HUMAN MOTION TRACKING WITH MULTIPLE CAMERAS USING A PROBABILISTIC FRAMEWORK FOR POSTURE ESTIMATION

A Thesis

Submitted to the Faculty

of

Purdue University

by

Ruth Devlaeminck

In Partial Fulfillment of the

Requirements for the Degree

of

Master of Science in Electrical and Computer Engineering

August 2006

Purdue University

West Lafayette, Indiana

To my mom, who helped me over all the hurdles

ACKNOWLEDGMENTS

I wis to thank my committee members who took time out of their schedules to help me with my thesis. I especially wish to acknowledge my major professor Avi Kak who encouraged me to go to grad school and helped me along that road.

I wish to thank Johnny Park for all of his help and guidance along the way. Without his help this would not have been possible.

I wish to thank all of the members of the RVL lab who were willing subjects for me.

I wish to thank all of my friends and family who put up with my disappearing for days at work and then calling them in the middle of the night for a break.

TABLE OF CONTENTS

LIST OF FIGURES

Figure Page

LIST OF TABLES

ABSTRACT

Devlaeminck, Ruth. M.S.E.C.E., Purdue University, August, 2006. Human Motion Tracking With Multiple Cameras Using a Probabilistic Framework for Posture Estimation. Major Professor: Avinash C. Kak.

The ability of machines to detect humans and recognize their activities in an environment is an essential part of allowing a machine to interact with humans easily and efficiently. The existing methods for detecting and analyzing human motion often require special markers attached to the human subjects, which prevents the widespread application of the technology. This thesis presents a system based on Zimmermann and Svoboda's work in probabilistic estimation of human motion parameters. Our system can track articulated human motion without relying on markers or special colored clothing. The system consists of two major components: the first component of the system extracts the silhouette of the person in the images acquired by multiple cameras in the environment, the second component of the system then estimates a set of parameters that describe the posture of the person by taking the extracted silhouettes from the recorded images and comparing them with the synthetic silhouettes generated by projecting a virtual 3D human model with a set of predicted posture parameters. The parameter estimation incorporates the physical limitations of the human body such as how far and how fast each joint can move. We show both quantitative and qualitative analysis of the system using synthetic and real images of a human. The experimental results indicate that our system is capable of estimating human postures without any special markers and that our approach has a potential to be used in real-time applications, unlike previous systems which work entirely offline.

1. INTRODUCTION

The existing methods for detecting and analyzing human motion often require special markers attached to the human subjects, which prevents the widespread application of the technology. This thesis presents a system based on Zimmermann and Svoboda's work in probabilistic estimation of human motion parameters. Our system can track articulated human motion without relying on markers or special colored clothing. We show both quantitative and qualitative analysis of the system using synthetic and real images of a human. The experimental results indicate that our system is capable of estimating human postures without any special markers and that our approach has a potential to be used in real-time applications, unlike previous systems which work entirely offline.

The ability of machines to detect humans and recognize their activities in an environment is an essential part of allowing a machine to interact with humans easily and efficiently. The analysis of images involving human motion tracking includes face recognition, hand gesture recognition, whole-body tracking, and articulated-body tracking. There are a wide variety of applications for human motion tracking, for a summary see Table 1.1.

A common application for human motion tracking is that of virtual reality. Human motion tracking can allow for more interactive virtual worlds, better character animation, etc. In order to create a physical presence in a virtual space, one needs to recover the body pose of the person in physical space. As interactive space on the Internet develops further, the addition of gestures, head pose, and facial expressions to the current system of text and icons in 2D spaces would allow for more complex communication.

Another very important application for human motion tracking is a "smart" surveillance system. A "smart" system describes a system that does more than mere motion detection, which could detect animals or wind instead of people. The first step would be to identify that the movement is a person, and then it could possibly do face recognition,

determine what the person is doing, or some other more advanced step. One might also wish to identify suspicious behavior, such as looking into cars in a parking lot, stealing items from a supermarket or department store, or leaving an unattended package in an airport or secure zone.

In the user-interface application domain, vision is useful to complement speech recognition and natural language understanding for better communication between humans and machines. Initially, the machine can only initiate communication if a human is present. More detailed cues can be obtained by recognizing who the user is, observing facial expressions and gestures and perhaps recalling some of the past interactions. Vision can also help speech recognition when there is a noisy environment, or if there is more than one person, identifying who the person is talking to. Vision can also be used to recognize sign-language and to control a system via gestures. Human motion analysis can also provide control for a remotely located implement, such as microscopic surgery.

In the context of motion analysis, a possible application is content-based indexing of sports video footage. For example, one might want to query a large video archive of tennis plays and ask “give me all the cases where player X came to the net and volleyed”. It could also be used as a personal trainer by analyzing inefficiencies in movement and where a move could be improved. It also has major applications in the medical field. Human motion analysis can help a physical therapist understand the progress of a patient and where improvement could be made.

A final application domain is that of model-based image coding. In a video phone setting, one can track faces in images and code them in more detail than the background. More ambitiously, one might try to recover a 3D head model initially and code only the pose and deformation parameters subsequently.

In all of the applications discussed above, a non-intrusive sensory method based on vision is preferable over a method which relies on markers attached to the bodies of the human subjects or a method based on active sensing. Additionally not all of the above applications can be accomplished by methods which require markers or active sensing [1].

Table 1.1
Applications of “Looking at People” primarily from [1]

general domain	**specific area**
virtual reality	-interactive virtual worlds -games -virtual studios -character animation -teleconferencing (e.g. film, advertising, home)
“smart” surveillance systems	-access control -parking lots -supermarkets, department stores -vending machines, ATMs -traffic -suspicious behavior identification
advanced user interfaces	-social interfaces -sign-language translation -gesture driven control -signaling in high-noise environments (e.g. airports, factories) -control remotely located implements
motion analysis	-content-based indexing of sports video footage -personalized training in golf, tennis, etc. -choreography of dance -clinical studies of orthopedic patients
model-based coding	-very low bit-rate video compression

The system consists of two major components: the first component of the system extracts the silhouette of the person in the images acquired by multiple cameras in the environment and generates a binary image, the second component of the system then estimates a set of parameters that describe the posture of the person by taking the extracted silhouettes from the recorded images and comparing them with the synthetic silhouettes generated by projecting a virtual 3D human model with a set of predicted posture parameters. The parameter estimation incorporates the physical limitations of the human body such as how far and how fast each joint can move. After the best set of angles is selected, the system can then display the 3D model with the current set of angles. This process is shown in Figure 1.1.

Many current human motion tracking systems that track articulated humans, rely on markers or special colored clothing. One of the contributions of this project is that it does not. Additionally this project has the potential to do this analysis in real time.

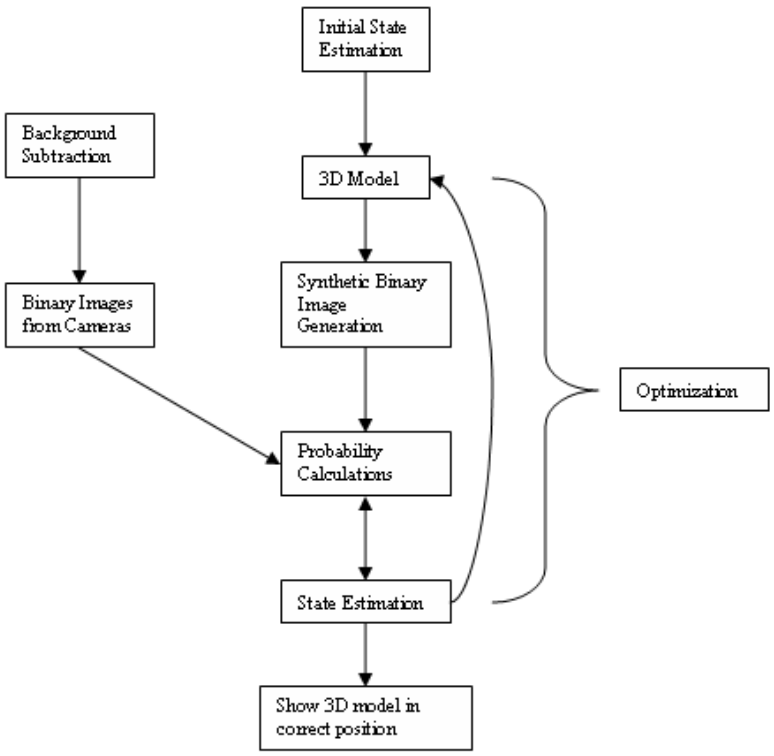


Figure 1.1. Human Motion Tracking Process for a Single Frame as Implemented in Our Algorithm

2. LITERATURE SURVEY

2.1 Assumptions

Various assumptions on the conditions of motion capture are commonly associated with individual contributions. The actual assumptions made characterize the various systems and can provide a useful reference for evaluation. The typical assumptions may be divided into two classes: movement assumptions and appearance assumptions. Movement assumptions restrict the movement of the subject and/or camera(s) involved. Appearance assumptions restrict aspects of the environment and the subject. A good table of the relevant assumptions and their associations was created by Moesland and Granum in [2] and has been recreated in Table 2.1. These assumptions are made to make the human motion capture problem tractable and are applied in varying number and selection. Which assumptions a particular system uses depends upon its goals. The complexity of a system is generally inversely proportional to the number of assumptions that are made. The more complex the system, the few assumptions are made.

2.2 2D Approaches With and Without Explicit Shape Models

One general approach to the analysis of human movement has been to bypass the pose recovery step altogether and to describe human movement in terms of simple low-level, 2D features from region of interest, for example the research done by Polana and Nelson [3]. Models for human bodies are then described in statistical terms derived from low-level features or by simple heuristics. One place this is especially popular is in the applications of hand pose estimation in sign language recognition and gesture-based dialogue management. For applications involving the analysis of hands, the hand is typically extracted using either background subtraction or skin color detection. The extracted features are

Table 2.1
The Typical Assumptions Made by Motion Capture Systems Listed

in Ranked Order According to Frequency from [2]

Assumptions related to movements	Assumptions related to appearance
1. The subject remains inside the workspace 2. None or constant camera movement 3. Only one person in the workspace at a time 4. The subject faces the camera at all times 5. Movements are parallel to the camera-plane 6. No occlusion 7. Slow and continuous movements 8. Only move one or a few limbs 9. The motion pattern of the subject is known 10. Subject moves on a flat ground plane	Environment 1. Constant lighting 2. Static background 3. Uniform background 4. Known camera parameters 5. Special Hardware Subject 1. Known start pose 2. Known subject 3. Markers placed on the subject 4. Special colored clothes 5. Tight-fitting clothes

based on hand shape, movement, and/or location of the interest region. Then x-y image moments and orientation histograms [4], rotationally invariant Zernike moments [5], motion trajectories of hand centroids [6] [7] [8] [9], or some other analysis is applied. Another technique is to superimpose a grid on the interest region, after possible normalization, whereupon in each tile of the grid a simple feature is computed and these features are combined to form a K x K feature vector to describe the state of movement a time *t*. Action classification is the based on hard-coded decision trees [6] [7] [8], nearest neighbor criteria [5] [3], or general pattern matching techniques for time-varying data. Another line of research involves the use of statistical shape models to detect and track the contours of hands or persons. For example, “Active Shape Models” can be used for this purpose [10]. However, the method has some drawbacks including that the features need to be present at all times (no occlusions), a good initial estimate must be available, and the chosen parametrization might include states which have implausible physical interpretations. 2D approaches without explicit shape models are also often used for pedestrian tracking. For example Baumberg and Hogg [11] used B-splines to represent the pedestrians.

Many systems that use explicit *a priori* knowledge of how the human body (or hand) appears in 2D, track and label body parts. Since self-occlusion makes this problem extremely difficult for arbitrary movements, most systems assume a certain type of movement. The human figure is usually extracted from the image using a background subtraction algorithm. A number of researchers have analyzed scenes involving human gait parallel to the image plane. The system then uses one of the following: hierarchical and articulated curve fitting with 2D ellipsoid [12], finding human silhouettes with deformable contours in X-T space [13], finding human silhouettes with deformable surfaces in X-Y-T space [14], obtaining a skeleton of the silhouette and matching it to a model stick figure [15], or detection of ribbons corresponding to the arms and legs to model the motion of the human [16]. Other groups work without any a priori knowledge of the movement being performed. The tracking can then be accomplished through the use of anti-parallel lines (apars) [17] [18]. One can also segment and label the body parts from a silhouette using a basic body model consisting of five U-shaped ribbons and a body trunk, various

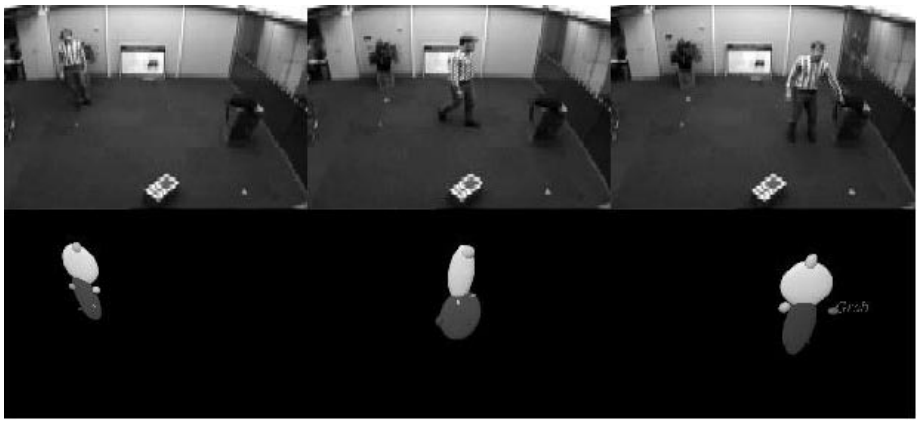


Figure 2.1. Detecting and tracking human “blobs” with the Pfinder system (work by Wren *et al*. [20])

joint and mid-points, plus structural constraints such as support [19]. Wren *et al*. [20] used image segmentation to generate foreground blobs which typically correspond to the person’s hands, head, feet, shirt and pants. See Figure 2.1 for results. Cai and Aggarwal [21] use a simplified head-trunk model to track humans across multiple cameras.

2.3 3D Approaches

In this section we will discuss 3D approaches to human motion tracking. Most of these approaches attempt to recover a 3D articulated pose over time, i.e. joint angles of a human model. Most approaches rely on some *a priori* knowledge about the kinematic and shape properties of the human body to make this problem tractable. The common assumptions that are made are outlined in section 2.1. The use of a 3D shape model can also predict events such as self occlusion and self collision. A general framework of human motion capture is illustrated in Figure 2.2. There are six major components. Initialization includes the actions needed to ensure that the system begins its operation with the correct interpretation of the scene. Initialization can also include the pre-processing of some data and/or the calibration of cameras, etc. The prediction component takes into account previous states up to time t to make a prediction for time $t + 1$. Prediction is usually done at a

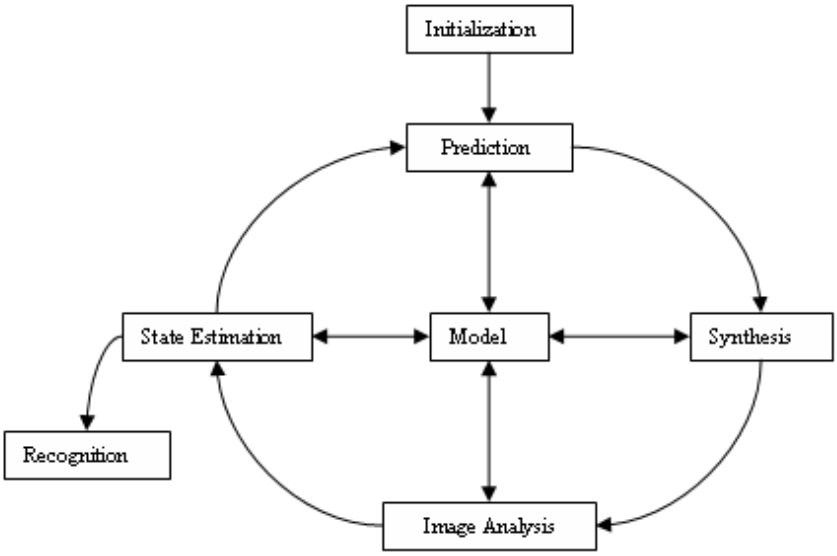


Figure 2.2. Model-based tracking (adapted from Gavrila [1])

high level (in state space) rather than at a low level (in image space). The synthesis component translates the prediction from state level to the measurement (image) level, which allows the image analysis component to selectively focus on a subset of regions and look for a subset of features. The state-estimation component computes the new state using the segmented image and the recognition component identifies a certain posture, gesture, or movement.

2.3.1 3D body modeling

3D graphical models for the human body generally consist of two components: a representation for the skeletal structure (the “stick figure”) and a representation of the flesh around it. The stick figure is simply a collection of segments and joint angles with various degrees of freedom at the articulation sites. Relevant rotations are generally described by their Euler angles [22] [23]. The representation for the flesh can either be surface based (using polygons) or volumetric based (e.g. using cylinders). Most

computer vision systems are concerned with recovering the 3D model from the images and therefore are more likely to use volumetric primitives to flesh out their model because of the lower number of parameters involved. Various models have used overlapping spheres [24], cylindrical primitives [25] [26] [27] [28] [29], and super-quadratics including cylinders, spheres, ellipsoids, and hyper-rectangles [30]. Additional flexibility can be achieved by allowing global deformations and/or local deformations on the super-quadratics [30] [31] [32] [33] [8].

2.3.2 3-D pose recovery and tracking

Some 3D pose recovery systems recover the pose from a monocular image sequence. These systems sometimes use a divide-and-conquer technique where an articulated object is decomposed into a number of primitive sub-parts; one solves for the motion and depth of the sub-parts and verifies whether the parts satisfy the necessary constraints. In order to avoid physically impossible combinations for verification, it is beneficial to propagate constraints from part to part. Some work has used perspective projection models [34] [35] [36]. Another technique for articulated pose recovery is based on a fusion of constraints and measurements using a Kalman filter framework [37]. Other approaches use parametrized models where the articulation constraints are encoded in the representation itself. One advantage to this system is that each representable state represents a physically valid pose, however one needs to work in very high dimensional space. Some systems using such paramtrized models make use of inverse kinematics [38] [26] [39] [40] [41] [42] [36], which can also use multiple cameras when no feature correspondence between cameras is assumed. Another approach using parametrized models does not attempt to invert a non-linear measurement equations. Instead, it uses the measurement equation directly to synthesize the model and uses a fitting measure between synthesized and observed features for feedback [25] [31] [27] [43] [44] [45] [29]. Pose-recovery can then be formulated as a search problem which entails finding the pose parameters of a graphical human model whose synthesized appearance is most similar to the actual appearance of

the real human. Most approaches to the measurement equation exist are based on occluding contours or regions. There has also been work on using depth data for articulated pose recovery [46].

2.3.3 Experimental results

Using a single camera to analyze the whole body some researchers have obtained interesting results. Hogg [27] and Rohr [29] deal with human gaits that are parallel to the image plane. The unconstrained upper-body motion tracking done by Downton and Drouet [25] gets lost due to propagation of errors. Goncalves *et al*. [26] track one arm while keeping the shoulder fixed at a known position. Using multiple cameras, researchers are able to track one arm [32], obtain the 3D locations of the face and hands [46], and Gavrila and Davis [31] show initial results on whole-body tracking using four cameras placed in the corners of the room. Since most of these approaches are working with real data, there has typically been no ground truth established, so the verification is difficult. When using more than one camera, people can at least visually verify the recovered pose along the depth dimension.

2.3.4 Action recognition

The recognition of an action or pose is often done in post processing and is often considered a classification problem based on the information from the pose-recovery step. Recognition often consists of matching an unknown test sequence with a library of labeled sequences. Commonly recognized actions include walking and running, and more advanced actions such as ballet steps have also been studied. Low level recognition is typically based on spatio-temporal data, such as spatio-temporal templates [47] and motion templates [3]. More high level methods are usually based on pose estimated data. Such methods use correlation [48], silhouette matching [49], phase-space representation [22], Dynamic Time Warping (DTW) [50] [51] [52] [53], Hidden Markov Models (HMMs) [54] [9] [55], and neural networks [15] [56]. Static recognition is concerned

with spatial data, one frame at a time. The information may be templates [57], transformed templates [58], normalized silhouettes [59], or postures [60]. The goal of static recognition is mainly to recognize various postures, e.g. pointing [61], standing and sitting [62], or specially defined postures used in interfaces [63] [4]. Such systems sometimes make use of attributes besides 3D pose such as the direction the person is facing, the contents of the room, etc.

2.4 Works with Direct Applications to Our Research

Gavrila and Davis [31] have done research into relatively unconstrained movement. They use four cameras to film the room, one in each corner, which generate four orthogonal views. The subject then faces one camera and two others give sideways views while one give a view from behind. They use a 3D human model which can predict events such as (self) occlusion and (self) collision. They seek to recover the 3D pose through time in terms of 3D joint angels defined with respect to a human centered 3D motion. Then the 3D joint angles can be used as features for movement matching. This project uses a 3D model composed of tapered super-quadratics and has 22 DOF. It has 3 DOF for positioning the root of the articulated structure, 3 DOF for the torso, 4 DOF for each arm and leg, and a rigid torso-head approximation. Their 3D models can be seen in Figure 2.3. The shape parameters are derived from the projections of occluding contours in two orthogonal views. They perform 2D segmentation on the two orthogonal views and then back-project the contours to obtain a set of 3D occluding contours. After that a coarse-to-fine search procedure is used over a reasonable range of parameter space to determine the best-fitting quadric. They use chamfer distance as a fitting metric between the fitted and back-projected occluding 3D contours. Their approach to pose recovery is based on a generate-and-test strategy. By using a multi-view approach, they achieve tighter 3D pose recovery and tracking of the subject than from using one view only. They synthesize the appearance of the human model for all the available views and evaluate the appropriateness of a 3D pose based on the similarity measures for the individual views. Gavrila and

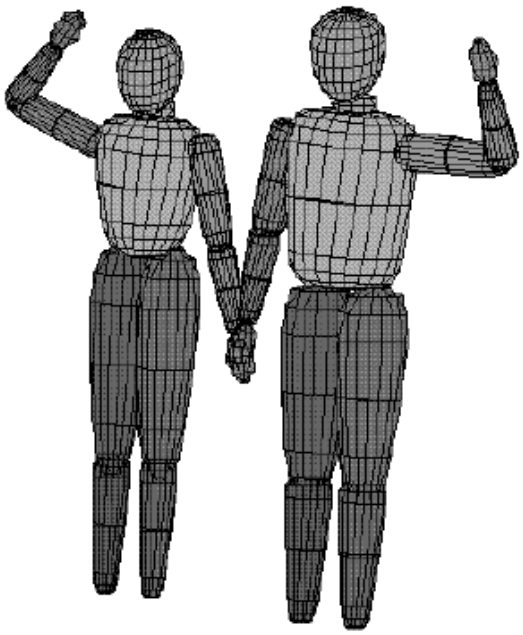


Figure 2.3. 3D models ELLEN and DARIU from [31]

Davis use a best-first search method. They assume that a reasonable initial estimation can be used during tracking or by a boot-strap method at start-up. To reduce the size of the search space, they use something called *search space decomposition*. They search for the head and torso, then the arms and torso twist, then the legs. For each search space, they assume that the best parameter values in the other search spaces are equal to their predicted values. They use a boot-strapping method for initialization. It currently handles the case where the moving objects (i.e. humans) do not overlap and are positioned against a stationary background. The procedure starts with background subtraction followed by a thresholding operation to determine the region of interest. Then using the binary image, the major axis of the region of interest is determined and used to find the 3D axis of the torso. Then the torso twist and arms and legs are searched for in a coarse-to-fine manner. They have applied their program to a Humans in Action database and have very promising results [31].

Carranza et. al., [64], deal with the idea of being able to view a prerecorded video sequence from any arbitrary location. In current visual media only a two dimensional view of the world is presented. The camera positions are unchangeable and determined by the directors. The goal of this project is to bring about a sense of immersion by giving the viewer the freedom to choose his or her viewpoint. They believe that a combination of marker-free motion capture and multi-view texture generation is highly effective for synthesizing novel views of a human in motion. The input to the system consists of a series of calibrated cameras which are located at approximately even intervals around a room. The silhouette is then extracted using a background subtraction algorithm. The human body model they are using is a publicly available VRML model, which is then adapted to the individual actors during the initialization phase. The generic model consists of 16 rigid body segments (head, upper arm, torso, etc.) and the model's kinematics are defined via an underlying skeleton consisting of 17 joints connecting bone segments with 35 parameters needed to completely define the body's pose. In order to accommodate a wide range of physical body types, they allow for deformation of each body segment. The motion capture subsystem tracks the body motion of the recorded actor over time. After an initialization step, the body pose parameters that maximize the overlap between projected model silhouettes and input camera silhouettes are estimated for every time step. The error metric that they use is a pixel-wise exclusive-or between the image silhouette and the rendered model silhouette in each input camera view. For initialization a global model position is computed by using a grid sampling of the parameter space. The fit is improved by optimizing over the pose parameters and joint scaling parameters in an iterative process that uses the same error metric. The optimization is carried out on the torso, then the head and hip joints, then the arms and legs, and then the hands and feet. After motion capture and pose calculation, the model has surface textures applied to it at each time point. After these calculations are performed, the view-dependent images can be calculated and rendered in real time. The system has been tested including by a ballet dancer and someone doing karate and it successfully tracked human motion involving fast arm movements, complex twisted poses of the extremities and full body turns.

Mangor and Theobalt [65] employ background subtraction for silhouette extraction and an exclusive-or as an error metric for model fitting. The model is initially fitted to the subject and then optimized in a hierarchical manner to calculate the pose at a given time. They use the VRML model discussed in [64]. They propose that the optimization algorithm be parallelized such that after the torso parameters are determined, the four extremities and the head could be calculated by different computers. They do this by windowing around each body part's approximate location and only sending that data to the PC doing the calculations. To reduce the influence of other body parts during separate body limb optimization, the entire geometry model is rendered once without the body segment in question, and all covered silhouette pixels are discarded. This parallelization increases the speed by between 4 and 7 times.

Zimmermann and Svoboda propose a method with a weak model which they then optimize over the entire sequence. Their model is composed of ellipsoids. They optimize over two types of human body model parameters: shape (e.g. length of the arm) and motion (e.g. the angle between the upper and lower arm). They assume that the shape parameters are constant throughout the sequence. The human body model parameters are represented by $\mathbf{\Theta}$ = { $\mathbf{m}_1$, $\mathbf{m}_2$, ..., $\mathbf{m}_n$, $\mathbf{s}$}, where $\mathbf{m}_i$ represents all of the motion parameters at time *i* and **s** represents the shape parameters. A multi-view sequence of images is represented by $\mathbf{Z}$ =$\{\mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_n\}$ where $\mathbf{z}_i$ is the set of ALL images from ALL cameras at time *i*. $\mathbf{z}_i$ is referred to as a multi-view frame and **Z** represents the entire multi-view sequence. If the shape parameters, **s**, are known, then estimation of motion parameters $\mathbf{m}_i$ in the frame *i* is based on posterior probability maximization. The posterior is calculated from the projection of the ellipsoidal model onto the cameras. The probability of parameters, given the multi-view frame $\mathbf{z}_i$, is inversely proportional to the sum of the distances between the borders of the silhouettes (i.e. segmented images) and the projected silhouettes. The goal is to find $\mathbf{\Theta}$ which maximizes $p(\mathbf{\Theta}\,|Z)$, our posterior probability. Since optimization over all of $\mathbf{\Theta}$ is technically infeasible, Zimmermann and Svoboda propose decomposing the sequence posterior into a multiplicative form that can be more easily optimized. They model the motion as a Markov process, therefore the motion parameter vector $\mathbf{m}_i$ at time

i is considered to only be dependent on the multi-view frame $\mathbf{z}_i$, the shape parameter **s**, and the previous motion parameter vector $\mathbf{m}_{i-1}$. This allows the sequence posterior to be expressed as

$$p(\mathbf{\Theta} \mid \mathbf{Z}) = p(\mathbf{m}_1 \mid \mathbf{s}, \mathbf{z}_1) \cdot p(\mathbf{m}_k, \mathbf{s} \mid \mathbf{z}_k) \cdot \prod_{i \neq k} p(\mathbf{m}_i \mid \mathbf{m}_{i-1}, \mathbf{s}, \mathbf{z}_i)$$

where $\mathbf{z}_k$ is a key frame. The algorithm can then optimize over $p(\mathbf{m}_k, \mathbf{s} \mid \mathbf{z}_k)$ to find **s**. Then given the shape parameters **s**, they can calculate the motion parameters for the entire sequence. Let us call $p(\mathbf{m}_i \mid \mathbf{m}_{i-1}, \mathbf{s}, \mathbf{z}_i)$ the frame posterior probabilities. Zimmerman and Svoboda have shown (by repeated applications of Bayes rule) that the frame posterior can be simplified as

$$p(\mathbf{m}_i \mid \mathbf{m}_{i-1}, \mathbf{s}, \mathbf{z}_i) = c \cdot p(\mathbf{z}_i \mid \mathbf{m}_i, \mathbf{s}) \cdot p(\mathbf{m}_i \mid \mathbf{m}_{i-1})$$

where c is a normalization constant that can be ignored in implementation. The likelihood, $p(\mathbf{z}_i \mid \mathbf{m}_i, \mathbf{s})$ can be computed by projecting the body model into the camera image planes and comparing the projections with the actual images. The transition probabilities $p(\mathbf{m}_i \mid \mathbf{m}_{i-1})$ represent our prior knowledge about how the human body moves and how the joint parameters can change from one time to the next. They also show that

$$p(\mathbf{m}_k, \mathbf{s} \mid \mathbf{z}_k) = c \cdot p(\mathbf{z}_k \mid \mathbf{m}_k, \mathbf{s}) \cdot p(\mathbf{s})$$

Zimmermann and Svoboda have shown good results in tracking a person as they walk through a room [66].

2.5 Alternative Approaches

Plänkers and Fua [67] did research into articulated soft objects. They believe that the use of cylinders or ellipsoids attached to an articulated skeleton is too oversimplified; they are too crude for precise recovery of both shape and motion. Instead they propose a framework which maintains the articulated skeleton, but replaces the simple geometric primitives with soft objects where each primitive defines a field function and the skin is taken

to be a level set of the sum of these fields. A simple representation of the process can be seen in Figure 2.4. Plänkers and Fua believe that implicit surface formulation has the following advantages: effective use of stereo and silhouette data, accurate shape description from a small number of parameters, and explicit modeling of 3D geometry. The human body model used here has 230 metaballs. In order to prevent body parts from blending into each other, the body is segmented into 10 distinct parts: upper and lower torso, upper and lower arms, and upper and lower legs. When computing the implicit surface, only the metaballs belonging to the same segment are taken into account. Body shape and position are controlled by a *state vector* Θ, which is the set of parameters defining joint locations and limb sizes. These are variable by subject, but are taken to be constant across a sequence. The expected output of the system is the instantiated state vector Θ. This is a highly nonlinear problem because the model consists of an articulated set of implicit surfaces. To solve this problem, they use the Levenberg-Marquart least-squares estimator to minimize the distance between observations and model. For initialization, they choose one frame in the sequence and the click on approximate locations of certain key joints in two images. This gives a rough scaling of the skeleton and an approximate model pose. The system then does frame-to-frame tracking and global fitting. For a given time step, the system extracts clouds of 3D points from their synchronized input video sequences using a correlation-based approach. The tracking process then adjusts the model's joint angles by using the least squares estimator previously described. The results from the tracking in all frames then serve as initialization for global fitting to refine the postures in all frames and to adjust the skeleton and/or metaball parameters to improve the model's shape. In global fitting they also use an objective function that favors constant angular speeds to help avoid local minima. The authors then use a trinocular video sequence of complex 3D motions to test their system. The system performs well and is able to correct when it has errors.

Mikic et. al. [68] propose the use of voxels to perform human motion tracking. The main components of their system are the 3D voxel reconstruction, model acquisition, and motion tracking. The 3D voxel reconstruction takes multiple synchronized and segmented video streams and computes the reconstruction of the shape represented by the foreground

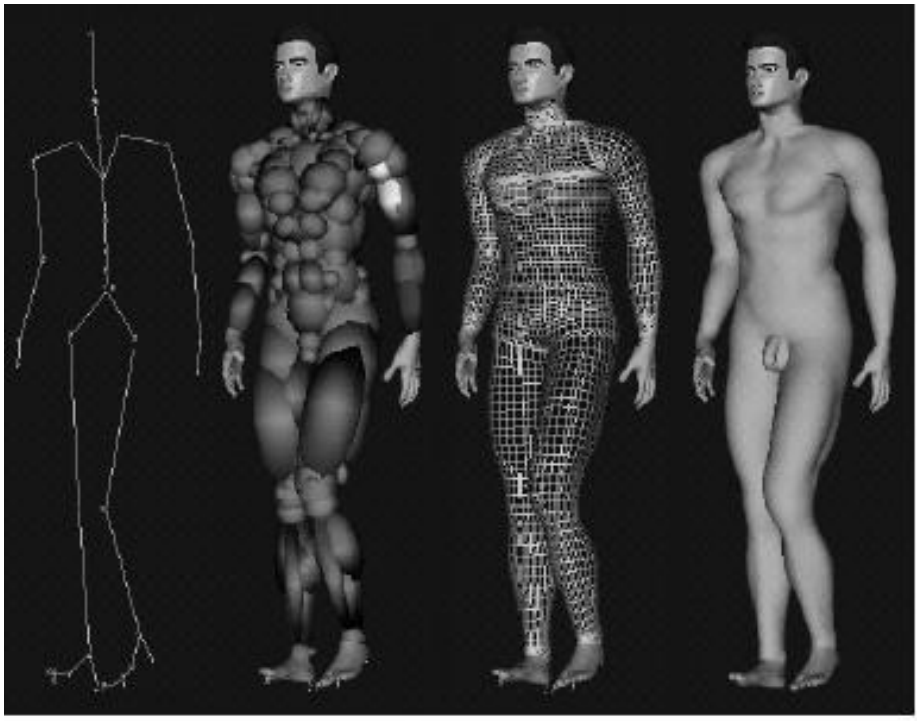


Figure 2.4. Articulated Soft Objects (a) Skeleton. (b) Volumetric primitives used to simulate muscles and fat tissue. (c) Polygonal surface representation of the skin. (d) Shaded Rendering. From [67].

pixels. In the first frame of a sequence, initial model acquisition is performed by using a simple template fitting and growing procedure to find different body parts. This procedure takes advantage of the prior knowledge of average sizes and shapes of different body parts. Then the model is refined using a Bayesian network that incorporates the knowledge of human body proportions into the estimates of body part sizes. The tracking procedure uses a predict and update method at each time. Using the prediction of the model position and configuration from the previous frame, the voxels in the new frame are assigned to one of the body parts. Measurements of location of specific points on the body are extracted from the labeled voxels and a Kalman filter is used to adjust the model position and configuration to best fit the extracted measurements. The voxel labeling procedure combines template fitting and distance minimizing approaches. This algorithm can handle large frame-to-frame displacements and results in robust tracking performance. The human body model is made up of ellipsoids and cylinders. The body model consists of five open kinematic chains: torso-head, torso-left arm, torso-right arm, torso-left leg, and torso-right leg. It is described using the twists framework developed in the robotics community. At initialization, the body model is modified to fit the current subject. The constraints that ensure kinematically valid postures which are allowed by the degrees of freedom in the joints and the connectedness between specific body parts are incorporated into the model. Voxel carving is performed using an octree. At each level a cube is tested to see if it lies entirely inside a silhouette or entirely outside a silhouette. If so, it is labeled as such, if not it subdivides further. Since this method wants to be able to use non-static cameras, they precompute a lookup table that pas points from undistorted sensor-plane coordinates to the image pixels. Then, the only computation that is performed at run-time is mapping from world coordinates to undistorted sensor-plane coordinates. A voxel is uniformally sampled and included in the 3D reconstruction if the majority of the sample points agree with all image silhouettes. A large disadvantage to this approach is that it requires a large number of cameras in order to be accurate.

3. THE ALGORITHM

3.1 Background Subtraction

The room that we are working in has a total of 12 fixed cameras. These cameras have been calibrated and the calibration information collected. They are configured such that most of the room has at least three cameras that can see it at all times, see Figures 3.1 and 3.2 for a model of the room with all of the cameras. This overlap of cameras gives us the ability to view a person from multiple directions, thereby allowing us to better calculate the joint angles. The more cameras that we have, the better we can predict the joint angles. At least two points in different 2D spaces are required in order to triangulate the 3D location of any point in space, so to do motion tracking, we require at least two views of the body part that we are tracking. In order to ensure that each body part that is being tracked is viewed by at least two cameras, a minimum of three cameras is employed.

The background subtraction algorithm assumes that there is a static background. To initialize the background subtraction algorithm, we determine the average RGB value for each pixel for each camera when the room is empty. Then, we are able to calculate the difference between the current image and the background image. A simple background subtraction algorithm classifies pixels as foreground if the difference between the RGB values of the background pixels and the current pixels exceeds a given threshold. In order to reduce the number of shadows that are calculated as foreground pixels, as seen in Figure 3.3, we use the normalized RGB values for the background pixel and the test pixel. Let

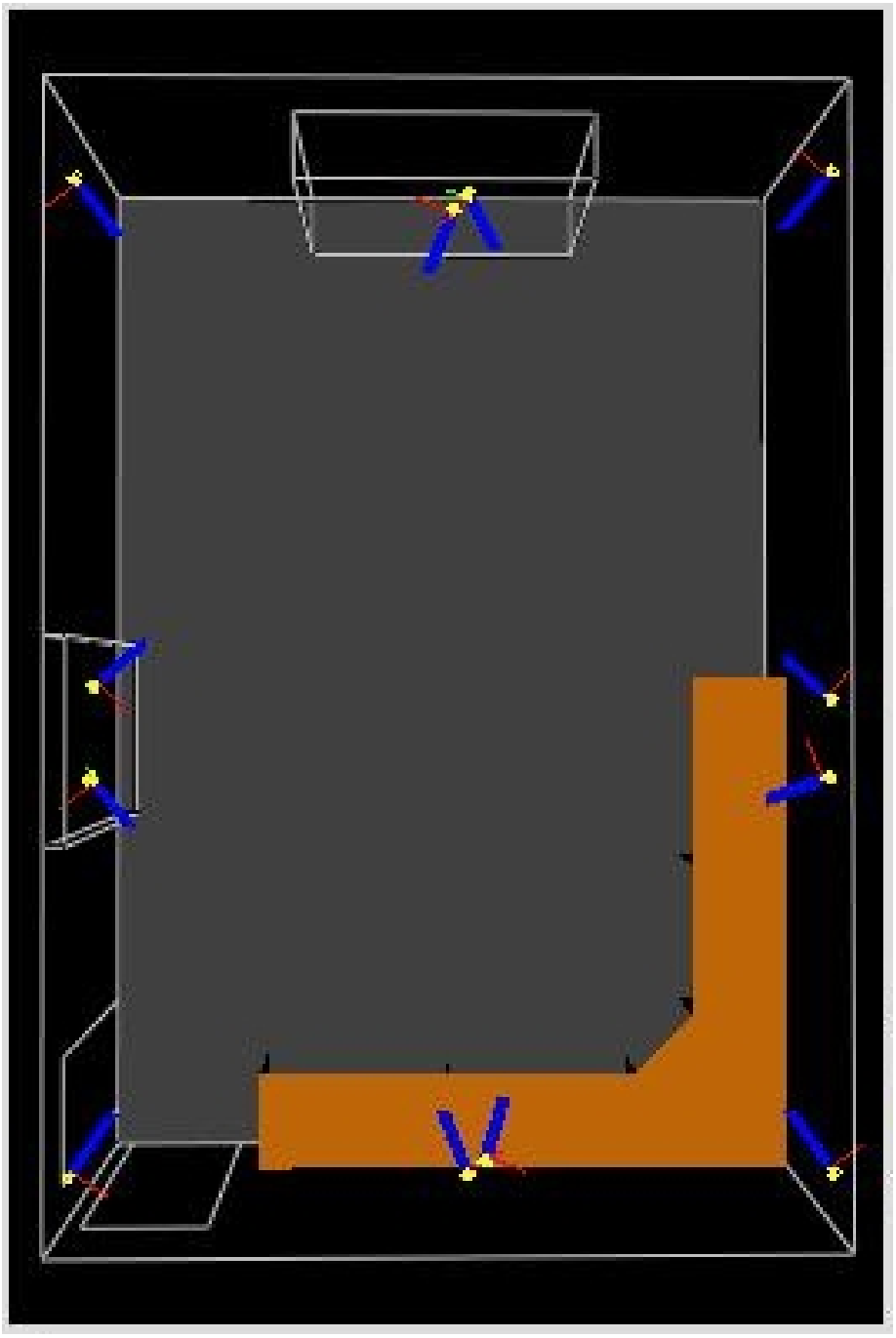


Figure 3.1. Room Model with Cameras Top View

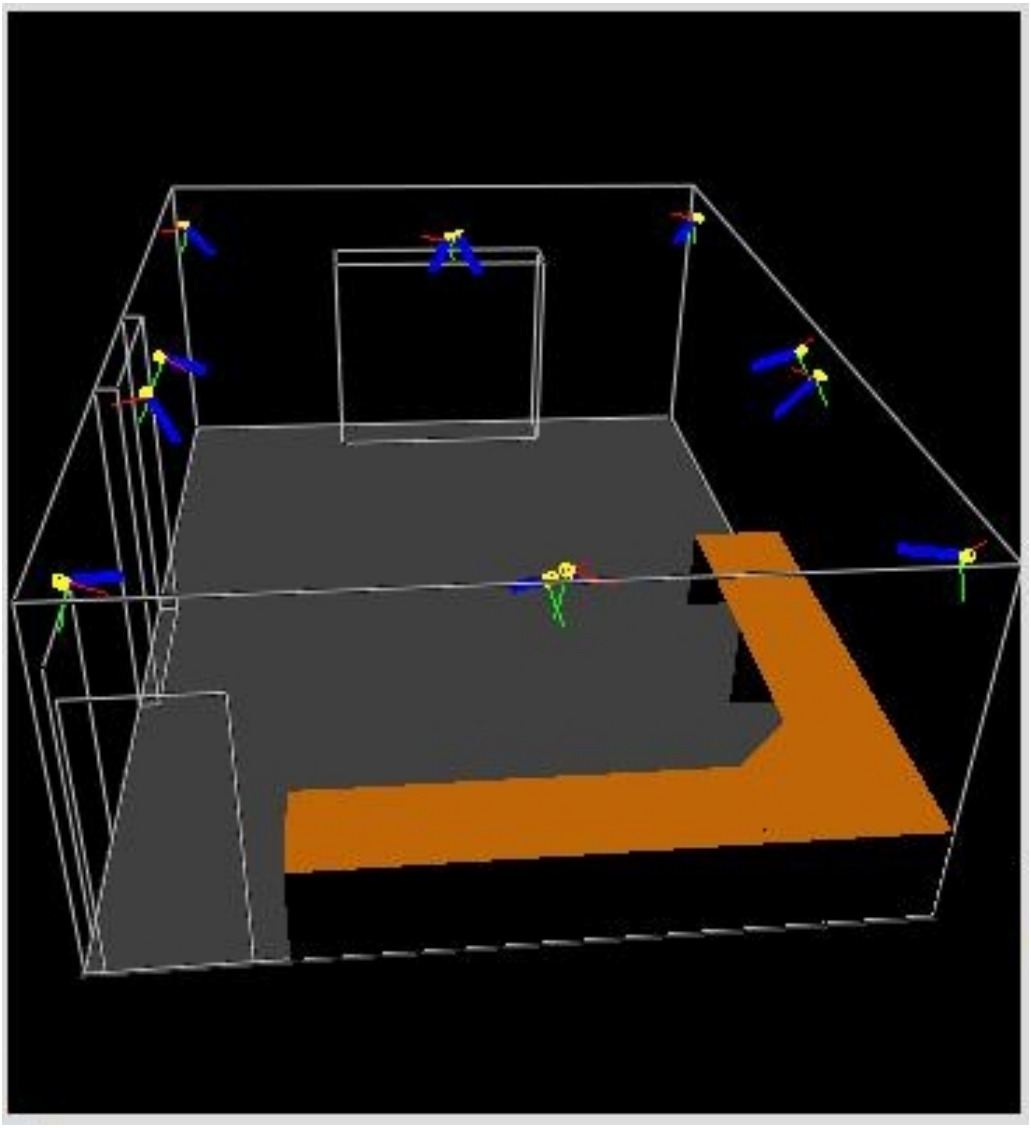


Figure 3.2. Room Model with Cameras Side View

Background Color Image with Subject Binary Background Subtraction

Figure 3.3. Original Background Subtraction Example

Background Color Image with Subject Binary Background Subtraction

Figure 3.4. Current Background Subtraction Example

NRback, NGback, and NBback be the normalized RGB values of the background pixel and let NRtest, NGtest, and NBtest be the normalized values of the test pixel.

$$\begin{aligned} if \quad & (NRback - NRtest)^2 + (NGback - NGtest)^2 + (NBback - NBtest)^2 \\ & > threshold\, then\, foreground \\ else\, if \quad & (NRback - NRtest)^2 + (NGback - NGtest)^2 + (NBback - NBtest)^2 \\ & < threshold\, then\, background \end{aligned}$$

The background subtraction algorithm generates a binary image in which the foreground pixels are white and the background pixels are black. This process is illustrated in Figure 3.4. Additionally, we have determined that our foreground blobs should exceed an experimentally determined area and the contours should be at least an experimentally determined length length. This eliminates much of the noise that is also seen in Figure 3.3. Additionally we now remove the radial distortion from the images prior to background subtraction.

3.2 3D Model and Binary Image Generation

In order to be able to determine the likelihood of a set of joint angles, we need a 3D human model. We have created an articulated 3D model using FLTK, a GUI toolkit for C++ [69]. The model is composed of tapered cylinders for each of the rigid parts of the body and spheres at the joints, as seen in Figure 3.5. The model currently has 25 degrees of freedom (DOF). The first three DOF represent the location of the model, that is its XYZ position in the world coordinate frame, as seen in Figure 3.6. The next three DOF represent the orientation of the entire model. The entire model can be rotated about the x-, y- and z-axis whose coordinate frame is located at the center of the model. Rotation about the x-axis is shown in Figure 3.7. Each of the joints has its own coordinate frame about which the attached body part(s) are rotated. Figures 3.8 through 3.10 show the individual coordinate axis at each joint. The red line is the x-axis, the green line is the y-axis, and the blue line is the z-axis. As the shoulder and hip joints rotate, the coordinate axis at the elbow and knee rotate with them. Each arm has three DOF at the shoulder, rotation

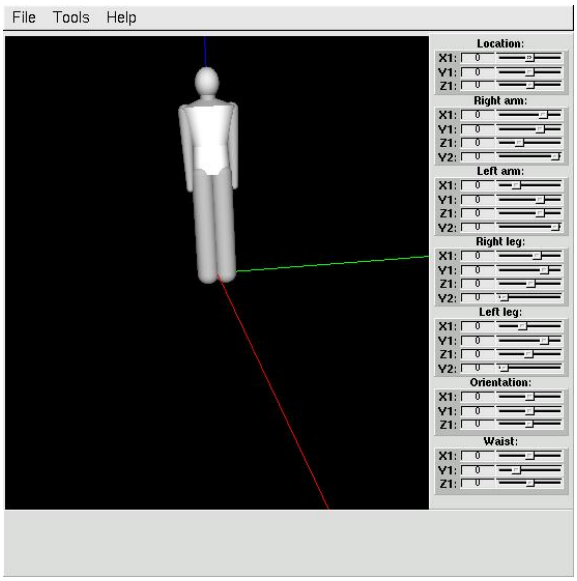


Figure 3.5. Human Model

about the x-, y-, and z-axis, and one DOF at the elbow, rotation about the y-axis. Figure 3.8 illustrates the rotation of the model's right arm about the x- and y-axis at the shoulder. Figure 3.9 illustrates rotation of the model's arms about the y-axis at the elbows and has the right shoulder also rotated about the z-axis. The legs work similarly to the arms and each has a total of 4 DOF. The torso can also be rotated about the x, y and z-axis, allowing the model to twist and bend at the waist as seen in Figure 3.10.

The 3D model and all of the joint angles are used to generate binary images for comparison with the binary background subtraction images from the cameras. When the cameras are calibrated, certain data is collected that describe the camera. This data includes both extrinsic parameters, like the location and orientation of the camera in the world coordinate system, and intrinsic parameters, like the focal length and field of view of the camera. Our algorithm uses this data to generate binary images of the 3D model from the point of view of each camera.

The location and orientation of the camera in the world coordinate system can be described by a 4x4 transformation matrix $\mathbf{M}$. The matrix ${}^{w}\mathbf{M}_{c}$ describes the transformation

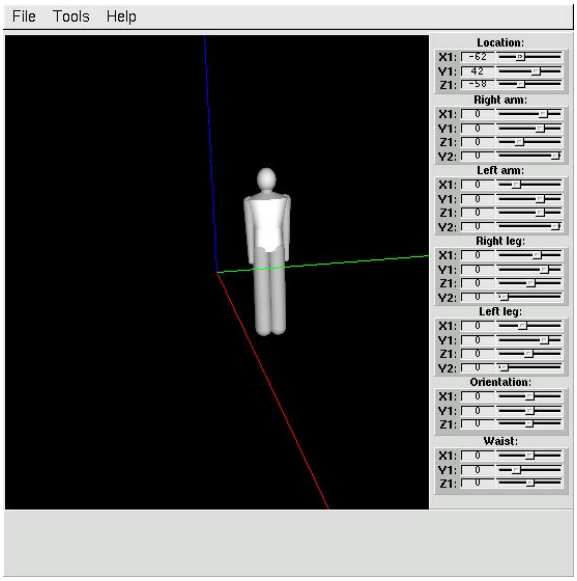


Figure 3.6. Human Model Change in Location

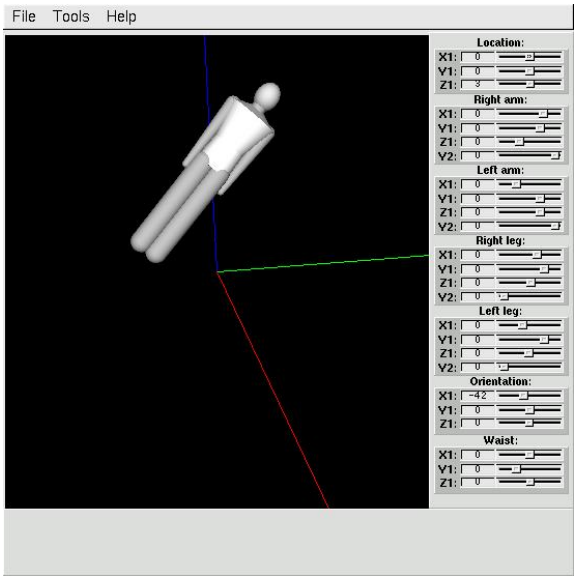


Figure 3.7. Human Model Rotated about the x-axis

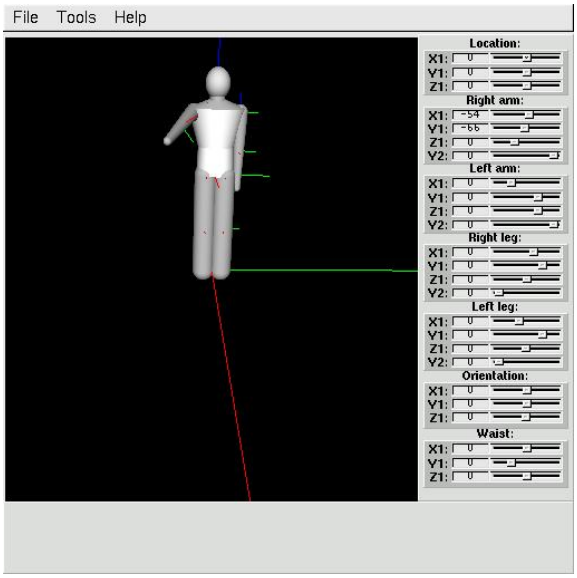


Figure 3.8. Human Model with Shoulder Rotated about the x- and y-axis

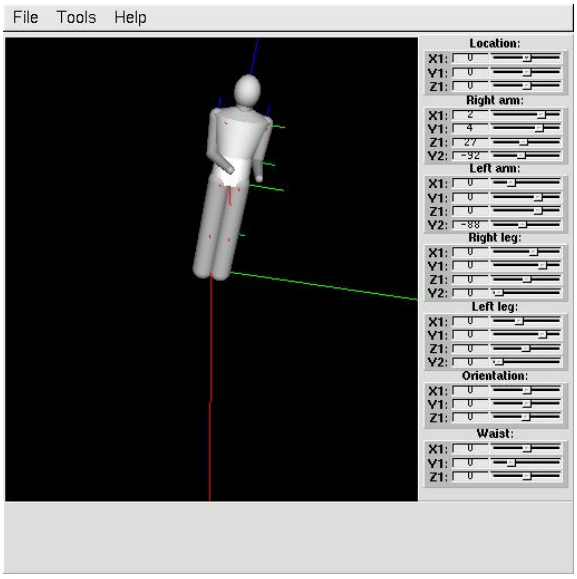
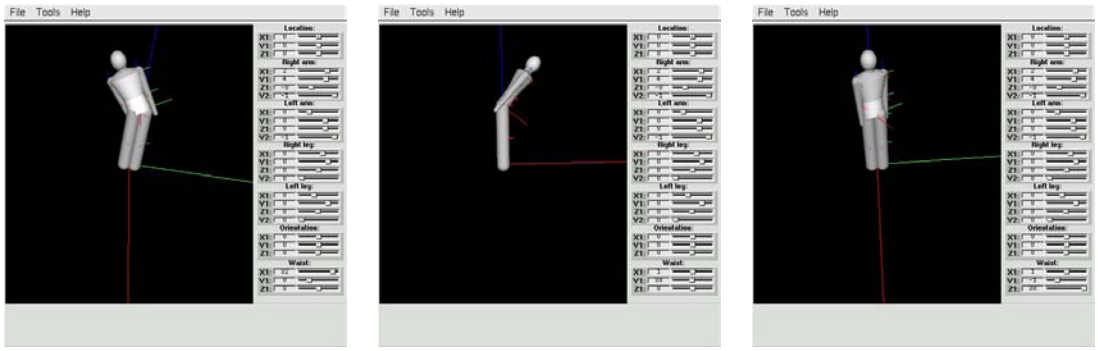


Figure 3.9. Human Model with Both Elbows Rotated about the y-axis and the Right Shoulder Rotated about the z-axis



(a) (b) (c)

Figure 3.10. Human Model with Rotation about the (a) x-axis, (b) y-axis, and (c) z-axis

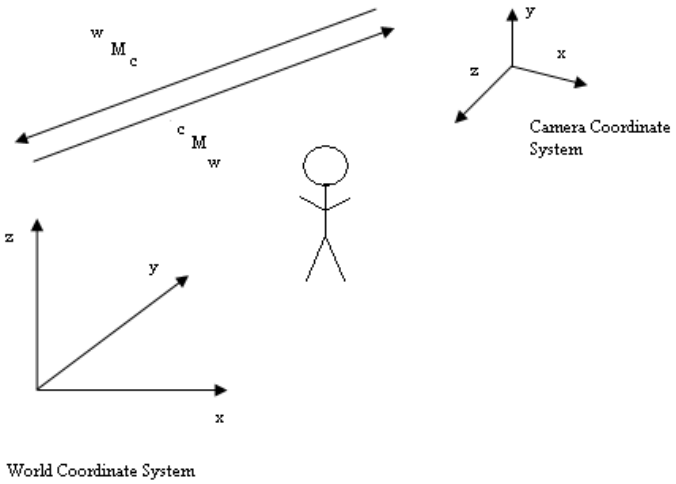


Figure 3.11. Coordinate Systems

from the world coordinate system to the cameras coordinate system, and the matrix ${}^{c}\mathbf{M}_{w}$ describes the transformation from the camera coordinate system to the world coordinate system, as seen in Figure 3.11. If **v** is a homogeneous coordinate representation of a vertex and **M** is a transformation matrix, then **Mv** is the vertex under transformation. This transformation is applied to all of the points on the 3D human model. The matrix **M** can be broken down into two parts. The upper left 3x3 matrix is also called the rotation matrix **R**. This matrix specifies which direction the camera is facing in the world coordinate system. It can be used to represent rotations about the x, y, and z-axis. For example if the camera were rotated about the world x-axis by θ then rotated about the world y-axis by $\phi$and then rotated about the world z-axis by ψ, the following ${}^{w}\mathbf{M}_{c}$ matrix would be generated:

$$\begin{bmatrix} cos\psi cos\phi + sin\psi sin\theta sin\phi & sin\psi cos\phi - cos\psi sin\theta sin\phi & cos\theta sin\phi & 0 \\ -sin\psi cos\theta & cos\psi cos\theta & sin\theta & 0 \\ sin\psi sin\theta cos\phi - cos\psi sin\phi & -cos\psi sin\theta cos\phi - sin\psi sin\phi & cos\theta cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.1}$$

The right column specifies the location of the camera in the world coordinates. It is known as the translation vector **T**. If the camera is at the world coordinate location (t_x, t_y, t_z) the following transformation matrix, ${}^{w}\mathbf{M}_{c}$, is generated:

$$\begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

If the camera is located at (t_x, t_y, t_z) and the orientation is described by the matrix 3.1, then each of the points on the 3D model is multiplied by the following matrix:

$$\begin{bmatrix} cos\psi cos\phi + sin\psi sin\theta sin\phi & sin\psi cos\phi - cos\psi sin\theta \cdot sin\phi & cos\theta sin\phi & t_x \\ -sin\psi cos\theta & cos\psi cos\theta & sin\theta & t_y \\ sin\psi sin\theta cos\phi - cos\psi sin\phi & -cos\psi sin\theta cos\phi - sin\psi sin\phi & cos\theta cos\phi & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Each of the values in this matrix is determined when the camera is calibrated. We then use that information to define a viewpoint and viewing direction in the 3D space that the

Figure 3.12. Generated Binary Image from Camera 200

human model is in. We also obtain certain intrinsic parameters having to do with the camera's focal length and field of view and use them to calculate the field of view that our simulated camera should have. Using all of this information, we are able to specify the view of the camera and generate a binary image of what the camera would see if it were looking into the simulated scene. For example, Figure 3.12 shows a binary view of the model as it is viewed by camera 200.

3.3 A Probabiliistic Framework for Estimating Pose Parameters

The goal of this project is to calculate the most likely set of pose parameters (the 25 degrees of freedom as explained in Section 3.2) given the segmented images obtained from the cameras as explained in Section 3.1. The pose parameters are denoted as $\mathrm{M} = (\mathbf{m}_1, \mathbf{m}_2, ..., \mathbf{m}_n)$ where $\mathbf{m}_i$ is the set of joint angles and other degrees of freedom at time *i*. Unlike the work done by Zimmerman and Svoboda [66], our algorithm assumes that we know the shape parameters ahead of time. The images from Section 3.1 are called a multiview segmented sequence of images denoted by $\mathbf{Z} = (\mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_n)$ where $\mathbf{z}_i$ denotes all of the images from all of the cameras at time *i*. In order to calcu-

late the most likely set of motion parameters, we want to optimize the sequence posterior probabilty, $p(\mathbf{M} \mid \mathbf{Z}) = p(\mathbf{m}_1, \mathbf{m}_2, ..., \mathbf{m}_n \mid \mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_n)$. The Chain Rule says $p(A, B, C) = p(C|A, B)p(B|A)p(A)$, so

$$\begin{aligned} p(\mathbf{M}, \mathbf{Z}) &= p(\mathbf{m}_n \mid \mathbf{m}_1, ..., \mathbf{m}_{n-1}, \mathbf{Z}) \cdot p(\mathbf{m}_1, ..., \mathbf{m}_{n-1}, \mathbf{Z}) \\ &= p(\mathbf{m}_n \mid \mathbf{m}_1, ..., \mathbf{m}_{n-1}, \mathbf{Z}) \cdot p(\mathbf{m}_{n-1} \mid \mathbf{m}_1, ..., \mathbf{m}_{n-2}, \mathbf{Z}) \cdot \\ &\qquad \cdots \cdot p(\mathbf{m}_2 \mid \mathbf{m}_1, \mathbf{Z}) \cdot p(\mathbf{m}_1 \mid \mathbf{Z}) \end{aligned} \tag{3.2}$$

We assume that human motion is a Markov process, the motion parameters $\mathbf{m}_i$ in frame *i* are considered to be dependent only on multiview frame $\mathbf{z}_i$ and preceding motion parameters $\mathbf{m}_{i-1}$. Therefore

$$p(\mathbf{m}_i \mid \mathbf{m}_1, ..., \mathbf{m}_{i-1}, \mathbf{Z}) = p(\mathbf{m}_i \mid \mathbf{m}_{i-1}, \mathbf{z}_i) \tag{3.3}$$

By applying the chain rule, Baye's rule and assuming that human motion is a Markov process we get

$$p(\mathbf{M} \mid \mathbf{Z}) = \frac{p(\mathbf{M}, \mathbf{Z})}{p(\mathbf{Z})} = \frac{p(\mathbf{m}_1 \mid \mathbf{z}_1) \prod_{i>1} p(\mathbf{m}_i \mid \mathbf{m}_{i-1}, \mathbf{z}_i)}{p(\mathbf{Z})} \tag{3.4}$$

By the chain rule

$$p(\mathbf{m}_i, \mathbf{m}_{i-1}, \mathbf{z}_i) = p(\mathbf{m}_i \mid \mathbf{m}_{i-1}, \mathbf{z}_i)p(\mathbf{m}_{i-1} \mid \mathbf{z}_i)p(\mathbf{z}_i) \tag{3.5}$$

and

$$p(\mathbf{m}_i, \mathbf{m}_{i-1}, \mathbf{z}_i) = p(\mathbf{z}_i \mid \mathbf{m}_i, \mathbf{m}_{i-1})p(\mathbf{m}_i \mid \mathbf{m}_{i-1})p(\mathbf{m}_{i-1}) \tag{3.6}$$

Therefore by combining equations 3.5 and 3.6 we have

$$p(\mathbf{m}_i \mid \mathbf{m}_{i-1}, \mathbf{z}_i) = \frac{p(\mathbf{z}_i \mid \mathbf{m}_i, \mathbf{m}_{i-1})p(\mathbf{m}_i \mid \mathbf{m}_{i-1})p(\mathbf{m}_{i-1})}{p(\mathbf{m}_{i-1} \mid \mathbf{z}_i)p(\mathbf{z}_i)} \tag{3.7}$$

Since the multiview image $\mathbf{z}_i$ is constant, $p(\mathbf{z}_i)$ is constant. $\mathbf{m}_{i-1}$ is independent of $\mathbf{z}_i$ therefore $p(\mathbf{m}_{i-1}|\mathbf{z}_i) = p(\mathbf{m}_{i-1})$ and $p(\mathbf{z}_i \mid \mathbf{m}_i, \mathbf{m}_{i-1}) = p(\mathbf{z}_i \mid \mathbf{m}_i)$. Therefore equation 3.7 reduces to

$$p(\mathbf{m}_i \mid \mathbf{m}_{i-1}, \mathbf{z}_i) \sim \frac{p(\mathbf{z}_i \mid \mathbf{m}_i)p(\mathbf{m}_i \mid \mathbf{m}_{i-1})p(\mathbf{m}_{i-1})}{p(\mathbf{m}_{i-1})} \tag{3.8}$$

which further reduces to

$$p(\mathbf{m}_i \mid \mathbf{m}_{i-1}, \mathbf{z}_i) \sim p(\mathbf{z}_i \mid \mathbf{m}_i)p(\mathbf{m}_i \mid \mathbf{m}_{i-1}) \tag{3.9}$$

Since optimizing $p(\mathbf{M} \mid \mathbf{Z})$ is techinically intractable, we divide the problem into tractable subproblems. As shown above $p(\mathbf{M} \mid \mathbf{Z})$ is the product of each $p(\mathbf{m}_i \mid \mathbf{m}_{i-1}, \mathbf{z}_i)$, therefore optimizing each $p(\mathbf{m}_i \mid \mathbf{m}_{i-1}, \mathbf{z}_i)$ gives a near global optimization of $p(\mathbf{M} \mid \mathbf{Z})$ according to Zimmermann and Svoboda [66]. By equation 3.9, optimizing $p(\mathbf{z}_i \mid \mathbf{m}_i)$ and $p(\mathbf{m}_i \mid \mathbf{m}_{i-1})$ optimizes $p(\mathbf{m}_i \mid \mathbf{m}_{i-1}, \mathbf{z}_i)$. Therefore we need to optimize $p(\mathbf{z}_i \mid \mathbf{m}_i)$and $p(\mathbf{m}_i \mid \mathbf{m}_{i-1})$.

$p(\mathbf{z}_i \mid \mathbf{m}_i)$is the probability of a multiview segmented image given a set of motion parameters. This can be interpreted as the probability of a multiview image given a generated set of multiview images which are created according to the process explained in Section 3.2. Let $\mathbf{z}_{ij}$ be the image from camera *j* at time *i* and $\mathbf{y}_{ij}$ be the synthetic image from camera *j* at time *i*. For each camera *j*, we calculate the chamfer distance between the contours in $\mathbf{z}_{ij}$ and $\mathbf{y}_{ij}$ and the normalized XOR between the images $\mathbf{z}_{ij}$ and $\mathbf{y}_{ij}$. The chamfer distance is a way of describing the similarity between two contours, which may or may not be the same size or shape. Let $P_{ij} = (p_1, p_2, \ldots, p_n)$ be the points on the contour of the background subtraction $\mathbf{z}_{ij}$ and let $Q_{ij} = (q_1, q_2, \ldots, q_m)$ be the points on the contour of $\mathbf{y}_{ij}$. Let $\|\cdot,\cdot\|$ be the Euclidean distance between two points. The chamfer distance is

$$\sum_{k=1}^{n} min(\| p_k, q_1 \|, \| p_k, q_2 \|, \ldots, \| p_k, q_m \|) \tag{3.10}$$

The normalized chamfer distance is

$$\frac{\sum_{k=1}^{n} min(\| p_k, q_1 \|, \| p_k, q_2 \|, \ldots, \| p_k, q_m \|)}{n} \tag{3.11}$$

This is illustrated in Figure 3.13. The probability of a segmented image given a set of motion parameters based on chamfer distances is then calculated by taking one minus the normalized chamfer distance. The smaller the chamfer distance, the larger the probability. For each estimated $\mathbf{m}_i$, there exists a different estimated set of $\mathbf{y}_i$, which gives a different set of Q_{ij} for each $\mathbf{m}_i$. However, $\mathbf{z}_i$ is constant and not dependent upon $\mathbf{m}_i$. Therefore the

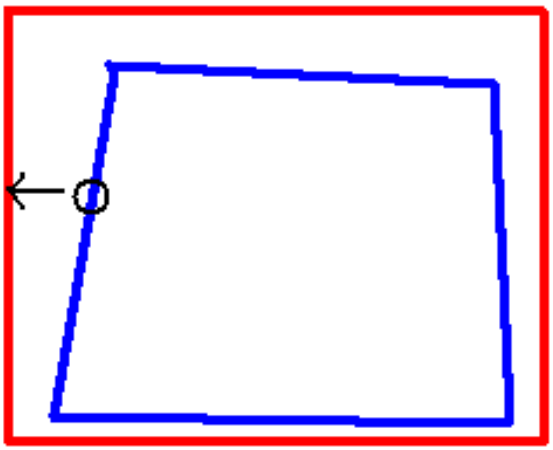


Figure 3.13. Chamfer Distance

set of P_{ij} is constant for time *i* which means that *n* is constant for each $\mathbf{z}_{ij}$. This means that

$$\begin{aligned} \frac{p(\mathbf{z}_i|\mathbf{m}_{ia})}{p(\mathbf{z}_i|\mathbf{m}_{ib})} &= \frac{1-normChamfer(\mathbf{z}_i,\mathbf{m}_{ia})}{1-normChamfer(\mathbf{z}_i,\mathbf{m}_{ib})} \\ &= \frac{n-n\cdot chamfer(\mathbf{z}_i,\mathbf{m}_{ia})}{n-n\cdot chamfer(\mathbf{z}_i,\mathbf{m}_{ib})} \\ &= \frac{1-chamfer(\mathbf{z}_i,\mathbf{m}_{ia})}{1-chamfer(\mathbf{z}_i,\mathbf{m}_{ib})} \end{aligned} \tag{3.12}$$

where $\mathbf{m}_{ia}$ and $\mathbf{m}_{ib}$ are two different estimates for $\mathbf{m}_i$. Equation 3.12 implies that since the ratio of the probabilities of two estimates for $\mathbf{m}_i$ is equal to the ratio of one minus the chamfer distances between the two estimates, that to maximize the probability is the same as minimizing the chamfer distance. To make the algorithm run faster, it seeks to minimize the chamfer distance between $\mathbf{z}_{ij}$ and $\mathbf{y}_{ij}$, rather than calculating and maximizing the probabilities. The probability of a $\mathbf{z}_{ij}$ given $\mathbf{y}_{ij}$ based on the normalized XOR function is equal to the number of pixels that are different between the two images and divided by the number of foreground pixels in $\mathbf{z}_{ij}$.

$p(\mathbf{m}_i|\mathbf{m}_{i-1})$is the probability of a set of motion parameters at time *i* given the set of motion parameters at time *i - 1*. This algorithm takes this to mean the probability that a set of motion parameters at time *i* is physically possible given a set of motion parameters at time *i - 1*. For rotation about each joint it determines whether or not the position is physically possible and whether they could have moved from the previous position into the

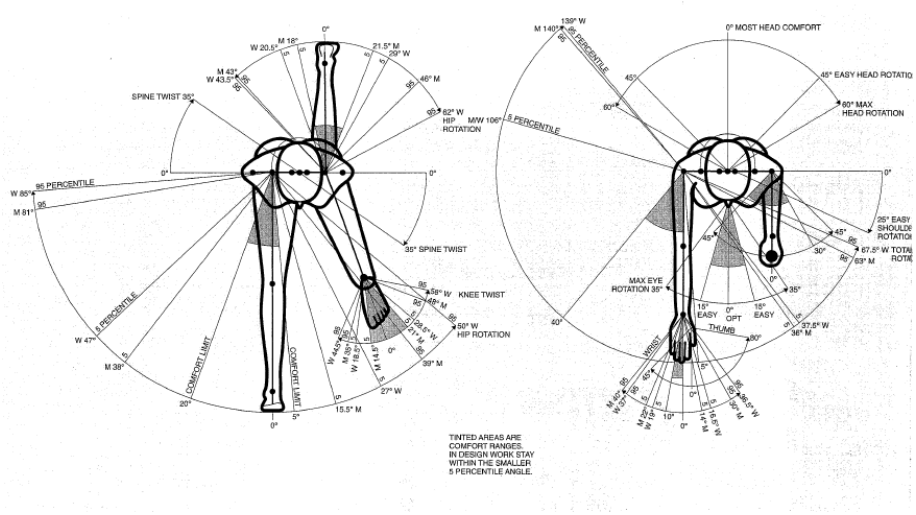


Figure 3.14. Overhead views of human from [70]

new position. The physical possibilites are determined based on simple joint limitations taken from Figures 3.14 and 3.15. The probability distributions are shown in Figure 3.16. The probabilities of changes in angle were determined experimentally. The distributions are each uniform functions over the area of possible changes.

In order for an optimization algorithm to be able to correctly choose the best set of angles, it must be the most probable set of angles. Additionally the surface represented by the curve of the probabilities must be relatively smooth. As seen in Figure 3.17, when we vary the rotation of the shoulder about the x- and y-axis by $\pm$ 20 degrees from it's actual location at -90 degrees about the x-axis and 0 degrees about the y-axis, the total chamfer distance trends towards the center in a relatively smooth fashion. The probabilities of each of the angles and the changes in angles are assumed to be one. However, when we vary the rotation about the shoulder at the y-axis and the elbow at the y-axis, our results are not as good. This is because it is difficult to visually distinguish the rotation about the elbow. The graph of the probabilities is shown in Figure 3.18. In Figure 3.19, we see two images from camera 201 and how close they are visually. The original has the elbow rotated -20 degrees about the y-axis and the shoulder not rotated about the y-axis at all. The test has

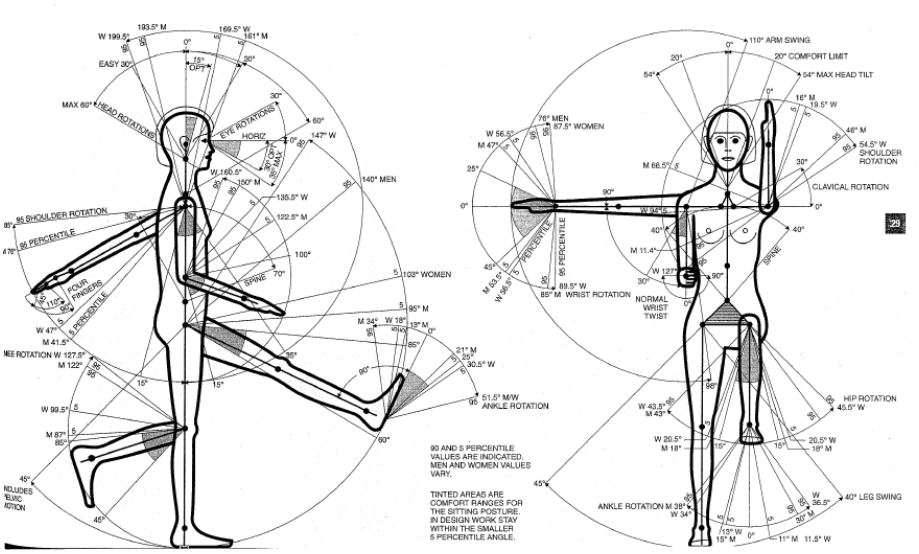


Figure 3.15. Side and front view of human from [70]

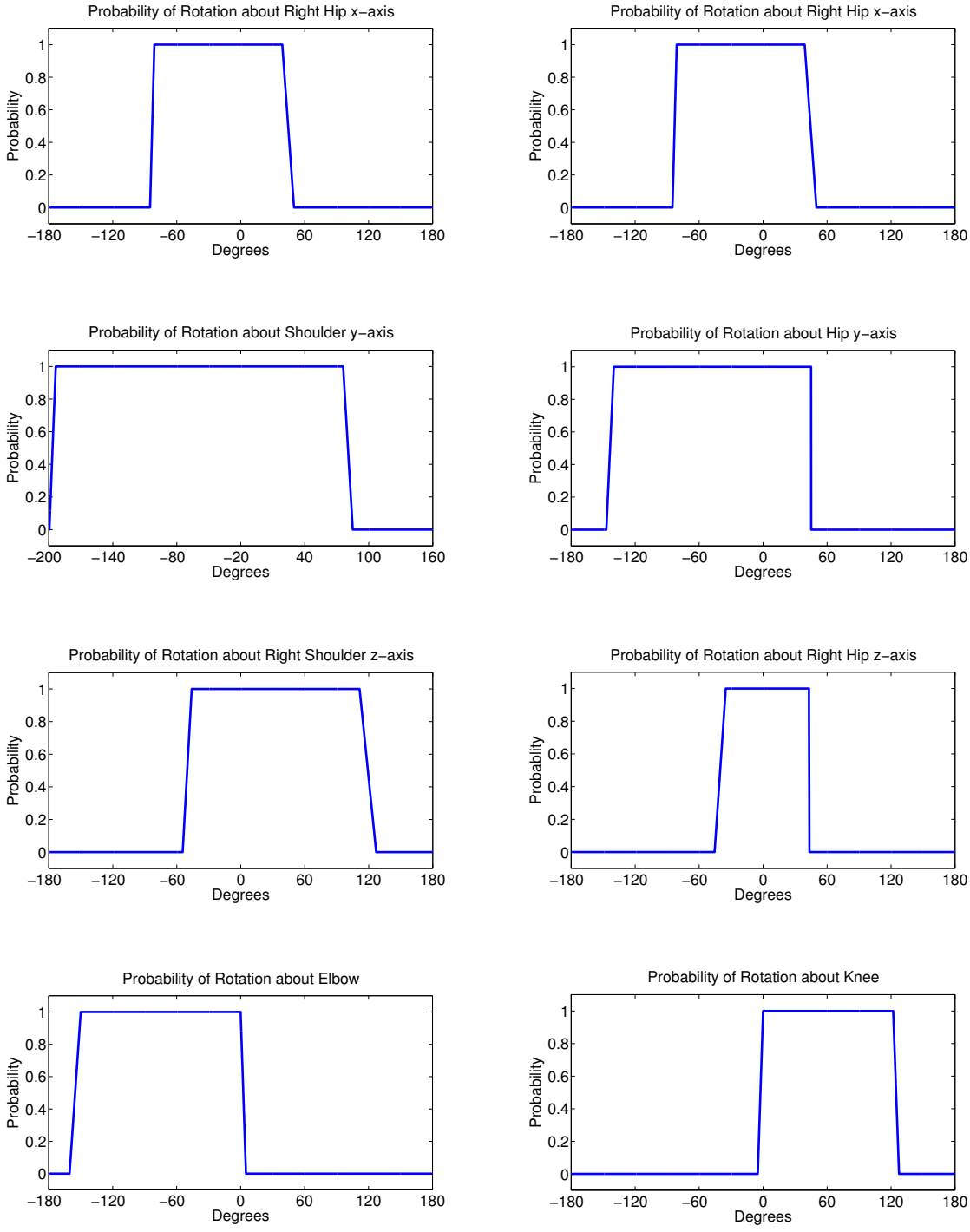


Figure 3.16. Probabilities for Joint Rotations

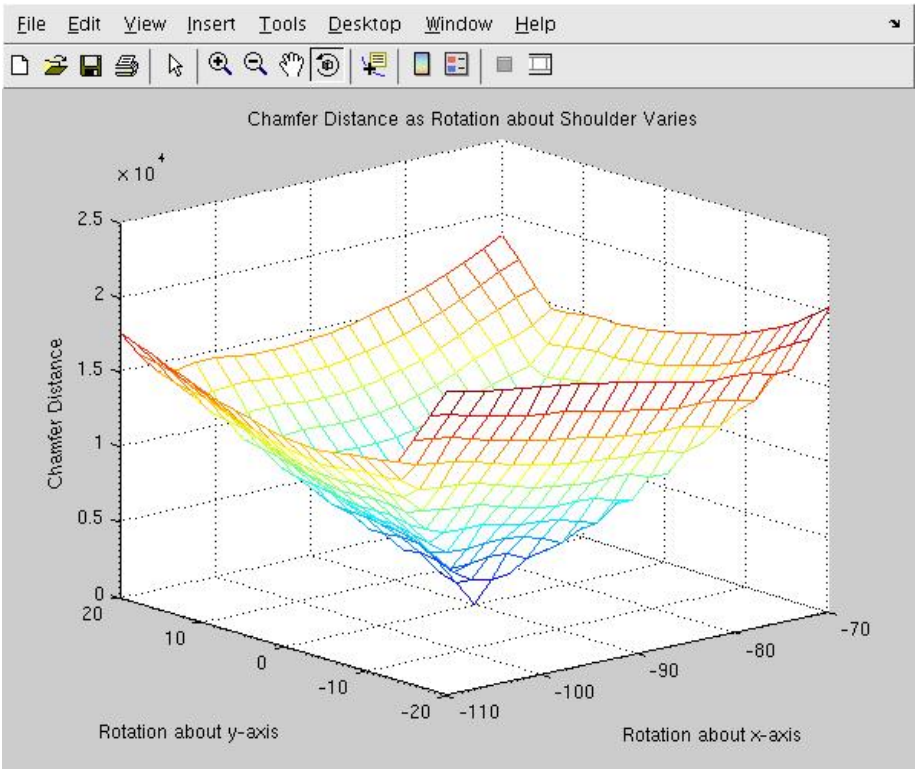


Figure 3.17. Chamfer Distance as Right Shoulder is Rotated

the shoulder rotated -20 degrees about the y-axis and the elbow rotated -4 degrees about the y-axis. This is why the chamfer distance is harder to use at this point.

3.4 Optimization Algorithm

In order to obtain the best set of angles, we need an optimization algorithm. The system presented in this thesis uses the open source algorithm `levmar` [71], which is an implementation of the Levenberg-Marquardt (LM) algorithm, an iterative technique that locates the minimum of a multivariate function that is expressed as the sum of squares of non-linear real-valued functions. It can be thought of as a combination of steepest descent and the Gauss-Newton method. Following is a brief explanation of the LM algorithm, and what those parameters symbolize in our algorithm.

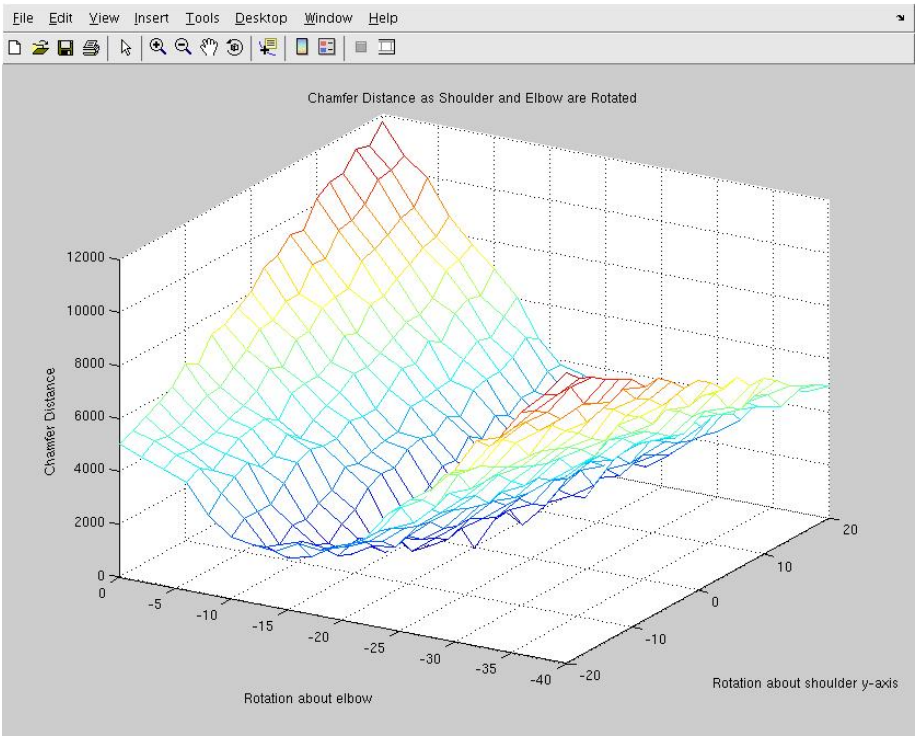


Figure 3.18. Chamfer Distance as Shoulder and Elbow are Rotated

Original Image Test Image

Figure 3.19. Minor Variations at the Elbow

Let vectors and arrays appear in boldface and let ^{T} be used to denote transposition. Also, $\| \cdot \|$ and $\| \cdot \|_\infty$ denote the 2nd and infinity norms respectively. Let $\mathbf{p} \in \Re^m$ denote the *parameter vector*, which in our algorithm is the set of angles. Let $\mathbf{x} \in \Re^n$ denote the *measurement vector*, which in our algorithm is a zero vector. Let *f* be an assumed functional relation which maps a parameter vector **p** to an estimated measurement vector $\hat{\mathbf{x}} = f(\mathbf{p})$, in our algorithm the dimension of $\mathbf{x}$ and $\hat{\mathbf{x}}$ two times the number of pose parameters being optimized across plus two times the number of cameras being used (n = 2m + 2 * numCameras). The first m parameters of the vector $\hat{\mathbf{x}}$ are equal to 1000 * (1 - probability that the angle is possible). The second m parameters of the vector $\hat{\mathbf{x}}$ are equal to 1000 * (1 - probability that the change in angle is possible). The next parameter is 1000 * normalized XOR function for the first camera. The following parameter is the chamfer distance for the first camera. These two parameters repeat for each camera that is being used. Further explainations of these functions are in Section 3.3. An initial parameter vector $\mathbf{p}_0$ (the estimated angle set, which is also the previous angle set) and a measured vector $\mathbf{x}$ (the vector we optimize towards) are provided by our algorithm. The LM algorithm tries to find $\mathbf{p}^+$ that best satisfies the functional relation *f*, i.e. minimizes the squared distance $\epsilon^T\epsilon$ with $\epsilon = \mathbf{x} - \hat{\mathbf{x}}$. The basis of the LM algorithm is a linear approximation to *f* in the neighborhood of **p**. For a small $\| \delta_\mathbf{p} \|$, a Taylor series expansion leads to the approximation

$$f(\mathbf{p} + \delta_\mathbf{p}) \approx f(\mathbf{p}) + \mathbf{J}\delta_\mathbf{p} \tag{3.13}$$

where **J** is the Jacobian matrix $\frac{\partial f(\mathbf{p})}{\partial \mathbf{p}}$. Like all non-linear optimization methods, LM is iterative: Initiated at the starting point $\mathbf{p}_0$, the method produces a series of vectors that converge towards a local minimizer for *f*. Therefore, at each step, it is required to find $\delta_\mathbf{p}$ that minimizes the quantity

$$\| \mathbf{x} - f(\mathbf{p} + \delta_\mathbf{p}) \| \approx \| \mathbf{x} - f(\mathbf{p}) + \mathbf{J}\delta_\mathbf{p} \| = \| \epsilon - \mathbf{J}\delta_\mathbf{p} \| \tag{3.14}$$

The sought $\delta_{\mathbf{p}}$ is the solution to the linear least-squares problem: the minimum is attained when $\mathbf{J}\delta_{\mathbf{p}} - \epsilon$ is orthogonal to the column space of $\mathbf{J}$. This leads to $\mathbf{J}^T\mathbf{J}\delta_{\mathbf{p}} = \mathbf{J}^T\epsilon = 0$, which yields $\delta_{\mathbf{p}}$ as the solution of the so-called normal equations in Equation 3.13:

$$\mathbf{J}^T\mathbf{J}\delta_{\mathbf{p}} = \mathbf{J}^T\epsilon \tag{3.15}$$

The matrix $\mathbf{J}^T\mathbf{J}$ in the left hand side of Equation 3.15 is the approximate Hessian, i.e. an approximation to the matrix of second order derivatives. The LM method actually solves a slight variation of Equation 3.15, known as the *augmented normal equations*

$$\mathbf{N}\delta_{\mathbf{p}} = \mathbf{J}^T\epsilon \tag{3.16}$$

where the off-diagonal elements of $\mathbf{N}$ are identical to the corresponding elements of $\mathbf{J}^T\mathbf{J}$ and the diagonal elements are given by $\mathbf{N}_{ii} = \mu + \left[\mathbf{J}^T\mathbf{J}\right]_{ii}$ for some $\mu > 0$. The strategy of altering the diagonal elements of $\mathbf{J}^T\mathbf{J}$ is called *damping* and μ is referred to as the *damping term*. If the unpdated parameter vector $\mathbf{p} + \delta_{\mathbf{p}}$ with $\delta_{\mathbf{p}}$ computed from Equation 3.16 leads to a reduction in the error ϵ, the update is accepted and the process repeats with a decreased damping term. Otherwise, the damping term is increased, the augmented normal equations are solved again and the process iterates until a value of $\delta_{\mathbf{p}}$ that decreases error is found. The process of repeatedly solving Equation 3.16 for different values of the damping term until an acceptable update to the parameter vector is found corresponds to one iteration of the LM algorithm.

In LM, the damping term is adjusted at each iteration to assure a reduction in the error ϵ. If the damping term is large, then matrix $\mathbf{N}$ in Equation 3.16 is nearly diagonal and the LM update step $\delta_{\mathbf{p}}$ is near the steepest descent direction. Moreover, the magnitude of $\delta_{\mathbf{p}}$ is reduced in this case. Damping also handles situations where the Jacobian is rank deficient and $\mathbf{J}^T\mathbf{J}$ is therefore singular. In this way, LM can defensively navigate a region of the parameter space in which the model is highly nonlinear. If the damping is small, the LM step approximates the exact quadratic step appropriate for a fully linear problem. LM is adaptive because it controls its own damping: it raises the damping if a step fails to reduce

ϵ; otherwise it reduces the damping. In this way LM is capable of alternating between a slow descent approach when far from the minimum and a fast convergence when it is in the minimum's neighborhood. The LM algorithm terminates when at least one of the following conditions is met:

- The magnitude of the gradient of $\epsilon^T\epsilon$, i.e. $\mathbf{J}^T\epsilon$ in the right hand side of Equation 3.15, drops below a user defined threshold ε_1
- The relative change in the magnitude of $\delta_{\mathbf{p}}$ drops below a user defined threshold ε_2
- The error $\epsilon^T\epsilon$ drops below a user defined threshold ε_3
- A maximum number of iterations k_{max} is completed

This summation has been taken from the paper by Lourakis [72]. This algorithm produces the best set of angles for a given multi-view image.

4. RESULTS

4.1 Synthetic Data[1]

We generated synthetic data and used it to test the accuracy of our system. We start the tracking with the arms strait out to the side and the legs both down. While the arms are extended away from the body, the system does a good job of tracking them, as shown in Tables 4.1 and 4.2. However, as the arms moves closer to the body, the tracking algorithm has a harder time tracking them accurately. Additionally, the algorithm is better able to track the right arm than the left arm, because it has a clearer view of the right arm than the left arm. As the left arm moves down it is obscured by the body, especially in camera 200, as shown in Figure 4.1. Without reinitializing the system, we continue to the data in Tables 4.3, 4.4, 4.5 and 4.6. As shown in Tables 4.3 and 4.4, the program continues to track the right arm well. The only place that it has problems is that when the elbow bends, the algorithm sometimes moves the entire arm up more instead of bending the elbow. The arms do look similar in each as shown in Figure 4.2, especially the right arm. The left arm continues not to track as well as the right arm, as seen in Tables 4.5 and 4.6. However, although the calculated rotations seems to be very different from the synthetic data, when you project the elbow into 3D space and calculate the Euclidean distance between the synthetic elbow and the generated elbow, the locations are much closer than one might assume. This is because rotations in 3D space are not unique. The points where the distance is the greatest are when the elbow bends. The left arm, like the right arm, tries to move the entire arm up instead of just the elbow. As seen in Tables 4.7, 4.8, 4.9, and 4.10, the algorithm is able to track the legs as well as the arms, although it does not do as well.

[1]For the tables in this section, we have abbreviated certain words. Right is abbreviated as R, Left as L. S stands for shoulder, E for Elbow, H for Hip and K for Knee. X, Y and Z stand for rotation about the relevent axis. For example RSX means the rotation in degrees of the right shoulder about the x-axis.

Table 4.1
Right Arm Rotated about x-axis.

Synthetic Test Data				Generated Data			
RSX	RSY	RSZ	REY	RSX	RSY	RSZ	REY
-90	0	0	0	-89.95	-0.02	0.00	-2.34
-110	0	0	0	-109.07	0.33	-3.03	-3.15
-130	0	0	0	-129.64	3.02	-0.29	-1.67
-150	0	0	0	-151.15	4.49	-0.93	-1.98
-180	0	0	0	-178.43	13.21	-1.27	-3.65
-180	0	0	0	-173.27	2.32	0.02	-6.41
-160	0	0	0	-158.91	1.69	1.12	-8.08
-130	0	0	0	-130.61	1.20	-3.17	-5.33
-110	0	0	0	-109.13	-2.00	-2.59	-4.51
-90	0	0	0	-90.25	0.79	-1.37	3.83
-80	0	0	0	-80.66	0.74	-1.38	-3.63
-60	0	0	0	-60.03	-0.15	-0.57	-2.51
-45	0	0	0	-46.71	0.11	-0.49	-2.25
-30	0	0	0	-30.44	-0.58	0.23	-2.29
-20	0	0	0	-22.01	-1.89	-0.70	-3.13
-10	0	0	0	-11.38	-1.84	-0.56	-2.40
0	0	0	0	-11.41	-2.53	-0.25	-2.51
-5	0	0	0	-6.04	-1.73	0.37	-2.54
0	0	0	0	-1.54	-2.53	1.03	-3.73

Although the algorithm fails to initially bring the right leg back down, it does correct itself later while the left leg is moving.

Table 4.2
Left Arm Rotated about x-axis

Synthetic Test Data				Generated Data			
LSX	LSY	LSZ	LEY	LSX	LSY	LSZ	LEY
90	0	0	0	90.01	0.00	0.00	-2.20
110	0	0	0	108.45	4.07	0.17	-1.57
130	0	0	0	130.99	0.36	0.32	-3.16
150	0	0	0	145.70	1.89	-0.53	-1.64
180	0	0	0	169.66	3.36	-0.42	0.47
180	0	0	0	178.03	4.38	0.76	-4.60
160	0	0	0	163.86	1.91	-0.24	-4.83
130	0	0	0	132.01	1.47	-5.30	-5.16
110	0	0	0	114.50	-2.86	-7.40	-4.98
90	0	0	0	102.07	-5.08	-7.20	-7.11
80	0	0	0	94.63	-7.53	-8.34	-8.44
60	0	0	0	84.70	-12.50	-7.83	-9.92
45	0	0	0	77.76	-18.81	-8.47	-10.88
30	0	0	0	71.33	-21.30	-8.16	-10.69
20	0	0	0	68.48	-23.35	-8.63	-7.02
10	0	0	0	64.95	-23.39	-11.31	-5.99
0	0	0	0	64.55	-23.26	-12.11	-5.96
5	0	0	0	62.36	-23.65	-8.61	-10.29
0	0	0	0	61.88	-25.34	-8.60	-10.27

(a) (b) (c)

Figure 4.1. Views from each camera (a) Camera 200 (b) Camera 201 (c) Camera 202

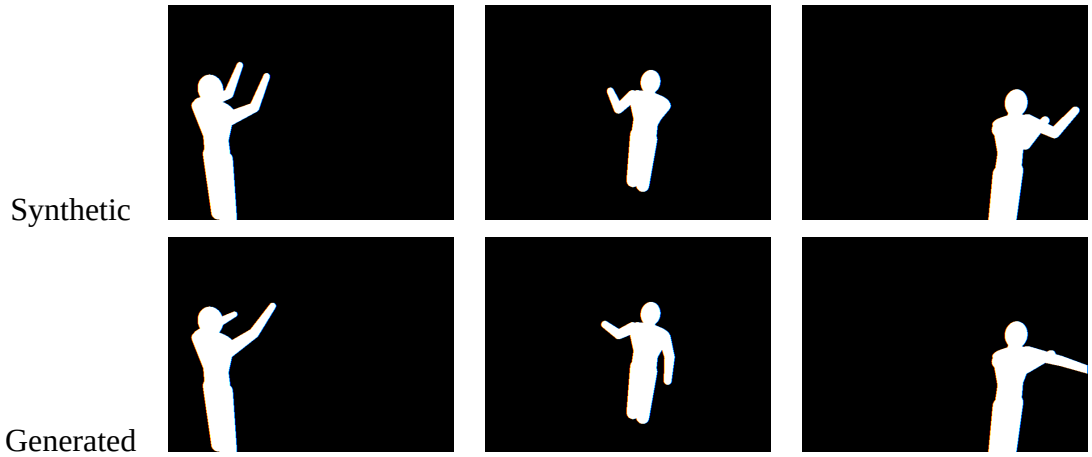


Figure 4.2. Elbows Bent

Table 4.3
Right Arm Rotated about y-axis

Synthetic Test Data				Generated Data			
RSX	RSY	RSZ	REY	RSX	RSY	RSZ	REY
0	0	0	0	-1.54	-2.57	1.03	-3.73
0	-10	0	0	1.67	-12.57	2.39	-4.07
0	-20	0	0	-0.74	-17.29	2.13	-4.31
0	-35	0	0	-15.46	-39.35	-2.00	-2.77
0	-50	0	0	-1.03	-49.04	-4.44	-2.56
0	-70	0	0	0.60	-67.00	-4.54	-6.52
0	-90	0	0	5.37	-89.78	-4.10	-13.51
0	-90	0	-10	6.16	-87.04	-4.58	-19.63
0	-90	0	-20	5.88	-90.35	-3.93	-21.71
0	-90	0	-30	5.24	-94.91	-2.96	-22.55
0	-90	0	-40	6.04	-99.57	-1.74	-24.68
0	-90	0	-50	5.44	-97.30	-0.76	-26.75
0	-90	0	-60	3.18	-102.93	0.46	-30.09
0	-90	0	-70	2.87	-102.78	2.57	-35.76
0	-90	0	-80	-1.60	-107.01	4.89	-38.79
0	-90	0	-90	-8.09	-109.54	7.66	-41.50
0	-90	0	-80	-8.76	-100.65	9.06	-37.95
0	-90	0	-70	-8.31	-93.53	9.05	-52.02
0	-90	0	-60	-6.96	-89.77	8.08	-51.21
0	-90	0	-50	-5.94	-87.01	6.85	-48.98
0	-90	0	-40	-5.11	-86.37	5.74	-47.71

Table 4.4
Right Arm Rotated about y-axis cont.

Synthetic Test Data				Generated Data			
RSX	RSY	RSZ	REY	RSX	RSY	RSZ	REY
0	-90	0	-30	-4.22	-84.20	4.67	-45.32
0	-90	0	-20	-3.69	-80.23	4.52	-40.94
0	-90	0	-10	-3.19	-76.32	4.61	-34.28
0	-90	0	0	-2.97	-75.64	4.58	-34.11
0	-80	0	0	-3.23	-68.33	3.57	-28.19
0	-60	0	0	-0.56	-52.70	4.99	-21.23
0	-30	0	0	11.03	-46.10	10.95	-25.54
0	-20	0	0	10.47	-38.33	10.16	-23.95
0	-10	0	0	11.87	-32.22	9.83	-22.55

Table 4.5
Left Arm Rotated about y-axis

Synthetic Test Data				Generated Data				Distance in inches
LSX	LSY	LSZ	LEY	LSX	LSY	LSZ	LEY	Left Elbow
0	0	0	0	61.88	-25.34	-8.60	-10.27	0.48
0	-10	0	0	60.08	-25.53	-7.56	-9.67	0.70
0	-20	0	0	59.74	-25.08	-7.39	-5.67	7.43
0	-35	0	0	18.05	-13.66	-7.07	-8.08	8.29
0	-50	0	0	23.35	-23.33	-6.33	-8.47	18.38
0	-70	0	0	30.91	-30.06	-5.50	-12.06	19.09
0	-90	0	0	43.45	-37.53	-5.22	-9.79	20.04
0	-90	0	-10	49.90	-40.91	-4.13	-10.52	11.83
0	-90	0	-20	60.35	-42.71	-3.92	-11.11	9.03
0	-90	0	-30	62.03	-48.49	-3.64	-11.43	1.17
0	-90	0	-40	73.98	-63.81	-3.44	-16.44	5.60
0	-90	0	-50	78.46	-62.64	-0.82	-12.46	3.23
0	-90	0	-60	84.33	-61.45	-0.16	-12.55	8.04
0	-90	0	-70	88.43	-61.27	1.35	-2.51	23.95
0	-90	0	-80	90.56	-64.34	1.37	-1.05	23.06
0	-90	0	-90	92.88	-62.98	0.98	-2.42	7.48
0	-90	0	-80	94.01	-67.18	0.99	-1.56	22.57
0	-90	0	-70	101.91	-67.78	-4.04	-2.36	24.12
0	-90	0	-60	102.46	-68.72	-4.19	-1.60	19.39
0	-90	0	-50	103.37	-68.58	-4.40	-2.12	24.10
0	-90	0	-40	103.12	-70.74	-4.51	-2.94	19.81

Table 4.6
Left Arm Rotated about y-axis cont.

Synthetic Test Data				Generated Data				Distance in inches
LSX	LSY	LSZ	LEY	LSX	LSY	LSZ	LEY	Left Elbow
0	-90	0	-30	103.71	-71.81	-4.30	-1.53	21.71
0	-90	0	-20	102.82	-71.57	-4.30	-1.62	6.23
0	-90	0	-10	95.46	-84.40	-3.87	-6.24	2.89
0	-90	0	0	96.57	-87.30	-4.13	-5.84	14.35
0	-80	0	0	96.34	-94.38	-4.64	-7.87	22.65
0	-60	0	0	97.41	-105.56	-4.42	-12.90	20.09
0	-30	0	0	92.74	-126.37	-0.97	-13.17	4.49
0	-20	0	0	94.36	-128.87	-2.24	-9.45	8.91
0	-10	0	0	94.28	-133.62	-0.83	-7.16	6.26

Table 4.7
Right Leg Rotated about y-axis

Synthetic Test Data				Generated Data			
RHX	RHY	RHZ	RKY	RHX	RHY	RHZ	RKY
0	0	0	0	0.00	0.00	0.00	0.00
0	-20	0	0	-6.86	-4.48	0.00	-0.63
0	-40	0	0	-1.19	-36.03	-0.22	-10.53
0	-60	0	0	-3.28	-49.59	-1.29	-11.54
0	-40	0	0	-0.34	-39.19	-0.87	-3.82
0	-20	0	0	-0.33	-37.88	-1.06	-3.71
0	0	0	0	-0.53	-36.87	-0.64	-3.74
0	0	0	0	-2.09	-34.84	-0.64	-4.96
0	0	0	0	-4.45	-36.83	-2.18	-3.46
0	0	0	0	-8.40	-37.40	-5.16	-2.70
0	0	0	0	0.90	-3.34	-11.78	5.27
0	0	0	0	2.89	-1.57	-11.86	6.82
0	0	0	0	2.89	-1.57	-11.86	6.82
0	0	0	10	2.86	-1.58	-11.87	6.80
0	0	0	20	5.01	1.72	-12.18	8.12
0	0	0	30	5.01	1.72	-12.18	8.12
0	0	0	40	3.55	4.64	-12.18	9.79
0	0	0	50	3.26	5.58	-12.78	10.47
0	0	0	60	1.94	6.98	-13.34	11.10
0	0	0	70	1.94	6.98	-13.34	11.10
0	0	0	80	-1.18	6.55	-11.90	10.81
0	0	0	90	-0.74	6.79	-13.37	9.13
0	0	0	80	-6.70	8.38	-13.76	11.50
0	0	0	70	-4.46	16.17	-19.51	14.02
0	0	0	60	-4.37	14.67	-21.70	15.67

Table 4.8
Right Leg Rotated about y-axis cont

Synthetic Test Data				Generated Data			
RHX	RHY	RHZ	RKY	RHX	RHY	RHZ	RKY
0	0	0	50	-5.45	12.48	-21.83	14.74
0	0	0	40	-5.75	11.05	-21.77	14.32
0	0	0	30	-5.95	8.82	-21.23	14.35
0	0	0	20	-4.66	6.05	-21.33	13.94
0	0	0	10	-5.09	2.82	-21.32	12.91
0	0	0	0	-3.52	0.21	-20.84	11.93
0	0	0	0	-1.84	-1.69	-20.87	10.41

Table 4.9
Left Leg Rotated about x-axis

Synthetic Test Data				Generated Data			
LHX	LHY	LHZ	LKY	LHX	LHY	LHZ	LKY
0	0	0	0	0.00	0.00	0.00	0.00
0	0	0	0	3.16	-2.57	0.00	2.82
0	0	0	0	4.13	-4.26	0.25	2.30
0	0	0	0	-1.05	0.44	-0.70	4.75
0	0	0	0	0.76	-0.82	-1.41	3.19
0	0	0	0	-1.23	-1.86	-1.51	6.62
0	0	0	0	-4.02	-2.24	-1.76	5.87
0	0	0	0	-5.72	-1.48	-1.65	7.92
20	0	0	0	7.40	-4.30	-0.54	3.52
40	0	0	0	27.16	-0.83	-1.15	7.71
30	0	0	0	31.40	-3.80	6.77	13.05
10	0	0	0	28.84	-4.98	6.64	12.01
0	0	0	0	28.84	-4.98	6.64	16.01
0	0	0	0	28.82	-4.99	6.66	16.02
0	0	0	0	26.44	-3.72	6.85	15.66
0	0	0	0	26.44	-3.72	6.85	19.66
0	0	0	0	25.78	-3.98	6.85	20.56
0	0	0	0	25.90	-4.77	6.80	20.18
0	0	0	0	25.79	-4.92	7.17	19.99
0	0	0	0	25.79	-4.92	7.17	23.99
0	0	0	0	23.64	-3.98	7.60	22.61
0	0	0	0	20.08	-2.53	6.24	27.84
0	0	0	0	12.18	1.42	6.63	35.17
0	0	0	0	-4.92	-5.74	1.51	32.15
0	0	0	0	-1.55	-4.51	-0.24	30.65

Table 4.10
Left Leg Rotated about x-axis cont

Synthetic Test Data				Generated Data			
LHX	LHY	LHZ	LKY	LHX	LHY	LHZ	LKY
0	0	0	0	-2.15	-4.32	-0.09	33.39
0	0	0	0	-2.53	-4.99	-0.01	36.77
0	0	0	0	-1.78	-4.37	-0.62	34.73
0	0	0	0	-3.00	-4.16	-0.44	34.81
0	0	0	0	-1.91	-4.76	-0.73	38.02
0	0	0	0	-1.92	-4.81	-0.72	37.07
0	0	0	0	-2.31	-4.52	-0.47	36.05

4.2 Real Data

The room that we collected data in has twelve cameras arranged throughout it. The cameras are divided into groups of three and each group of cameras is connected to one computer. Each computer has the ability to synchronize the cameras that are connected to it. However the cameras that are connected to different computers cannot currently be synchronized. Therefore all of the real data that has been collected and analyzed is based on three synchronized cameras that are connected to one PC. These cameras are numbered 200, 201, and 202. For data collection, we chose a position where most of the person could be seen in all three cameras. Figure 4.3 shows a series of images and the generated 3D models for those images. The images in Figures 4.3 and 4.4 were analyzed without reinitializing the system. In Figure 4.4 the arms move from the sides to the front and the algorithm sucessfully tracks the arms to the front. Figure 4.5 shows the ability of the system to track the leg as it moves forward and back down. Like the synthetic results, it also tracks the leg moving out well, but doesn't do as well at bringing the leg back down.

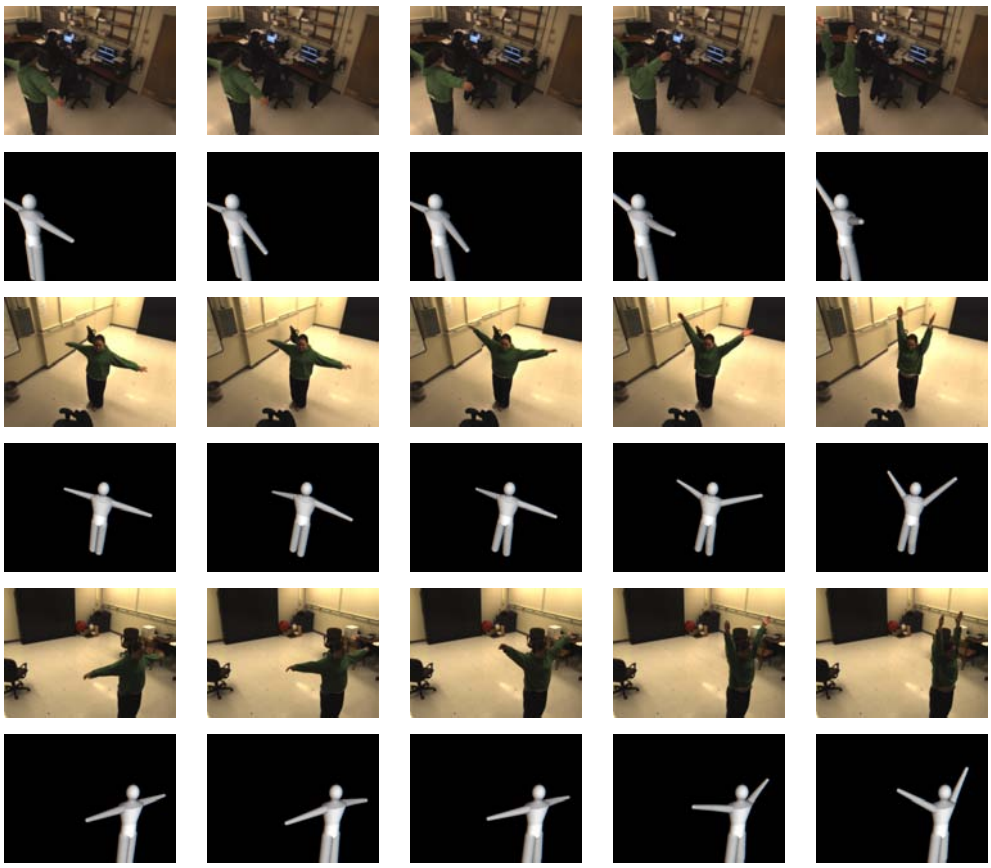


Figure 4.3. Results Tracking Arms in x-axis

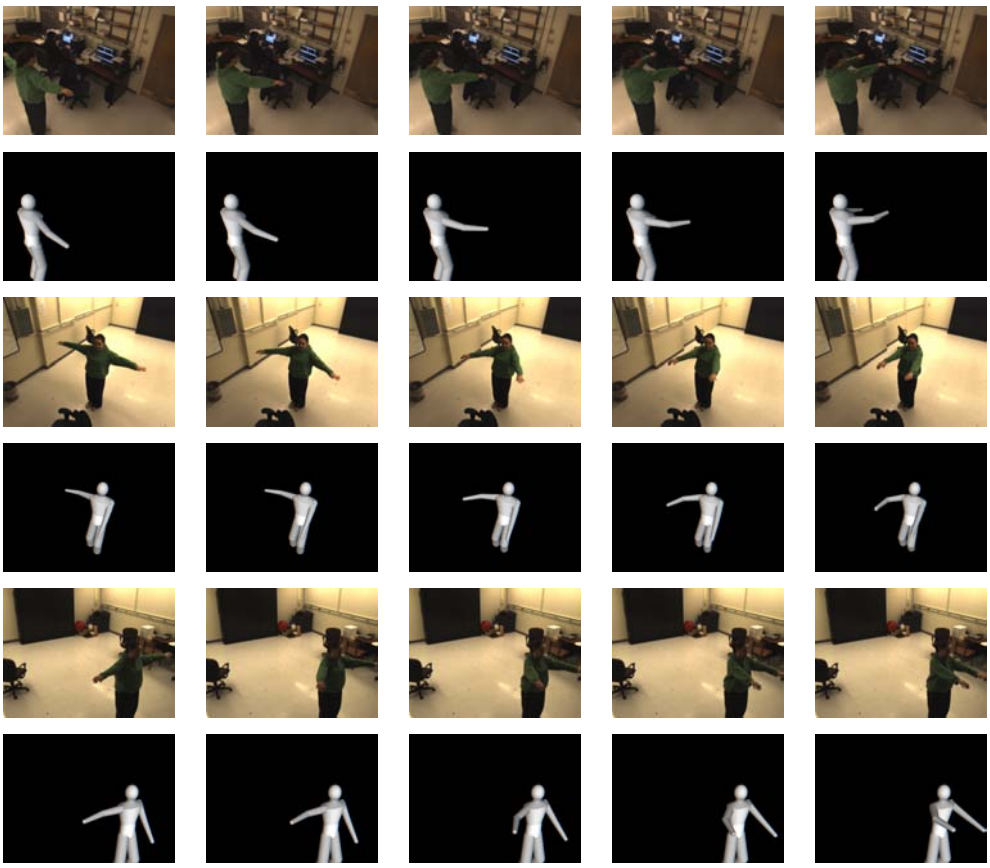


Figure 4.4. Results Tracking Arms from Out to Sides to Front

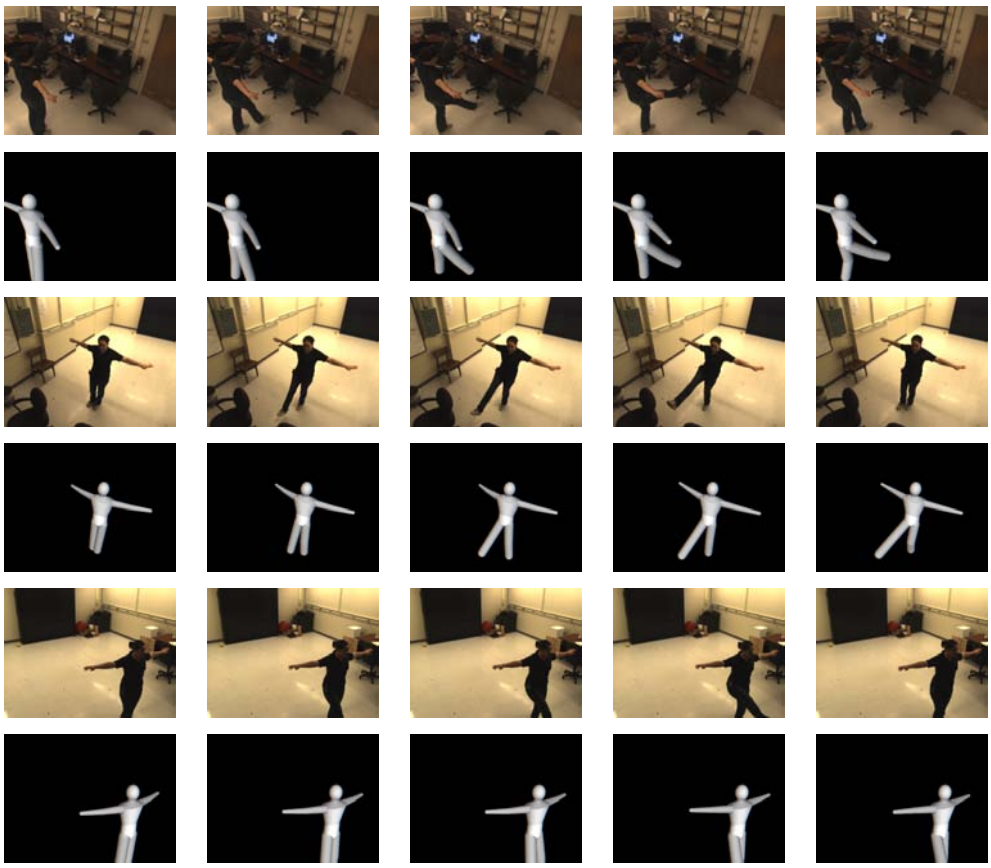


Figure 4.5. Results Tracking Right Leg

5. CONCLUSIONS

This thesis has presented a system for tracking articulated human motion without relying on markers or special colored clothing. The system first extracts a silhouette of the person in the room using background subtraction. This silhouettte is then used to generate a set of parameters which describe the pose of the person. The parameter estimation incorporates the physical limitations of the human body such as how far and how fast each joint can move. We have shown that the system is very accurate through the use of synthetic data. We have also shown that they system works for actual data by analyzing data collected by cameras. Both show very promising results.

There are a number of ways that this system could be improved in the future. Currently our system optimizes across the 16 parameters that describe the movement of the arms and legs. The torso of the person is assumed to be fixed and the location and orientation are known. Future works would include optimizing across all of the degrees of freedom explained in Section 3.2. This would allow the person to move throughout the room and the algorithm to track this movement. Another way that the system could be improved would be to design a model that could be varied to better fit the subject. Right now we can scale the model as a whole, but we could make the individual body parts scaleable so that someone with longer or shorter arms or legs could be tracked well.

The biggest improvement to this system would be to make it work in real time. It is designed in such a way that this should be possible in the future. One of the major modifications to make this possible to be to parallelize the optimization algorithms. For example, once the location and orientation of the torso is determined, the optimization of the pose parameters of the arms and the legs could be carried out on separate machines, with up to four computers performing computations at once.

LIST OF REFERENCES

LIST OF REFERENCES

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

[2] T. Moeslund and E. Granum, "A survey of computer vision-based human motion capture," *Computer Vision and Image Understanding: CVIU*, vol. 81, no. 3, pp. 231–268, 2001.

[3] R. Polana and R. Nelson, "Low level recognition of human motion," in *IEEE Workshop on Motion of Non-Rigid and Articulated Objects*, pp. 83–88, IEEE, 1994.

[4] W. Freeman, K. Tanaka, and J. Ohta, "Computer vision for computer games," in *International Workshop on Automatic Face and Gesture Recognition*, pp. 100–105, IEEE, October 1996.

[5] E. Hunter, J. Schlenzig, and R. Jain, "Posture estimation in reduced-model gesture input systems," in *International Workshop on Automatic Face and Gesture Recognition*, pp. 290–295, IEEE, June 1995.

[6] C. Charayaphan and A. Marble, "Image processing system for interpreting motion in american sign language," *Journal of Biomedical Engineering*, vol. 14, no. 15, pp. 419–425, 1992.

[7] J. Davis and M. Shah, "Gesture recognition," Tech. Rep. CS-TR93 -11, 1993.

[8] A. Pentland, "Automatic extraction of deformable models," *International Journal of Computer Vision*, vol. 4, pp. 107–126, 1990.

[9] T. Starner and A. Pentland, "Real-time american sign language recognition from video using hidden markov models," in *International Symposium on Computer Vision*, pp. 265–270, IEEE, November 1995.

[10] T. Cootes, C. Taylor, D. Cooper, and J. Graham, "Active shape models - their training and application," *Computer Vision and Image Understanding*, vol. 61, no. 1, pp. 38–59, 1995.

[11] A. Baumberg and D. Hogg, "An efficient method for contour tracking using active shape models," Tech. Rep. 94.11, April 1994.

[12] A. Geurtz, *Model-based Shape Estimation*. Phd thesis, Polytechnic Institute of Lausanne, Department of Electrical Engineering, 1993.

[13] S. Niyogi and E. Adelson, "Analyzing and recognizing walking figures in xyt," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 469–474, IEEE, June 1994.

[14] S. Niyogi and E. Adelson, "Analyzing gait with spatiotemporal surfaces," in *IEEE Workshop on Motion of Non-Rigid and Articulated Objects*, pp. 64–69, IEEE, November 1994.

[15] Y. Guo, G. Xu, and S. Tsuji, "Understanding human motion patterns," in *12th International Conference on Pattern Recognition*, pp. 325–329, January 1994.

[16] I.-C. Chang and C.-L. Huang, "Ribbon-based motion analysis of human body movements," in *International Conference on Pattern Recognition*, pp. 436–440, August 1996.

[17] S. Kurakake and R. Nevatia, "Description and tracking of moving articulated objects," in *International Conference on Computer Vision and Pattern Recognition*, pp. 491–495, IEEE, 1992.

[18] W. Long and Y. Yang, "Log-tracker, an attribute-based approach to tracking human body motion," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 5, no. 3, pp. 439–458, 1991.

[19] M. Leung and Y. Yang, "First sight: A human body outline labeling system," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, pp. 359–377, April 1995.

[20] C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland, "Pfinder: Real-time tracking of the human body," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 780–785, 1997.

[21] Q. Cai and J. Aggarwal, "Tracking human motion using multiple cameras," in *International Conference on Pattern Recognition*, pp. 68–72, August 1996.

[22] L. Campbell and A. Bobick, "Recognition of human body motion using phase space constraints," in *International Conference on Computer Vision*, pp. 624–630, June 1995.

[23] M. Spong and M. Vidyasagar, *Robot dynamics and control*. New York: John Wiley and Sons, 1991.

[24] J. O'Rourke and N. Badler, "Model-based image analysis of human motion using constraint propagation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 2, pp. 522–536, November 1980.

[25] A. Downton and H. Drouet, "Model-based image analysis for unconstrained human upper-body motion," in *IEEE International Conference on Image Processing and its Applications*, pp. 274–277, IEEE, 1992.

[26] L. Goncalves, E. Bernardo, E. Ursella, and P. Perona, "Monocular tracking of the human arm in 3d," in *International Conference on Computer Vision*, pp. 764–770, IEEE, June 1995.

[27] D. Hogg, "Model-based vision: A program to see a walking person," *Image and Vision Computing*, vol. 1, no. 1, pp. 5–20, 1983.

[28] D. Marr and H. Nishihara, "Representation and recognition of the spatial organization of three-dimensional shapes," in *Royal Society of London B*, pp. 269–294, 1978.

[29] K. Rohr, "Towards model-based recognition of human movements in image sequences," *Computer Vision, Graphics, and Image Processing: Image Understanding*, vol. 59, pp. 94–115, January 1994.

[30] A. Barr, "Global and local deformations of solid primitives," *Computer Graphics*, vol. 18, no. 3, pp. 21–30, 1984.

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

[32] I. Kakadiaris and D. Metaxas, "Model-based estimation of 3d human motion with occlusion based on active multi-viewpoint selection," in *Conference on Computer Vision and Pattern Recognition*, pp. 81–87, June 1996.

[33] D. Metaxas and D.Terzopoulos, "Shape and nonrigid motion estimation through physics-based synthesis," *IEEE Transactions on Pattern Recognition and Machine Intelligence*, vol. 15, no. 6, pp. 580–591, 1993.

[34] Z. Chen and H.-J. Lee, "Knowledge-guided visual perception of 3d human gait from single image sequence," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 22, pp. 336–342, March/April 1992.

[35] R. Holt, T. Huang, A. Netravali, and R. Qian, "Determining articulated motion from perspective views: A decomposition approach," *Pattern Recognition*, vol. 30, pp. 1435–1449, 1997.

[36] J. Zhao, *Moving Posture Reconstruction from Perspective Projections of Jointed Figure Motion*. Phd thesis, University of Pennsylvania, Department of Computer and Information Science, 1993.

[37] Y. Hel-Or and M. Werman, "Constraint fusion for recognition and localization of articulated objects," *International Journal of Computer Vision*, vol. 19, no. 1, pp. 5–28, 1996.

[38] B. Dorner, "Hand shape identification and tracking for sign language interpretation," in *Looking at People, International Joint Conference on Artificial Intelligence*, 1993.

[39] J. Rehg and T. Kanade, "Visual tracking of high dof articulated structures: an application to human hand tracking," in *European Conference on Computer Vision*, pp. 35–46, 1994.

[40] J. Rehg and T. Kanade, "Model-based tracking of self-occluding articulated objects," in *International Conference on Computer Vision*, pp. 612–617, 1995.

[41] J. Wang, G. Lorette, and P. Bouthemy, "Analysis of human motion: A model-based approach," in *7th Scandinavian Conf. Image Analysis*, pp. 1142–1149, 1991.

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

[43] J. Kuch and T. Huang, "Vision based hand modeling and tracking for virtual teleconferencing and telecollaboration," in *International Conference on Computer Vision*, pp. 666–671, June 1995.

[44] J. Ohya and F. Kishino, "Human posture estimation from multiple images using genetic algorithm," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 750–753, IEEE, 1994.

[45] F. Perales and J. Torres, "A system for human motion matching between synthetic and real images based on a biomechanic graphical model," in *IEEE Workshop on Motion of Non-Rigid and Articulated Objects*, pp. 83–88, IEEE, 1994.

[46] A. Azarbayejani and A. Pentland, "Real-time self-calibrating stereo person tracking using 3-d shape estimation from blob features," in *International Conference on Pattern Recognition*, pp. 627–632, August 1996.

[47] O. Chomat and J. Crowley, "Recognizing motion using local appearance," in *International Symposium on Intelligent Robotic Systems*, pp. 271–279, 1998.

[48] A. Bharatkumar, K. Daigle, M. Pandy, Q. Cai, and J. Aggarwal, "Lower limb kinematics of human walking with the medial axis transformation," in *Workshop on Non-Rigid Motion*, pp. 70–76, IEEE, November 1994.

[49] T. Darrell, P. Maes, B. Blumberg, and A. Pentland, "A novel environment for situated vision and behavior," in *Workshop On Visual Behaviors*, pp. 68–72, IEEE, June 1994.

[50] A. Bobick and A. Wilson, "A state-based technique for the summarization and recognition of gesture," in *5th International Conference on Computer Vision*, pp. 382–388, IEEE, June 1995.

[51] T. Darrell and A. Pentland, "Space-time gestures," in *Conference on Computer Vision and Pattern Recognition*, pp. 335–340, IEEE, June 1993.

[52] D. Gavrila, *Vision-based 3-D tracking of humans in action*. Phd thesis, University of Maryland, Department of Computer Science, 1996.

[53] K. Takahashi, S. Seki, H. Kojima, and R. Oka, "Recognition of dexterous manipulations from time-varying images," in *IEEE Workshop on Motion of Non-Rigid and Articulated Objects*, pp. 23–28, IEEE, 1994.

[54] L. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–285, 1989.

[55] J. Yamato, J. Ohya, and K. Ishii, "Recognizing human action in time-sequential images using a hidden markov model," in *IEEE International Conference on Computer Vision*, pp. 379–385, IEEE, 1992.

[56] M. Rosenblum, Y. Yacoob, and L. Davis, "Human emotion recognition from motion using a radial basis function network architecture," in *IEEE Workshop on Motion of Non-Rigid and Articulated Objects*, pp. 43–49, IEEE, November 1994.

[57] C. Sul, K. Lee, and K. Wohn, "Virtual stage: A location-based karaoke system," *IEEE Multimedia*, vol. 5, no. 2, 1998.

[58] M. Oren, C. Papageorgiou, P. Sinha, E. Osuna, and T. Poggio, "Pedestrian detection using wavelet templates," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 193–199, IEEE, June 1997.

[59] I. Haritaoglu, D. Harwood, and L. Davis, "Ghost: A human body part labeling system using silhouettes," in *14th International Conference on Pattern Recognition*, pp. 77–82, 1998.

[60] L. Campbell and A. Bobick, "Using phase space constraints to represent human body motion," in *International Workshop on Automatic Face and Gesture Recognition*, pp. 338–343, 1995.

[61] R. Kahn, M. Swain, P. Prokopowicz, and R. Firby, "Gesture recognition using the perseus architecture," in *International Conference on Computer Vision and Pattern Recognition*, pp. 734–741, IEEE, 1996.

[62] C. Attwood, G. Sullivan, and K. Baker, "Model-based recognition of human posture using single synthetic images," in *Fifth Alvey Vision Conference*, pp. 25–30, University of Reading, UK, 1989.

[63] J. Amat, A. Casals, and M. Frigola, "Stereoscopic system for human body tracking in natural scenes," in *International Workshop on Modeling People*, pp. 70–78, IEEE, September 1999.

[64] 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.

[65] M. Mangor and C. Theobalt, "Model-based analysis of multi-video data," in *6th IEEE Southwest Symbosium on Image Analysis and Interpretation*, pp. 41–45, IEEE, March 2004.

[66] K. Zimmermann and T. Svoboda, "Probabilistic estimation of articulated body model from multiview data," in *3rd European Medical and Biological Engineering Conference*, pp. 1727–1983, International Federation for Medical and Biological Engineering, 2005.

[67] R. Plänkers, P. Fua, and N. D'Apuzzo, "Automated body modeling from video sequences," in *ICCV Workshop on Modeling People*, September 1999.

[68] I. Mikic, M. Trivedi, E. Hunter, and P. Cosman, "Human body model aquisition and tracking using voxel data," *International Journal of Computer Vision*, vol. 53, no. 3, pp. 199–223, 2003.

[69] "Fast light toolkit." [web page] `http://www.fltk.org/index.php/`.

[70] A. Tilley, *The measure of man and woman*. New York: John Wiley and Sons, 2002.

[71] M. Lourakis, "levmar: Levenberg-marquardt nonlinear least squares algorithms in C/C++." [web page] `http://www.ics.fourth.gr/~lourakis/levmar/`, July 2004.

[72] M. Lourakis, "A brief description of the levenberg-marquardt algorithm implemented by levmar." [web page] `http://www.ics.forth.gr/~lourakis/levmar/levmar.pdf`, 2005.