{m} W_{ik}$ and $D_{ij} = 0$ for $i \neq j$.

3. Compute the Laplacian matrix, L, so that $L = D - W$.

4. Compute d eigenvectors of L, $\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_d$, where $L\boldsymbol{x}_i = \lambda_i \boldsymbol{x}_i$ and

$$0 = \lambda_0 < \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_d \leq \cdots \leq \lambda_{m-1}.$$

5. Node i is mapped to $\boldsymbol{y}_i$, where

$$Y_{m \times d} = \begin{pmatrix} \boldsymbol{x}_1 & \boldsymbol{x}_2 & \cdots & \boldsymbol{x}_d \end{pmatrix} = \begin{pmatrix} \boldsymbol{y}_1' \\ \vdots \\ \boldsymbol{y}_m' \end{pmatrix}.$$

Table 4.1: Mapping nodes to Laplacian Eigenspace

is minimized, where W_{ij} is defined in Table 4.2. They impose the constraint $Y^{\mathsf{T}}Y = I$ to remove an arbitrary scaling factor. In addition to the distance minimizing property, the Laplacian Eigenvectors also possess certain properties that are described in Section 4.2, and motivate our segmentation algorithm described in 4.4.

4.2 Properties of Laplacian eigenvectors

We make the following observations about the Laplacian Eigenvectors based on the example in Figure 4.1 and justify them by analyzing the properties of the Laplacian Eigenvectors of special types of graphs such as the *extended star* graph and the *grid* graph.

1. Nodes on different chains are mapped to points on different curves in LE such that each of the curves can be discriminated from the others. We note that the discriminative capability of the LE transform improves with the dimension of the eigenspace. We observe that the position of each node along the 1D curve also encodes the position of that node along the 1D body part.

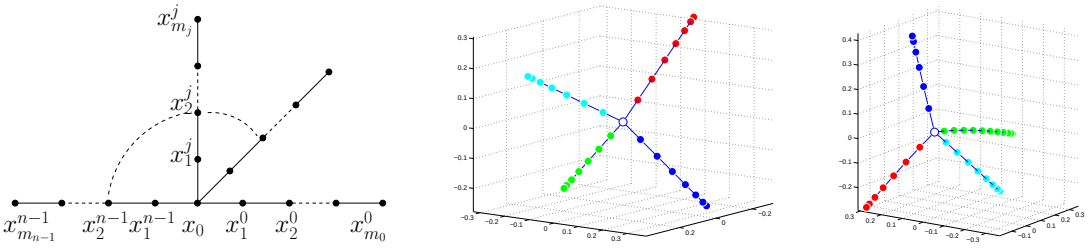
2. Nodes belonging to a given chain are mapped to points along a smooth 1D curve irrespective of the thickness of the chain to which they belong as shown in Figure 4.1 (b)-(c). The 1D structure is retained in the higher dimensions.

The first observation is justified using *extended star* graphs in 4.2.1 and the second is justified using *grid* graphs in 4.2.2.

4.2.1 Eigenvectors of extended star graphs

We define extended star graphs as graphs that are composed of $n > 2$ chains (or path graphs) connected at one end to a common node as illustrated in Figure 4.2. The path graph, P_m, is a graph on m vertices with an edge set $E_{P_m} = \{(i, i+1)|i =$

$0, 1, \cdots, m-2\}$. The ring graph, R_m, has the edge set $E_{R_m} = \{(i, (i+1) \bmod m) | i = 0, 1, \cdots, m-1\}$. These two graphs, and the path graph in particular, are the basic building blocks for the special graphs that we consider and we describe the properties of their Laplacian Eigenvectors in Appendix B.



(a) Extended star graph (b) Symmetric case (c) Asymmetric case

Figure 4.2: The structure of the extended star graph with the nodes labeled in (a). Examples of both symmetric and asymmetric extended star graphs with $n = 4$ in LE. (b) $m_j = 7$ in the symmetric case, and (c) $m_j = \{11, 8, 7, 6\}$ in the asymmetric case. Nodes belonging to different chains are colored differently.

The j^{th} chain has $m_j + 1$ nodes (including the common node) and hence there are a total of $r + 1$ nodes where $r = \sum_{j=0}^{n-1} m_j$. Let $\boldsymbol{x} = \begin{pmatrix} x_0 & x_1 & \cdots & x_r \end{pmatrix}^\mathsf{T}$ be an eigenvector of the Laplacian matrix. The node with index $\sum_{l=0}^{j-1} m_l + i$, *i.e.*, the i^{th} node ($i = 1, 2, \cdots, m_j$) in the j^{th} chain ($j = 0, 1, \cdots, n-1$) is labeled $x_i^{(j)}$ for the sake of clarity in representation. $x_0^{(j)}$ ($j = 0, 1, \cdots, n-1$) represents x_0. The graph is asymmetric in general, *i.e.*, $m_i \neq m_j$ for $i \neq j$. The graph is symmetric if $m_0 = \cdots = m_{n-1} = m$. We analyze the structure of the eigenvectors corresponding to the smallest non-zero eigenvalues in both the general asymmetric case and the special symmetric case.

The Laplacian Eigenvector, $\boldsymbol{x}$, needs to satisfy $L\boldsymbol{x} = \lambda\boldsymbol{x}$, from the rows of

which we get the following equations.

$$nx_0 - \sum_{j=0}^{n-1} x_1^{(j)} = \lambda x_0 \tag{4.2}$$

$$2x_i^{(j)} - x_{i-1}^{(j)} - x_{i+1}^{(j)} = \lambda x_i^{(j)}, \qquad 1 \le i < m_j,\, 0 \le j \le n-1 \tag{4.3}$$

$$x_{m_j}^{(j)} - x_{m_j-1}^{(j)} = \lambda x_{m_j}^{(j)}, \qquad 0 \le j \le n-1 \tag{4.4}$$

We note that (4.3) and (4.4) are similar to the equations for the path graph, P_m, and the ring graph, R_m, respectively (Appendix B). We can verify by substitution that the set of equations in (4.3) is satisfied by a solution of the form

$$x_i^{(j)} = \beta^j \sin\left(\theta^j + \varphi i\right) \qquad \text{for } 0 \le i \le m_j, 0 \le j \le n-1. \tag{4.5}$$

The corresponding eigenvalue, λ, is given by $2 - 2\cos\varphi$. We can show that there exists r eigenvectors of the form described in (4.5) and therefore *all* eigenvectors satisfy (4.5). We note that (4.5) satisfies (4.4) if $x_m^{(j)} = x_{m+1}^{(j)}$, *i.e.*,

$$\beta^j \sin\left(\theta^j + \varphi m_j\right) = \beta^j \sin\left(\theta^j + \varphi(m_j+1)\right) \tag{4.6}$$

$$\theta^j = \frac{\pi}{2} - \frac{\varphi}{2}\left(2m_j + 1\right) \tag{4.7}$$

Since $x_0^{(0)}$, $x_0^{(1)}$, $\cdots$, $x_0^{(n-1)}$ all represent the same point, we have

$$\beta^0 \sin(\theta^0) = \beta^1 \sin(\theta^1) = \cdots = \beta^{n-1} \sin(\theta^{n-1}). \tag{4.8}$$

Finally, substituting (4.5) in (4.2), we have

$$n\beta^0 \sin\theta^0 - \sum_{j=0}^{n-1} \beta^j \sin\left(\theta^j + \varphi\right) = (2 - 2\cos\varphi)\beta^0 \sin\theta^0. \tag{4.9}$$

We consider the general asymmetric case in Section 4.2.1.1 and the special symmetric case in Section 4.2.1.2.

4.2.1.1 Asymmetric extended star graph

In the general asymmetric case, $m_i \neq m_j$. If $\sin\theta^j = 0$ for some j, then from (4.8), we have $\sin\theta^j = 0\ \forall j$. If $\sin\theta^j \neq 0$, then it follows from (4.8) that $\beta^j = 1/\sin\theta^j$

and from (4.9), we get

$$n - \sum_{j=0}^{n-1} \frac{\sin(\theta^j + \varphi)}{\sin(\theta^j)} - (2 - 2\cos\varphi) = 0, \qquad \text{or} \tag{4.10}$$

$$n - \sum_{j=0}^{n-1} \left(\cos\varphi + \sin\varphi \cot\theta^j\right) - 2 + 2\cos\varphi = 0, \qquad \text{or} \tag{4.11}$$

using (4.7)

$$\sum_{j=0}^{n-1} \left(\left(1 - \frac{2}{n}\right)(1 - \cos\varphi) - \sin\varphi \tan(\varphi l_j) \right) = 0, \qquad \text{or} \tag{4.12}$$

$$\sum_{j=0}^{n-1} f(\varphi, l_j) = 0, \tag{4.13}$$

where $l_j = m_j + 1/2$ and

$$f(\varphi, l_j) = \left(1 - \frac{2}{n}\right)(1 - \cos\varphi) - \sin\varphi \tan(\varphi l_j). \tag{4.14}$$

We prove in Appendix C that $f(\varphi, l_j)$ is monotonically decreasing in $[0, \pi]$ except at points of discontinuity that occur at $\pi(2k+1)/(2l_j)$ for $k = 0, 1, \cdots, m_j - 1$. The sum of monotonically decreasing functions is also monotonically decreasing. The eigenvalue, $2 - 2\cos\varphi$, is a monotonically increasing function of φ in $[0, \pi]$. Therefore, the smallest eigenvalues correspond to the smallest values of φ that satisfy (4.13). Let $m_0 > m_1 > \cdots > m_{n-1}$. Examining the interval $\left[0, \frac{2\pi}{2\max(m_j)+1}\right]$, we see that if $m_0 > m_1 > \cdots > m_{n-1} > m_0/2$, then there is exactly one solution for (4.13) in each of the $n-1$ intervals $\left[\frac{\pi}{2l_{j-1}}, \frac{\pi}{2l_j}\right]$ for $j = 1, 2, \cdots, n-1$. Figure 4.4 plots $\sum_{j=0}^{n-1} f(\varphi, l_j)$ and $\sum_{j=0}^{n-1} f'(\varphi, l_j)$ for the asymmetric example in Figure 4.2 (c) and clearly illustrates the monotonic nature of $\sum_{j=0}^{n-1} f(\varphi, l_j)$ as well as the location of the solutions. Let the solution in the k^{th} interval be φ_k and $\lambda_k = 2 - 2\cos\varphi_k$. We have $0 < \varphi_1 < \varphi_2 < \cdots < \varphi_{n-1} < \pi$, and thus $\lambda_1 < \lambda_2 < \cdots < \lambda_{n-1}$. We therefore have

$$\frac{\pi}{2l_{k-1}} < \varphi_k < \frac{\pi}{2l_k}, \tag{4.15}$$

and substituting in (4.7),

$$\frac{\pi}{2}\left(1-\frac{l_j}{l_k}\right) < \theta^j < \frac{\pi}{2}\left(1-\frac{l_j}{l_{k-1}}\right). \tag{4.16}$$

Considering the θ^j for the k^{th} eigenvector, we see that

$$\theta^0 < \cdots < \theta^{k-1} < 0 < \theta^k < \cdots < \theta^{n-1} \tag{4.17}$$

$$\sin\theta^0 < \cdots < \sin\theta^{k-1} < 0 < \sin\theta^k < \cdots < \sin\theta^{n-1} \tag{4.18}$$

We see that for the first eigenvector, corresponding to the smallest eigenvalue,

$$\beta^0 = 1/\sin\theta^0 < 0, \tag{4.19}$$

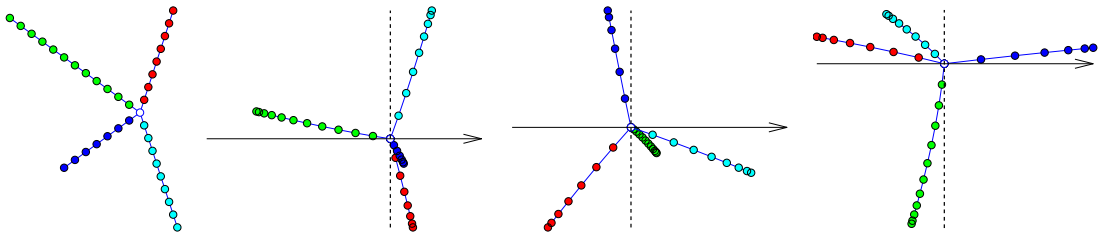
while

$$\beta^j = 1/\sin\theta^j > 0 \qquad\qquad \text{for } j = 1, \cdots, n-1. \tag{4.20}$$

Thus, substituting the value of β^j in (4.5), we see that the eigenvector corresponding to the smallest eigenvalue separates the longest chain from the rest. Similarly, the eigenvector corresponding to the second smallest eigenvalue separates the two longest chains from the rest and so on. Thus, we are able to discriminate between n chains using the eigenvectors corresponding to the $n-1$ smallest eigenvalues. Figure 4.2 (c) illustrates the plot of an asymmetric extended tree graph with $n = 4$. The nodes are plotted using the first $n-1$ eigenvectors. Figure 4.3 illustrates how the i^{th} eigenvector for the same example separates the i longest chain(s) from the remaining chain(s).

In case there are multiple chains with the same length, *i.e.*, $m_{j_1} = m_{j_2} = \cdots = m_{j_q} = m$, there exist eigenvalues $2-2\cos(k\pi/(2m+1))$ each with multiplicity $q-1$ in addition to the eigenvalues described above. The eigenvectors corresponding to the eigenvalue $2-2\cos(\pi/(2m+1))$ are of the form

$$x_i^{(j)} = \begin{cases} \beta^{(j)}\sin\left(\pi i/(2m+1)\right) & \text{if } j = j_1, j_2, \cdots, j_q, \\ 0 & \text{otherwise,} \end{cases} \tag{4.21}$$



(a) Graph (b) Eigenvector 1 (c) Eigenvector 2 (d) Eigenvector 3

Figure 4.3: Chain discrimination based on length: We plot the Laplacian eigenvectors for the graph in (a). The lengths of the different chains are 12 (green), 9 (cyan), 8 (red) and 7 (blue). (b), (c), and (d) are the plots of the first, second and third eigenvectors on the x-axis. We note that the first eigenvector separates the longest chain from the rest. Similarly, the second eigenvector separates the two longest chains from the rest and the third eigenvector separates the longest three from the remaining one.

where the $\beta^{(j)}$ are determined as described in the symmetric case in the following subsection. Therefore, the q chains of length m are discriminated by the $q-1$ eigenvectors corresponding to the eigenvalue $2-2\cos(\pi/(2m+1))$ of multiplicity $q-1$. While we have not explicitly dealt with ring graphs, the eigenvectors of ring graphs have a very similar structure to that of path graphs (Appendix B), and it is possible to establish similar results.

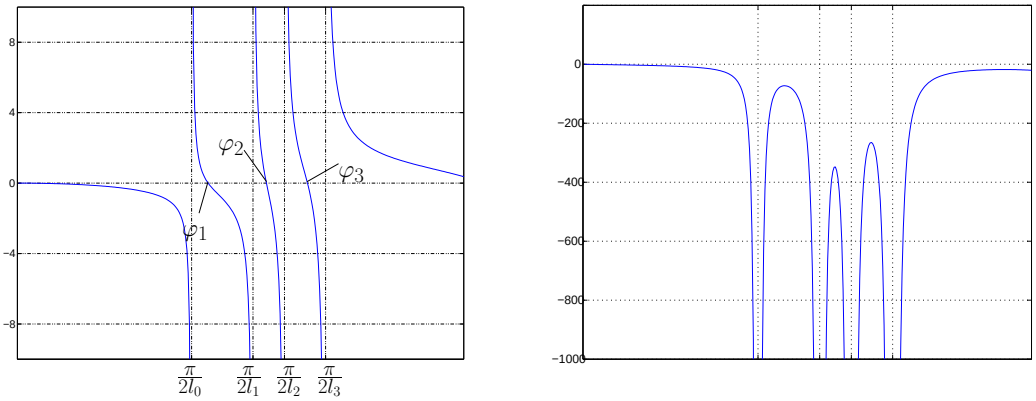
4.2.1.2 Symmetric extended star graph

In the symmetric extended star graph case, we have $m_j = m \ \forall j$ and it follows from (4.7) that $\theta_j = \theta$. Substituting in (4.9), we get

$$n\beta^{(0)} \sin\theta - \sin(\theta+\varphi) \sum_{j=0}^{n-1} \beta^{(j)} = (2-2\cos\varphi)\, \beta^{(0)} \sin\theta. \tag{4.22}$$

The solution to (4.22) depends on whether $\sin\theta = 0$ or $\sin\theta \neq 0$. If $\sin\theta = 0$, we have from (4.7),

$$\varphi = k\pi/(2m+1) \qquad \text{for } k = 1, 3, 5, \cdots \tag{4.23}$$



(a) $\sum_{j=0}^{n-1} f(\varphi, l_j)$ (b) $\sum_{j=0}^{n-1} f'(\varphi, l_j)$

Figure 4.4: The plots correspond to the example in Figure 4.2 (c) with $n = 4$ chains and $m_0 = 11, m_1 = 8, m_2 = 7, m_3 = 6$. The first $n-1$ intervals and the corresponding solutions φ_1, φ_2, and φ_3 are marked. We note that $\sum_{j=0}^{n-1} f'(\varphi, l_j) < 0$.

and substituting in (4.5), we get

$$x_i^{(j)} = \beta^{(j)} \sin\left(k\pi i/(2m+1)\right) \qquad \text{for } k = 1, 3, 5, \cdots \tag{4.24}$$

The eigenvalue is given by $2 - 2\cos\left(k\pi/(2m+1)\right)$. Equation (4.24) satisfies the conditions in (4.3) and (4.4). Substituting $x_0 = x_0^{(j)} = 0$ and $x_i^{(j)} = \beta_l^j \overline{x}_i$ in (4.2), we get $\sum_{j=0}^{n-1} \beta_l^j = 0$. We consider a set of eigenvectors, $\boldsymbol{x}_l$ given by

$$\boldsymbol{x}_l = \begin{pmatrix} 0 & \beta_l^0 \overline{\boldsymbol{x}}^{\mathsf{T}} & \beta_l^1 \overline{\boldsymbol{x}}^{\mathsf{T}} & \cdots & \beta_l^j \overline{\boldsymbol{x}}^{\mathsf{T}} & \cdots & \beta_l^{n-1} \overline{\boldsymbol{x}}^{\mathsf{T}} \end{pmatrix}^{\mathsf{T}}, \tag{4.25}$$

where $\overline{\boldsymbol{x}} = \begin{pmatrix} \overline{x}_1 & \overline{x}_2 & \cdots & \overline{x}_m \end{pmatrix}^{\mathsf{T}}$, $\overline{x}_i = \sin\left(\pi k i/(2m+1)\right)$, and $x_i^{(j)} = \beta_l^j \overline{x}_i$. In order to ensure that the eigenvectors form an orthonormal set, we require that

$$\begin{pmatrix} \boldsymbol{x}_1 & \boldsymbol{x}_2 & \cdots \end{pmatrix}^{\mathsf{T}} \begin{pmatrix} \boldsymbol{x}_1 & \boldsymbol{x}_2 & \cdots \end{pmatrix} = I. \tag{4.26}$$

Let $\overline{\boldsymbol{x}}^{\mathsf{T}} \overline{\boldsymbol{x}} = c$. Then substituting (4.25) in the above equation, we get

$$c^2 \begin{pmatrix} \boldsymbol{\beta}_1 & \boldsymbol{\beta}_2 & \cdots \end{pmatrix}^{\mathsf{T}} \begin{pmatrix} \boldsymbol{\beta}_1 & \boldsymbol{\beta}_2 & \cdots \end{pmatrix} = I, \text{ where } \boldsymbol{\beta}_l = \begin{pmatrix} \beta_l^0 & \cdots & \beta_l^{n-1} \end{pmatrix}^{\mathsf{T}}. \tag{4.27}$$

Since $\sum_{j=0}^{n-1} \beta_i^j = 0$, the $\boldsymbol{\beta}_l$ must also satisfy $\boldsymbol{\beta}_l^\mathsf{T}\mathbf{1} = 0$. Thus, we are interested in vectors that, along with $\mathbf{1}$, form an orthonormal basis for $\mathbb{R}^n$. These vectors form the columns of the orthonormal matrix

$$B = \begin{pmatrix} \frac{\mathbf{1}}{\sqrt{n}} & c\boldsymbol{\beta}_1 & c\boldsymbol{\beta}_2 & \cdots \end{pmatrix} \tag{4.28}$$

and we have $BB^\mathsf{T} = B^\mathsf{T}B = I$. We can obtain $n-1$ such vectors, $\boldsymbol{\beta}_1, \cdots, \boldsymbol{\beta}_{n-1}$ and thus we have $n-1$ eigenvectors by (4.25). Let $\boldsymbol{b}_j = \begin{pmatrix} \beta_1^j & \beta_2^j & \cdots & \beta_{n-1}^j \end{pmatrix}^\mathsf{T}$. Then the j^{th} row of B is given by $\begin{pmatrix} 1/\sqrt{(n)} & \boldsymbol{b}_j^\mathsf{T} \end{pmatrix}$. Let us consider the mapping of nodes to $\mathbb{R}^{n-1}$ using the $n-1$ eigenvectors $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_{n-1}$. The i^{th} node on the j^{th} chain is mapped to $\overline{x}_i \begin{pmatrix} \beta_1^j & \beta_2^j & \cdots & \beta_{n-1}^j \end{pmatrix}^\mathsf{T} = \overline{x}_i \boldsymbol{b}_j$. Thus all points on the j^{th} chain are mapped to the line $\alpha \boldsymbol{b}_j$ in $\mathbb{R}^{n-1}$. We can compute the angle between any two lines as $\cos^{-1}\left(\boldsymbol{b}_i^\mathsf{T}\boldsymbol{b}_j / \left(\|\boldsymbol{b}_i\|\|\boldsymbol{b}_j\|\right)\right)$. Since $BB^\mathsf{T} = I$, considering the $(i,j)^{\text{th}}$ element of BB^T, we have

$$\begin{pmatrix} 1/\sqrt{n} & \boldsymbol{b}_i^\mathsf{T} \end{pmatrix} \begin{pmatrix} 1/\sqrt{n} \\ \boldsymbol{b}_j \end{pmatrix} = 1/n + \boldsymbol{b}_i^\mathsf{T}\boldsymbol{b}_j = \delta_{ij}. \tag{4.29}$$

It follows from (4.29) that for $i \neq j$ and $i = j$ respectively, that

$$\boldsymbol{b}_i^\mathsf{T}\boldsymbol{b}_j = -1/n \tag{4.30}$$

and

$$\|\boldsymbol{b}_i\| = \|\boldsymbol{b}_j\| = \sqrt{(n-1)/n}. \tag{4.31}$$

We therefore have

$$\frac{\boldsymbol{b}_i^\mathsf{T}\boldsymbol{b}_j}{\|\boldsymbol{b}_i\|\|\boldsymbol{b}_j\|} = -\frac{1}{n-1}. \tag{4.32}$$

We thus see that the nodes are mapped to n lines in $\mathbb{R}^{n-1}$, such that the dot product of the direction of any pair of lines is negative and almost tending to zero with increasing n. In other words, the lines corresponding to different chains

are almost orthogonal to each other in $\mathbb{R}^{n-1}$. This structure is optimal for discriminating the nodes belonging to different chains. Figure 4.2 (b) illustrates an example of a symmetric extended tree graph with $n = 4$ in $\mathbb{R}^{n-1}$.

It remains to be shown that the smallest $n-1$ non-zero eigenvalues correspond to those above, *i.e.*, the case when $\theta = 0$. If $\sin\theta \neq 0$ in (4.22), it follows from (4.8) and the fact $\theta_0 = \cdots = \theta_{n-1} = \theta$, that

$$\beta^0 = \beta^1 = \cdots = \beta^{n-1}. \tag{4.33}$$

We can then divide (4.22) by $\beta^0 \sin\theta$ to obtain

$$n - n\frac{\sin(\theta+\varphi)}{\sin(\theta)} = 2 - 2\cos\varphi. \tag{4.34}$$

The form (4.34) is similar to (4.10), and in a fashion similar to (4.10)-(4.14), we simplify (4.34) to obtain

$$\left(1 - \frac{2}{n}\right)(1-\cos\varphi) - \sin\varphi\tan(\varphi l) = 0 \qquad \text{or} \tag{4.35}$$

$$f(\varphi, l) = 0 \tag{4.36}$$

where $l = m + 1/2$. We show in Appendix C that $f(\varphi, l)$ is monotonically decreasing in $[0, \pi]$ except for the discontinuities $\varphi = \pi(2k+1)/(2m+1)$ for $k = 0, 1, \cdots, m-1$. The smallest possible value of φ that is a solution for $f\left(\varphi, m + \frac{1}{2}\right) = 0$ lies in the interval $[\pi/(2m+1), 3\pi/(2m+1)]$ and and hence cannot be smaller than $\varphi = \pi/(2m+1)$. Thus, the $n-1$ smallest non-zero eigenvalues correspond to the case where $\theta = 0$.

4.2.2 Eigenvectors of grid graphs

Let $G = (V, E)$ and $H = (W, F)$ be graphs. Then $G \times H$ is the graph with vertex set $V \times W$ and edge set, $((v_1, w), (v_2, w))$ and $((v, w_1), (v, w_2))$ where $(v_1, v_2) \in E$ and $(w_1, w_2) \in F$. We use the following theorem [41] in our analysis.

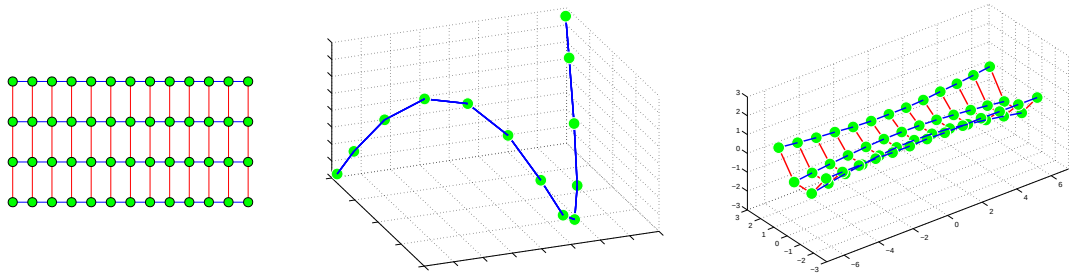
Theorem 1 *Let $G = (V, E)$ and $H = (W, F)$ be graphs with Laplacian eigenvalues $\lambda_0, \ldots, \lambda_m$ and $\mu_0, \ldots, \mu_n$ respectively. Then, for each $0 \leq i < m$ and $0 \leq j < n$, $G \times H$ has an eigenvector z of eigenvalue $\lambda_i + \mu_j$ such that $z(v, w) = x_i(v)y_j(w)$.*

Let G and H be path graphs of length m and n respectively, where $(k+1)n > m > kn$ and $k \in \mathbb{N}$. Then $\lambda_i = 2 - 2\cos(i\pi/2m)$ and $\mu_j = 2 - 2\cos(j\pi/2n)$. $G \times H$ is a grid graph with grid dimensions $m \times n$. Clearly, the larger the value of k, the "longer" the object. The term "long" here refers to the ratio of its largest dimension to the second largest dimension. We then have

$$0 = \mu_0 = \lambda_0 < \lambda_1 < \ldots < \lambda_k < \mu_1 < \lambda_{k+1} < \cdots . \tag{4.37}$$

Thus, the smallest k eigenvalues are $\mu_0 + \lambda_1, \cdots, \mu_0 + \lambda_k$ and the corresponding eigenvectors are $z(v, w) = x_i(v)y_j(w) = x_i(v)$. Thus, all points along the width of the object are *mapped to the same point* in k-dimensional LE and the nodes map to a smooth 1D curve in LE. We can easily see that the same results hold for 3D grid graphs as well where m and n are the largest and the second largest dimensions. The above result is illustrated in Figure 4.5, where all nodes along the width of the chain are mapped to the same point. This underlines the property of LE transform to map chains whose length is greater than their width to a 1D curve in eigenspace.

Combining the properties shown in 4.2.1 and 4.2.2, if we have n "long" chains (with the lengths greater than their widths) to segment, we need to map the nodes to LE of dimension $n - 1$. If we map to LE of higher dimensions, the nodes still retain their 1D structure as long as the chains are sufficiently long. The number of eigenvectors that can be used depends on the ratio of the length of the chains to their width. The greater the ratio, the greater the number of eigenvectors that can be used with the chains preserving their 1D structure in eigenspace.



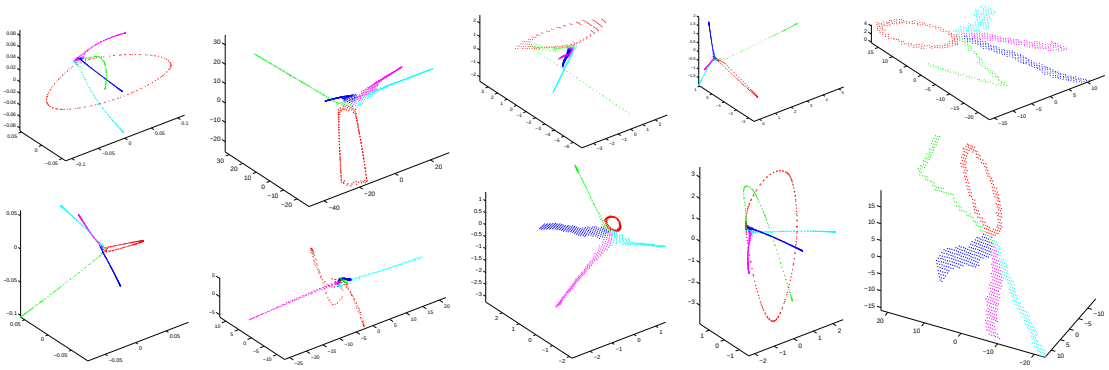
(a) Grid graph (b) Laplacian Eigenspace (c) Isomap

Figure 4.5: Example of a grid graph with length (13) three times greater than the width (4). The first three eigenvectors map all nodes along the width onto the same point and the corresponding structure in 3D LE is perfectly 1D.

4.3 Comparison with other manifold techniques

We observe the embedding of the data in example of Figure 4.1 (a) obtained from different manifold techniques using the program provided by Todd Witterman.[1] A comparison of the embedding obtained using techniques such as Laplacian Eigenmaps, Isomaps [73], Locally Linear Embedding [56], Diffusion maps [48] and Multi-dimensional Scaling [17], is presented in Figure 4.6. The Local Tangent Space Alignment mapping (LTSA) [82] did not result in any kind of structure and is not presented in the figure. We note that from the point of mapping to a 1D curve, LE map and the Diffusion map perform very well. Diffusion map embedding is a variation of LE map and uses a Gaussian kernel of width σ to construct a weighted graph and normalizes the Laplacian operator. Isomap tries to preserve the geodesic distance between the nodes and hence does not map the nodes onto a 1D curve as efficiently as LE map which does not try to preserve distances of any kind. We specifically compare Isomaps to Laplacian Eigenmaps in Figure 4.7. The objective is to segment nodes according to the chains to which they belong. We note that LE map does a much better job of mapping the nodes from the three chains to 1D curves than does the Isomap. We also note that

[1] Available at http://www.math.umn.edu/∼wittman/mani/.



(a) Laplacian (b) Isomap (c) LLE (d) Diffusion Map (e) MDS

Figure 4.6: Comparison of manifold techniques: The nodes in the first six dimensions of the embedding space using different techniques. The nodes correspond to the example in Figure 4.1.

the Isomap (Figure 4.7 (c)) has no structure in higher dimensions unlike LE map (Figure 4.7 (f)).

We measure quantitatively the 1D nature of the curves in LE and Isomap as follows. We consider nodes in chain 2 in the example illustrated in Figure 4.7. The position of node i along the chain is denoted by the site parameter t_i. In this example t_i happens to be the position of node i along the vertical axis and is marked for some of the nodes in Figure 4.7 a. Let node i map to $\boldsymbol{y}_i^L$ in LE and to $\boldsymbol{y}_i^I$ in the Isomap. We compute 1D splines ($\boldsymbol{f}^I$ and $\boldsymbol{f}^L$) to fit the nodes in Laplacian Eigenspace and Isomap cases in order to minimize $\sum_i e_i^L$ and $\sum_i e_i^I$ respectively which are given by

$$e_i^I = \|\boldsymbol{y}_i^I - \boldsymbol{f}^I(t_i)\| \tag{4.38}$$

and

$$e_i^L = \|\boldsymbol{y}_i^L - \boldsymbol{f}^L(t_i)\|. \tag{4.39}$$

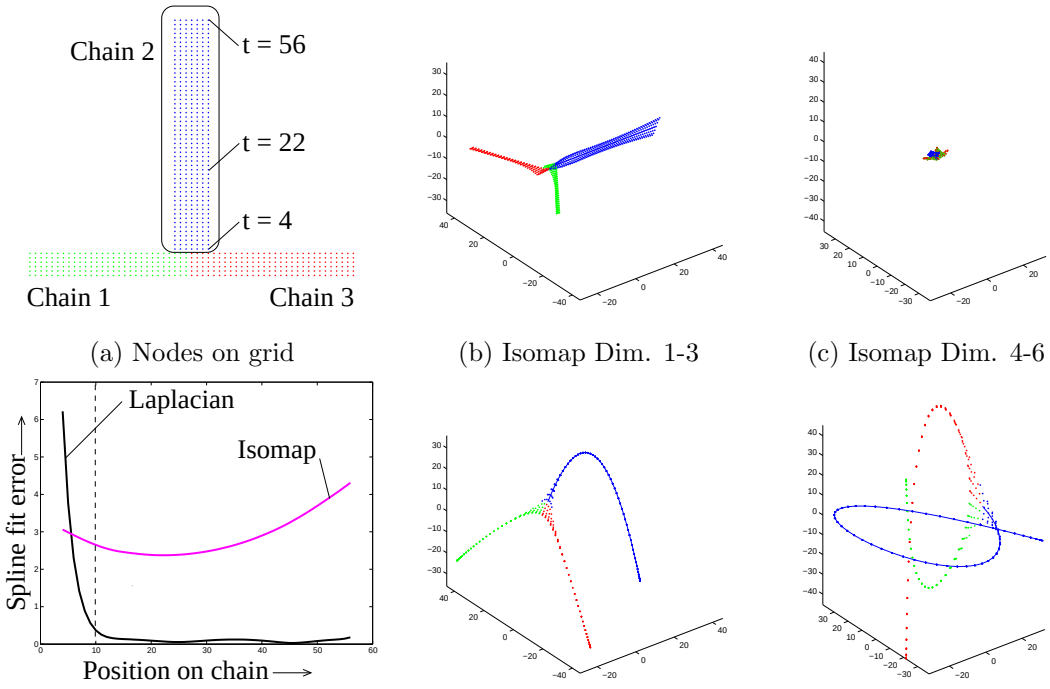
e_i^I and e_i^I are plotted in Figure 4.7 (d) versus the site location t. As can be seen in (a), the value of t denotes the distance of the node from the junction. We note

Method	Dimensions 1-3			Dimensions 1-6		
	L	MSE	MSE/L	L	MSE	MSE/L
LE	106.12	0.143	1.35e-03	265.86	1.471	5.532e-03
Isomap	54.36	8.019	1.48e-01	55.17	9.689	1.756e-01

Table 4.2: Comparison of LE and Isomap: We measure the 1D nature of the nodes in LE and Isomap for the example in Figure 4.7. L is the length of the spline used to fit the voxels and MSE is the Mean Squared Error in the spline fit. MSE/L is a measure of the 1D structure of the nodes.

that, for nodes on the right side of the vertical dashed line $e^L(t)$ is very small. However, on the left hand side of the dashed line, *i.e.*, as the nodes approach the junction $e^L(t)$ starts rapidly increasing. This plot indicates that a 1D spline is an excellent choice to fit nodes in Laplacian Eigenspace, except near a junction. The spline fit error is therefore an excellent indicator of the position of a voxel with respect to a junction. In the case of Isomap, on the other hand, the spline fit error is more or less constant for the length of the chain, and is not negligible compared to the length of the spline. Table 4.2 compares the spline fit error voxels on the right side of the vertical dashed line for both Isomap and LE with respect to the length of their respective splines. We observe that the errors in the case of Isomap are *two orders of magnitude* greater than the spline fit error for LE as is clearly obvious in Figure 4.7 (d).

We also compare the LE map and Isomap using a real example in Figure 4.8. It is obvious that as far as segmenting using 1D splines is concerned, the LE provides a much better mapping than does Isomap. We also note the nodes retain some structure in higher dimensions in LE, unlike Isomap. In addition to the properties described in the previous section, the LE map has the following advantages. The neighborhood matrix, W, is easily and efficiently computed as the points lie on a grid. The mappping is global in nature. It is not necessary to know the exact number of chains that we need to segment as the 1D structure is retained in higher



(a) Nodes on grid (b) Isomap Dim. 1-3 (c) Isomap Dim. 4-6

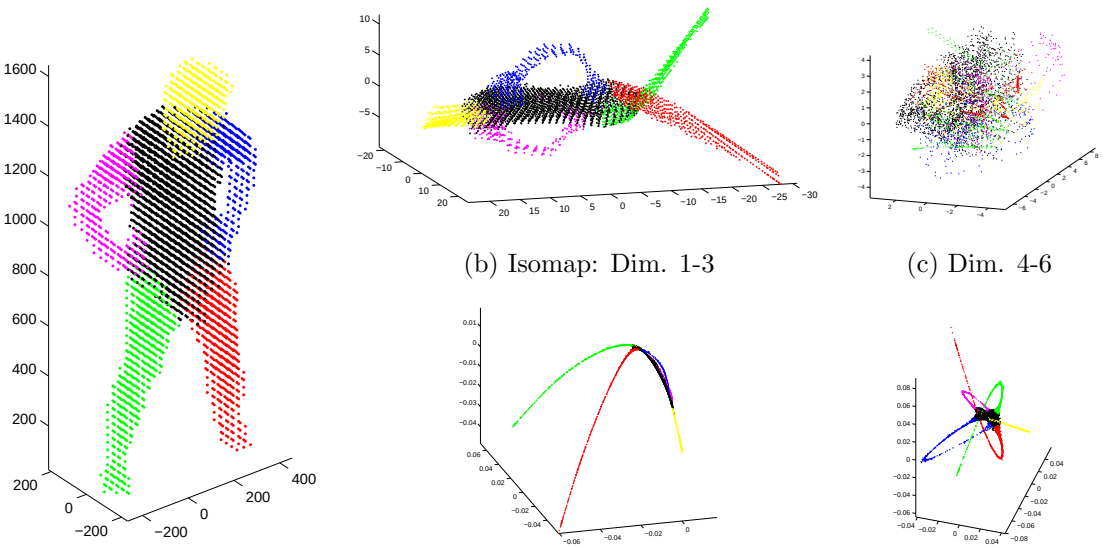
(d) Error (e) LE Dim. 1-3 (f) LE Dim. 4-6

Figure 4.7: Isomap versus LE: The Isomap embedding is compared to the Laplacian embedding for the example in (a). The 1D structure is not retained in higher dimensions in the case of Isomap as it tries to preserve geodesic distances.

dimensions. We can, therefore, map to a higher dimensional space than strictly necessary. For e.g., if we wish to segment n chains, all of whose lengths are at least twice as much as their widths, we can map them to LE whose dimension is between $n-1$ and $2n$. Indeed, in the case of human subjects we map nodes to 6D LE, although the maximum number of chains at a junction is four.

4.4 Human body segmentation in Laplacian Eigenspace

We visualize the human body as being composed of several articulated chains connected at joints as described in Chapter 3. We have shown in the previous sections that the non-rigid chains of an object form smooth 1D curves in Laplacian Eigenspace. This is true of the human body, as the lengths of the articulated chains



(a) Normal (b) Isomap: Dim. 1-3 (c) Dim. 4-6

(d) LE: Dim. 1-3 (e) Dim. 4-6

Figure 4.8: Comparison of LE and Isomaps using a real example

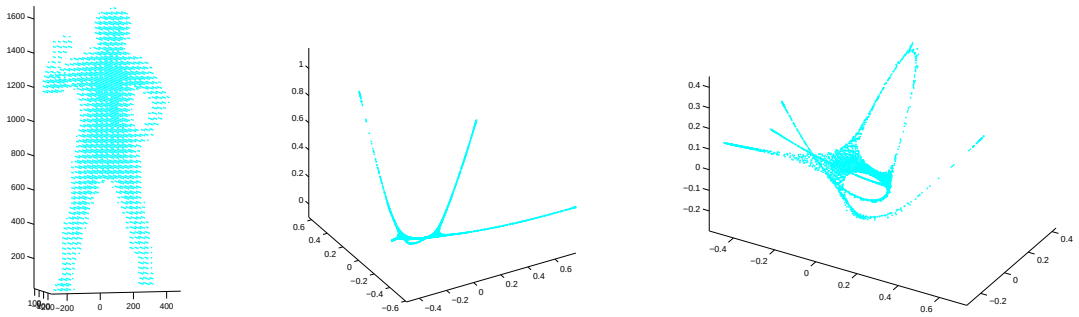
in the case of the human body are greater than their widths. In fact, the lengths of the limbs are much greater than their widths. Since the transform is based on a graph structure, the edges of which are determined by the neighborhood relations between voxels in normal 3D space, it is minimally affected by the articulation at joints. However, at points where three or more such segments meet, for e.g., at the neck joint (head, two arms and trunk), the nodes lose their 1D structure and diverge in different directions as shown in the previous sections. We exploit this structure of the voxels in LE in order to perform segmentation by fitting a different 1D spline to each articulated chain. This process also enables us to obtain the position of the nodes along their respective articulated chains. All operations described below are performed in LE. We describe the segmentation algorithm using the example in Figure 4.9 which is voxel data obtained from a real video sequence. The example considers the subject in a pose with self-contact in order to illustrate the power of the segmentation process. The steps in the segmentation algorithm are listed below and are described in the sections that follow.

1. *Initialization*: The initialization step describes the process of identifying a set of voxels to begin growing the spline.

2. *Spline Fitting*: The spline fitting step describes how to estimate a spline to fit the given set of voxels.

3. *Propagation*: The propagation step describes the growth of the spline.

4. *Termination*: The termination step describes the conditions under which the growth of the spline is terminated.

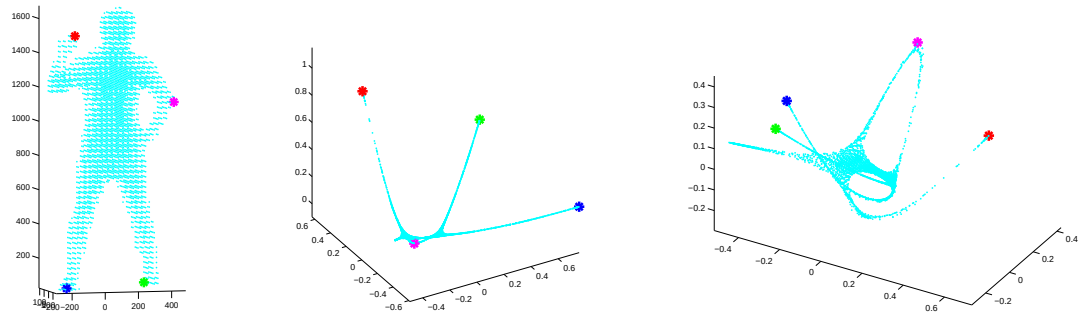


(a) Normal (b) Dim. 1-3 (c) Dim. 4-6

Figure 4.9: Voxels in normal space and eigenspace: (b) and (c) illustrate the structure in eigenspace and represent eigenvectors 1-3 and 4-6 respectively.

4.4.1 Initialization

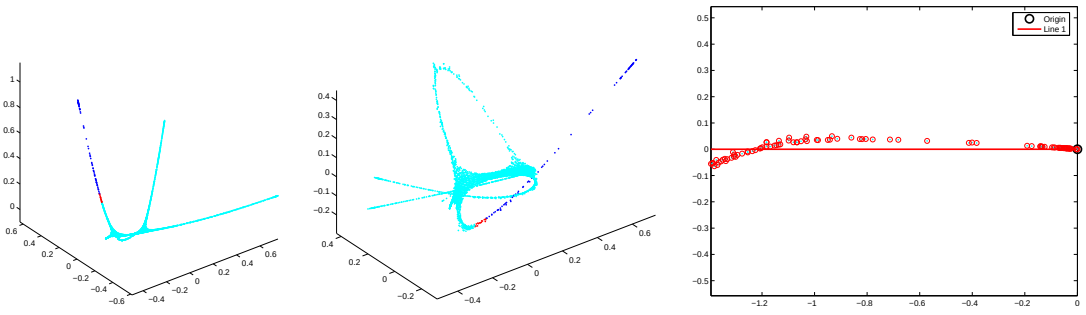
We can classify the articulated chains into two types according to whether they are connected to other chains at one end (Type 1) or both ends (Type 2). In the example in Figure 4.10 (a), the two legs, head, and one of the arms are of Type 1, *i.e.*, one end of the chain is free, and the left arm and the trunk are of Type 2, *i.e.*, both ends are attached to other chains. For Type 1 chains, we note that the node at the free end is farthest from other chains. However, for Type 2 chains, the node that is farthest from other chains lies in the middle of the chain. In order to initialize the spline, we begin with the node that is farthest from all existing



(a) Normal (b) Dim. 1-3 (c) Dim. 4-6

Figure 4.10: Voxels in normal space and eigenspace: The red, green, blue, magenta asterisks denote the starting node for the first, second, third and fourth splines.

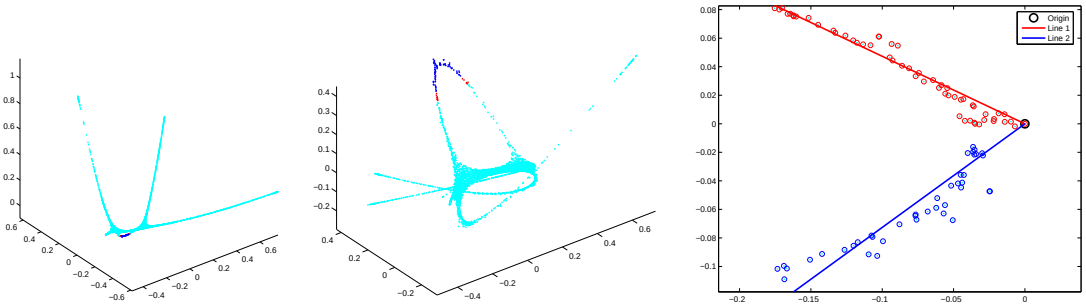
chains. To begin with, in the absence of existing splines, we begin with the node that is farthest from the origin. The initial node is denoted by the red asterisk in Figure 4.10. The starting node for the second, third and the fourth splines are denoted by the green, blue and magenta asterisks.



(a) Dim. 1-3 (b) Dim. 4-6 (c) Line fit

Figure 4.11: Fitting lines to nodes for Type 1 chains

We obtain a set of nodes that are closest to the initial node as can be seen in Figure 4.11 (a-b) and Figure 4.12 (a-b) for the Type 1 and Type 2 cases respectively. We then determine if the initial node lies at one end of the curve or in the middle by computing the number of lines in space that can be fit to the cluster of nodes. We find N_0 closest nodes (Euclidean distance), $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_{N_0}$, to the initial node, $\boldsymbol{y}_0$ and perform PCA (Principal Component Analysis) on $\boldsymbol{y}_i - \boldsymbol{y}_0$



(a) Dim. 1-3 (b) Dim. 4-6 (c) Line fit

Figure 4.12: Fitting lines to nodes for Type 2 chains

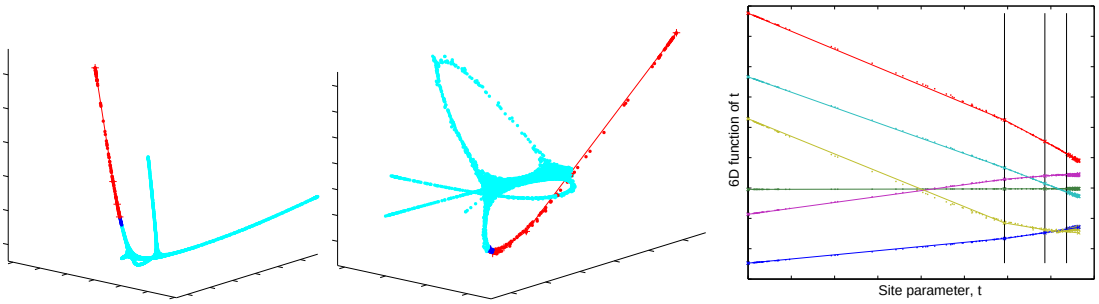
to find the two biggest *principal components* $\boldsymbol{u}^{(a)}$ and $\boldsymbol{u}^{(b)}$. The N_0 closest nodes are marked in blue in Figure 4.11 (a-b) while the first two principal components of the N_0 nodes are plotted in Figure 4.11 (c). We find the principal directions (lines) which are linear functions of the two principal components. In the Type 1 case (Figure 4.11 (c)), there is only one direction and we grow a single spline from that point. In the Type 2 case (Figure 4.12), there are two principal directions, as we start in the middle, and we grow a spline in each direction independently.

4.4.2 Spline fitting

We illustrate the spline fitting procedure using Figure 4.13. Given a set of nodes and the principal axis as computed in the previous section, we project each node, $\boldsymbol{y}_i$, onto the principal axis to obtain its site value t_i. The cluster of nodes and the principal axes are plotted in Figure 4.13. The nodes which are 6D vectors are plotted against their site parameters, t, in Figure 4.13 (c). A 6D spline, $\boldsymbol{f}^{\mathrm{L}}$, can be computed to minimize the error given by

$$\sum_i \|\boldsymbol{f}^N(t_i) - \boldsymbol{y}_i\|^2. \tag{4.40}$$

The spline computed is a cubic spline with two continuous derivatives and is computed using the MATLAB spline toolbox function *spap2*.



(a) Dim. 1-3 (b) Dim. 4-6 (c) Fitting spline

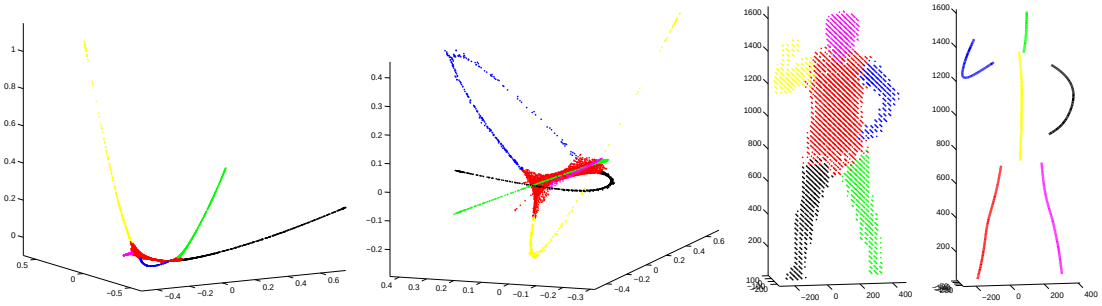
Figure 4.13: Spline fitting: We grow the spline by adding nodes at the growing end. The blue nodes denote the nodes that are added to the existing nodes (red). We fit a spline in each dimension in (c). x-axis represents the site parameter t, and the y-axis represents the location of node in different dimensions.

4.4.3 Propagation

We propagate the spline by adding nodes that are closest to the growing end of the spline, *e.g.*, the blue nodes in Figure 4.13 (a)-(b). The principal axis used to compute the site value is recomputed locally using the additional nodes. When the angle between the recomputed principal axis and the previous principal axis, exceeds $\theta_{\max}$, a new principal axis and a corresponding pivot point is computed. This is so that the principal axis adapts to the curvature of the voxels. The black vertical lines in Figure 4.13 (c) denote the boundaries of different principal axes. $\theta_{\max}$ is an adaptive threshold and is computed as

$$\theta_{\max} = \theta_{\text{MAX}} \frac{e_{\text{old}}}{(e_{\text{old}} + 5e_{\text{new}})}, \tag{4.41}$$

where $\theta_{\text{MAX}} = 15°$. e_{old} is the spline fit error if the old principal axis is used and e_{new} is the spline fit error if the new principal axis is used. When the curvature of the spline is high, $e_{\text{new}} \ll e_{\text{old}}$ and $\theta_{\max} \approx \theta_{\text{MAX}} = 15°$. However, when the nodes diverge (*e.g.*, at a junction,) $e_{\text{new}} \approx e_{\text{old}}$ and $\theta_{\max} \approx \theta_{\text{MAX}}/6 = 2.5°$. The maximum angle between adjacent principal axes is therefore 15° when the curve is strongly 1D, and 3° when it is not.



(a) Dim. 1-3 (b) Dim. 4-6 (c) Segmented (d) Skeleton

Figure 4.14: Segmentation and registration in Eigenspace: The nodes are segmented in LE (a)-(b). The labels are represented in the original 3D space in (c). The computed skeleton is presented in (d).

4.4.4 Termination

A node is considered an outlier if the spline fit error of that node exceeds a fixed threshold, $CL\sqrt{d}$, where $C = 0.005$, L is the length of the average spline in LE (set to 1 as we have normalized the LE such that $\boldsymbol{y}_i \in [0,1]^6$) and d is the dimension of the LE. In other words, a node is an outlier if it does not lie close to the computed 1D spline in LE. The number of outliers increases rapidly at a junction because the nodes diverge in widely different directions. When the number of outlier nodes is greater than 5, we stop growing the spline and begin growing the next spline. We can use our knowledge about the human body and stop the spline fitting procedure when six splines have been discovered. In general the stopping criteria can be varied according the requirements of the application, *e.g.*, when no more 1D curves can be discovered. We show in Figure 4.14 (b-d) the successful segmentation of the voxels into different articulated chains although there is contact between the palm and the hip.

4.5 Constructing the skeleton curve

We compute the position of each node along the articulated chain implicitly as part of the segmentation algorithm. We note that the position of the node along the 1D curve in LE can be used to determine the position of the node *along the corresponding articulated chain in normal space* as well. For each node $\boldsymbol{y}_i$ in LE belonging to a segmented spline, we have the site parameter t_i. Let the position of the i^{th} node in normal space be given by $\boldsymbol{v}_i$; we then associate t_i with $\boldsymbol{v}_i$ and compute a smoothing spline in normal space using the set of nodes $(t_i, \boldsymbol{v}_i)$. The computed skeleton curve for the segmented example is illustrated in Figure 4.14 (d). The skeleton curve spline, $\boldsymbol{f}^{\text{S}}$, seeks to minimize the error given by

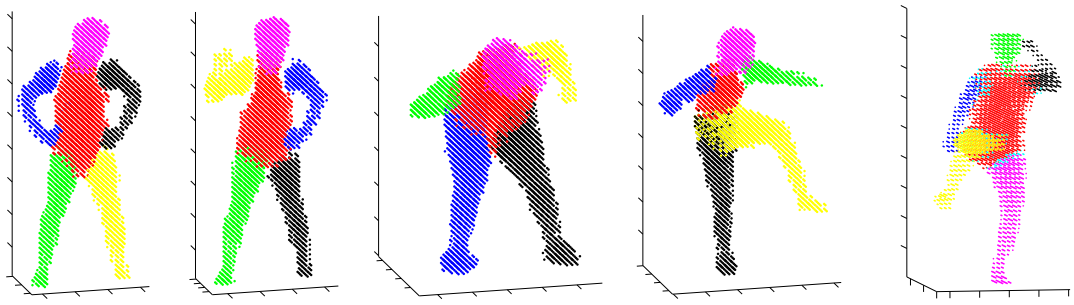
$$\sum_i \left\| \boldsymbol{f}^{\text{S}}(t_i) - \boldsymbol{v}_i \right\|^2. \tag{4.42}$$

We thus compute the skeleton curve for each of the splines in normal space. Type 1 chains contain a single spline. Type 2 chains contain two splines which are merged together to form a single spline. We now have a set of splines and can use a top-down approach to register each of the segmented chains as well as identify the pose.

4.6 Experimental results

We provide segmentation results on voxels obtained from different sources and under different conditions. We typically use a minimum of eight cameras in order to obtain reasonable quality voxel reconstruction. If there are fewer cameras, the space carving or voxel reconstruction tends to be of poor quality and may have severe defects such as "ghost" limbs, *i.e.*, limbs that do not really exist. The HumanEvaII data-set [61] contains sequences captured using four cameras and the corresponding voxels are of poor quality as can be observed in Figure 4.16. We map voxels to 6D LE in the case of voxels computed from 3D laser scans and sequences

captured from eight or more cameras. When the voxel is of poorer quality, we map voxels to 5D LE. We provide results on three different sequences. The first sequence was captured using 12 cameras and the voxel reconstruction is fairly accurate. We present segmentation results on four different subjects with different BMI (Body Mass Index) in Figure 4.15. We also consider challenging poses where there is self contact and algorithms like Isomap typically fail (Figure 4.15 (a)-(b)). The second set of sequences is from the HumanEvaII data-set.[2] The results are for one subject and are presented in Figure 4.16. The third set of sequences are voxels obtained using 3D laser scan meshes. The results for four subjects are presented in Figure 4.17. The voxel reconstruction in this case appears to be the most accurate due to the high resolution of the original 3D laser scans.



(a) Subject A (b) Subject A (c) Subject B (d) Subject C (e) Subject D

Figure 4.15: Segmentation results for different subjects and poses using voxels computed from 12 cameras.

[2]Available at http://vision.cs.brown.edu/humaneva/index.html.

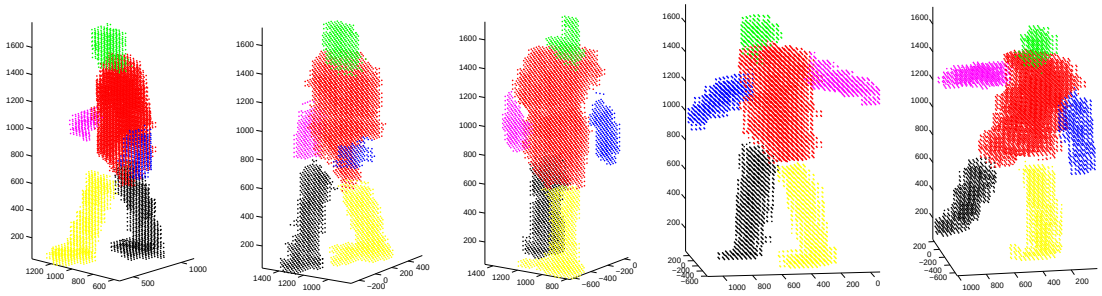
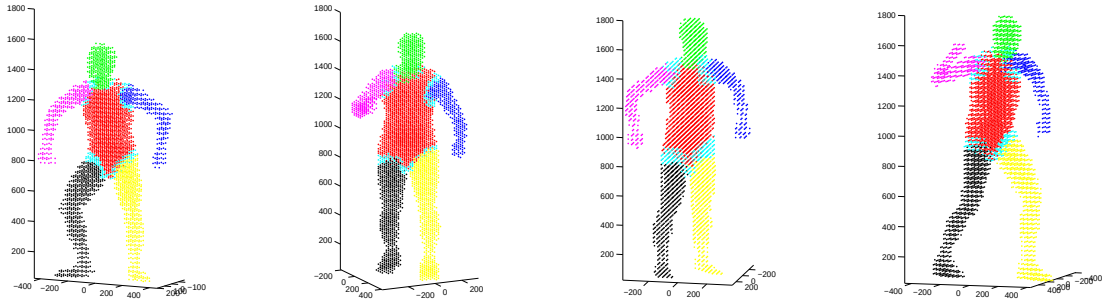


Figure 4.16: Segmentation results for the same subject using voxels computed 4 cameras in the HumanEvaII data set. The segmentation was performed in 5D LE, due to poor quality of voxel reconstruction.



(a) Subject A (b) Subject E (c) Subject F (d) Subject G

Figure 4.17: Segmentation results for different subjects using voxels computed 3D laser scans.

Chapter 5

Model and pose initialization

Having used a bottom-up segmentation algorithm to segment the voxel data of a human subject into its component articulated chains, we proceed to use our knowledge of the structure of the human body to guide us for pose and model estimation routines. Figure 5.1 illustrates the pipeline in the pose and model estimation process from voxel data. We have a frame of voxel data in Figure 5.1 (a) that has been segmented in Figure 5.1 (b)-(c). We also have the skeleton curve of each segmented articulated chain in Figure 5.1 (d). Since the segmentation algorithm was a bottom-up procedure it is necessary for us to perform registration, *i.e.*, we need to identify the segmented chains. Our model consists of six articulated chains labeled b_1, b_2, b_3, b_4, b_5, and b_6 that correspond to the trunk, head, right arm, left arm, right leg and left leg respectively as illustrated in Figure 5.1 (d). It remains for us to register the segmented chains to $b_1, \cdots, b_6$, and thereafter estimate the pose and the human body model parameters. We note that, while the registration in most poses, for instance the pose in Figure 5.2 (b), can be straightforward, our registration algorithm can also deal with poses where there is self contact as in Figure 5.2 (c). Given the human body model parameters, we can estimate the pose parameters. However, to begin with, we need to estimate the human body model parameters and the pose parameters simultaneously. We use a hierarchical estimation algorithm wherein we estimate the human body model parameters and pose beginning with a simple model with few parameters and proceeding progressively to the complete human body model described in Section 3.2.1. Our simplest human body model is a skeleton model that depends on a single parameter, the stature of the subject. A slightly more complex model is a skeleton model that

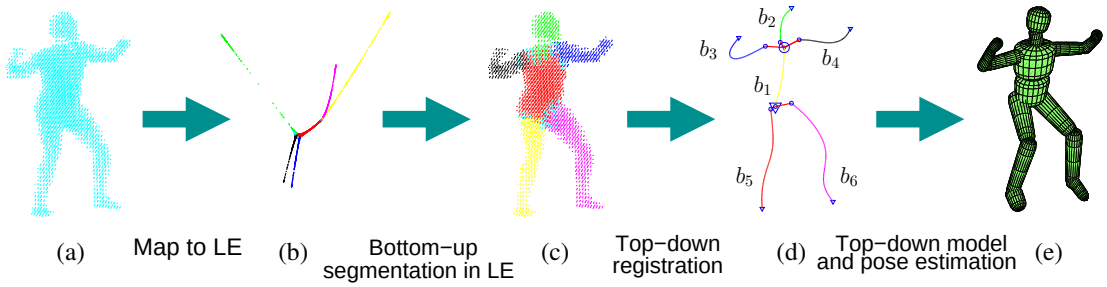


Figure 5.1: Block diagram illustrating the segmentation, registration and model estimation pipeline: We have segmented the voxel data into different articulated chains and need to register the segmented chains to the human body based on their properties and mutual connectivity. We can then estimate the human body model and pose parameters

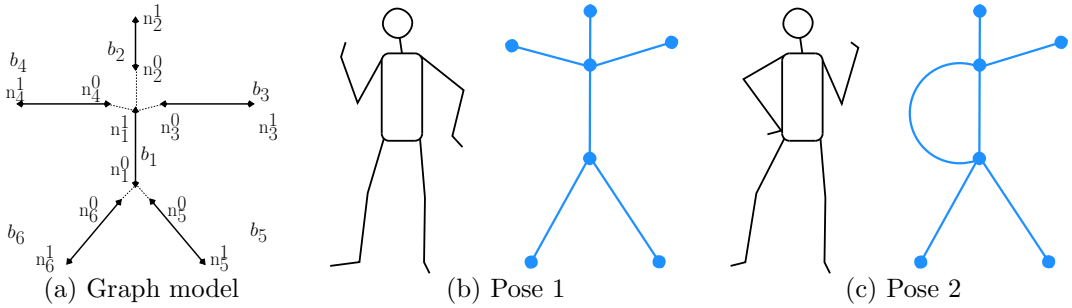
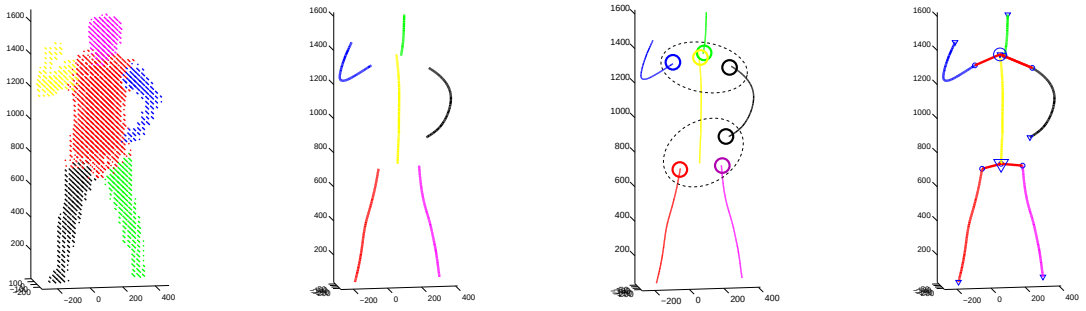


Figure 5.2: (a) Human body model consisting of six articulated chains. Each articulated chain b_i, has two vertices n_i^0 and n_i^1. The diagram illustrates the connectivity between the nodes of each articulated chain. (b) and (c) denote various poses and the corresponding connectivity of the component articulated chains.

allows flexibility in the location of the individual joint locations subject to symmetry constraints. The complete model includes the joint locations as well as the parameters of the super-quadric shape for each body segment.

We describe the probabilistic registration algorithm in Section 5.1. We then describe our hierarchical pose and body model estimation algorithm in Section 5.2. We present the results of the pose and body model estimation algorithms for voxel data obtained from different sources and under different conditions in Section 5.3.



(a) Segmented (b) Skeleton (c) Graph (d) Connected

Figure 5.3: Registration in LE: The labels are represented in the original 3D space in (a). The computed skeleton curve is presented in (b) and the two joints in (c). The correct registration is shown in (d).

5.1 Probabilistic registration

Figure 5.3 illustrates the registration task at hand. Having segmented the voxel data into its articulated chains, we are faced with the task of identifying them. We cannot resolve all ambiguities based on the connection between the spline segments alone as Figure 5.3 (c) illustrates. The trunk (yellow spline) is connected in exactly the same way as the left arm (black spline) and we need to make decisions considering all the possibilities. The probabilistic registration method described in this section considers both the individual properties of each segment as well as their mutual connectivity in order to find the most probable registration which is illustrated in Figure 5.3 (d). We obtained six splines in the previous section and we denote the i^{th} spline as s_i with nodes at each end (n_i^0 and n_i^1). Type 1 splines have a connected node at one end and Type 2 splines have connected nodes at both ends. We can then assign a probability to the "joint" between connected nodes n_i^k and n_j^l as

$$\Pr\left(\mathrm{n}_i^k \leftrightarrow \mathrm{n}_j^l\right) \propto \begin{cases} e^{-\frac{d(\mathrm{n}_i^k, \mathrm{n}_j^l)}{d_{\text{mean}}}} & \text{if } \mathrm{n}_i^k \text{ and } \mathrm{n}_j^l \text{ are connected,} \\ 0 & \text{otherwise,} \end{cases} \tag{5.1}$$

where $d(\mathrm{n}_i^k, \mathrm{n}_j^l)$ denotes the Euclidean distance between the nodes in LE. For each connected node, we compute the distance to the closest node, and d_{mean} is the mean of this distance for all connected nodes. We note that some of these "joints" are true joints but some of them are pseudo joints caused by contact between body parts, *e.g.*, between the hip and the left palm in Figure 5.3.

We wish to register the s_i to the known body chains b_i in Figure 5.2 (a). We denote possible registrations as a permutation $(j_1, j_2, \ldots, j_6)$ of $(1, 2, \ldots, 6)$ which indicates that $s_{j_i} = b_i$. The probability of the registration as given in (5.2), is the product (5.3) of the probability of each chain match being correct ($\Pr(b_i = s_{j_i})$) and the probability of the connection between the appropriate nodes.

$$\Pr((j_1, \ldots, j_6)) = \Pr(b_1 = s_{j_1}, \ldots, b_6 = s_{j_6}) \tag{5.2}$$

$$\begin{aligned} &= \Big(\Pr\left(n_{j_1}^0 \leftrightarrow n_{j_5}^0\right) \Pr\left(n_{j_1}^0 \leftrightarrow n_{j_6}^0\right) \Pr\left(n_{j_1}^1 \leftrightarrow n_{j_2}^0\right) \\ &\qquad \Pr\left(n_{j_1}^1 \leftrightarrow n_{j_3}^0\right) \Pr\left(n_{j_1}^1 \leftrightarrow n_{j_4}^0\right) \Big) \left(\prod_{i=1}^{6} \Pr(s_{j_i}) = b_i \right). \end{aligned} \tag{5.3}$$

The length and the thickness (or "girth") for each chain is obtained using the computed skeleton curve, $\boldsymbol{f}^{\mathrm{S}}$, computed in Section 4.5. The length of the chain is the length of the spline. The "girth" of the chain is the mean of the error of the spline reconstruction, $\|\boldsymbol{f}^{\mathrm{S}}(t_i) - \boldsymbol{v}_i\|^2$ also computed in (4.42) in Section 4.5. Let l_k and g_k be the length and "girth" of k^{th} chain. We normalize them by the maximum lengths respectively, *i.e.*, $l_k = l_k / \max\{l_k\}$ and $g_k = g_k / \max\{g_k\}$, and sort them so that

$$\left\{l_1^S, \ldots, l_6^S\right\} = \mathrm{sort}(\{l_1, \ldots, l_6\}) \tag{5.4}$$

and

$$\left\{g_1^S, \ldots, g_6^S\right\} = \mathrm{sort}(\{g_1, \ldots, g_6\}). \tag{5.5}$$

Note that b_1, b_2, are the trunk and head, b_3, b_4, the arms, and b_5, b_6, the legs. The

probability that a segmented chain s_k is actually b_i, $\Pr(s_k = b_i)$, is computed as

$$\Pr(s_k = b_i) = \begin{cases} c_{\text{trunk}} e^{-5|g_k - g_{\text{trunk}}|}, & i = 1, \\ c_{\text{head}} e^{-5|l_k - l_{\text{head}}|}, & i = 2, \\ c_{\text{ARM}} e^{-2|l_k - l_{\text{ARM}}| - 2|g_k - g_{\text{ARM}}|}, & i = 3, 4, \\ c_{\text{LEG}} e^{-2|l_k - l_{\text{LEG}}| - 2|g_k - g_{\text{LEG}}|}, & i = 5, 6, \end{cases} \tag{5.6}$$

where c_{trunk}, c_{head}, c_{ARM}, and c_{LEG} are normalizing constants and

$$\begin{aligned} g_{\text{trunk}} &= g_6^S, \\ l_{\text{head}} &= l_1^S, \\ l_{\text{ARM}} &= (l_3^S + l_4^S)/2, \\ g_{\text{ARM}} &= (g_1^S + g_2^S)/2, \\ l_{\text{LEG}} &= (l_5^S + l_6^S)/2, \\ g_{\text{LEG}} &= (g_3^S + g_4^S)/2. \end{aligned} \tag{5.7}$$

The key idea here is that we expect

$$l_{\text{head}} < l_{\text{trunk}} < l_{\text{ARM}} < l_{\text{LEG}} \tag{5.8}$$

and

$$g_{\text{ARM}} < g_{\text{LEG}} < g_{\text{head}} < g_{\text{trunk}}. \tag{5.9}$$

We therefore set $g_{\text{ARM}} = (g_1^S + g_2^S)/2$, *i.e.*, the mean of the two smallest values of "girth". The other values are set using similar reasoning.

We find the probability of all permutations and select the most probable permutation as the correct registration. We also require that the probability of registration be greater than a threshold and impose the additional implicit constraint that there exist exactly six segments. In fact for most poses (where there is no "loop-back"), the only chain that has non-zero probability of connections at both nodes is the trunk and therefore the number of permutations is greatly

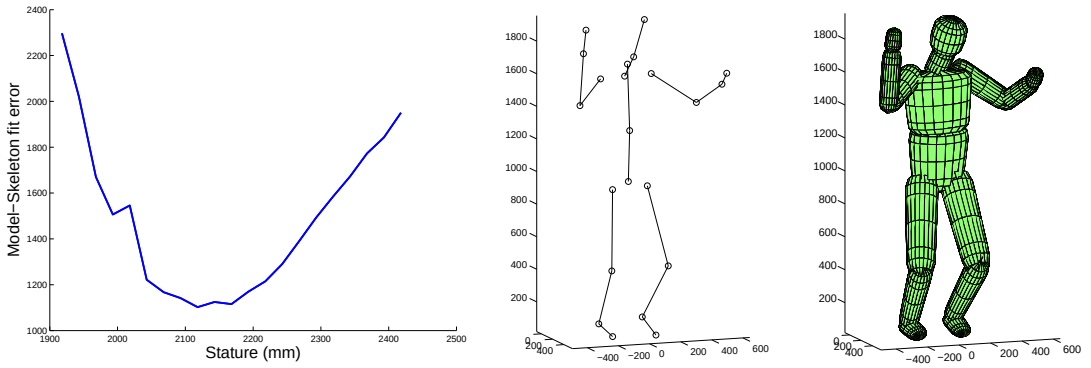
reduced. For the example in Figure 5.3 (c), the yellow and black chain have equal probability of being identified as the trunk based on the connections alone. The properties of the individual chains help discriminate between the trunk and the arms. We select the frames with the best registration probabilities for use in model estimation routines, while the other frames are discarded as unsuitable.

5.2 Pose and model estimation

As mentioned earlier, the model estimation routine is closely tied to the pose estimation routine, and we estimate both pose and the body model parameters simultaneously. Once the body model parameters are known, we can keep the body model fixed and use the same procedure to estimate the pose. The two sets of parameters that we would like to estimate are the pose vector and the body structure (joint locations and super-quadric parameters). We use a hierarchical approach, beginning with a skeletal model (joint locations and limb lengths) illustrated in Figure 5.4 (b) and then increase model complexity and refine parameters to obtain a volumetric model (super-quadric parameters) illustrated in Figure 5.4 (c). The joint locations cannot be reliably estimated from a single frame or pose; the reliability of the estimate typically depends on the articulation at the joint. We therefore use a set of key frames where registration is successful in order to estimate the body model parameters. These key frames are spread apart temporally so that a set of distinct poses is obtained.

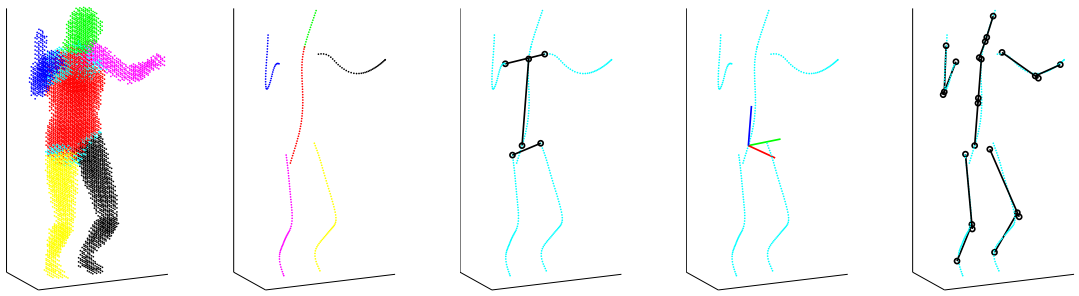
The stature (or height) of the subject is a key parameter that is strongly related to a number of human body model parameters, such as the lengths of long bones in the body [50]. Anthropometric studies have been performed on certain demographic groups to study the relationship between stature and the long bones in the body [12,37]. These studies indicate that we can estimate the lengths of the large bones for an average human subject from the stature parameter alone. We can construct a skeleton model for the average subject as a function of the stature

by scaling the limb lengths and the joint locations by the ratio of the stature of the subject to the stature of the average human. We describe how to initialize the pose using the registered skeleton curves in Section 5.2.1. We estimate the skeleton model parameters by minimizing the error between the estimated human skeleton model error and the computed skeleton curves. We describe how to compute the error of a given skeleton model in Section 5.2.2. In the first step of the model estimation routine, we find the optimal stature for the subject using the skeleton model as described in Section 5.2.3. In the second step, described in Section 5.2.4, we optimize for the joint locations based on the skeleton model. In the third step, described in Section 5.2.5, we estimate and optimize for the super-quadric parameters using the full super-quadric model. We use functions provided in the MATLAB optimization toolbox in order to perform the non-linear optimizations.



(a) Fit error versus stature (b) Skeleton model (c) Super-quadric model

Figure 5.4: Hierarchical human body model estimation: (a) We first estimate the stature parameter of the subject. (b) The stature is used to build a skeleton model, whose parameters are optimized. (c) Finally, we estimate the parameters of a complete super-quadric model



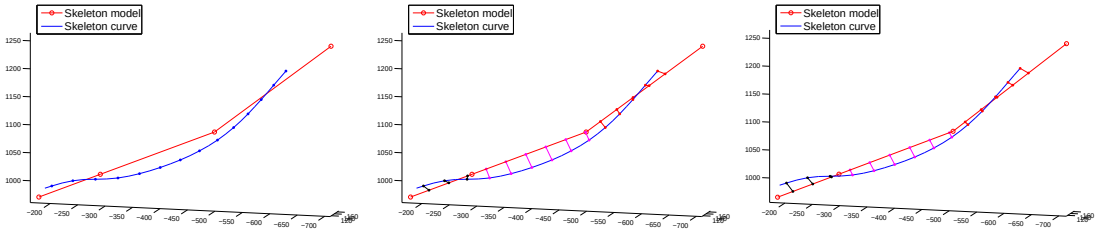
(a) (b) (c) (d) (e)

Figure 5.5: Pose initialization using registered skeleton curves.

5.2.1 Pose initialization

We describe the pose initialization step, given registered skeleton curves, using a successfully segmented and registered frame illustrated in Figure 5.5. The skeleton curve is sampled at regular intervals of 20mm to obtain a set of ordered points for each body chain (trunk, head, two arms and two legs). The sampled skeleton curve is illustrated in the images in Figure 5.5 (c)-(e). Hereafter, we use the term skeleton curve to mean the sampled skeleton curve.

The pose is computed using the skeleton curves and is initialized in two steps. First, the pose of the trunk is determined and second, the pose of the remaining five articulated segments is computed. The z-axis of the trunk is aligned with the skeleton curve of the trunk as marked in Figure 5.5 (c). The y-axis of the trunk is in the direction of the line joining the right pelvic joint to the left pelvic joint in the actual model. This direction is set to be the average of the rays from the right to left shoulder joint and from the right to left pelvic joint on the skeleton curve marked in Figure 5.5 (c). The x-axis points in the forward direction. This direction is estimated using the direction of the feet and is orthogonal to the computed yz plane. The location of the origin is set to be near one end of the skeleton curve of the trunk. The xyz axis orientation that describes the pose of the trunk is illustrated in Figure 5.5 (d). Once the trunk pose has been estimated,



(a) (b) (c)

Figure 5.6: Computing the distance between skeleton curve and skeleton model: (a) denotes sample points on skeleton curve. (b) denotes the distance to the closest point on skeleton model before optimization. (c) denotes the same after optimization.

the joint locations at the pelvis, shoulders and neck are fixed. It is then possible to estimate the pose of each of the articulated chains independently using the error described in Section 5.2.2. The objective is to compute the pose of the skeleton model, so that the distance between the points on the skeleton curve and the skeleton model is minimized. The initial estimate of the pose is illustrated in Figure 5.5 (e).

5.2.2 Computing skeleton fit error

Consider a set of ordered points $\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n$, on a skeleton curve corresponding, for instance, to the arm as in Figure 5.6. The corresponding skeleton model for the arm consists of three line segments, L_1, L_2, and L_3. We compute the distance, e_i^j, between $\boldsymbol{x}_i$ and the closest point on line segment L_j. We then assign each point to a line segment. Since the set of points on the skeleton curve is ordered, we impose the constraint that the assignment is performed in a monotonic manner, *i.e.*, points $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_{n_1}$ are assigned to L_1, points $\boldsymbol{x}_{n_1+1}, \cdots, \boldsymbol{x}_{n_2}$ are assigned to L_2 and points $\boldsymbol{x}_{n_2+1}, \cdots, \boldsymbol{x}_n$ are assigned to L_3. For a given value of n_1, n_2 is chosen so that the distance between points $\boldsymbol{x}_{n_1}$ and $\boldsymbol{x}_{n_2}$ is approximately equal to the length of line segment L_2. For the above assignment, the distance between the skeleton curve is given by the vector $(\, e_1^1 \cdots e_{n_1}^1 \; e_{n_1+1}^2 \cdots e_{n_2}^2 \; e_{n_2+1}^3 \cdots e_n^3 \,)'$. n_1 and

n_2 are chosen so as to minimize the sum of the elements in the vector.

Let us denote a given skeleton curve as $\mathcal{C}$ and the body model as $\mathcal{M}$. The pose parameter is given by $\mathbf{\Phi}$. Given a skeleton model, skeleton curve and the pose vector, we can compute the fit error of the skeleton model. We compute the fit error of the pose and model with the skeleton curve as $\boldsymbol{h}(\mathcal{C}, \mathcal{M}, \mathbf{\Phi})$, where the function $\boldsymbol{h}$ is a vector that comprises the error between the six skeleton curves and the corresponding line segments in the skeleton model as described above. We can therefore minimize the function $\boldsymbol{h}'\boldsymbol{h}$ by varying either the pose $\mathbf{\Phi}$ or the model parameters $\mathcal{M}$ using non-linear optimization. We drop the dependence of $\boldsymbol{h}$ on $\mathcal{C}, \mathcal{M}$ and $\mathbf{\Phi}$ in subsequent sections.

5.2.3 Estimation of skeleton model from stature

In the first iteration of the algorithm, we optimize for the stature of the subject. We use the relationship between the stature and the length of the limbs for an average human. We can compute the complete skeleton model for an average subject given the stature. Given the length of the limbs in the curve skeleton, we can use the inverse relation to compute the height. We use the following formula to compute the stature from the length of the skeleton curve of a limb.

$$\text{stature} = \begin{cases} 2.89 * \text{length(Limb)}, & \text{if Limb is an arm,} \\ 1.69 * \text{length(Limb)}, & \text{if Limb is a leg.} \end{cases} \tag{5.10}$$

The estimated stature is the median of the stature estimated for all the limbs in the set of key frames. We construct a skeleton model for different values of stature in the region of the estimated stature and compute $\boldsymbol{h}'\boldsymbol{h}$ versus the stature and select that value of stature for which $\boldsymbol{h}'\boldsymbol{h}$ is the minimum. $\mathbf{\Phi}$ is initialized using the routine described in Section 5.2.1.

A plot of the skeleton fit error versus the stature allows us to determine the best stature value to initialize the skeleton model. One such plot for a synthetic sequence is presented in Figure 5.4 (a). The initial stature estimated was 2168mm

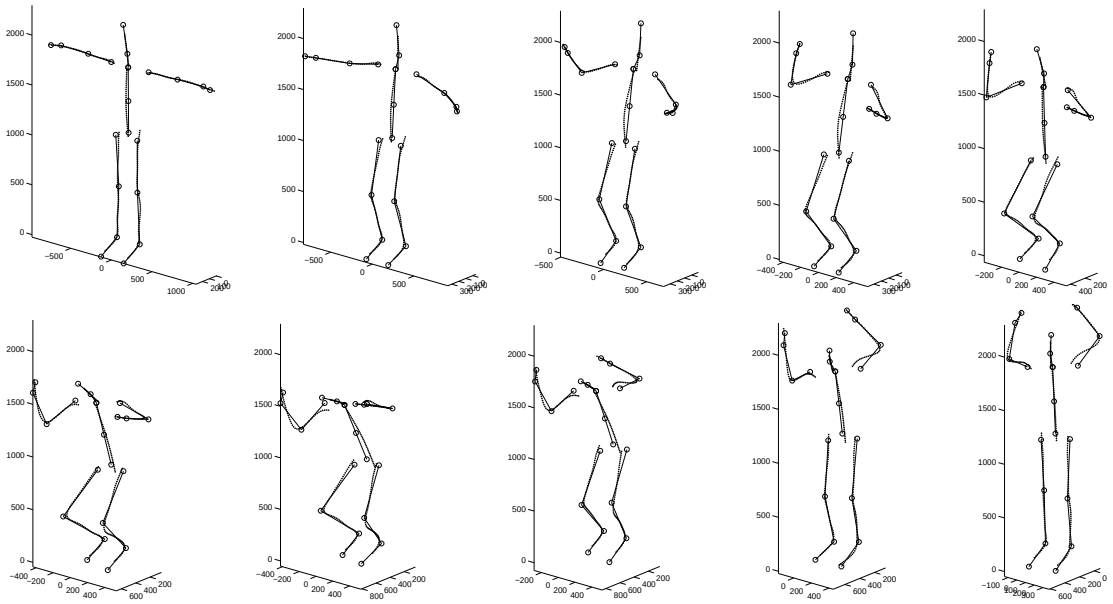


Figure 5.7: Fit of skeleton model after the stature has been optimized: The model $\mathcal{M}^0$ is super-imposed on the skeleton curves.

while the correct stature is 2100mm. It should be emphasized that the computation of the stature is in the context of its relationship to the body structure of the subject; we do not have any use for the stature in itself. The initial skeleton segments have been super-imposed on the computed skeleton curves in Figure 5.7. The optimization results in an optimal value of the stature, using which we can compute the initial pose estimate and the initial skeletal model estimate $\mathcal{M}^0$.

5.2.4 Optimization of joint locations

In the second step, we optimize the skeleton model parameters directly rather than through the stature parameter. $\mathcal{M}$ is allowed to vary in a bounded region centered around $\mathcal{M}^0$. The optimization is performed with bounded input. The pose and the body model parameters are alternately varied, *i.e.*, we optimize for $\mathcal{M}$ while keeping $\mathbf{\Phi}$ fixed and optimize for $\mathbf{\Phi}$ while keeping $\mathcal{M}$ fixed. We begin with the optimal pose and model estimated in the previous section. We label

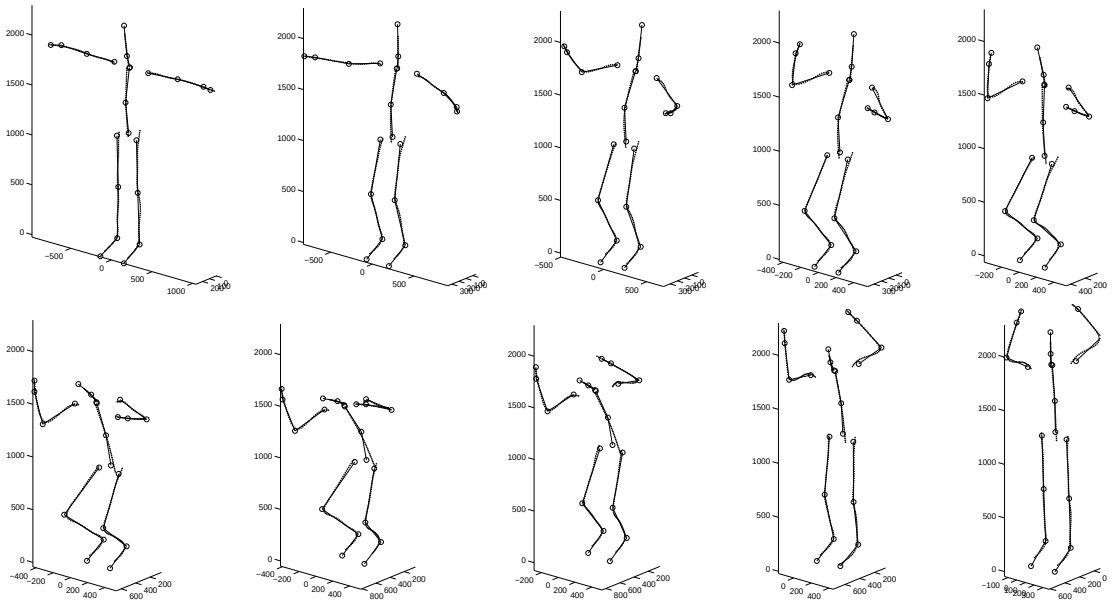
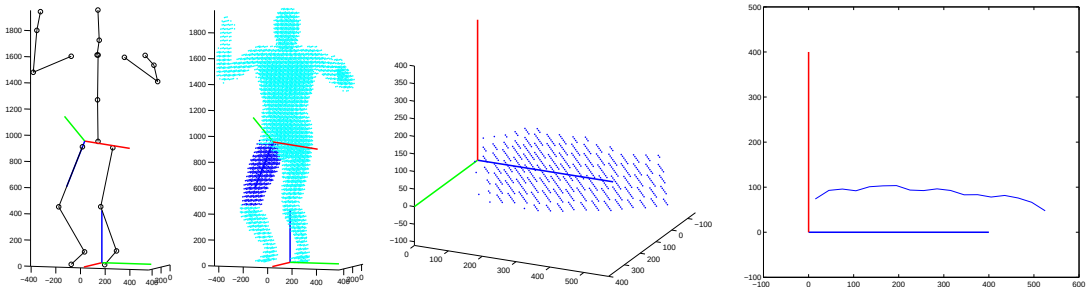


Figure 5.8: Fit of skeleton model after the joint locations have been optimized: The model $\mathcal{M}^1$ is super-imposed on the skeleton curves.

the optimal model parameters after optimization of the joint locations as $\mathcal{M}^1$. The skeleton model super-imposed on the curve skeleton after joint locations are optimized is presented in Figure 5.8.

5.2.5 Estimation of super-quadric parameters

The super-quadric parameters for the trunk, head, arm, forearm, thigh, and leg are estimated from voxels as these body segments are large enough to be estimated using the voxel resolution in our experiments. At this stage, we know which body segment on the skeleton model each point on the skeleton curve is closest to. Since we also know the location of each voxel along the skeleton curve, we can associate each voxel to a body segment as illustrated in Figure 5.9. Each articulated chain can therefore be segmented into its component rigid segments. Using the estimated joint angles, the orientation of the coordinate frame attached to the component segment can also be computed. For a given body segment, the pose is normalized using the body coordinate frame, so that the body segment is



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

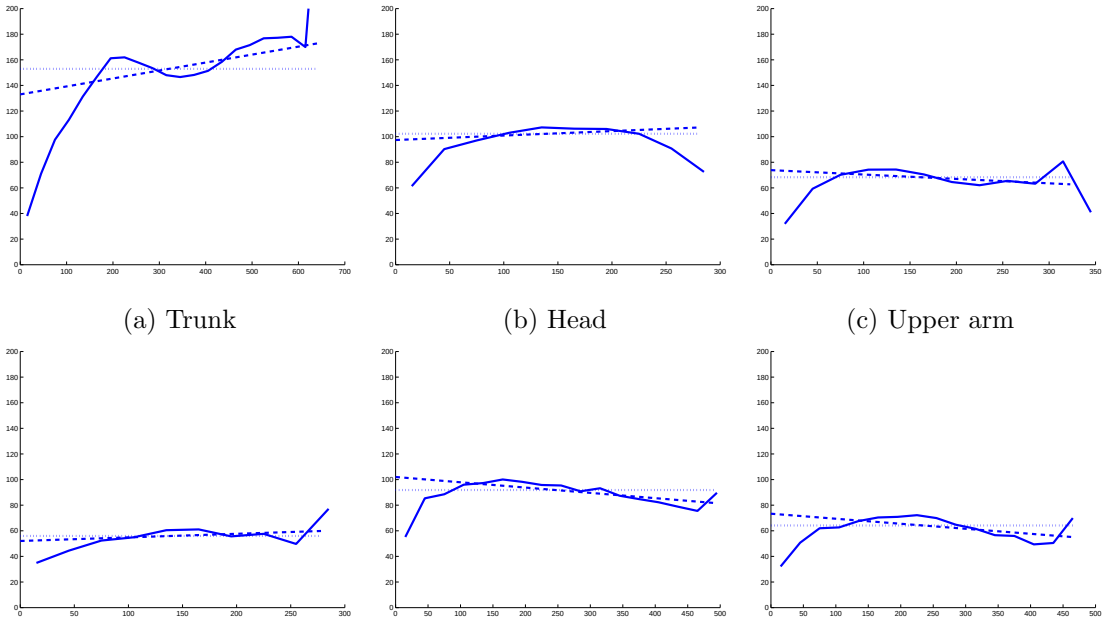
Figure 5.9: We classify a set of voxels using their position along the skeleton curve as belonging to the different segments in an articulated chain. (a) shows the coordinate frame representing the position of the right thigh, (b) shows the set of voxels associated with the thigh, (c) shows the voxels in the thigh transformed to the coordinate frame of the thigh, and (d) shows the radial profile along the z-axis.

positioned at the origin and aligned with the z-axis as in Figure 5.9 (c).

We compute the area of the cross-section of the voxels, A_z, (plane parallel to the xy-plane) at regularly spaced points along the z-axis. We assume that the cross-section is a disc and find the radius, r, from the area using the relation $A = \pi r^2$. An equivalent ellipse with equal area would be such that $x_0 y_0 = r^2$. We compute the radius at different points along the z-axis, , which we refer to as the radial profile, as

$$r_z = \sqrt{A_z/\pi}. \tag{5.11}$$

The radial profile for the thigh segment in a frame is illustrated in Figure 5.9 (d). The radial profile is computed in all the key frames for each body segment. The median radial profiles for some of the body segments are presented in Figure 5.10. The length, radius and the scale parameters of the body segment are computed from the median radial profile. We set $x_0 = y_0 = r$ in (3.2) for all body segments except the trunk and head, for which we determine the x_0 and y_0 parameters of the super-quadric in the following manner. We obtain the xy-histogram, $I(x, y)$, a function whose value at (x_i, y_i) is given by the number of voxels that have x and



(a) Trunk (b) Head (c) Upper arm

(d) Forearm (e) Thigh (f) Leg

Figure 5.10: Radial profiles of different body segments: The solid line is the median radial profile. The dotted line is the super-quadric radius with scale parameter set to zero. The dashed line is the super-quadric radius with estimated scale parameter. The x-axis of the plots is the distance in mm along the z-axis of the body segment coordinate system. The y-axis of the plots is the radius value also in mm.

y coordinates given by x_i and y_i respectively. The set of all points that lie inside the ellipse with parameters a and b is given by

$$S_{a,b} = \left\{ (x, y) : \left(\frac{x}{a}\right)^2 + \left(\frac{y}{b}\right)^2 < 1 \right\}. \tag{5.12}$$

We find the values of x_0 and y_0 that satisfy the constraint, $x_0 y_0 = r^2$ and maximize the function

$$\sum_{(x,y) \in S_{x_0,y_0}} I(x, y), \tag{5.13}$$

to estimate the x_0 and y_0 parameters.

Finally, we refine the pose using the super-quadric body segments and the voxels directly. The objective is to obtain the pose that maximizes the overlap

between the super-quadric model and the voxels. The pose is refined by bounded optimization of the pose parameter to minimize the "distance" between the voxels and the super-quadric model. This "distance" depends on the position of each voxel with respect to the closest super-quadric. The distance of a voxel is set to e^0 if the voxel is on the surface and e^{-1} if it is on the axis of the super-quadric. The distance increases exponentially as the voxel is farther from the surface of the super-quadric. The distance vector, $\boldsymbol{d}$ comprises of the distance of each voxel with respect to the super quadric and is given by

$$\boldsymbol{d} = \begin{pmatrix} d_1 & d_2 & \cdots & d_N \end{pmatrix}', \tag{5.14}$$

where

$$d_i = \min\left(d_i^{(1)}, d_i^{(2)}, \cdots, d_i^{(J)}\right). \tag{5.15}$$

$d_i^{(j)}$ is the distance of the i^{th} voxel with respect to the j^{th} body segment and given as

$$d_i^{(j)} = \begin{cases} \exp\left(r_i^j - q_i^j\right), & \text{if } 0 \le z_i^j \le z_0^j \\ \exp\left(r_i^j + p_i^j\right), & \text{otherwise,} \end{cases} \tag{5.16}$$

where

$$p_i^j = \min\left(\left|z_0^j - z_i^j\right|, \left|z_i^j\right|\right), \tag{5.17}$$

$$r_i^j = \sqrt{\left(\frac{x_i^j}{x_0^j}\right)^2 + \left(\frac{y_i^j}{y_0^j}\right)^2}, \text{ and} \tag{5.18}$$

$$q_i^j = \sqrt{\left(1 + s^j \frac{z_i^j}{z_0^j}\right)\left(1 - \left(1 - 2\frac{z_i^j}{z_0^j}\right)^d\right)}. \tag{5.19}$$

(x_i^j, y_i^j, z_i^j) are the voxel coordinates in the coordinate system of the j^{th} body segment and $(x_0^j, y_0^j, z_0^j, s^j, d^j)$ are the super-quadric parameters of the j^{th} body segment. Although the distance function appears complicated it is merely a measure of how close the voxel is to the central axis of the super-quadric. The optimal pose is the pose that minimizes $\boldsymbol{d}'\boldsymbol{d}$.

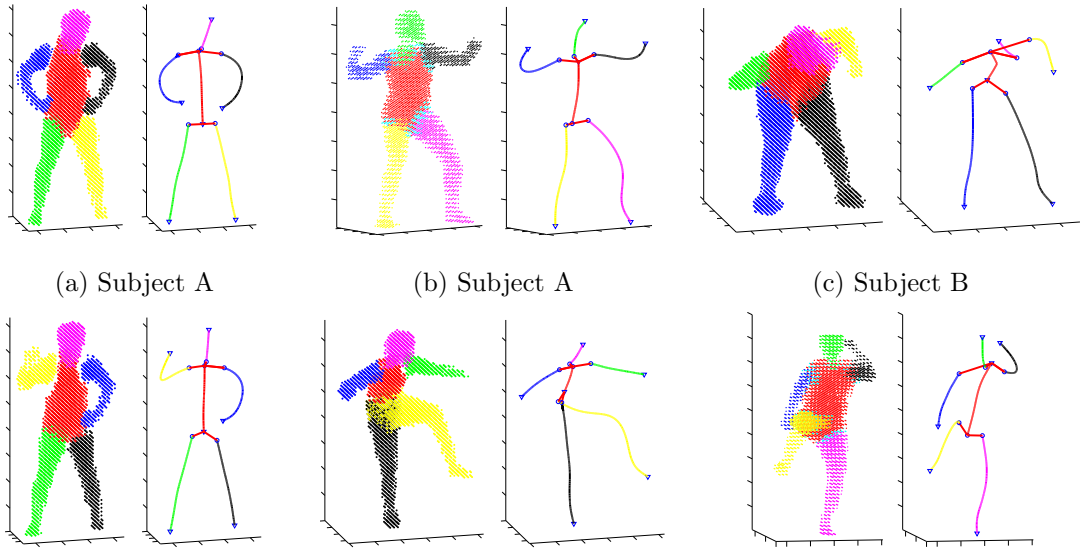
5.3 Experimental results

We present results of our experiments on synthetic data obtained from animation models, as well as real data obtained both from 3D laser scans and synchronized video sequences. The registration and model estimation results using voxels obtained from video data (which includes different subjects and poses with self-contact) are presented in Section 5.3.1. We also present pose estimation results on two sequences from the HumanEvaII database in Section 5.3.2. We also present the results of the model estimation algorithm on other sources such as 3D laser scan data in Section 5.3.3 as well as synthetic data in Section 5.3.4. These different sources result in voxel data with varying degrees of accuracy. Voxels of dimension 30mm $\times$ 30mm $\times$ 30mm were used in the experiments.

5.3.1 Registration of segmented voxels

The results of the registration algorithm on different subjects in both simple and difficult poses are presented in Figure 5.11. The voxels in this case were computed using images from multiple cameras. Gray scale images were captured from $N_{\text{cam}} = 12$ calibrated cameras. We have successfully performed segmentation and registration in the case of self contact as illustrated in Figure 5.11 (a) and (d), which other algorithms, such as [13], do not address. This probabilistic registration allows us to reject improbable registrations based on the estimated connections between the segments as well as lets us use prior knowledge of the properties of the different segments as well as the graph describing their connectivity.

We also present results on the model estimation and pose initialization. Given that the quality of the voxels construction is relatively inferior due to space carving and background subtraction artifacts, we used 20 frames in the human body model estimation algorithms. The results of the human body model estimation for different subjects are presented in Figure 5.12.



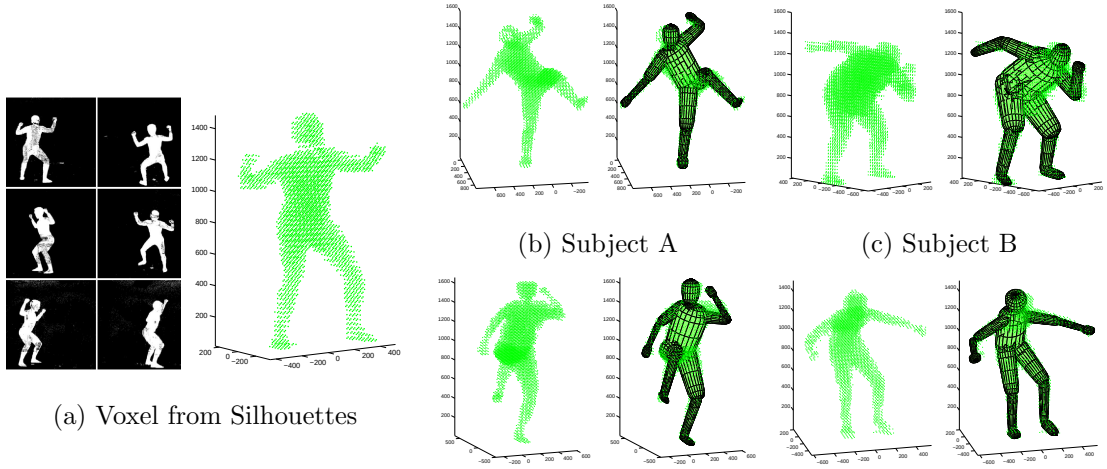
(a) Subject A (b) Subject A (c) Subject B

(d) Subject A (e) Subject C (f) Subject D

Figure 5.11: Registration for different subjects and poses.

5.3.2 HumanEvaII data

We present results of the segmentation and registration algorithm on two sequences from the HumanEvaII dataset in Figure 5.14. We map the nodes to 5D LE, as the accuracy of voxel reconstruction is low, and we do not gain much by mapping to a higher dimensional space. The algorithm does not find the requisite number of body segments in the majority of the frames principally due to two reasons. The arms are too close to the body and obscured in a majority of the cameras and are undetected, or segmented limbs are rejected due to the length of their curve skeleton being too short. The voxel reconstruction algorithm also creates a "ghost limb" as an artifact of the space carving algorithm in certain configurations of the subject with respect to the cameras. It should be noted that both these problems can be alleviated by using more cameras. The problematic frames are rejected automatically. We report results on the Walking (Frames 1-350) and Balancing (800-1222) subsets. A total of 68 frames (around 9% of the total) were segmented and registered.



(a) Voxel from Silhouettes

(b) Subject A

(c) Subject B

(d) Subject D

(e) Subject C

Figure 5.12: Human body model estimation for different subjects from video sequences.

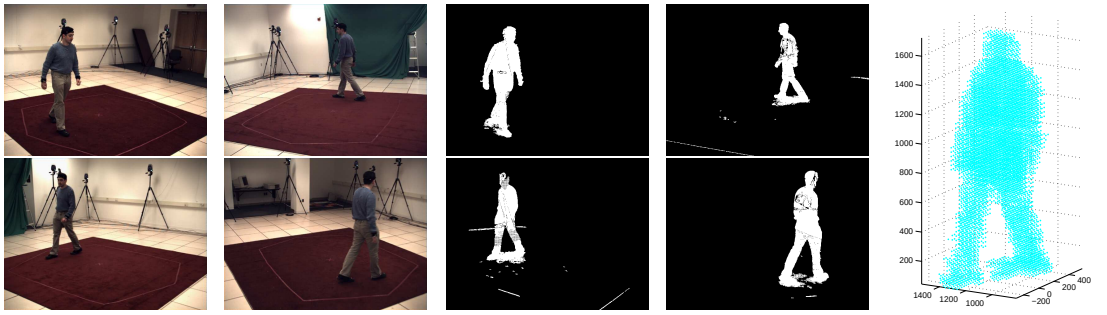


Figure 5.13: Sample frame from the HumanEvaII data-set, with original images, and corresponding foreground images and computed voxel data.

The two subsets from the sequence we use are *Walking* (subset 1) and *Balancing* (subset 2). The error numbers reported are in MSE per joint location. We note that the error is for the *full* body pose, which consists of 20 joints on the human body. Figure 5.15 provides the plots of the error for the different frames that are temporally separate in space. The *absolute* error and the *relative* error are plotted and are similar with the absolute error being slightly less in both the cases. The means of the errors are plotted in dashed lines. Table 5.1 contains a summary of the pose estimation error. The number of frames for which the pose has been computed is also listed.

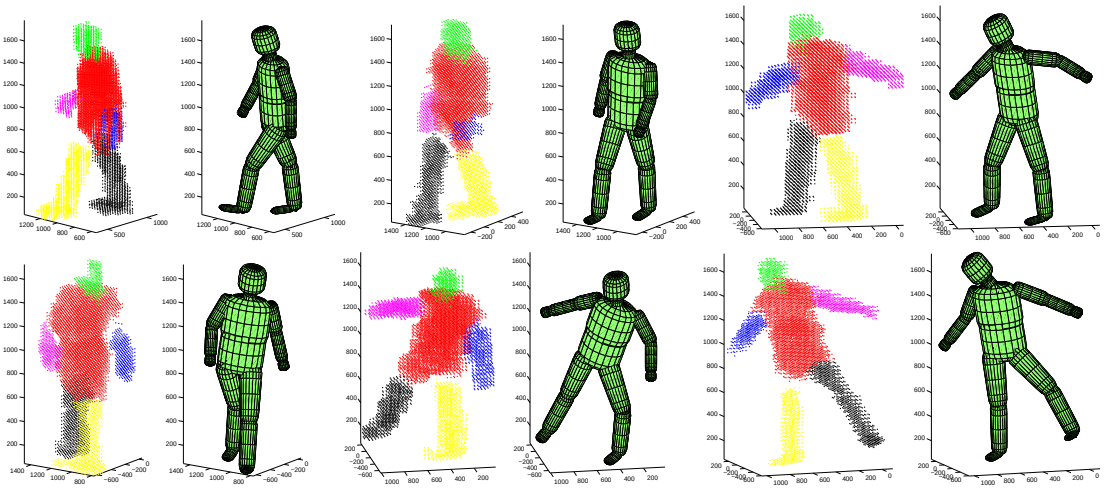


Figure 5.14: Pose estimation results from HumanEvaII sequence.

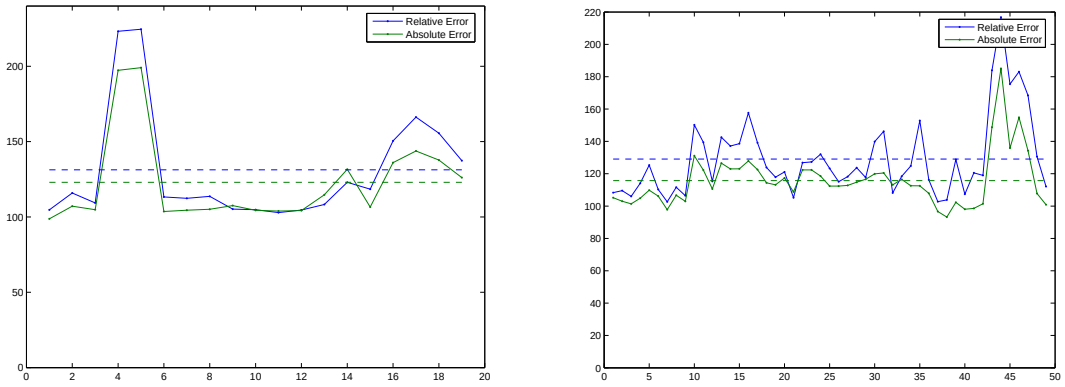


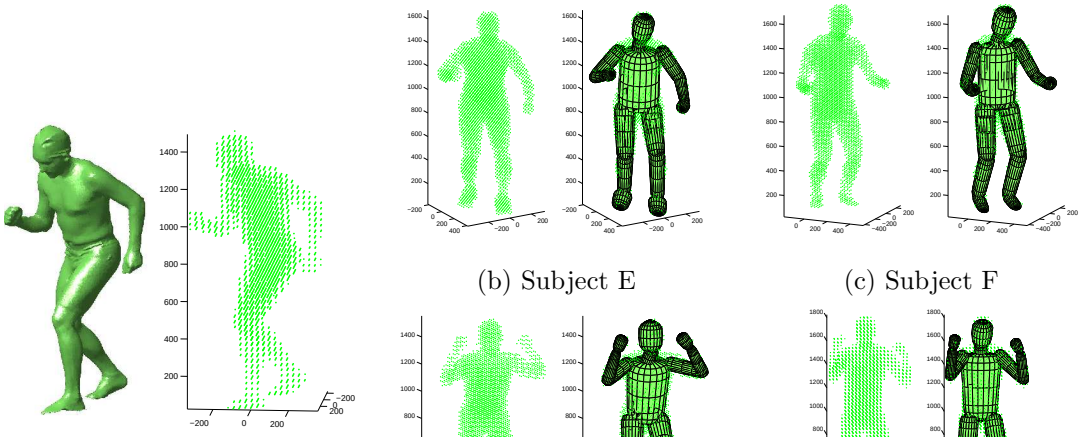
Figure 5.15: Plot of the error per frame for the two sequences from the HumanEvaII data set. The error is in mm.

5.3.3 3D scan data

We also tested our human body model estimation algorithm on different subjects using laser scan data which provide 3D meshes. Voxels are computed from 3D meshes by determining if nodes on a regular 3D grid lie inside the mesh structure. The subject in each case strikes different poses that exercise different joint angles. The subjects are of different heights and build. A set of five different poses was used to estimate the human body pose. Each pose is quite different from the other and the 3D scans are relatively accurate and we were thus able to estimate

Subset	Number of Frames	Relative Error		Absolute Error	
		Mean	Median	Mean	Median
1	19	131.3	113.7	123.0	107.1
3	49	129.1	123.3	115.7	112.8

Table 5.1: Pose error per frame: The pose error is the MSE of all joints per frame and is in mm.



(a) Voxel from scan

(b) Subject E

(c) Subject F

(d) Subject A

(e) Subject G

Figure 5.16: Human body model estimation from 3D scan obtained for different subjects.

the human body model parameters from fewer frames. The results of the human body model for different subjects are presented in Figure 5.16. This experiment illustrates that human body model estimation can be performed using a limited number of frames, provided the poses are varied.

5.3.4 Synthetic data

We provide results on human body model estimation using a synthetic sequence that has been generated from a known model and known motion sequence described by the standard BVH format. A sample 3D frame and the corresponding

voxel data is presented in Figure 5.17 along with the estimated model in different poses. The human body parameters as well as the pose parameters are known and we can compare the estimated human body model and the motion parameters with the ground truth values. We note that the known body parameters are only the joint locations and not the shape parameters; the 3D animation is a smooth fairly realistic mesh as can be seen in the above figure and is very similar to real data. The sequence had 120 frames, and the six different chains were correctly segmented and registered in 118 of the 120 frames. We used 10 equally spaced frames as the key frames in our human body model estimation algorithm. The human body model used in this estimation used two rigid segments for the trunk. The human body model used in the other experiments use one rigid segment for the trunk. The pose was computed for all the 118 frames using the estimated model. The errors in the joint angles at the important joints are compared in Table 5.2. The error is in degrees and computed as

$$\text{Joint angle error} = \cos^{-1}\left(\boldsymbol{n}_G'\boldsymbol{n}_E\right), \tag{5.20}$$

where $\boldsymbol{n}_G$ and $\boldsymbol{n}_E$ are the actual and estimated unit vectors describing the direction of the segment at the joint.

Optim.	Statistic	Trunk	L Should.	L Elbow	R Should.	R Elbow	L Hip	L Knee	R Hip	R Knee
Skeleton	Mean	1.24	8.80	4.20	8.61	5.21	4.09	4.04	3.97	4.82
	Median	1.20	8.51	3.89	8.66	4.98	3.33	3.71	2.68	3.33
Super-quadric	Mean	1.25	7.78	4.25	10.04	5.09	4.18	4.66	3.70	4.96
	Median	1.20	8.13	4.14	9.67	4.97	3.41	4.68	3.22	4.35

Table 5.2: Joint angle error for skeleton and super-quadric optimization. The joint angle errors are in degrees.

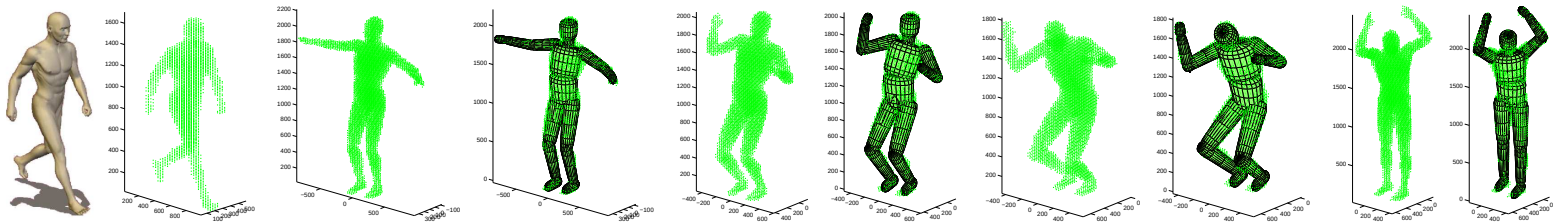


Figure 5.17: Human body model estimation for synthetic sequence: While the sequence itself is synthetic, the voxel data is of the same resolution and is not very different from real data. Four images from the sequence are presented with the super-quadric model super-imposed with the voxel data.

Chapter 6

Pose tracking using multiple cues

Pose initialization and pose tracking are different problems and are typically tackled using different techniques, although initialization techniques may be adapted to perform tracking. In a sense, pose tracking is easier as only an incremental change in pose needs to be estimated. A number of algorithms assume that the pose is initialized and attempt to solve the tracking problem. On the other hand, initialization methods, that estimate pose using a single frame, typically are not able to estimate the pose of the subject in all the frames in a sequence. We, therefore, need to combine elements of both pose initialization and pose tracking to estimate the pose in an entire sequence. Having performed model and pose estimation, or initialization, we describe a tracking algorithm that builds on our initialization work in this chapter.

Much of the work in the past has focussed on using either motion or shape cues in order to track the pose. We present a pose estimation and tracking algorithm that combines both 3D and 2D shape cues as well as 2D image motion cues to track pose in a sequence using an articulated model of the subject. The human body model has been described in detail in Section 3.2.1. We represent the pose, $\mathbf{\Phi}$, in a parametric form as described in Section 3.2.3. In the registration method described in the previous chapter, we have success with frames where all six articulated chains have been segmented and the probability of registration is high. However, in typical sequences, the spatial registration routine fails when body segments are too close to each other or when errors in the 2-D silhouette estimation cause holes and gaps in the voxel reconstruction. We extend the registration method to deal with those frames that have only been partially segmented. We

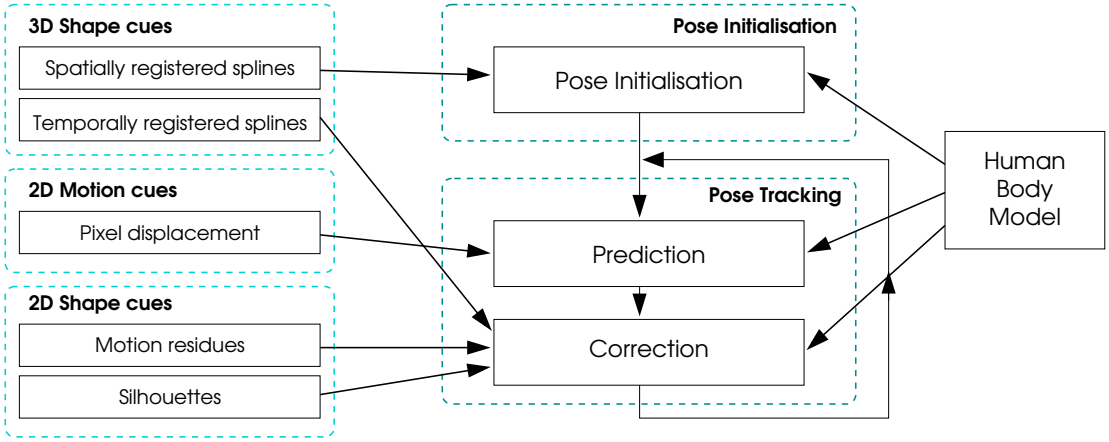


Figure 6.1: The outline of the tracking algorithm illustrating the role of different cues in the different steps: initialization, prediction, and correction.

use the temporal relationships between articulated chains to register them to body segments in as many of the remainder of the frames as possible. These segmented articulated chains serve as shape cues in the tracking algorithm as well. The block diagram of the tracking algorithm is presented in Figure 6.1. We initialize our tracking algorithm with a frame that has been completely segmented and registered, and for which we can estimate the pose. We propose a two-part tracking algorithm that uses both motion and shape cues and consists of a predictor and corrector. The tracker performs the following steps at each time instant.

- Compute 2D pixel displacement between frames at times t and $t+1$. Use the pixel displacement of all body segments in all images to compute change in 3D pose. The predicted pose is computed using the estimated change in pose.

- For all articulated chains whose skeleton curves are present, perform pose correction using skeleton curves (3D shape cues).

- For articulated chains whose skeleton curves are not available, perform pose correction using 2D shape cues such as silhouettes and motion residues.

The performance of a tracking system typically improves with the number of

independent observations, and to that end our system uses different kinds of cues that can be estimated from the images. We use both motion cues, in the form of pixel displacements, as well as shape cues, such as skeleton curves, silhouettes, and "motion residues", hereafter collectively referred to as shape cues. The motion and shape cues complement each other and work together to alleviate the drift and local minima problem that are manifest when they are applied independently. Since we use motion and shape cues in our tracking algorithm, we are able to better deal with cases where the body segments are close to each other, such as when the arms are close to the side of the body. Purely silhouette-based and voxel-based methods typically experience difficulties in such cases. Indeed, we use a voxel-based algorithm to initialize the pose and commence the tracking, but the registration algorithm used in the initialization is successful in only a limited number of frames as mentioned earlier. Silhouette or edge-based methods also have the weakness that they will not be able to deal with rotation around the axis of the body segment. We also propose a smoothing algorithm that smooths the trunk pose, a 6D vector. This is an optional step in the algorithm and improves the performance of the tracker. Since the trunk forms the root of the kinematic chain in our model, smoothing only the trunk pose leads to a smoother estimate of the pose of the entire body. The smoothed trunk estimate is used as an input to the tracking algorithm; it is not a post processing step.

We show that it is possible to estimate 3D pose change from pixel displacement under a projective transformation in Section 6.1. We describe the temporal registration of skeleton curves in Section 6.2. The details of the tracking algorithm including the initialization, prediction, correction, and smoothing routines, are presented in Section 6.3. Finally, we present results of the tracking algorithm on different sequences in Section 6.4. In our experiments, we use sequences that have been captured using eight cameras that are placed around the subject and cover a variety of motions such as swinging arms in wide arcs, walking in a straight line and walking in a circles. The tracking algorithm successfully tracks the pose

through the entire sequence, some of which extend for more than 10 seconds.

6.1 Pose estimation from pixel displacement

We derive a comprehensive formulation to estimate whole body 3D pose from pixel displacement of different body segments in images obtained from multiple cameras. Existing algorithms perform monocular tracking, or use the orthographic projection model for the cameras. We formulate the instantaneous pixel velocity as a linear function of *pose velocity* under a perspective projection model for the cameras. Thereafter, we measure pixel displacement and iteratively estimate the change in 3D pose using our formulation. We can easily incorporate measurements from multiple cameras in our formulation and thus overcome the problem of kinematic singularity, leading to a robust estimate.

We consider a point on the i^{th} segment, and its projection onto the camera image. We show that the 3D point velocity and the corresponding 2D pixel velocity of this point are linear functions of the *pose velocity* in Section 6.1.1 and Section 6.1.2 respectively. It is easier to measure *pixel displacement* than *pixel velocity* when we are given two frames. This is because the measured pixel velocity is a measurement of the *optical flow*, rather than the actual motion of the pixel. We therefore estimate and use the *pixel displacement* between two frames instead. We then estimate the *change in pose* using the measured pixel *displacement* using an iterative method described in Section 6.1.3.

6.1.1 Point velocity as a function of pose velocity

We first define certain notations and representations that we use in our formulation. We use the notation and some results from Murray *et al.* [46]. We can express the homogeneous transformation matrix, G, as a function of the twist vector $\boldsymbol{\xi} = \left(\begin{smallmatrix} p \\ \omega \end{smallmatrix}\right)$ as

$$G = e^{\hat{\boldsymbol{\xi}}} = I + \hat{\boldsymbol{\xi}} + \frac{1}{2!}\hat{\boldsymbol{\xi}}^2 + \frac{1}{3!}\hat{\boldsymbol{\xi}}^3 + \cdots, \tag{6.1}$$

where

$$\hat{\boldsymbol{\xi}} = \begin{pmatrix} 0 & -\omega_3 & \omega_2 & p_1 \\ \omega_3 & 0 & -\omega_1 & p_2 \\ -\omega_2 & \omega_1 & 0 & p_3 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{6.2}$$

We define the $\vee$ (vee) operator to extract the 6-dimensional vector which parameterizes a twist, and is the inverse operator of $\wedge$ (hat), so that $\hat{\boldsymbol{\xi}}^{\vee} = \boldsymbol{\xi}$.

Let us consider a point $\boldsymbol{q}$, given by $\boldsymbol{q}^{(i)}$ in the coordinate frame of segment i and $\boldsymbol{q}^{(j)}$ in the coordinate frame of segment j. We then have

$$\boldsymbol{q}^{(i)} = G_{ij}\boldsymbol{q}^{(j)}, \tag{6.3}$$

where G_{ij} has been defined in (3.6) in Section 3.2.3. We consider the motion of frame j with respect to frame i. Since we are only concerned with the instantaneous motion, we can assume that the motion is described by a constant twist, $\boldsymbol{\xi}$, so that

$$g_{ij}(t) = g_{ij}(0)e^{\hat{\boldsymbol{\xi}}t} = G_{ij}e^{\hat{\boldsymbol{\xi}}t}. \tag{6.4}$$

The twist vector is what we refer to as *pose velocity*, and we can see that if we estimate the pose velocity, we can estimate the change in 3D pose, which is the multiplicative factor $e^{\hat{\boldsymbol{\xi}}t}$ in (6.4). The t parameter is a scalar and denotes time. We use $g_{ij}(t)$ to denote the transformation as a function of time. G_{ij} is a constant and is given by $g_{ij}(0)$. We consider motion at $t = 0$ without loss of generality. Considering the velocity of the point $\boldsymbol{q}$ in frame i, we have

$$\dot{\boldsymbol{q}}^{(i)} = \dot{g}_{ij}\boldsymbol{q}^{(j)} + g_{ij}\dot{\boldsymbol{q}}^{(j)} \tag{6.5}$$

$$= \dot{g}_{ij}\boldsymbol{q}^{(j)}, \tag{6.6}$$

where the second equation follows because the point is fixed in frame j and therefore $\dot{\boldsymbol{q}}^{(j)} = 0$. Substituting (6.4) in (6.6), we get

$$\dot{\boldsymbol{q}}^{(i)}(t) = G_{ij}\hat{\boldsymbol{\xi}}e^{\hat{\boldsymbol{\xi}}t}\boldsymbol{q}^{(j)}. \tag{6.7}$$

We thus have

$$\dot{\boldsymbol{q}}^{(i)}(0) = G_{ij}\hat{\boldsymbol{\xi}}e^{\hat{\boldsymbol{\xi}}0}\boldsymbol{q}^{(j)} = G_{ij}\hat{\boldsymbol{\xi}}\boldsymbol{q}^{(j)} = G_{ij}\Upsilon(\boldsymbol{q}^{(j)})\boldsymbol{\xi}, \tag{6.8}$$

where

$$\Upsilon(\boldsymbol{q}) \triangleq \begin{pmatrix} 1 & 0 & 0 & 0 & -q_3 & q_2 \\ 0 & 1 & 0 & q_3 & 0 & -q_1 \\ 0 & 0 & 1 & -q_2 & q_1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}. \tag{6.9}$$

Assuming there are a total of m segments, and given a point, $\boldsymbol{q}$, on the i^{th} segment, we have

$$\boldsymbol{q}^{(0)} = g_{0i}\boldsymbol{q}^{(i)} = g_{01}g_{12}\cdots g_{(i-1)i}\boldsymbol{q}^{(i)}. \tag{6.10}$$

It follows that

$$\dot{\boldsymbol{q}}^{(0)} = \left(\dot{g}_{01}g_{12}\cdots g_{(i-1)i} + g_{01}\dot{g}_{12}\cdots g_{(i-1)i} + \cdots + g_{01}g_{12}\cdots \dot{g}_{(i-1)i}\right)\boldsymbol{q}^{(i)} \tag{6.11}$$

$$= \Big(g_{01}\hat{\boldsymbol{\xi}}^{(1)}g_{12}\cdots g_{(i-1)i} + g_{01}g_{12}\hat{\boldsymbol{\xi}}^{(2)}\cdots g_{(i-1)i} + \cdots + g_{01}g_{12}\cdots g_{(i-1)i}\hat{\boldsymbol{\xi}}^{(i)}\Big)\boldsymbol{q}^{(i)} \tag{6.12}$$

$$= g_{01}\hat{\boldsymbol{\xi}}^{(1)}\boldsymbol{q}^{(1)} + g_{02}\hat{\boldsymbol{\xi}}^{(2)}\boldsymbol{q}^{(2)} + \cdots + g_{0i}\hat{\boldsymbol{\xi}}^{(i)}\boldsymbol{q}^{(i)} \tag{6.13}$$

$$= g_{01}\Upsilon(\boldsymbol{q}^{(1)})\boldsymbol{\xi}^{(1)} + g_{02}\Upsilon(\boldsymbol{q}^{(2)})\boldsymbol{\xi}^{(2)} + \cdots + g_{0i}\Upsilon(\boldsymbol{q}^{(i)})\boldsymbol{\xi}^{(i)} \tag{6.14}$$

$$= \begin{pmatrix} g_{01}\Upsilon(\boldsymbol{q}^{(1)}) & g_{02}\Upsilon(\boldsymbol{q}^{(2)}) & \cdots & \overbrace{g_{0i}\Upsilon(\boldsymbol{q}^{(i)})}^{\text{col. } 6(i-1)+1 \text{ to } 6i} & \underbrace{0_{4\times 6(m-i)}}_{\text{col. } 6i+1 \text{ to } 6m} \end{pmatrix} \begin{pmatrix} \boldsymbol{\xi}^{(1)} \\ \boldsymbol{\xi}^{(2)} \\ \vdots \\ \boldsymbol{\xi}^{(m)} \end{pmatrix} \tag{6.15}$$

$$= F(\boldsymbol{\Phi}, \boldsymbol{q})\boldsymbol{\Xi}, \tag{6.16}$$

where $\boldsymbol{\Xi} = \begin{pmatrix} \boldsymbol{\xi}^{(1)} \\ \vdots \\ \boldsymbol{\xi}^{(m)} \end{pmatrix}$. Let the pose at time $t = 0$ be $\boldsymbol{\Phi}(0) = \begin{pmatrix} \boldsymbol{\varphi}^{(1)}(0) \\ \vdots \\ \boldsymbol{\varphi}^{(m)}(0) \end{pmatrix}$ and the pose at time $t = 1$ be $\boldsymbol{\Phi}(1) = \begin{pmatrix} \boldsymbol{\varphi}^{(1)}(1) \\ \vdots \\ \boldsymbol{\varphi}^{(m)}(1) \end{pmatrix}$. We note that the pose at time $t + 1$ is given by

$$g(t) = e^{\hat{\boldsymbol{\varphi}}(t+1)} = g(0)e^{\hat{\boldsymbol{\xi}}t} = e^{\hat{\boldsymbol{\varphi}}(0)}e^{\hat{\boldsymbol{\xi}}t}. \tag{6.17}$$

We therefore have

$$\boldsymbol{\varphi}(t) = \boldsymbol{\varphi}(0) \oplus \boldsymbol{\xi}, \tag{6.18}$$

where the $\oplus$ operation is defined as

$$\boldsymbol{\varphi} \oplus \boldsymbol{\xi} \triangleq \left(\ln e^{\hat{\boldsymbol{\varphi}}} e^{\hat{\boldsymbol{\xi}}}\right)^{\vee}. \tag{6.19}$$

The pose at $t = 1$ for each segment i in the body is then given by

$$\boldsymbol{\varphi}^{(i)}(1) = \boldsymbol{\varphi}^{(i)}(0) \oplus \boldsymbol{\xi}, \qquad \text{for } i = 1, 2, \cdots, m. \tag{6.20}$$

We can represent the set of operations in (6.20) using the abbreviated version

$$\boldsymbol{\Phi}_{t+1} = \boldsymbol{\Phi}_t \oplus \boldsymbol{\Xi}, \tag{6.21}$$

where the upper case Greek letters $\boldsymbol{\Phi}$ and $\boldsymbol{\Xi}$ refer to the vector stack of the poses of the individual segments represented by lower case Greek letters $\boldsymbol{\varphi}$ and $\boldsymbol{\xi}$.

6.1.2 Pixel velocity as a function of pose velocity

We show in [70] that if we use a perspective projection to project the 3D point on to the camera, the resulting pixel velocity is still a linear function of the pose velocity. Let $P_{3\times 4} = \begin{pmatrix} \boldsymbol{P}_1' \\ \boldsymbol{P}_2' \\ \boldsymbol{P}_3' \end{pmatrix}$ be the projection matrix, then the pixel value is given in homogeneous coordinates by $\boldsymbol{q}^{(c)} = P\boldsymbol{q}^{(0)}$. The superscript, (c), denotes

the camera coordinate system. Let $\boldsymbol{u}$ be the pixel coordinates in inhomogeneous coordinate system. We then have

$$\boldsymbol{u} = \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = \frac{1}{q_3^{(c)}} \begin{pmatrix} q_1^{(c)} \\ q_2^{(c)} \end{pmatrix} = \frac{1}{\boldsymbol{P'}_3 \boldsymbol{q}^{(0)}} \begin{pmatrix} \boldsymbol{P'}_1 \\ \boldsymbol{P'}_2 \end{pmatrix} \boldsymbol{q}^{(0)}. \tag{6.22}$$

We then have the pixel velocity given by

$$\dot{\boldsymbol{u}} = \frac{1}{\boldsymbol{P'}_3 \boldsymbol{q}^{(0)}} \begin{pmatrix} \boldsymbol{P'}_1 \\ \boldsymbol{P'}_2 \end{pmatrix} \dot{\boldsymbol{q}}^{(0)} - \frac{\dot{\boldsymbol{q}}^{(0)}}{\left(\boldsymbol{P'}_3 \boldsymbol{q}^{(0)}\right)^2} \begin{pmatrix} \boldsymbol{P'}_1 \\ \boldsymbol{P'}_2 \end{pmatrix} \boldsymbol{q}^{(0)} \tag{6.23}$$

$$= \frac{1}{\boldsymbol{P'}_3 \boldsymbol{q}^{(0)}} \left(\begin{pmatrix} \boldsymbol{P'}_1 \\ \boldsymbol{P'}_2 \end{pmatrix} - \frac{1}{\boldsymbol{P'}_3 \boldsymbol{q}^{(0)}} \begin{pmatrix} (\boldsymbol{P'}_1 \boldsymbol{q}^{(0)}) \boldsymbol{P'}_1 \\ (\boldsymbol{P'}_2 \boldsymbol{q}^{(0)}) \boldsymbol{P'}_2 \end{pmatrix} \right) \dot{\boldsymbol{q}}^{(0)} \tag{6.24}$$

$$= E(P, \boldsymbol{q}) \dot{\boldsymbol{q}}^{(0)} \tag{6.25}$$

$$= E(P, \boldsymbol{q}) F(\boldsymbol{\Phi}, \boldsymbol{q}) \boldsymbol{\Xi}, \tag{6.26}$$

where $E(P, \boldsymbol{q})$ represents the matrix in (6.25) and (6.26) is obtained by combining (6.16) and (6.25) to express the 2D pixel velocity as a linear function of the 3D pose velocity, $\boldsymbol{\Xi}$.

6.1.3 Estimating pose change from pixel displacement

We can estimate $\boldsymbol{\Xi}$ from pixel velocity using the inverse of (6.26) and thereafter compute the new pose, $\boldsymbol{\Phi}_{t+1}$, from $\boldsymbol{\Xi}$ and $\boldsymbol{\Phi}_t$ using (6.21). However, we measure pixel displacement from the images, and hence we use a first order approximation of the pixel velocity in order to compute the change in pose. Given a set of points, we can compute the projection of each of these points for all the cameras as a function of the pose $\boldsymbol{\Phi}$. We refer to this stacked vector as $C(\boldsymbol{\Phi})$. We can also compute the matrix $D(\boldsymbol{\Phi}) = E(P, \boldsymbol{q}) F(\boldsymbol{\Phi}, \boldsymbol{q})$ and describe it as a function of $\boldsymbol{\Phi}$. D and C are functions of both the point coordinates and the projection matrices besides $\boldsymbol{\Phi}$, but as these are fixed for a given frame, we do not explicitly denote them for the sake of simplicity. We therefore have the pixel location and velocity

given as

$$\boldsymbol{u} = C(\boldsymbol{\Phi}) \tag{6.27}$$

and

$$\dot{\boldsymbol{u}} = D(\boldsymbol{\Phi})\boldsymbol{\Xi}. \tag{6.28}$$

The state vector in our state-space formulation is $\boldsymbol{\Phi}_t$ and the state update and observation equations are given by (6.29)-(6.30).

$$\text{State update} : \boldsymbol{\Phi}_{t+1} = \boldsymbol{\Phi}_t \oplus \boldsymbol{\Xi}_t \tag{6.29}$$

$$\text{Observation} : \Delta\boldsymbol{u} = \boldsymbol{u}_{t+1} - \boldsymbol{u}_t \approx D(\boldsymbol{\Phi})\boldsymbol{\Xi} \tag{6.30}$$

(6.30) follows from the first order Taylor series approximation

$$\boldsymbol{u}_{t+1} = \boldsymbol{u}_t + \dot{\boldsymbol{u}}_t + \frac{1}{2}\ddot{\boldsymbol{u}}_t + \cdots \approx \boldsymbol{u}_t + \dot{\boldsymbol{u}}_t. \tag{6.31}$$

We can then use an iterative algorithm to estimate the pose from the given pixel displacement using the steps described in Table 6.1. We have several pixel displacement measurements from multiple cameras and the estimation equation is highly over-constrained and we can perform a least squares estimate in step 5.

6.2 Temporal registration of skeleton curves

Given a sequence of frames, we can typically segment and register the voxel data for a subset of the frames using the *single frame* registration algorithm presented in Section 5.1. The pose can then be initialized for the frames belonging to this subset using the algorithm presented in Section 5.2. An example of a successfully segmented and registered frame is presented in Figure 6.2. The single frame registration method does not succeed in all frames possibly due to errors in the voxel reconstruction or segmentation or a complex pose. Two examples where registration of skeleton curves to articulated chains in a stand-alone frame fails

1. Let $\mathbf{\Phi}_t$ be the estimated pose at time t.
2. Let $\mathbf{\Phi}_{t+1}^0 = \mathbf{\Phi}_t$ and $k = 0$.
3. Let $k = k + 1$.
4. Let $\Delta \boldsymbol{u}^{(k)} = \Delta \boldsymbol{u} - (C(\mathbf{\Phi}^{(k)}) - C(\mathbf{\Phi}_t))$
5. Compute $\mathbf{\Xi}^{(k)} = \left(D(\mathbf{\Phi}^{(k)})' D(\mathbf{\Phi}^{(k)}) \right) D(\mathbf{\Phi}^{(k)})' \Delta \boldsymbol{u}^{(k)}$.
6. Update $\mathbf{\Phi}_{t+1}^{(k+1)} = \mathbf{\Phi}_{t+1}^{(k)} \oplus \mathbf{\Xi}^{(k)}$.
7. If converged, or k exceeds *number_iterations*, go to 3.
8. Set $\mathbf{\Phi}_{t+1} = \mathbf{\Phi}_{t+1}^{(k+1)}$.

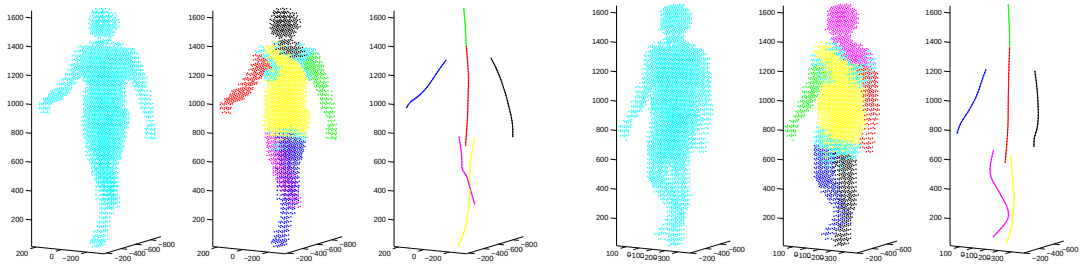
Table 6.1: Algorithm for estimating 3D pose using pixel displacement

are illustrated in 6.3. In one of the examples, the head is missing, due to errors in background subtraction, and in the other *seven*, instead of *six*, spline segments are discovered. It is very useful, from a tracking point of view, to register at least some of the skeleton curves in a given frame. We describe a temporal registration method which can be used in conjunction with the *spatial* registration method presented earlier, to register skeleton curves using their temporal relation.

Let $\mathcal{S}^A = \{\boldsymbol{x}_1^A, \boldsymbol{x}_2^A, \cdots, \boldsymbol{x}_{n_A}^A\}$ and $\mathcal{S}^B = \{\boldsymbol{x}_1^B, \boldsymbol{x}_2^B, \cdots, \boldsymbol{x}_{n_B}^B\}$ be the set of points belonging to skeleton curves $\mathcal{S}^A$ and $\mathcal{S}^B$ respectively. The distance between skeleton curves $\mathcal{S}^A$ and $\mathcal{S}^B$ is given by

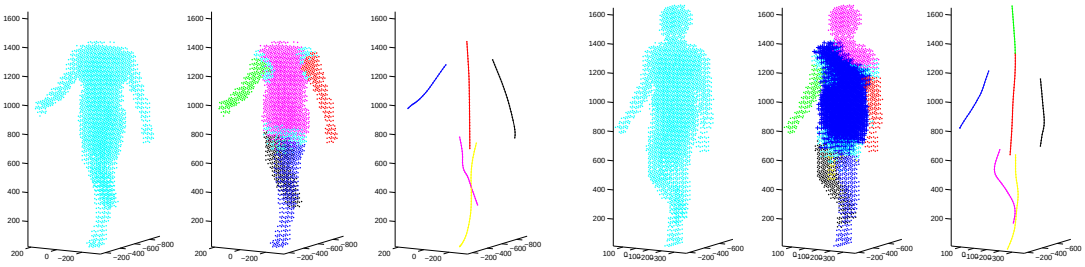
$$d(\mathcal{S}^A, \mathcal{S}^B) = \frac{1}{n_A + n_B} \left(\sum_{i=1}^{n_A} \min_j (\|\boldsymbol{x}_i^A - \boldsymbol{x}_j^B\|) + \sum_{i=1}^{n_B} \min_j (\|\boldsymbol{x}_i^B - \boldsymbol{x}_j^A\|) \right). \quad (6.32)$$

Let us assume that frames at time t_0 and t_1 are registered using the spatial registration method referred to in the previous section. We need to register skeleton curves for the frames between t_0 and t_1. Let $\mathcal{R}_t^i$ represent the reference (or registered) skeleton curve for the i^{th} articulated chain at time instant t. The



(a) (b)

Figure 6.2: Example of registered frames: The various stages from segmentation and registration to pose initialization.



(a) (b)

Figure 6.3: Examples of unregistered frames: The first example has a missing segment (head) and the second example has an extra segment which is an artifact of the voxel reconstruction.

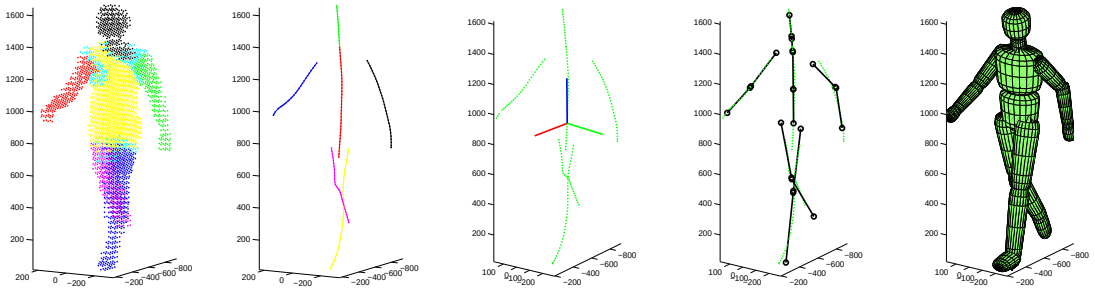
1. Set $t = t_0$. Set $\mathcal{R}_t^i = \mathcal{S}_t^i$ for $i = 1, \cdots, 6$. Note that this frame has been already registered.
2. Set $t = t + 1$.
3. Let the skeleton curves in the current frame be $\mathcal{S}_t^1, \cdots, \mathcal{S}_t^{N_t}$. Note that the N_t may not be equal to six.
4. For each $\mathcal{R}_{t-1}^i$, find the closest curve, $\mathcal{S}_t^{r_i}$, if it exists, such that $d(\mathcal{S}_t^{r_i}, \mathcal{R}_{t-1}^i) < d_{\text{THRESHOLD}}$ and the mapping is unique.
5. If $\mathcal{R}_{t-1}^i$ has a registered candidate, then set $\mathcal{R}_t^i = \mathcal{S}_t^{r_i}$, else set $\mathcal{R}_t^i = \mathcal{R}_{t-1}^i$.
6. If $t = t_1 - 1$, stop, else go to Step 2.

Table 6.2: Temporal registration of skeleton curves

forward temporal registration algorithm is presented in Table 6.2. We typically set $d_{\text{THRESHOLD}} = 50$mm. The same algorithm can be used to perform reverse temporal registration as well, *i.e.*, we start at $t = t_1$ and proceed backwards in time. Any skeleton curve that is not registered to the same articulated chain in the forward and backward temporal registration process is said to be unregistered.

6.3 Tracking algorithm

We describe the complete tracking algorithm which consists of the predictor-corrector framework, beginning with initialization. We assume that the human body model is available for the subject in the sequence. We track the pose starting at the first frame in which the pose can be initialized. The pose initialization procedure in the context of tracking is briefly explained in Section 6.3.1. Typically, the pose can be initialized in several frames in a sequence and the majority of the skeleton curves in the remainder of the frames can be registered using the



(a) Segmentation (b) Registration (c) Initialize pose (d) Optimize pose (e) Final pose

Figure 6.4: Pose initialization for tracker using computed model.

temporal registration algorithm described in Section 6.2. Our tracking algorithm consists of two steps, a prediction step and a correction step. Given the pose at t, we predict the pose at time $t+1$ using motion cues as described in Section 6.3.2. We correct the predicted pose using the available 2D and 3D shape cues as described in Section 6.3.3. Finally, we describe an optional smoothing routine in Section 6.3.4.

6.3.1 Pose initialization for tracker

To briefly summarize our initialization algorithm, we begin with a bottom-up segmentation (Figure 6.4 (a)) and perform registration of skeleton curves (Figure 6.4 (b)). We then initialize the trunk pose using previously estimated human body model parameters for the subject (Figure 6.4 (c)) and optimize the pose parameters using skeleton curves and the skeleton model (Figure 6.4 (d)) and finally the voxels and super-quadric model (Figure 6.4 (e)). We note that the pose initialization is performed by minimizing $\boldsymbol{h}'\boldsymbol{h}$ as described in Section 5.2.2, except that the body model parameters are fixed in this instance. We note that even with partial registration, we can still obtain an *initial estimate* of the pose of the trunk using the skeleton curves of the trunk and either two arms or two limbs.

6.3.2 Pose prediction using motion cues

In order to estimate the motion of the whole body, we first project each body segment onto each image. We call this step pixel-body registration. We thus have a pixel mask for each segment in each of the images. We compute the pixel displacement for each mask segment in each image using the rigid body motion model. We then combine the pixel displacement for a set of bodies in all the images into a single matrix equation using which we estimate the change in 3D pose.

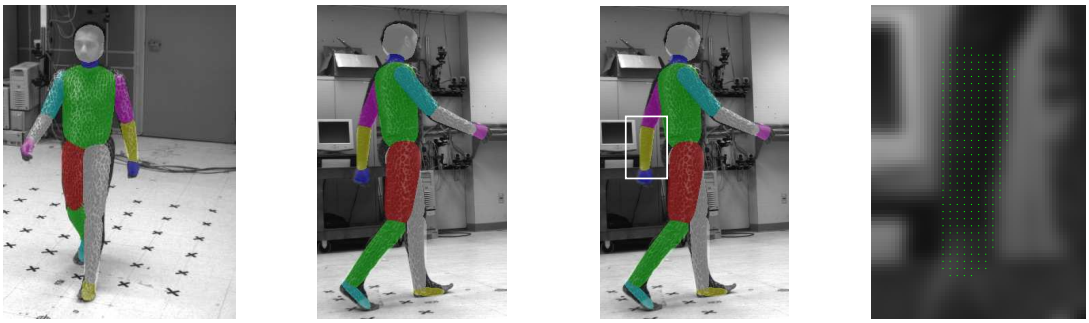
6.3.2.1 Pixel-body registration

We register each pixel in each image to its 3D coordinate and determine the body segment it belongs to. We can thus obtain a 2D mask for each body segment in each image and we can impose a rigid motion model for all pixels belonging to the same segment (mask). In order to determine the correspondence between a pixel and the body segments in a given image, we convert the super-quadric representing each body segment into a triangular mesh and project it onto the image. We can compute the depth at each pixel by interpolating the depths of the triangle vertices.[1] Since we can compute the depth at a pixel for different body segments, we can resolve self-occlusion. Figure 6.5 illustrates the projection of the body segments onto images from two cameras. Different colors denote different body segments. We can also compute 3D coordinates of the points corresponding to the pixels using similar interpolation techniques.

6.3.2.2 Estimating pixel displacement

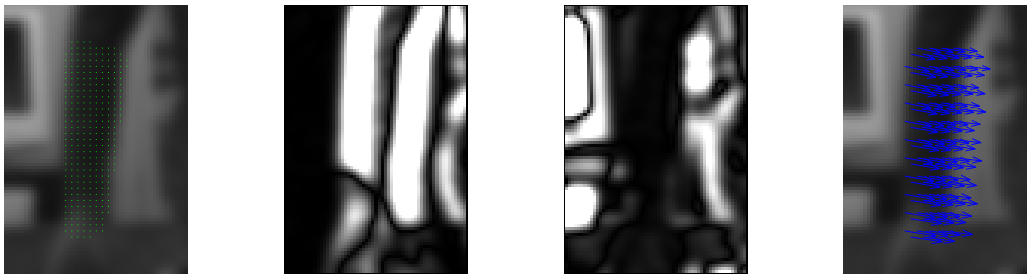
We use a parametric rigid motion model for all the pixels belonging to the same body segment in an image. The displacement, $\delta\boldsymbol{u}$, at a pixel $\boldsymbol{u}$ is a function of

[1] Since we convert the super-quadric to triangular meshes, we can easily extend our algorithm to use mesh-based models instead of super-quadrics.



(a) View 1 (b) View 2 (c) Sample block (d) Mask

Figure 6.5: Pixel registration showing the mask of left elbow.



(a) Mask (b) Image Difference (c) Motion Residue (d) Flow

Figure 6.6: Pixel displacement and motion residue

$\boldsymbol{\psi} = (\boldsymbol{\Delta}, \theta, s)$ where $\boldsymbol{\Delta}$ is the displacement, θ is the rotation and s is the scale parameter for the motion of the mask. The displacement at a pixel, $\boldsymbol{u}$, is therefore given by

$$\Delta\boldsymbol{u} = \boldsymbol{\delta}(\boldsymbol{u}, \boldsymbol{\psi}) = s \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} (\boldsymbol{u} - \boldsymbol{u}_0) + \boldsymbol{\Delta}, \tag{6.33}$$

where $\boldsymbol{u}_0$ denotes the projection of the joint location for the body segment. We prefer the above parametric representation to an affine model as we can set meaningful upper and lower bounds on each parameter. Let $\boldsymbol{u}_1, \cdots, \boldsymbol{u}_n$ be the pixels registered to a given segment and illustrated in Figure 6.6 (a). We compute that value of $\boldsymbol{\psi} \in [\boldsymbol{\psi}_0 - \boldsymbol{\psi}_B, \boldsymbol{\psi}_0 + \boldsymbol{\psi}_B]$ for the segment that minimizes the residue given

by $\boldsymbol{e}^{\mathsf{T}}\boldsymbol{e}$, where $\boldsymbol{\psi}_B$ denotes the bounds on the motion that we impose, and the j^{th} element of $\boldsymbol{e}$ is given as

$$e_j = I_t(\boldsymbol{u}_j) - I_{t+1}(\boldsymbol{u}_j + \boldsymbol{\delta}(\boldsymbol{u}_j, \boldsymbol{\psi})). \tag{6.34}$$

A value of $\boldsymbol{\psi} = \mathbf{0}$ implies no motion. Figure 6.6 illustrates the pixel displacement computation and the concept of "motion residue". Figure 6.6 (a) is the smoothed intensity image at time t. Figure 6.6 (b) is the difference between the intensity images at time t and $t+1$. This is the same as the "motion residue" for $\boldsymbol{\psi} = \mathbf{0}$. We note that if the actual motion of the pixel agrees with the estimated motion, then the motion residue for the pixel is close to zero, otherwise it is generally a non-zero value. We note that the motion $\boldsymbol{\psi} = \mathbf{0}$ agrees with the motion of the background pixels (the region left of the mask) which is stationary. However, $\boldsymbol{\psi} = \mathbf{0}$ does not agree with the motion of the foreground pixels. Figure 6.6 (c) is the difference between the image at time t warped according to the estimated motion and the image at $t + 1$ and is the "motion residue" for the optimal $\boldsymbol{\psi}$. We note that the estimated motion agrees with the actual motion for the pixels in the mask, but does not agree with the motion for the background pixels. The value of the pixels in the region of the mask is close to zero where the estimated pixel displacement agrees with the actual pixel displacement. Thus, the "motion residue" provides us with a rough delineation of the location of the body segment at time $t+1$, even when the original mask does not exactly match the body segment. Figure 6.6 (d) illustrates the computed pixel displacement for pixels in the mask.

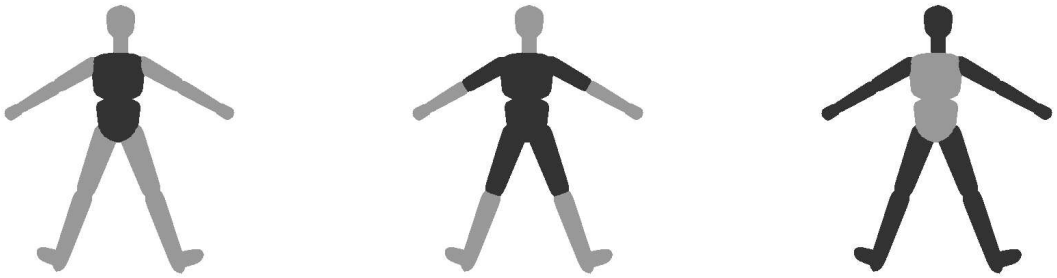
6.3.2.3 Pose prediction

We need to predict the pose $\boldsymbol{\Phi}_{t+1}$ given $\boldsymbol{\Phi}_t$ from the pixel displacement. The basic relationship between the pose velocity, $\boldsymbol{\Xi}$, and the pixel velocity, and how we compute $\boldsymbol{\Phi}_{t+1}$ from the pixel displacement was described in Section 6.1. Since we have registered pixels to 3D points on the model, and the camera calibration parameters, we can compute the matrix $D(\boldsymbol{\Phi})$ in (6.28) between the pixel velocity

and the pose in pose. While our body model and our pose estimation algorithm allows rotation and translation for each joint, we set the translational component to zero at most joints as we find that, in practice, the estimation is more robust when the number of translational parameters are minimum [70]. We, therefore, allow only the base body to translate freely. We allow the shoulder joint to translate under the following constraint, where $\boldsymbol{p}^{\text{SHOULDER}}$ is the translation at the shoulder joint.

$$\|\boldsymbol{p}^{\text{SHOULDER}}\| \leq 20mm \tag{6.35}$$

We estimate the whole body pose of the subject in multiple steps, starting at the root of the kinematic chain as illustrated in Figure 6.7. In the first step, we estimate the pose for the segments belonging to the trunk, in the second we include the first segment in all the limbs, and in the final step we estimate the pose for all the segments save the trunk segments.

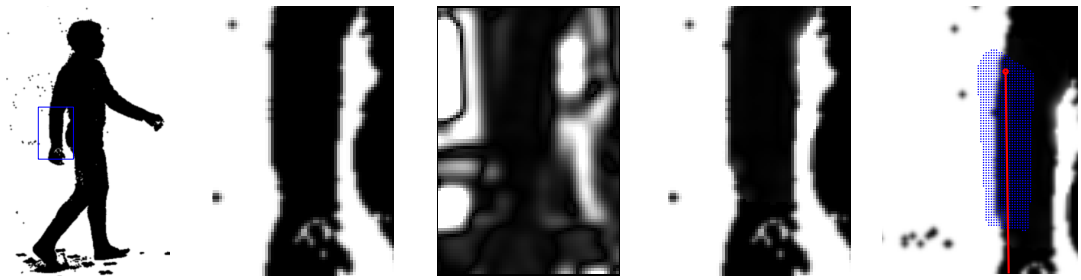


(a) Step 1 (b) Step 2 (c) Step 3

Figure 6.7: We estimate the motion beginning at the base of the kinematic chain, *i.e.*, the trunk and propagate the motion along the chains in further steps. The segments for which the pose is computed at a given step are colored in dark gray, while the remaining segments are in light gray.

6.3.3 Pose correction using shape cues

The pose can be corrected for all the articulated chains in a given frame that have been registered (spatially or temporally) using the 3D shape cues (skeleton curves). The pose of each articulated chain can be corrected by minimizing $\boldsymbol{h}'\boldsymbol{h}$ as described in Section 5.2.2, the only difference being that in this instance we optimize for pose alone, keeping the body model parameters fixed. The pose parameter search space is bounded and centered around the pose predicted using motion cues. In the absence of 3D shape cues for an articulated chain, we can use 2D shape cues in the form of silhouettes and motion residues. This allows us to use the predictor-corrector framework irrespective of which shape cues are available. The "motion residue" for the example in Section 6.3.3 is presented in Figure 6.6 (d). The "motion residue" for a given segment provides us with the region that agrees with the motion of the mask and helps us spatially delineate the segment in the image.



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

Figure 6.8: Obtaining unified error image for the forearm: (a) and (b) denote the silhouette at $t+1$, (c) the motion residue, (d) the combined energy image and (e) the mask.

We combine the "motion residue" and the silhouette as shown in Figure 6.8 to form an error image for that segment. We now have the pixel-wise error image for each camera and a given segment as well as a mask for the body segment for the body segment for a given image as illustrated in Figure 6.8 (e). For a

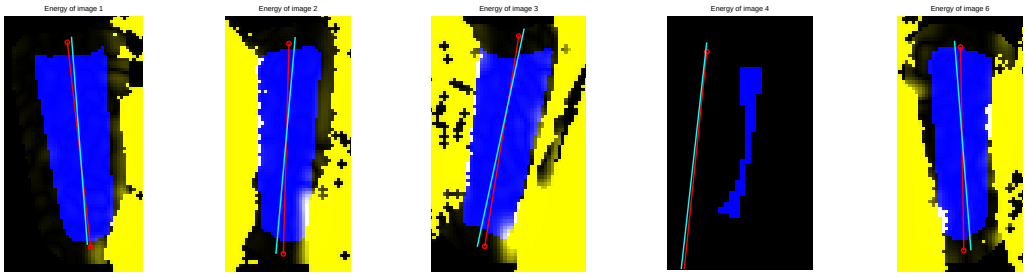


Figure 6.9: Minimum error configuration: It does not matter if the object is occluded or nearly occluded in some of the images.

given 3D pose, $\boldsymbol{\varphi}$, of the segment we can project the axis onto each image. The red line in Figure 6.8 (e) denotes the 3D axis of the segment denoted in blue. For a new value of the pose we get a different axis (*e.g.*, the cyan line). The 2D motion can be represented by a displacement and rotation of this axis.[2] We compute the error of a 2D pose by warping the segment mask according to the mentioned displacement and rotation and summing the value of all the pixels in the error image that belong to the mask. We can then express the error of each 3D pose directly by stacking the error of the pixels belonging to that segment in each image. We minimize this error function in a pose parameter space that is centered around the predicted pose using non-linear optimization functions. We illustrate the results of the pose correction for the above example in Figure 6.9. The red line represents the initial position of the axis of the body segment and the cyan line represents the corrected position.

6.3.4 Pose smoothing

It is often beneficial to perform temporal smoothing on the pose vector as it uses the redundancy in the temporal remain to reduce the error in the pose estimate. It also serves to stabilize the pose estimate. We propose an optional routine

[2]We ignore the change in scale.

that acts on the pose of the root segment of the kinematic chain. It is difficult to smooth the entire pose vector due to the articulated constraints between the segments, and we therefore restrict the smoothing to the pose of the root segment of the chain as it has an impact on the pose of all the body segments. We can obtain an estimate of the pose of the trunk using even partially registered skeleton curves as explained in Section 6.3.1. The trunk pose is smoothed using the cubic smoothing spline with the trunk poses obtained from the available skeleton curves in the sequence. We note that the trunk pose can be interpolated for frames missing the trunk pose using the estimated spline parameters. The translational components of the pose of the trunk for one of the test sequences is presented in Figure 6.10. The translational components are given by $\boldsymbol{p} = \begin{pmatrix} p_1 \\ p_2 \\ p_3 \end{pmatrix}$.

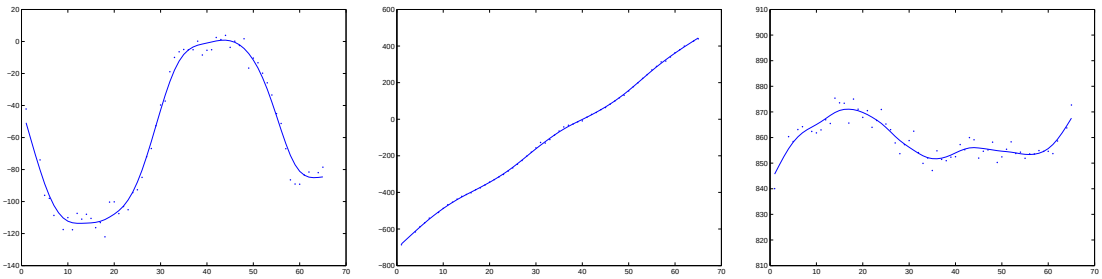


Figure 6.10: The translational components of the pose of the trunk in sequence 1 are presented in the above three images along with the smoothed and interpolated components.

6.4 Experimental results

We performed tracking on sequences where the subject performs different kinds of motion. The experiments were performed using gray-scale images obtained from eight cameras with a spatial resolution of 648×484 at a frame rate of 30 frames per second. The external and internal camera calibration parameters for all the cameras were obtained as described in Section 3.1.

We present results for three sequences that include the subject walking in a

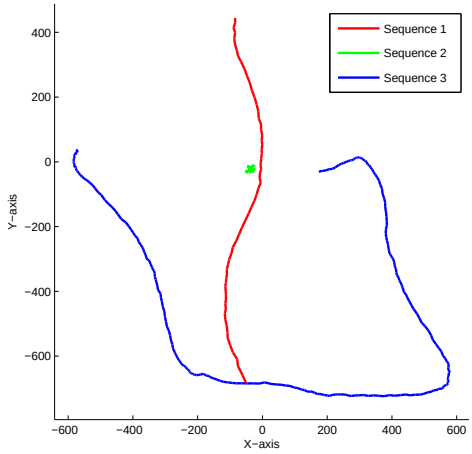
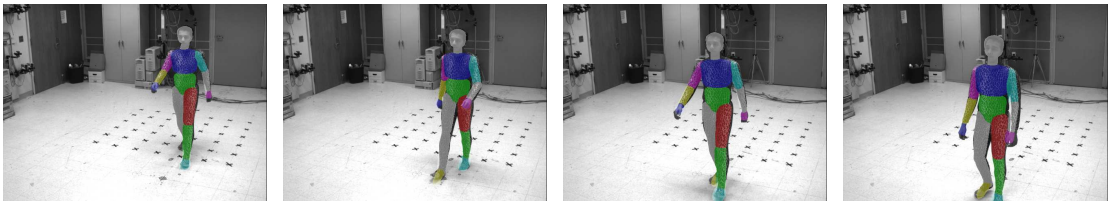


Figure 6.11: The position of the trunk of the subject is plotted in the world reference frame. Sequence 1 has the subject walking in a straight line, Sequence 2 has the subject swinging arms, and Sequence 3 has the subject moving randomly in a roughly circular path with sharp turns.

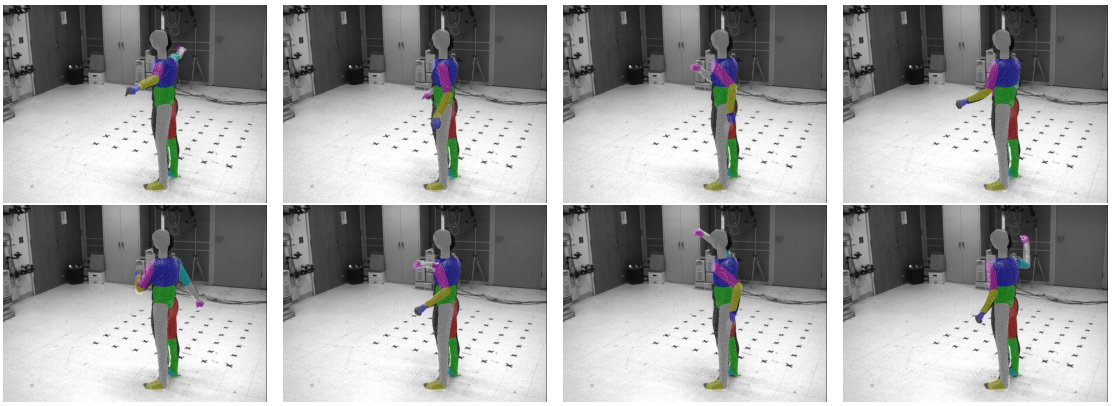
straight line (65 frames, 2 seconds) in Figure 6.12, swinging the arms in a wide arc (300 frames, 10 seconds) in Figure 6.13, and walking in a roughly circular path (300 frames, 10 seconds) in Figure 6.14. Figure 6.11 illustrates the motion of the base body in the world reference frame in the three sequences. Our results show that using only motion cues for tracking causes the pose estimator to drift and lose track eventually, as we are estimating only the *difference in the pose* and the error accumulates. This underlines the need for correcting the predicted pose using shape cues and we observe that the correction step of the algorithm prevents drift in the tracking. We illustrate the results of the tracking algorithm by super-imposing the tracked body model onto the image for two of the eight cameras. The body parts are successfully tracked in the three sequences.



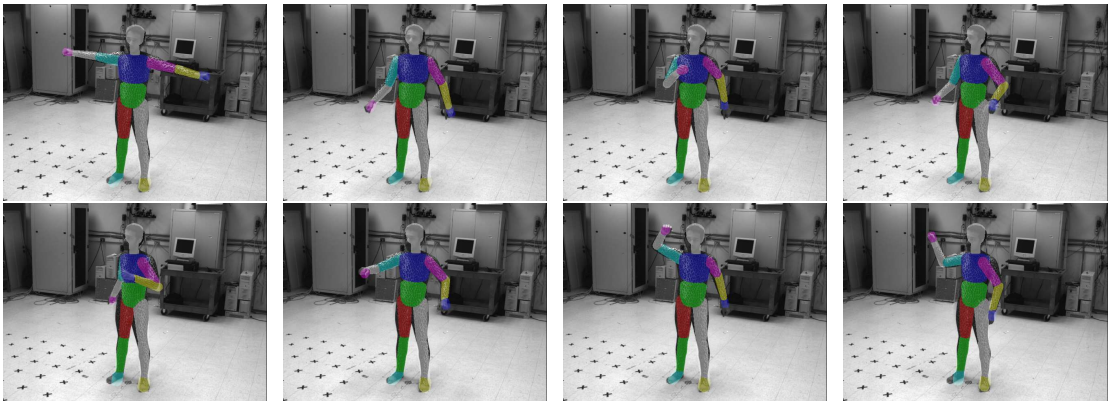
Sequence 1: Images from camera 1

Sequence 1: Images from camera 3

Figure 6.12: Tracking results for sequence 1

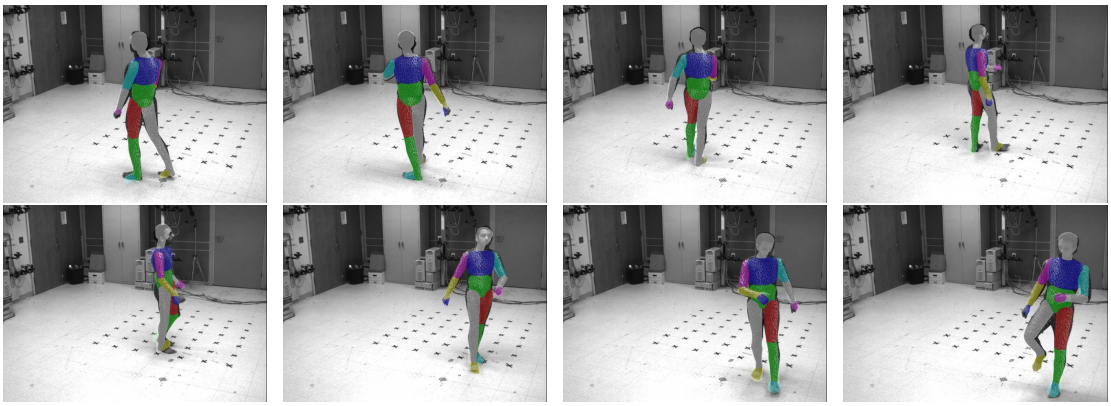


Sequence 2: Images from camera 1

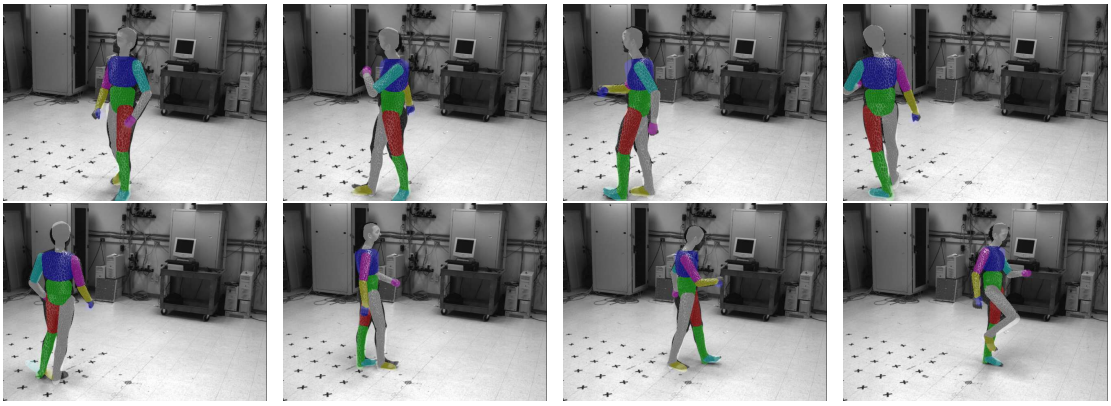


Sequence 2: Images from camera 3

Figure 6.13: Tracking results for sequence 2



Sequence 3: Images from camera 1



Sequence 3: Images from camera 3

Figure 6.14: Tracking results for sequence 3

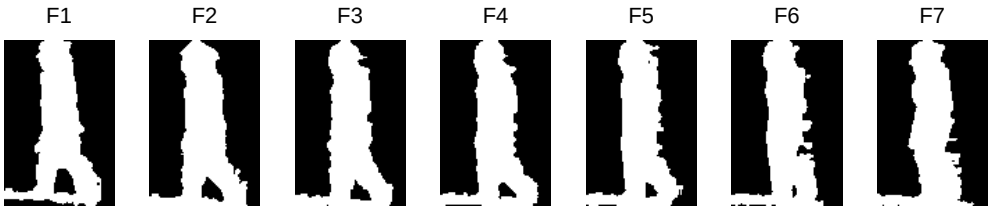
Chapter 7

Conclusion and future directions

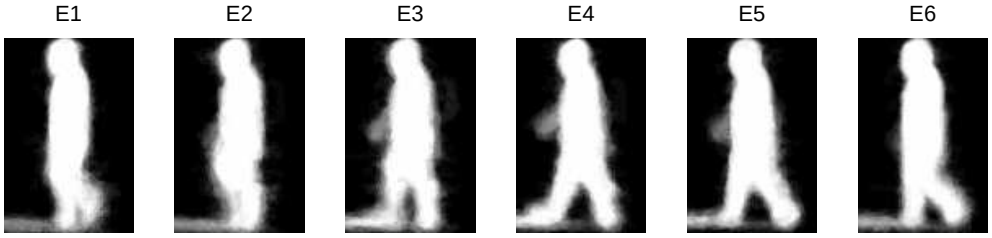
In summary, we have presented a complete markerless motion capture system. We describe a flexible and scalable human body model and present a novel bottom-up segmentation algorithm particularly well suited for segmenting the volumetric human body data into its component articulated chains. The algorithm is able to segment voxel data of human subjects at the joints in the body and is able to handle complex poses. We then use our knowledge of the structure of the human body and use a top-down approach to register the segmented chains to the human body model. We initialize the pose and estimate the human body model parameters for each subject using frames from a subset of the video sequence or 3D scan data. Finally, we track the pose in the sequence using the estimated human body model and the initialized pose for a frame. We combine both spatial and motion cues in the tracker to overcome the twin problems of local minima and drift respectively. We have shown how to apply the above algorithms to estimate human body models, initialize and track pose on different kinds of input data and for different subjects.

In this chapter, we present some research and application directions that extend the markerless motion capture system that we have presented. We touch upon the huge potential of a markerless motion capture system to analyze human motion, including gait, and its applications in biomechanics and surveillance in Section 7.1. We explore the extension of the segmentation algorithm using depth images besides the intensity images in Section 7.2. We also discuss the extension of the components of our system as well as their integration into a near real-time system in Section 7.3.

7.1 Human motion analysis



(a) Observations



(b) Exemplars

Figure 7.1: (a) Set of frames used as observation in the HMM method [31] to recognize subjects using their gait (b) *Exemplars*, which represent the states of the HMM used to model the shape and gait of each subject.

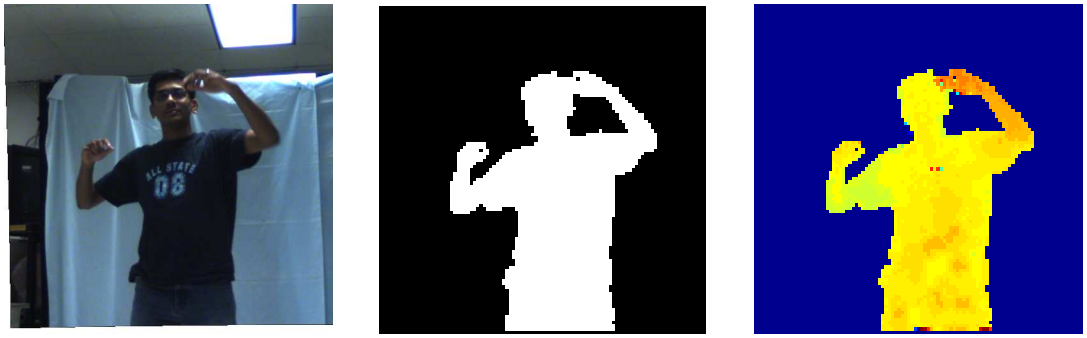
Marker-based techniques are already in wide use in biomechanical research to analyze gait and human motion. However, as described in Chapter 1, marker-based motion capture methods have several short-comings that limit their use on a large scale. Mündermann *et al.* [15] and Corazza *et al.* [16] have studied the feasibility of accurately measuring 3D human body kinematics through a marker-less system for musculo-skeletal biomechanics. Much work has also been done in the field of computer vision and image processing on gait analysis using images from a single camera to tackle problems such as person identification and general surveillance. Murray performed some pioneering work [45] in which gait patterns for pathologically abnormal patients were compared with those of pathologically normal people. There has been considerable work in biomechanics and psychology since then to suggest that there is much information in gait that can be used

for analyzing gender, identity, and abnormalities that may be caused by injuries. Nixon *et al.* [49] study gait as a biometric and survey techniques in computer vision that analyze gait. We have also analyzed gait in video sequences for the purpose of person identification using a Hidden Markov Model (HMM) to model both the shape and the dynamics [31, 30, 70]. While these methods use images from a single camera as input, the HMM framework itself is quite general and is able to use other inputs such as the 3D pose vector, $\mathbf{\Phi}$, which not only separates the shape and pose information but also offers a richer description of the motion. Figure 7.1 illustrates some sample observations and estimated exemplars used in the HMM framework for human identification [31]. It is possible to apply existing shape and gait analysis techniques, which use images and silhouettes as input observations obtained from a single camera, to the 3D pose vector that can be extracted motion capture methods. These methods have applications in fields ranging from biomechanics to surveillance, and human-computer interaction.

7.2 Depth images for pose estimation

There exist applications, such as human computer interaction, where it is not feasible to have multiple calibrated cameras in a suitable configuration all around the subject. In such cases, it is extremely helpful if depth information is available, so that we can deal with segmentation, kinematic singularities, and occlusion. Depth information can also be useful in motion capture and to reduce the number of cameras required. There exist cameras that provide range data (depth) in a scene using new technology such as shuttered light pulse [24] or by measuring the amount of time that light takes to reach each pixel[1]. Alternatively, we can replace each camera with a stereo pair in order to compute the depth using disparity measures. Obtaining direct depth information is usually very advantageous in that it provides complete 3D coordinates of pixels and it is an independent source

[1] http://www.canesta.com/



(a) Left image (b) Foreground (c) Disparity map

Figure 7.2: Combining depth disparity and foreground silhouette

of information that can be used in conjunction with the intensity map to aid in low-level vision problems such as segmentation. We present an example where we can perform segmentation using both the silhouette and the depth information in Figure 7.2. The depth information can be used to perform segmentation to remove the false connection between the head and the arm as illustrated in Figure 7.3 which denotes the neighborhood relation between nodes in normal space that has been computed based on distance *and* depth map. The corresponding segmentation in Laplacian Eigenspace is presented in Figure 7.4. The issues to be explored in this area are the computation of the W matrix using depth disparity values and self-occlusion.

7.3 Extension and tight integration of system

While we have all the important components of a markerless motion capture system, there remains the challenge of tightly integrating these components into a completely automated system. One of the challenges is to construct a pipeline to compute the voxel data for a given sequence from multiple cameras under different environmental conditions within the scope of the project. The pre-processing operations for each camera, including voxel reconstruction, are largely independent of the images from other cameras, and hence it is possible to perform the

pre-processing operations in a highly parallel manner. The design of a parallel architecture for voxel computation offers the advantage of making the time required for the voxel computation step independent of the number of cameras, and thus provides the possibility of *real-time* feedback of the output of the algorithm. For certain applications, especially in sports analysis, we need to incorporate external objects such as golf clubs or tennis rackets into our human body model so that we can correctly perform segmentation and registration of voxels. Figure 7.5 illustrates two views of a golf player in mid-swing. The system will need to handle the close proximity of limbs as well as the presence of an external object; the golf club. Finally, we can incorporate probabilistic tracking, such as a particle filter, in our algorithm, in order to make the tracking more robust and better integrated. We can leverage the multiple cues used in our tracker to efficiently search the pose space in the particle filter.

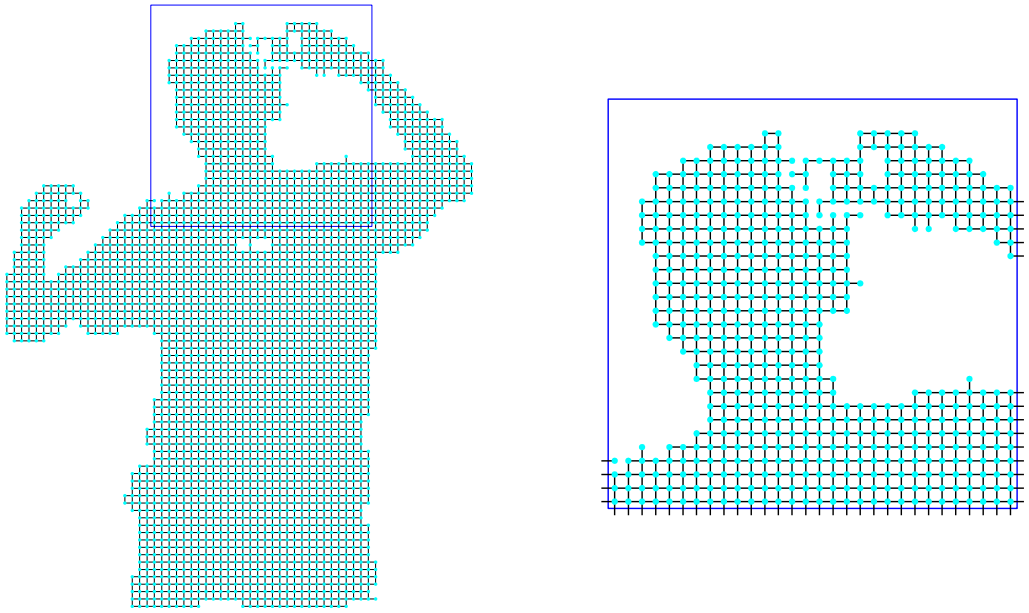
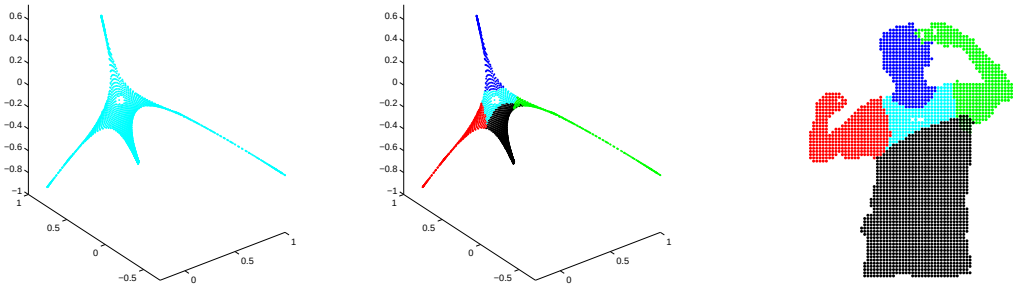


Figure 7.3: Computing W using disparity map and grid and neighbors: An edge is placed between two nodes if they are neighbors and there is no depth discontinuity between them. The close up shows that pixels on the left arm are not connected to pixels on the head due to the depth discontinuity between them.



(a) (b) (c)

Figure 7.4: Segmentation in Laplacian Eigenspace.

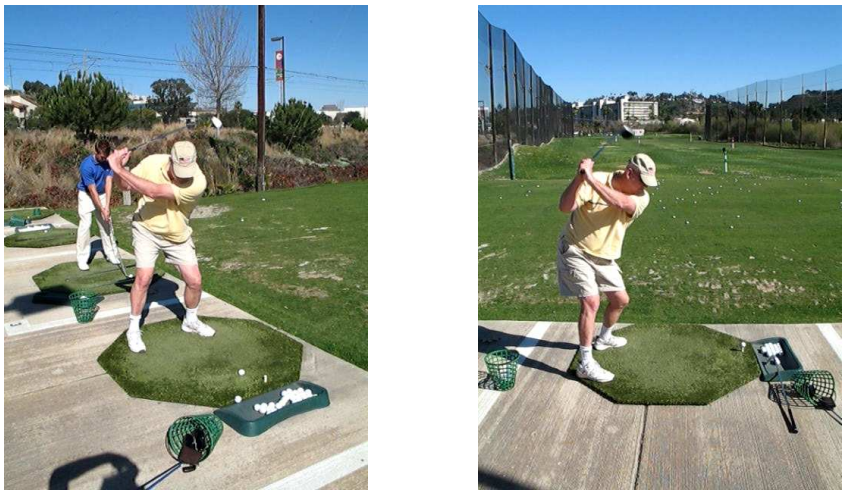


Figure 7.5: Analysis of a golf swing: We need to be able to handle the close proximity of the limbs, the connection between the two palms as well as the presence of a golf club in our human body model, segmentation and registration.

Appendix A

Portable motion capture system

We briefly describe the schematic of the portable motion capture system that we have designed and built. The system is called *Hydra* and is under construction. The current capture facility consists of ten Pixelink PL-A742 cameras connected to two workstations that control the cameras as well as store the images from the cameras. The cameras are connected to the workstations through firewire cables and to each other through custom-fitted ethernet cables and trigger-buffer boards. The cameras can be triggered individually using software trigger or synchronously using either an internal (generated by a camera) or an external (generated by an external device) trigger signal buffered through the buffer boards. The schematic of the facility is presented in Figure A.1. The details of the system can be obtained in a technical report which is in preparation [71] and also the website.[1]

[1] http://www.cfar.umd.edu/users/aravinds/research/hydra.html

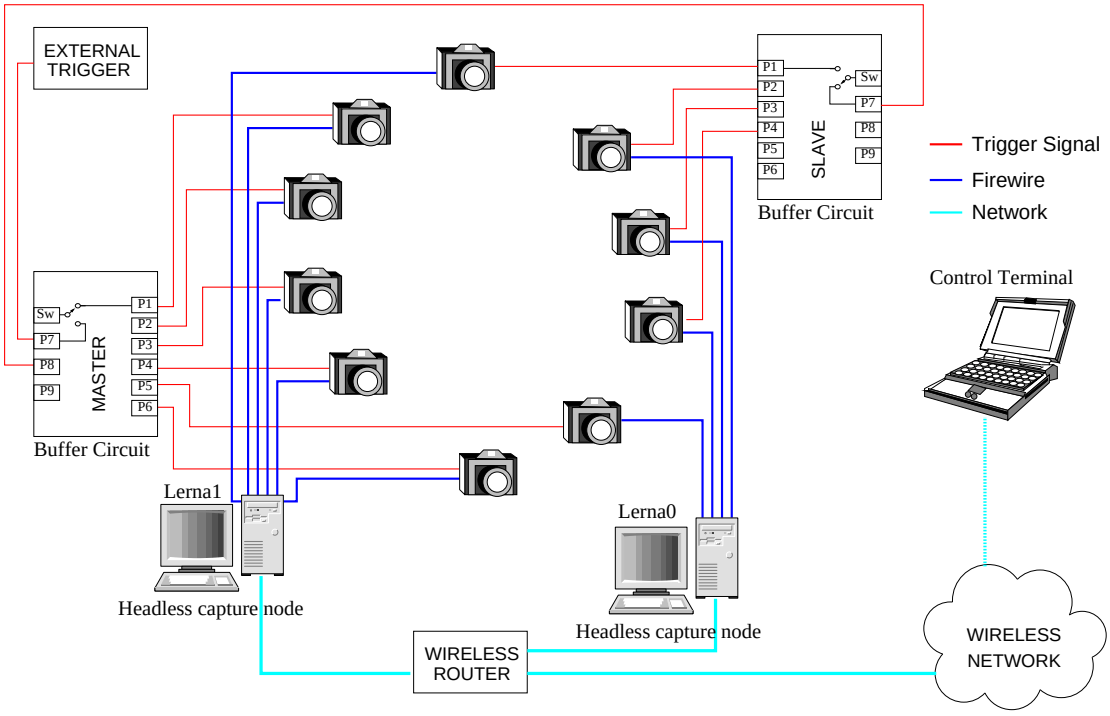
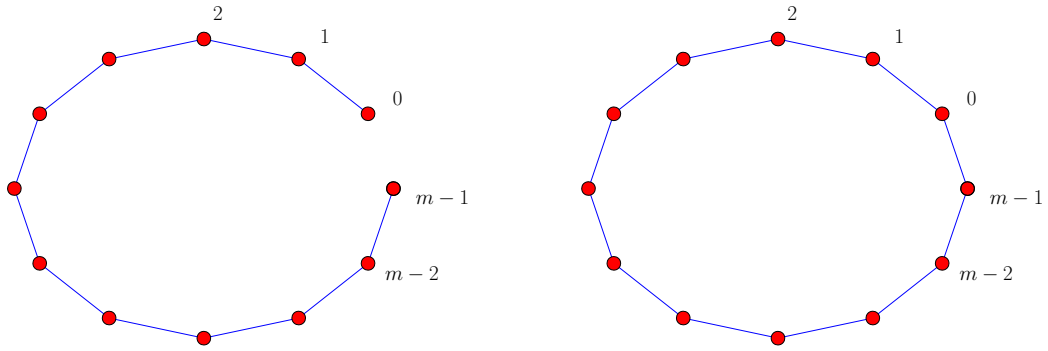


Figure A.1: Hydra Schematic: Layout for ten cameras with two workstations

Appendix B

Eigenvectors of simple graphs

We describe the Laplacian eigenvectors of two simple graphs on m vertices, namely the path graph, P_m and the ring graph, R_m. The two graphs are very similar and illustrated in Figure B.1.



(a) Path graph (b) Ring graph

Figure B.1: Illustration of a path graph and ring graph on m vertices.

B.1 Eigenvectors of Ring graph

We consider a ring graph on m vertices or nodes, the Laplacian eigenvectors of which need to satisfy

$$2x_i - x_{i-1} - x_{i+1} = \lambda x_i, \qquad 0 \leq i \leq m-1 \tag{B.1}$$

where

$$x_i = x_{i \bmod m}. \tag{B.2}$$

Substituting $x_i = \sin(\theta + i\varphi)$ in (B.1) leads to

$$2\sin(\theta + i\varphi) - \sin(\theta + (i-1)\varphi) - \sin(\theta + (i+1)\varphi) = \lambda \sin(\theta + i\varphi). \quad \text{(B.3)}$$

We get $\lambda = 2 - 2\cos(\varphi)$ using the following trigonometric identities.

$$\sin A + \sin B = 2\sin\frac{A+B}{2}\cos\frac{A-B}{2} \quad \text{(B.4)}$$

$$\cos A + \cos B = 2\cos\frac{A+B}{2}\cos\frac{A-B}{2} \quad \text{(B.5)}$$

In order to satisfy (B.2), we note that the solution is of the form

$$x_i = \sin(\theta + 2\pi ki/m) \qquad i = 0, 1, \cdots, \lfloor m/2 \rfloor \quad \text{(B.6)}$$

where θ can take any value. The eigenvalues are given by $2 - 2\cos(2\pi k/m)$.

B.2 Eigenvectors of Path graph

We consider the path graph on m vertices, P_m. The nodes, x_i, need to satisfy the following set of equations.

$$x_0 - x_1 = \lambda x_0 \quad \text{(B.7)}$$

$$x_{m-1} - x_{m-2} = \lambda x_{m-1} \quad \text{(B.8)}$$

$$2x_i - x_{i-1} - x_{i+1} = \lambda x_i \qquad 0 < i < m-1 \quad \text{(B.9)}$$

Considering solutions of the form $x_i = \sin(\theta + \varphi i)$ for $i = 0, 1, \cdots, m-1$, we need $x_{m-1} = x_m$ and $x_0 = x_{-1}$ or

$$\sin(\theta + \varphi(0)) = \sin(\theta + \varphi(-1)) \quad \text{(B.10)}$$

and

$$\sin(\theta + \varphi(m-1)) = \sin(\theta + \varphi(m)). \quad \text{(B.11)}$$

These conditions are satisfied if we choose $\varphi = \pi k/m$ and $\theta = \pi/2 + \pi k/2m$.

$$x_i = \cos(\pi k/2m + \pi ki/m) \text{ for } i = 0, 1, \ldots, m-1 \quad \text{(B.12)}$$

The eigenvalues are given by $\lambda_k = 2 - 2\cos \pi k/m$. The values of k range from $0, 1, 2, \ldots, m-1$.

Appendix C

Laplacian eigenvalues of extended tree graphs: Solutions for $f(\varphi, l)$ in $[0, \pi]$

We explore the solutions for $f(\varphi, l) = 0$ in the interval $[0, \pi]$, as the solutions give us the eigenvalues as well as the structure of the eigenvectors of the extended star graph, whose properties we explore in Section 4.2. The function $f(\varphi, l)$ is given by

$$f(\varphi, l) = (1 - 2/n)(1 - \cos\varphi) - \sin\varphi \tan(\varphi l), \tag{C.1}$$

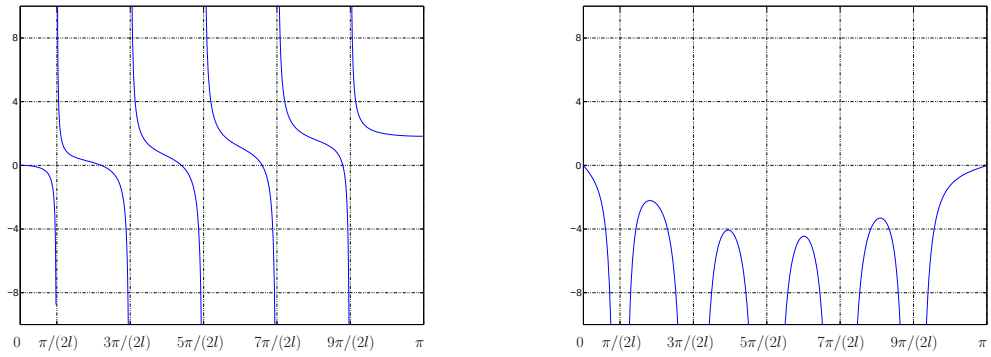
where $l = m + 1/2$, and $m \in \mathbb{N}$. We see that $f(0, l) = 0$ and $f(\pi, l) = 2(1 - 2/n) - 1/l \geq 0$, with the equality holding $\iff$ $m = 1$ and $n = 3$. Hereafter, we ignore the l parameter in $f(\varphi, l)$. We have

$$\lim_{\varphi \to \pi} \sin\varphi \tan(\varphi l) = \lim_{\psi \to 0} \sin\psi \cot(\psi l) \tag{C.2}$$

$$= \lim_{\psi \to 0} \sin\psi \cos(\psi l)/\sin(\psi l) \tag{C.3}$$

$$= \psi/(\psi l) = 1/l. \tag{C.4}$$

where we substitute $\psi = \pi - \varphi$. We see that $f(\varphi)$ has discontinuities at $\tan(\varphi l) = \pm\infty$ or $\varphi = \pi(2k + 1)/(2m + 1)$ for $k = 0, 1, \cdots, m - 1$. We show that $f'(\varphi) < 0$ in the continuous intervals between these discontinuities and hence $f(\varphi)$ is monotonically decreasing in each of the continuous intervals in $[0, \pi]$. $f(\varphi)$ and $f'(\varphi)$ are plotted in Figure C.1 for $m = 5$. The value of $f(\varphi)$ switches from $-\infty$ to $+\infty$ at the discontinuities. Thus we see that there is exactly one solution for $f(\varphi) = 0$ in each of the $m - 1$ intervals $\left[\pi\frac{2k+1}{2m+1}, \pi\frac{2k+3}{2m+1}\right]$ for $k = 0, 1, \cdots, m - 2$. We now show that $f'(\varphi) < 0$ for $\varphi \in [0, \pi]$.



(a) $f(\varphi)$ (b) $f'(\varphi)$

Figure C.1: The function $f(\varphi)$ and $f'(\varphi)$ for $m = 5$. The discontinuities of $f(\varphi)$ occur at $\pi(2k+1)/(2m+1)$ for $k = 1, 2, \cdots, m-1$. We see that $f'(\varphi) < 0$ and $f(\varphi)$ is monotonically decreasing function except at the points of discontinuity.

$$f'(\varphi) = (1 - 2/n)\sin\varphi - \sin\varphi \sec^2(\varphi l) l - \cos\varphi \tan(\varphi l) \tag{C.5}$$

$$= (1 - 2/n)\sin\varphi - \sin\varphi (1 + \tan^2(\varphi l)) l - \cos\varphi \tan(\varphi l) \tag{C.6}$$

$$= (1 - 2/n - l - l\tan^2(\varphi l))\sin\varphi - \cos\varphi \tan(\varphi l) \tag{C.7}$$

$$= -\left((l + l\tan^2(\varphi l) + 2/n - 1)\sin\varphi + \cos\varphi \tan(\varphi l)\right) \tag{C.8}$$

where (C.6) follows by substituting $\sec^2(\theta) = 1 + \tan^2(\theta)$. We therefore need to show that

$$\sin\varphi \left(l + l\tan^2(\varphi l) + 2/n - 1\right) > -\cos\varphi \tan(\varphi l) \qquad \text{or} \tag{C.9}$$

$$\sin\varphi \left(l + l\tan^2(\varphi l) - 1\right) > -\cos\varphi \tan(\varphi l), \tag{C.10}$$

where the last equation follows as $2/n \sin\varphi > 0$. We note that in the interval under consideration, *i.e.* $[0, \pi]$, that $\sin\varphi > 0$, and hence

$$\sin\varphi \left(l + l\tan^2(\varphi l) - 1\right) > 0. \tag{C.11}$$

We now proceed to prove that (C.10) holds in the interval $[0, \pi]$, by considering different sub-intervals in the following paragraphs.

- **Interval** $0 < \varphi < \pi/(2l)$:

  We have $-\cos\varphi\tan\varphi < 0$ and hence (C.10).

- **Interval** $\pi - \pi/(2l) < \varphi < \pi$:

  Substituting $\varphi = \pi - \varphi$ in (C.10), we need to show for $0 < \varphi < \pi/(2l)$ that

$$\sin\varphi\left(l + l\cot^2(\varphi l) - 1\right) > \cos\varphi\cot(\varphi l). \tag{C.12}$$

  We have

$$\begin{aligned}\sin\varphi\left(l + l\cot^2(\varphi l) - 1\right) &> l\sin\varphi\cot^2(\varphi l) && \text{(C.13)}\\ &= \frac{l\sin\varphi}{\sin(\varphi l)}\cos(\varphi l)\cot(\varphi l) && \text{(C.14)}\\ &> \cos\varphi\cot(\varphi l) && \text{(C.15)}\end{aligned}$$

  where the last statement follows because $l\sin\varphi/\sin(\varphi l) > 1$ for $0 \le \varphi \le \pi/2$.

- **Interval** $\pi/(2l) < \varphi < \pi - \pi/(2l)$

  We have

$$\begin{aligned}\sin\varphi\left(l + l\tan^2(\varphi l) - 1\right) &= |\cos\varphi||\tan(\varphi l)||\tan\varphi|\left((l-1)|\tan(\varphi l)|^{-1}\right. \\ &\quad \left. + l|\tan(\varphi l)|\right) && \text{(C.16)}\\ &\ge |\cos\varphi||\tan(\varphi l)|\frac{\pi}{2l}\left((l-1)|\tan(\varphi l)|^{-1}\right. \\ &\quad \left. + l|\tan(\varphi l)|\right) && \text{(C.17)}\\ &> |\cos\varphi||\tan(\varphi l)|. && \text{(C.18)}\end{aligned}$$

  (C.17) holds because $|\tan\varphi| > \min\varphi, \pi - \varphi$ for $\varphi \in [0, \pi]$, and (C.18) holds

because

$$\frac{\pi}{2l}\left((l-1)|\tan(\varphi l)|^{-1}+l|\tan(\varphi l)|\right)=\frac{\pi}{2}\left(\frac{(l-1)}{l}|\tan(\varphi l)|^{-1}+|\tan(\varphi l)|\right) \tag{C.19}$$

$$\geq \frac{\pi}{2}\left(\frac{1}{3}|\tan(\varphi l)|^{-1}+|\tan(\varphi l)|\right) \tag{C.20}$$

$$\geq \frac{\pi}{2}\frac{2}{\sqrt{3}} \tag{C.21}$$

$$> 1. \tag{C.22}$$

Bibliography

[1] J. Aggarwal and Q. Cai, "Human motion analysis: A review," *Computer Vision and Image Understanding*, vol. 73, no. 3, pp. 428–440, 1999.

[2] D. Anguelov, D. Koller, H. Pang, P. Srinivasan, and S. Thrun., "Recovering articulated object models from 3-D range data," in *Proc. of the Conference on Uncertainty in Artificial Intelligence*, Banff, Canada, 2004, pp. 18–26.

[3] N. I. Badler, C. B. Phillips, and B. L. Webber, *Simulating Humans*. Oxford University Press, Oxford, UK, 1993.

[4] J. L. Barron, D. J. Fleet, and S. S. Beauchemin, "Performance of optical flow techniques," *International Journal of Computer Vision*, vol. 12, no. 1, pp. 43–77, 1994.

[5] M. Belkin and P. Niyogi, "Laplacian eigenmaps for dimensionality reduction and data representation," *Neural Computation*, vol. 15, no. 6, pp. 1373–1396, 2003.

[6] M. Brand, "Charting a manifold," in *Proc. of the Conference on Neural Information Processing Systems*, Vancouver, BC, Canada, December 2002.

[7] C. Bregler and J. Malik, "Tracking people with twists and exponential maps," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, Santa Barbara, CA, USA, June 1998, pp. 8–15.

[8] G. Brostow, I. Essa, D. Steedly, and V. Kwatra, "Novel skeletal representation for articulated creatures," in *Proc. of the European Conference on Computer Vision*, vol. 3, Prague, Czech Republic, May 2004, pp. 66–78.

[9] J. Carranza, C. Theobalt, M. Magnor, and H. Seidel, "Freeviewpoint video of human actors," *ACM Transactions on Graphics*, vol. 22, no. 2, pp. 569–577, 2003.

[10] T.-J. Cham and J. M. Rehg, "A multiple hypothesis approach to figure tracking," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2, Ft. Collins, CO, USA, June 1999, pp. 239–245.

[11] K. Cheung, S. Baker, and T. Kanade, "Shape-from-silhouette of articulated objects and its use for human body kinematics estimation and motion capture," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1, Madison, USA, June 2003, pp. 77–84.

[12] B. Y. Choi, Y. M. Chae, I. H. Chung, and H. S. Kang, "Correlation between the postmortem stature and the dried limb-bone lengths of korean adult males," *Yonsei Medical Journal*, vol. 38, no. 2, pp. 79–85, 1997.

[13] C.-W. Chu, O. C. Jenkins, and M. J. Mataric, "Markerless kinematic model and motion capture from volume sequences." in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2, Madison, USA, June 2003, pp. 475–482.

[14] F. R. K. Chung, *Spectral Graph Theory*. American Mathematical Society, Providence, 1997.

[15] S. Corazza, L. Mündermann, and T. P. Andriacchi, "The evolution of methods for the capture of human movement leading to markerless motion capture for biomechanical applications," *Journal of Neuroengineering and Rehabilitation*, vol. 3, no. 6, 2006.

[16] S. Corazza, L. Mündermann, and T. P. Andriacchi, "A markerless motion capture system to study musculoskeletal biomechanics: visual hull and simulated annealing approach." *Annals of Biomedical Engineering*, vol. 34, no. 6, 2006.

[17] T. Cox and M. Cox, *Multidimensional Scaling*. Chapman and Hall, London, 1994.

[18] Q. Delamarre and O. Faugeras, "3D articulated models and multi-view tracking with silhouettes," in *Proc. of the International Conference on Computer Vision*, vol. 2, Kerkyra, Corfu, Greece, September 1999, pp. 716–721.

[19] D. Demirdjian, T. Ko, and T. Darrell, "Constraining human body tracking," in *Proc. of the International Conference on Computer Vision*, vol. 2, Nice, France, October 2003, pp. 1071–1078.

[20] A. Elad and R. Kimmel, "On bending invariant signatures for surfaces," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, pp. 1285–1295, 2003.

[21] A. E. Engin and S. T. Tumer, "Three-dimensional kinematic modelling of the human shoulder complex," *Journal of Biomechanical Engineering*, vol. 111, pp. 107–112, 1989.

[22] D. M. Gavrila, "The visual analysis of human movement: A survey," *Computer Vision and Image Understanding*, vol. 73, no. 1, pp. 82–98, 1999.

[23] D. Gavrila and L. Davis, "3-D model-based tracking of humans in action: A multi-view approach," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, 1996, pp. 73–80.

[24] H. González-Banõs and J. Davis, "Computing depth under ambient illumination using multi-shuttered light," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2, Washington, DC, USA, Jun 2004, pp. 234–241.

[25] K. Halvorsen, T. Arndt, H. Rosdahl, and A. Thorstensson, "Evaluation of three different models of the shoulder kinematics," in *Proc. of the International Symposium on 3D Human Movement*, Tampa, FL, USA, March 2004, pp. 128–131.

[26] X. He, S. Yan, Y. Hu, P. Niyogi, and H.-J. Zhang, "Face recognition using laplacianfaces," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 3, pp. 328–340, 2005.

[27] S. X. Ju, M. J. Black, and Y. Yacoob, "Cardboard people: A parameterized model of articulated image motion," in *Proc. of the International Conference on Automatic Face and Gesture Recognition*, Killington, Vermont, USA, October 1996, pp. 38–44.

[28] I. A. Kakadiaris and D. Metaxas, "3D human body model acquisition from multiple views," in *Proc. of the International Conference on Computer Vision*, Boston, MA, USA, June 1995, pp. 618–623.

[29] I. A. Kakadiaris and D. Metaxas, "Model-based estimation of 3D human motion," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 12, pp. 1453–1459, December 2000.

[30] A. Kale, A. Sundaresan, A. RoyChowdhury, and R. Chellappa, *Handbook on Pattern Recognition and Computer Vision*. World Scientific Publishing Company Pvt. Ltd., 2005, ch. Gait-Based Human Identification From A Monocular Video Sequence.

[31] A. A. Kale, A. Sundaresan, A. N. Rajagopalan, N. P. Cuntoor, A. K. R. Chowdhury, V. Krüger, and R. Chellappa, "Identification of humans using gait," *IEEE Transactions on Image Processing*, vol. 13, no. 9, pp. 1163–1173, September 2004.

[32] N. Krahnstoever and R. Sharma, "Articulated models from video," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1, Washington, DC, USA, June 2004, pp. 894–901.

[33] S. Lafon and A. B. Lee, "Diffusion maps and coarse-graining: A unified framework for dimensionality reduction, graph partitioning, and data set parameterization," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 9, pp. 1393–1403, 2006.

[34] X. Lan and D. P. Huttenlocher, "A unified spatio-temporal articulated model for tracking." in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1, Washington, DC, USA, June 2004, pp. 722–729.

[35] E.-J. Marey, *Le Mouvement*. Masson, Paris, 1894.

[36] D. Marr, *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information.* W. H. Freeman and Company, New York, 1982.

[37] M. C. D. Mendonca, "Estimation of height from the length of long bones in a portugese adult population," *American Journal of Physical Anthropology*, vol. 112, pp. 39–48, 2000.

[38] I. Mikić, M. Trivedi, E. Hunter, and P. Cosman, "Human body model acquisition and tracking using voxel data," *International Journal of Computer Vision*, vol. 53, no. 3, 2003.

[39] T. Moeslund and E. Granum, "Multiple cues used in model-based human motion capture," in *Proc. of the International Conference on Face and Gesture Recognition*, Grenoble, France, March 2000, pp. 362–367.

[40] T. Moeslund and E. Granum, "A survey of computer vision-based human motion capture," *Computer Vision and Image Understanding*, pp. 231–268, 2001.

[41] B. Mohar, *Graph Theory, Combinatorics, and Applications.* J. Wiley, New York, 1991, vol. 2, ch. The Laplacian Spectrum of Graphs, pp. 871–898.

[42] G. Mori and J. Malik, "Estimating human body configurations using shape context matching," in *Proc. of the European Conference on Computer Vision*, 2002, pp. 666–680.

[43] D. Morris and J. M. Rehg, "Singularity analysis for articulated object tracking," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, Santa Barbara, CA, USA, June 1998, pp. 289–297.

[44] L. Mündermann, S. Corazza, and T. Andriacchi, "Accurately measuring human movement using articulated ICP with soft-joint constraints and a repository of articulated models," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, Minneapolis, MN, USA, June 2007.

[45] M. P. Murray, "Gait as a total patern of movement," *American Journal of Physical Medicine*, vol. 46, no. 1, pp. 290–332, 1967.

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

[47] E. Muybridge, *The Human Figure in Motion*. Dover Publications, 1901.

[48] B. Nadler, S. Lafon, R. Coifman, and I. Kevrekidis, "Diffusion maps, spectral clustering and the eigenfunctions of fokker-planck operators," in *Proc. of the Conference on Neural Information Processing Systems*, Vancouver, British Columbia, Canada, December 2005.

[49] M. S. Nixon, T. N. Tan, and R. Chellappa, *Human Identification Based on Gait (The Kluwer International Series on Biometrics)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2005.

[50] A. Ozaslan, H. T. M. Yasar Iscan, Inci Oxaslan, and S. Koc, "Estimation of stature from body parts," *Forensic Science International*, vol. 132, no. 1, pp. 40–45, 2003.

[51] R. Plänkers and P. Fua, "Articulated soft objects for video-based body modeling," in *Proc. of the International Conference on Computer Vision*, vol. 1, Vancouver, British Columbia, Canada, July 2001, pp. 394–401.

[52] D. Ramanan and D. A. Forsyth, "Finding and tracking people from the bottom up," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2, Madison, WI, USA, June 2003, pp. 467–474.

[53] J. Rehg, D. D. Morris, and T. Kanade, "Ambiguities in visual tracking of articulated objects using two- and three-dimensional models," *International Journal of Robotics Research*, vol. 22, no. 6, pp. 393 – 418, June 2003.

[54] X. Ren, A. C. Berg, and J. Malik, "Recovering human body configurations using pairwise constraints between parts," in *Proc. of the International Conference on Computer Vision*, vol. 1, Beijing, China, October 2005, pp. 824–831.

[55] K. Rohr, *Human Movement Analysis Based on Explicit Motion Models*. Kluwer Academic, 1997.

[56] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, no. 5500, pp. 2323–2326, 2000.

[57] B. Schölkopf, A. Smola, and K.-R. Müller, "Nonlinear component analysis as a kernel eigenvalue problem," *Neural Computation*, vol. 10, pp. 1299–1319, 1998.

[58] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 888–905, 2000.

[59] H. Sidenbladh, M. J. Black, and D. J. Fleet, "Stochastic tracking of 3D human figures using 2D image motion," in *Proc. of the European Conference on Computer Vision*, vol. 2, Dublin, Ireland, June 2000, pp. 702–718.

[60] L. Sigal, S. Bhatia, S. Roth, M. J. Black, and M. Isard, "Tracking loose-limbed people," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1, Washington, DC, USA, June 2004, pp. 421–428.

[61] L. Sigal and M. Black, "Humaneva: Synchronized video and motion capture dataset for evaluation of articulated human motion," Brown University, Technical report CS-06-08, 2006.

[62] L. Sigal, M. Isard, B. H. Sigelman, and M. J. Black, "Attractive people: Assembling loose-limbed models using non-parametric belief propagation." in *Proc. of the Conference on Neural Information Processing Systems*, 2003.

[63] C. Sminchisescu and B. Triggs, "Kinematic jump processes for monocular 3D human tracking," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1, Madison, WI, USA, June 2003, pp. 69–76.

[64] C. Sminchisescu and B. Triggs, "Covariance scaled sampling for monocular 3D body tracking," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1, Kauai, HI, USA, Dec 2001, pp. 447–454.

[65] A. Sundaresan and R. Chellappa, "Acquisition of articulated human body models using multiple cameras," in *Proc. of the Conference on Articulated*

Motion and Deformable Objects, Port d'Andratx, Mallorca, Spain, July 2006, pp. 78–89.

[66] A. Sundaresan and R. Chellappa, "Multi-camera tracking of articulated human motion using motion and shape," in *Proc. of the Asian Conference on Computer Vision*, vol. 2, Hyderabad, India, January 2006, pp. 131–140.

[67] A. Sundaresan and R. Chellappa, "Segmentation and probabilistic registration of articulated body model," in *Proc. of the International Conference on Pattern Recognition*, vol. 2, Hong Kong, China, August 2006, pp. 92–96.

[68] A. Sundaresan and R. Chellappa, "Model driven segmentation and registration of articulating humans in laplacian eigenspace," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2007, (Under revision).

[69] A. Sundaresan and R. Chellappa, "Multi-camera tracking of articulated human motion using shape and motion cues," *IEEE Transactions on Image Processing*, 2007, (In preparation).

[70] A. Sundaresan, A. RoyChowdhury, and R. Chellappa, "Multiple view tracking of human motion modelled by kinematic chains," in *Proc. of the IEEE International Conference on Image Processing*, vol. 2, Barcelona, Catalonia, Spain, September 2004, pp. 93–96.

[71] A. Sundaresan, J. Sherman, F. McCall, and R. Chellappa, "Hydra: The portable camera capture facility for motion analysis," University of Maryland Institute for Advanced Computer Studies, College Park, MD, Tech. Rep., 2007, (In preparation).

[72] T. Svoboda, D. Martinec, and T. Pajdla, "A convenient multi-camera self-calibration for virtual environments," *PRESENCE: Teleoperators and Virtual Environments*, vol. 14, no. 4, August 2005.

[73] J. B. Tenenbaum, V. de Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, no. 5500, pp. 2319–2323, 2000.

[74] C. Theobalt, J. Carranza, M. A. Magnor, and H.-P. Seidel, "Combining 3D flow fields with silhouette-based human motion capture for immersive video," *Graphical Models*, vol. 66, no. 6, pp. 333–351, 2004.

[75] S. Wachter and H.-H. Nagel, "Tracking persons in monocular image sequences," *Computer Vision and Image Understanding*, vol. 74, no. 3, pp. 174–192, June 1999.

[76] L. Wang, W. Hu, and T. Tan, "Recent developments in human motion analysis," *Pattern Recognition*, vol. 36, no. 3, pp. 585–601, March 2003.

[77] Y. Weiss, "Segmentation using eigenvectors: A unifying view," in *Proc. of the International Conference on Computer Vision*, vol. 2, Kerkyra, Corfu, Greece, September 1999, pp. 975–982.

[78] N. Werghi, Y. Xiao, and P. Siebert, "A functional-based segmentation of human body scans in arbitrary postures," *IEEE Transactions on Systems, Man and Cybernetics, Part B*, vol. 36, no. 1, pp. 153–165, Feb 2006.

[79] Y. Xiao, P. Siebert, and N. Werghi, "Topological segmentation of discrete human body shapes in various postures based on geodesic distance," in *Proc. of the International Conference on Pattern Recognition*, vol. 3. Cambridge, England, UK: IEEE Computer Society, August 2004, pp. 131–135.

[80] M. Yamamoto and K. Koshikawa, "Human motion analysis based on a robot arm model," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, Maui, HI, June 1991, pp. 664–665.

[81] M. Yamamoto, A. Sato, S. Kawada, T. Kondo, and Y. Osaki, "Incremental tracking of human actions from multiple views," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, Santa Barbara, CA, USA, June 1998, pp. 2–7.

[82] Z. Zhang and H. Zha, "Principal manifolds and nonlinear dimensionality reduction via tangent space alignment," *SIAM Journal on Scientific Computing*, vol. 26, no. 1, pp. 313–338, 2004.