

# MOTION CAPTURE FROM SINGLE VIDEO SEQUENCE

A THESIS

SUBMITTED TO THE DEPARTMENT OF COMPUTER ENGINEERING  
AND THE INSTITUTE OF ENGINEERING AND SCIENCE  
OF BILKENT UNIVERSITY  
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS  
FOR THE DEGREE OF  
MASTER OF SCIENCE

By

İbrahim Demir

August, 2006

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

---

Assoc. Prof. Dr. Uğur Gdkbay(Advisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

---

Prof. Dr. Enis etin

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

---

Asst. Prof. Dr. Selim Aksoy

Approved for the Institute of Engineering and Science:

---

Prof. Dr. Mehmet Baray  
Director of the Institute

ABSTRACT

MOTION CAPTURE FROM SINGLE VIDEO  
SEQUENCE

İbrahim Demir  
M.S. in Computer Engineering  
Supervisor: Assoc. Prof. Dr. Uğur Gündükbay  
August, 2006

3D human pose reconstruction is a popular research area since it can be used in various applications. Currently most of the methods work for constrained environments, where multi camera views are available and camera calibration is known, or a single camera view is available, which requires intensive user effort. However most of the currently available data do not satisfy these constraints, thus they cannot be processed by these algorithms. In this thesis a framework is proposed to reconstruct 3D pose of a human for animation from a sequence of single view video frames. The framework for pose construction starts with background estimation. Once the image background is estimated, the body silhouette is extracted by using image subtraction for each frame. Then the body silhouettes are automatically labeled by using a model-based approach. Finally, the 3D pose is constructed from the labeled human silhouette by assuming orthographic projection. The proposed approach does not require camera calibration. The proposed framework assumes that the input video has a static background and it has no significant perspective effects and the performer is in upright position.

*Keywords:* motion capture, framework, single camera, uncalibrated camera, vision-based, animation.

## ÖZET

# TEK VIDEO DİZİSİNDEN HAREKET YAKALANMASI

İbrahim Demir

Bilgisayar Mühendisliği, Yüksek Lisans

Tez Yöneticisi: Assoc. Prof. Dr. Uğur GÜDÜKBAY

Ağustos, 2006

Üç boyutlu insan pozunun elde edilmesi çeşitli uygulama alanlarının olmasından dolayı popüler bir konudur. Şu anda var olan yöntemlerin çoğunun çeşitli kısıtlamalara dayanması uygulamada bazı zorluklar doğurmaktadır. Bu yöntemlerin getirmiş olduğu kısıtlamalar birden fazla kamera görüntüsüne ve kamera ayarlarının bilinmesine ihtiyaç duyulması veya tek kamera görüntüsü nedeniyle kullanıcının yoğun çabasına ihtiyaç duyulmasıdır. Çoğu zaman bu kısıtlamalardan dolayı mevcut veriler üzerinde bu yöntemler uygulanamamaktadır. Bu tezde bu kısıtlamalara bağlı olmadan tek kameradan elde edilen videodan üç boyutlu animasyonda kullanılacak insan pozunun elde edilmesi için bir çerçeve çalışma önerilmektedir. Buna göre video karelerindeki arka plan hesaplanmakta, daha sonra görüntü çıkarma yöntemi ile her bir video karesindeki insan silueti bulunmaktadır. Daha sonra vücudun parçaları iki boyutlu resim üzerinde otomatik olarak tanımlanmaktadır. Son adım olarak da vücut parçaları tanımlanmış olan silüetten dikey izdüşüm kullanılarak üç boyutlu poz oluşturulmaktadır. Bu tezde sunulan yöntem çoklu video zorunluluğu veya kamera kalibrasyonu gerektirmemektedir. Bu tezde anlatılan çerçeve çalışma videonun değişmeyen bir arka planın olması, videonun dikey izdüşümünü etkileyecek perspektif etkisinin fazla olmaması ve görüntüdeki kişinin ayakta olması varsayımlarına dayanmaktadır.

*Anahtar sözcükler:* hareket yakalanması, çerçeve, tek kamera, görme temelli, animasyon.

## **Acknowledgement**

I would like to express my thanks and gratitude to Assoc. Prof. Dr. Uğur Gdkbay for giving me the opportunity to do a graduate study with him.

Special thanks to my family and my friends for their endless support.

# Contents

- 1 Introduction** **1**
  - 1.1 Organization of the Thesis . . . . . 2
  
- 2 Background and Related Work** **3**
  - 2.1 Human Motion Control Techniques . . . . . 3
    - 2.1.1 Kinematics . . . . . 3
    - 2.1.2 Dynamics . . . . . 4
  - 2.2 Motion Capture . . . . . 4
    - 2.2.1 Non-vision Based Motion Capture . . . . . 5
    - 2.2.2 Vision-Based Motion Capture with Markers . . . . . 6
    - 2.2.3 Vision-Based Motion Capture without Markers . . . . . 8
  - 2.3 Single Image Processing . . . . . 9
  
- 3 Motion Capture and Animation System** **13**
  - 3.1 Proposed Framework . . . . . 15
    - 3.1.1 Human Modeling . . . . . 15

<i>CONTENTS</i>	vii
3.1.2 Background Estimation . . . . .	20
3.1.3 Silhouette Extraction . . . . .	21
3.1.4 2D Pose Extractor . . . . .	21
3.1.5 3D Pose Estimation . . . . .	37
<b>4 Experimental Results</b>	<b>38</b>
4.1 Visual Results . . . . .	38
4.2 Performance Analysis . . . . .	39
4.3 Effectiveness of the Proposed Framework . . . . .	39
<b>5 Conclusions and Future Work</b>	<b>51</b>
<b>Bibliography</b>	<b>53</b>

# List of Figures

2.1	Projection of a line (adapted from [16]). . . . .	10
2.2	The ambiguity under orthographic projection (adapted from [12]).	11
3.1	Proposed framework for human motion capture from single video sequence and animation using the motion capture data. . . . .	14
3.2	Human model tree. . . . .	17
3.3	The human model used in our implementation. . . . .	18
3.4	The iterative estimation of the torso angle. . . . .	25
3.5	The iterative estimation of the upper left leg angle. . . . .	26
3.6	The iterative estimation of the upper right leg angle. . . . .	27
3.7	The crossed left and right lower legs. . . . .	32
3.8	The iterative estimation of the foreshortening ratio of the torso. .	36
3.9	The iterative estimation of the foreshortening ratio of the upper arm. . . . .	36



4.1	The results obtained by applying the proposed framework to the frames of a public walking video. The input frame (top), the extracted silhouette (middle), and the reconstructed 3D pose of the human in the video (bottom). . . . .	41
4.1	(continued). . . . .	42
4.1	(continued). . . . .	43
4.1	(continued). . . . .	44
4.1	(continued). . . . .	45
4.2	The results obtained by applying the proposed framework to the frames of a public dancing video. The input frame (top), the extracted silhouette (middle), and the reconstructed 3D pose of the human in the video (bottom). . . . .	46
4.2	(continued). . . . .	47
4.2	(continued). . . . .	48
4.2	(continued). . . . .	49
4.2	(continued). . . . .	50

# List of Tables

3.1	The ratios of the lengths of different parts of the human model to the length of the human model. . . . .	17
3.2	The ratios of the radii of different parts of the human model to the length of the human model. . . . .	19
3.3	The joint limits in the human model. . . . .	19

# List of Algorithms

1	The algorithm to find how much the given segment(s) rotated around the normal axis of the image. . . . .	24
2	The algorithm to find the limb angles. . . . .	29
3	The algorithm to find the arm angle. . . . .	30
4	The algorithm to find the elbow corner. . . . .	31
5	The algorithm to detect whether the lower legs cross each other. . . . .	33
6	The algorithm to find the absolute foreshortening. . . . .	35

# Chapter 1

## Introduction

Human pose reconstruction is a popular research area since it can be used in various applications. Motion capture and motion synthesis are expensive and time consuming tasks for articulated figures, such as humans. Human pose estimation based on computer vision principles is inexpensive and widely applicable approach. In computer vision literature the term human motion capture is usually used in connection with large scale body analysis ignoring the fingers, hands and the facial muscles, which is the case in our work.

The motion capture problem we try to solve can be defined as follows: given a single stream of video frames of a performer, compute a 3D skeletal representation of the motion of sufficient quality to be useful for animation. The animation generation is an application of motion capture where the required accuracy is not as high as in some other applications, such as medicine [8].

In this thesis, a model-based framework is proposed to reconstruct the 3D pose of a human for animation from a sequence of video frames obtained from a single view. The proposed framework for pose reconstruction starts with background estimation. Once the background is estimated, the body silhouette is extracted for each frame by using image subtraction. Then, the 2D body segments are automatically labeled on the human body silhouette by using a model-based approach. Finally, the 3D pose is constructed from the labeled human silhouette

by assuming orthographic projection. The approach proposed in this thesis does not require camera calibration and it uses a video sequence obtained from a single camera. The proposed framework assumes that the input video has a static background and it has no significant perspective effects and the performer is in upright position.

Our approach for constructing the 3D pose is close to the approach defined in [16], but in our approach the necessary user interaction is significantly reduced. In order to construct the 3D pose, the joint coordinates of the human figure are needed. Unlike the other approach [16] which gets the corresponding points from the user input, our approach computes these points automatically.

We tested the proposed framework on various video sequences and obtained reasonable constructions of the human figure motion.

## 1.1 Organization of the Thesis

This thesis is organized as follows: Chapter 2 introduces human motion capture techniques and gives a summary of the related work in this area. In Chapter 3, we propose our framework for human motion capture from a video sequence obtained from a single camera and animation by using the captured motion data. In Chapter 4, we give visual results for different videos produced by the proposed framework. We also interpret the results and comment on the performance of the proposed framework. Finally, Chapter 5 gives conclusions.

# Chapter 2

## Background and Related Work

This chapter discusses human motion control techniques, motion capture techniques in general, non-vision based motion capture techniques, vision-based motion capture techniques with and without markers and single image processing techniques to find correspondence points in an image.

### 2.1 Human Motion Control Techniques

There are mainly two motion control techniques for animating articulated figures: *kinematics* and *dynamics*. Kinematics methods use time-based joint rotation values to control the animation while dynamics methods use force-based simulation of movements [11]. Creating data for these techniques can be done manually by talented animators or can be captured automatically by different types of devices [17, 18]. Motion control techniques are summarized in the following sections.

#### 2.1.1 Kinematics

Kinematics approaches obtain the motion parameters by considering position, velocity, and acceleration without being concerned with the forces that cause the

motion. Kinematics methods can be classified into two parts: *Forward Kinematics* and *Inverse Kinematics*. *Forward Kinematics* directly sets the motion parameters like the position, the orientation of joints at specific times for each joint, whereas *Inverse Kinematics* uses nonlinear programming techniques and the positions of the end-effectors to determine the position and the orientation of the joints in the hierarchy.

### 2.1.2 Dynamics

Dynamics approaches obtain the motion parameters by using dynamic motion equations considering the forces, the torques, and the physical properties of the objects. The dynamic methods can be classified into two parts: *Forward Dynamics* and *Inverse Dynamics*. *Forward Dynamics* computes the motion parameters by applying forces on the objects, whereas *Inverse Dynamics* computes the necessary forces for a specific motion.

## 2.2 Motion Capture

Motion capture is an attractive way of creating the motion parameters for computer animation. It can provide the realistic motion parameters. It permits an actor and a director to work together to create a desired pose, that may be difficult to describe with enough specificity to have an animator recreate manually [8]. The application areas of motion capture techniques can be summarized as follows [15]:

***Virtual reality:*** interactive virtual environments, games, virtual studios, character animation, film, advertising

***Smart surveillance systems:*** access control, parking lots, supermarkets, vending machines, traffic.

***Advanced user interfaces:*** advanced user interfaces.

***Motion analysis and synthesis:*** annotations of videos, personalized training , clinical studies of medicine.

Motion capture is an effective way of creating the motion data for an animation [8]. Quality for animations caused challenging requirements on capture systems. To date, capture systems that meet these requirements have required specialized equipment that is expensive. Computer vision can make animation data easier to obtain.

Ideally, the capture of motion data should be easily available, inexpensive. Using standard video cameras is an attractive way of providing these features. The use of a single camera is a particularly attractive way. It offers the lowest cost, simplified setup, and the potential use of legacy sources such as films [8].

Human motion capture systems generate data that represents measured human movement, based on different technologies. According to used technology, human motion capture systems can be classified as non-vision based, vision based with markers, vision based without markers.

### 2.2.1 Non-vision Based Motion Capture

In non-vision based systems, sensors are attached to the human body to collect movement information. Some of them have a small sensing footprint that they can detect small changes such as finger or toe movement [19]. Each kind of sensor has advantages and limitations [15].

Advantages of magnetic trackers:

- real-time data output can provide immediate feedback,
- no post processing is required,
- they are less expensive than optical systems,
- no occlusion problem is observed,



- multiple performers are possible.

Disadvantages of magnetic trackers:

- the trackers are sensitivity to metal objects,
- cables restricts the performers,
- they provide lower sampling rate than some optical systems,
- the marker configurations are difficult to change.

Advantages of electromechanical body suits:

- they are less expensive than optical and magnetic systems,
- real-time data is possible,
- no occlusion problem is observed,
- multiple performers are possible.

Disadvantages of electromechanical body suits:

- they provide lower sampling rate,
- they are difficult to use due to the amount of hardware,
- configuration of sensors is fixed.

### 2.2.2 Vision-Based Motion Capture with Markers

In 1973, Johansson explored his famous Moving Light Display (MLD) psychological experiment to perceive biological motion [10]. In the experiment, small reflective markers are attached to the joints of the human performers. When the

patterns of the movements are observed, the integration of the signals coming from the markers resulted in recognition of actions. Although the method faces the challenges such as errors, non-robustness and expensive computation due to environmental constraints, mutual occlusion and complicated processing, many marker based tracking systems are available in the market.

This is a technique that uses optical sensors, e.g. cameras, to track human movements, which are captured by placing markers upon the human body. Human skeleton is a highly articulated structure and moves in three-dimension. For this reason, each body part continuously moves in and out of occlusion from the view of the cameras, resulting in inconsistent and unreliable motion data of the human body.

One major drawback of using optical sensors and markers, they cannot sense joint rotation accurately. This is a major drawback in representing a real 3D model [20]. Optical systems has advantages and limitations [15].

Advantages of optical systems are as follows:

- they are more accurate,
- larger number of markers are possible,
- no cables restricts the performers,
- they produces more samples per second,

Disadvantages of optical systems :

- they require post-processing,
- they are expensive (between 100,000 and 250,000),
- occlusion is a problem in these systems.
- environment of the capturing must be away from yellow light and reflective noise,

### 2.2.3 Vision-Based Motion Capture without Markers

The marker-based tracking systems are restrictive to some extent due to the mounted markers as discussed in the previous Section 2.2.2. As a less restrictive motion capture technique, markerless-based systems are capable of overcoming the mutual occlusion problem as they are only concerned about boundaries or features on human bodies. This is an active and promising but also challenging research area in the last decade. The research with respect to this area is still ongoing [19].

The markerless-based motion capture technique exploits external sensors like cameras to track the movement of the human body. A camera can be of a resolution of a million pixels. This is one of the main reasons that optical sensors attracted people's attention. However, such vision-based techniques require intensive computational power [5].

As a commonly used framework, 2D motion tracking only concerns the human movement in an image plane, although sometimes people intend to project a 3D structure into its image plane for processing purposes. This approach can be catalogued with and without explicit shape models [19].

The creation of motion capture data from a single video stream seems like a plausible idea. People are able to watch a video and understand the motion, but clearly, the computing the human motion parameters from a video stream is a challenging task [8].

Vision-based motion capture techniques usually include *initialization* and *tracking* steps.

#### 2.2.3.1 Initialization

A system starts its operation with correct interpretation of the current scene. The initialization requires camera calibration, adaptation to scene characteristics and model initialization. Camera calibration is defined as parameters that are

required for translating a point in a 3D scene to its position in the image. Some systems find initial pose and increment it from frame to frame whereas in other systems the user specifies the pose in every single frame. Some systems have special initialization phase where the start pose is found automatically whereas in others the same algorithm is used both for initialization and pose estimation [14].

### 2.2.3.2 Tracking

Tracking phase extracts specific information, either low level, such as edges, or high level, such as head and hands. Tracking consists of three parts [14]:

1. *Figure-ground segmentation*: the human figure is extracted from the rest of the image.
2. *Representation*: segmented images are converted to another presentation to reduce the amount of information.
3. *Tracking over time*: how the subject should be tracked from frame to frame.

## 2.3 Single Image Processing

In this section, we explain how to find a corresponding point in an image according to orthographic projection as explained in [16].

A point in 3D when projected to orthographic scene can be represented by 2D coordinates. Let  $(X, Y, Z)$  be a point in the 3D world and  $(u, v)$  be its projection on the 2D plane.

$$\begin{pmatrix} u \\ v \end{pmatrix} = s \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \quad (2.1)$$

These two points can be expressed with Equation 2.1 [16] under scaled orthographic projection. Figure 2.1 shows the projection of a line with length  $l$ , onto the image under scaled orthographic projection. The projection of the two end points  $(X_1, Y_1, Z_1)$  and  $(X_2, Y_2, Z_2)$  are represented by  $(u_1, v_1)$  and  $(u_2, v_2)$  on the image respectively. If the scale factor  $s$  of the projection model is known, we can calculate the relative depth of the line denoted by  $\partial Z$  as shown in Equation 2.2 [16].

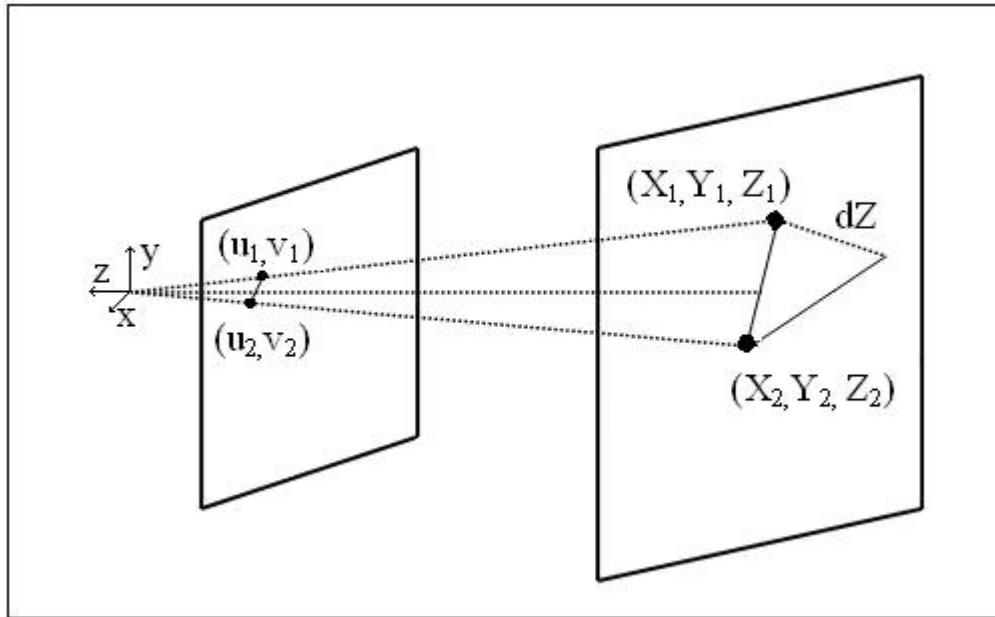


Figure 2.1: Projection of a line (adapted from [16]).

$$\begin{aligned}
 l^2 &= (X_1 - X_2)^2 + (Y_1 - Y_2)^2 + (Z_1 - Z_2)^2 \\
 (u_1 - u_2) &= s(X_1 - X_2) \\
 (v_1 - v_2) &= s(Y_1 - Y_2) \\
 \partial Z &= (Z_1 - Z_2) \\
 \partial Z &= \sqrt{l^2 - \frac{((u_1 - u_2)^2 + (v_1 - v_2)^2)}{s^2}}
 \end{aligned}
 \tag{2.2}$$

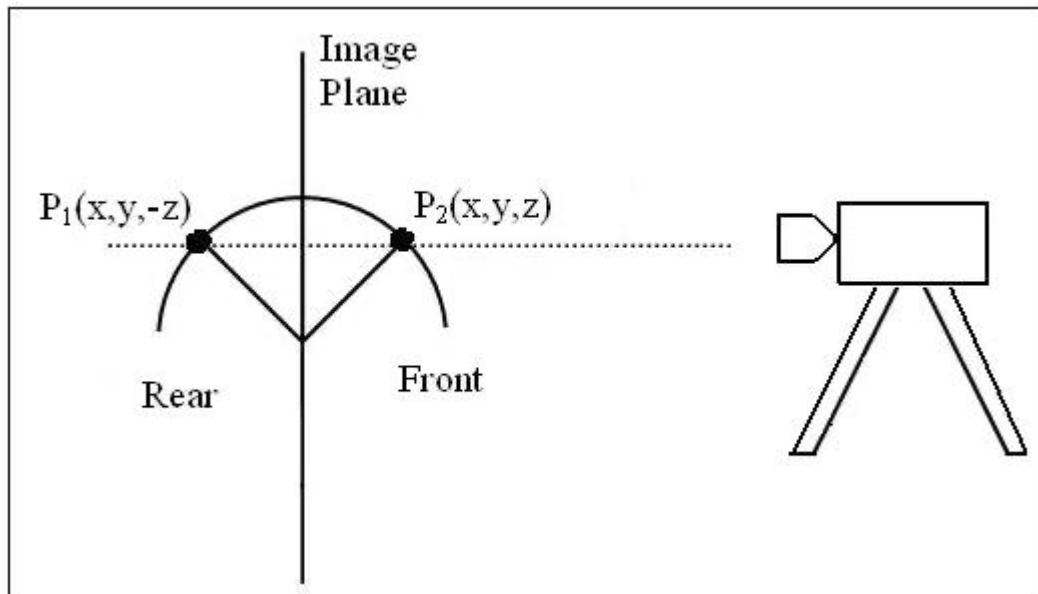


Figure 2.2: The ambiguity under orthographic projection (adapted from [12]).

These analysis shows us how to compute the 3D corresponding of a point on the image as a function of the scale parameter  $s$ . In this thesis we use the height of human observed on the image to compute scale parameter  $s$ . We use the  $s$  parameter found from the height to all body parts to find 3D joint coordinates. For the computed value of  $s$  two distinct solutions are possible. We do not know whether the each segment on the image is at rear or at front. This ambiguity is shown in Figure 2.2.

## Chapter 3

# Motion Capture and Animation System

Capturing the human motion parameters has various challenging problems. Thus, the solution for such a problem requires various methods and algorithms. In this section we give an overview of our framework, and the details of each stage.

The proposed framework finds the posture of a human if the joint correspondences on the image and the depth direction of the each body segment are provided. The user specifies the joint correspondences on the image and the depth direction of body segments by using the mouse. In [16], including the joint correspondences, are provided by the user. Our aim is to use image processing algorithms and computer graphics and computer vision techniques to extract the human motion parameters (namely body postures) with minimal user efforts. When the joint angles are calculated, remaining information needed to construct 3D pose is the depth information (inward or outward) of each body part, which rarely changes during a video frame sequence. The user only has to interact whenever a body segment changes its depth direction.



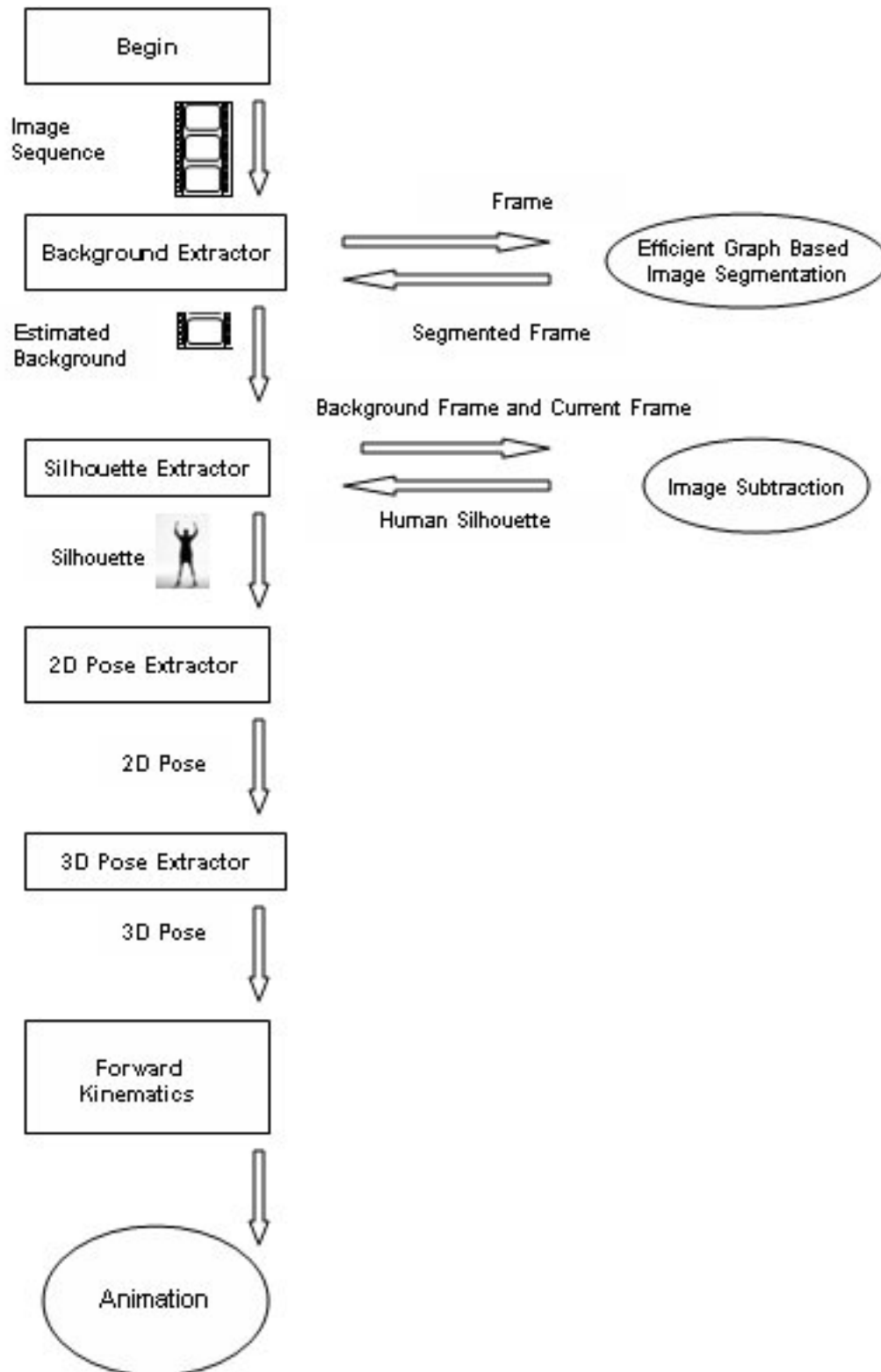


Figure 3.1: Proposed framework for human motion capture from single video sequence and animation using the motion capture data.

## 3.1 Proposed Framework

The proposed framework for human motion capture and animation is shown in Figure 3.1. For capturing the 3D pose from the given image sequence the first stage estimates the background of the video to find the silhouette of the performer on each frame of the video. Then, we find the silhouette of the performer by subtracting the estimated background from each frame. The extracted silhouette of the performer is given as input to the 2D pose extractor. The 2D pose extractor finds the joint coordinates of the performer on the image by using a model-based approach. A 2D stick human model (see Figure 3.3) is fitted onto the silhouette. In this way, the joint coordinates of the performer are matched to the joint coordinates of the 2D stick human model. The joint coordinates of the 2D stick model will be given as input to the 3D pose estimator. The 3D pose estimator computes the 3D joint configurations from the 2D joint coordinates and the depth information of the human model for each body part. Finally, the 3D human model is animated based on the 3D joint configurations computed from the video sequence.

### 3.1.1 Human Modeling

The computer generated human model should make simple the accurate positioning of the segments during motion, deform the skin where the muscles and tissue are taken into account realistically during the movement, generate realistic facial expressions, realistic modeling of hair, etc. All of these issues are research topics in their own right [13]. There are standardization efforts to address these issues, such as MPEG-4 [1] or Humanoid Animation (H-Anim) standard [2] developed by The Humanoid Animation Working Group of Web3D Consortium. Since the aim of our work is to extract the posture of a human in an image sequence, we do not go into details of human body modeling and animation. But human body model used to find human pose must be explained before going further. We describe the human model used for finding the human pose below.

Human model is represented as a collection of simple rigid objects connected by joints in a hierarchical manner. These models, called articulated bodies, can have various degrees of articulation. The number and the hierarchy of joints and limbs and the degrees of freedom (DOF) of the joints determine the complexity of model. The DOF of a joint is the independent position variable that is necessary to specify the state of a joint. The joints can rotate in one, two, or three orthogonal directions. The number of these orthogonal directions determines DOF of a joint. A human skeleton may have many DOFs. However, when the number of DOFs increases, the methodology, which is used for controlling the joints, becomes more complex.

For the sake of our model-based framework, we have to think about tradeoff between the accuracy of the representation and the number of parameters for the model that needs to be estimated. In our work, we are interested in large body movements of human motion. Hands or facial expressions are not considered. To reduce the computational complexity of the model we use a simple 3D articulated human model to capture the motion. Our articulated human model consists of 10 cylindrical parts representing head, torso, right upper leg, right lower leg, left upper leg, left lower leg, right upper arm, right lower arm, left upper arm, and left lower arm (see Figure 3.2). Each cylindrical part has two parameters: *radius* and *length*. For each cylindrical part there are up to three rotation parameters:  $\theta_X$ ,  $\theta_Y$ , and  $\theta_Z$ . Totally, there are 23 DOFs for the human model: 3 DOFs for the global positioning of the human body, 1 DOF for the head, 3 DOFs for the torso, 2 DOFs for the right upper leg, 2 DOFs for the right lower leg, 2 DOFs for the left upper leg, 2 DOFs for the left lower leg, 2 DOFs for the right upper arm, 2 DOFs for the right lower arm, 2 DOFs for the left upper arm, 2 DOFs for the left lower arm (see Figure 3.3).

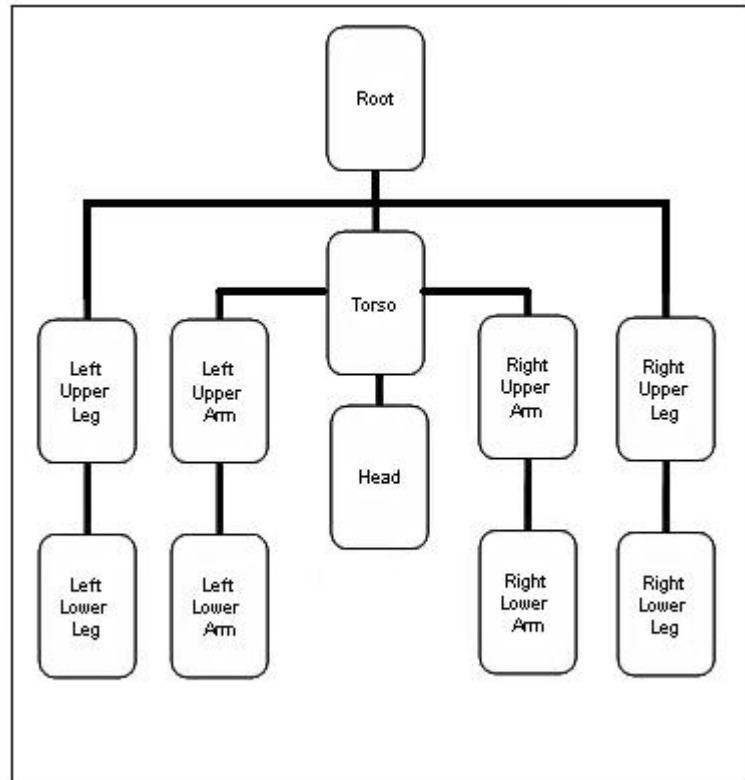


Figure 3.2: Human model tree.

Table 3.1: The ratios of the lengths of different parts of the human model to the length of the human model.

Length	Ratio
Height	175/175
Head	25/175
Torso	52/175
Upper arm	25/175
Lower arm	35/175
Upper leg	46/175
Lower leg	52/175

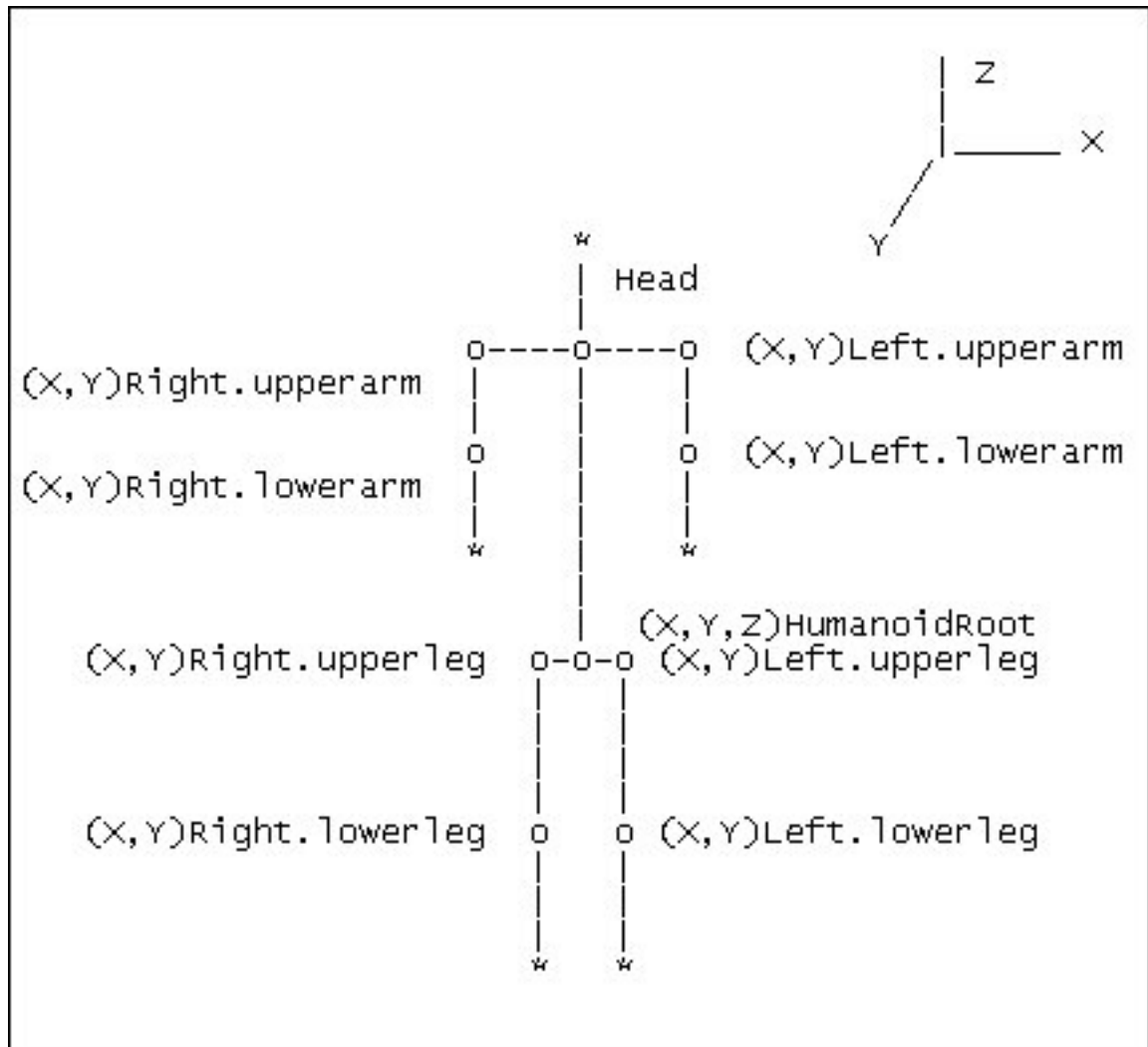


Figure 3.3: The human model used in our implementation.

Table 3.2: The ratios of the radii of different parts of the human model to the length of the human model.

Radius	Ratio
Head	20/175
Torso	40/175
Upper arm	10/175
Lower arm	10/175
Upper leg	20/175
Lower leg	20/175

Table 3.3: The joint limits in the human model.

Body part	Start limit (degrees)	End limit (degrees)
Neck	45	135
Waist	45	135
Right hip	240	300
Left hip	240	300

### 3.1.2 Background Estimation

We need to estimate background to extract the silhouette of the performer on the video. If the single video is prepared by the user then we can get exact background of video easily. If the video we are working is taken from a public resource we have to estimate the background. We do that by finding the regions on video frames that do not change. We label these regions as background. By analyzing a few frames, the background is estimated. We use an efficient graph-based image segmentation algorithm to decrease the lighting effect when deciding whether a region is changing or not [7]. Pixels that are inside the same segment are set to the same color.

#### 3.1.2.1 Efficient Graph Based Image Segmentation

We use the graph-theoretic approach proposed by Felzenszwalb and Huttenlocher for image segmentation [7]. In this approach, they define a predicate for measuring the evidence for a boundary between two regions using a graph-based representation of the image. They measure the evidence for a boundary by comparing intensity differences across the boundary and the intensity differences between the neighboring pixels within each region. This segmentation algorithm produces segmentations that satisfy global properties.

An undirected graph  $G = (V, E)$  is constructed for the image. Each vertex  $v_i \in V$  corresponds to a pixel in the image and each edge  $(v_i, v_j) \in E$  corresponds to the pairs of neighboring pixels. Each edge  $(v_i, v_j) \in E$  has a weight  $w(v_i, v_j)$ , which is a non-negative measure of dissimilarity between the two pixels of the edge. A segmentation  $S$  is a partition of the vertex set  $V$  into components such that each component  $C \in S$  corresponds to a connected component in a graph  $G' = (V, E')$  where  $E' \subseteq E$ .

In other words, any segmentation is a subset of the edges in  $E$ . In general, we want the elements in a component to be similar and the elements in different components to be dissimilar. This means that the edges between two vertices in

the same component should have relatively low weights, and the edges between vertices in different components should have higher weights.

The algorithm is computationally efficient, running in  $O(n \log n)$  time for  $n$  image pixels and with low constant factors, which makes it suitable for video applications.

### 3.1.3 Silhouette Extraction

Silhouette is the main feature extracted from the video frames and used in our framework. Our 2D pose extractor takes the silhouette of performer on the frame as input and process the silhouette to find 2D joint coordinates of the performer.

We use the background estimation method explained in the previous section to detect the silhouette on a frame. We use simple thresholding image subtraction to detect the body silhouette on the image. The main goal of the image subtraction algorithms is to detect foreground objects. In our case, the foreground object is the silhouette of the performer. We detect the silhouette on a frame as the difference between the frame and the background. We apply the threshold value to decide whether pixel of the frame belongs to the silhouette or the background. If the absolute value of the difference between the pixel of the frame with the background pixel is greater than the specified threshold, then the pixel is taken as a silhouette pixel, otherwise the pixel is regarded as a background pixel.

### 3.1.4 2D Pose Extractor

The aim of this part to find the joint coordinates of the performer by using the human silhouette. We extract the joint coordinates of human actor by using a model-based technique. We model the human as an assembly of cylinders. We match the silhouette with the human model iteratively. After fitting the human model onto the silhouette, we can use the joint coordinates of the human model as the joint coordinates of the performer.



During the 2D pose extraction, we do not know whether a leg is a left leg (arm) or right leg (arm). When we talk about the 2D pose extraction we use left leg to mean the leg that is on the left of image. The distinction of left and right is actually done during the 3D pose estimation. For the 2D pose extraction we have to be sure that the segments on the left of image are identified as left and the segments on the right of the image are identified as right.

The process of finding the 2D pose starts by detecting the torso location. We locate the  $y$  coordinate of the torso from the relative ratios shown in Table 3.1. We take the horizontal middle point of the silhouette as  $x$  coordinate of the torso. Then by using Algorithm 1, we find how much the torso is rotated around the normal axis of image plane, as shown in Figure 3.4. Then by using Algorithm 1, we detect the head rotation. We analyze the contour of the silhouette by using Algorithm 2 to find how much the lower left leg angle, lower right leg angle, upper left arm angle, upper right arm angle, lower left arm angle, lower right arm angle rotated around the normal axis of the image plane. Then, we find the rotation angle of the upper left leg and the upper right leg by Algorithm 1, as shown in Figures 3.5 and 3.6. After finding each rotation angle we find the foreshortening of each segment by using Algorithm 6. Finding foreshortening of the upper arm and the torso is shown in Figures 3.9 and 3.8, respectively.

#### 3.1.4.1 Finding Orientation of Body Parts

Algorithm 1 finds how much one or more body segments rotated around the normal axis of the image. The technique used in Algorithm 1 is close to the method used in [9]. Before going further we have to define the similarity between to images.

#### 3.1.4.2 Measuring Similarity

We measure similarity between two images  $I_1$  and  $I_2$  by an operator  $S(I_1, I_2)$  as described in [9]. The similarity operator only considers the area difference

between the two shapes; i.e, the ratio of the positive error  $p$  (represents the ratio of the number of pixels in the silhouette but not in the human model to the total number of pixels of the human model and the silhouette) and the negative error  $n$  (represents the ratio of the number of pixels in the human model but not in the silhouette to the total number of pixels of the human model and the silhouette) which are calculated as

$$p = \frac{(I_1 \cap I_2^C)}{(I_1 \cup I_2)} \quad (3.1)$$

$$n = \frac{(I_2 \cap I_1^C)}{(I_1 \cup I_2)} \quad (3.2)$$

where  $I^C$  denotes the complement of  $I$ . The similarity between the two shapes  $I_1$  and  $I_2$  is calculated as

$$S(I_1, I_2) = e^{-p-n} (1 - p) \quad (3.3)$$

Algorithm 1 takes the fragment of the human silhouette that contains the body segments to be searched as input. The algorithm also takes the lower and upper joint angle limits. The lower and upper joint angle limits used in our implementation are listed in Table 3.3.

```

S ← PartialSilhouetteContainingSegmentsToBeSearched;
startAngle ← startAngle;
endAngle ← endAngle;
numberOfDivision ← 8;
minimumStepAngle ← 1°;
searchAngleInterval ← (endAngle − startAngle);
stepAngle ← (searchAngleInterval/numberOfDivision);
angleThatMaximizes ← (endAngle + startAngle)/2;
maximumSimilarity ← 0;
while searchAngleInterval > minimumStepAngle do
  | angForBegin ← angleThatMaximizes − searchAngleInterval/2;
  | if angForBegin < startAngle then
  |   | angForBegin ← startAngle;
  | end
  | angForEnd ← angleThatMaximizes + searchAngleInterval/2;
  | if angForEnd > endAngle then
  |   | angForEnd ← endAngle;
  | end
  | if stepAngle is 0 then
  |   | stepAngle ← minimumStepAngle;
  | end
  | angleForBodyPart ← angForBegin;
  | while angleForBodyPart ≤ angForEnd do
  |   | currentModelPose ← DrawBodyPart();
  |   | similarityResult ←
  |   | MeasureSimilarity(S, currentModelPose);
  |   | if similarityResult > maximumSimilarity then
  |   |   | maximumSimilarity ← similarityResult;
  |   |   | angleThatMaximizes ← angleForBodyPart;
  |   | end
  |   | angleForBodyPart ← angleForBodyPart + StepAngle;
  | end
  | searchAngleInterval ←
  | 2 × (searchAngleInterval/numberOfDivision);
  | stepAngle ← 2 × (stepAngle/numberOfDivision);
end
return angleThatMaximizes

```

**Algorithm 1:** The algorithm to find how much the given segment(s) rotated around the normal axis of the image.

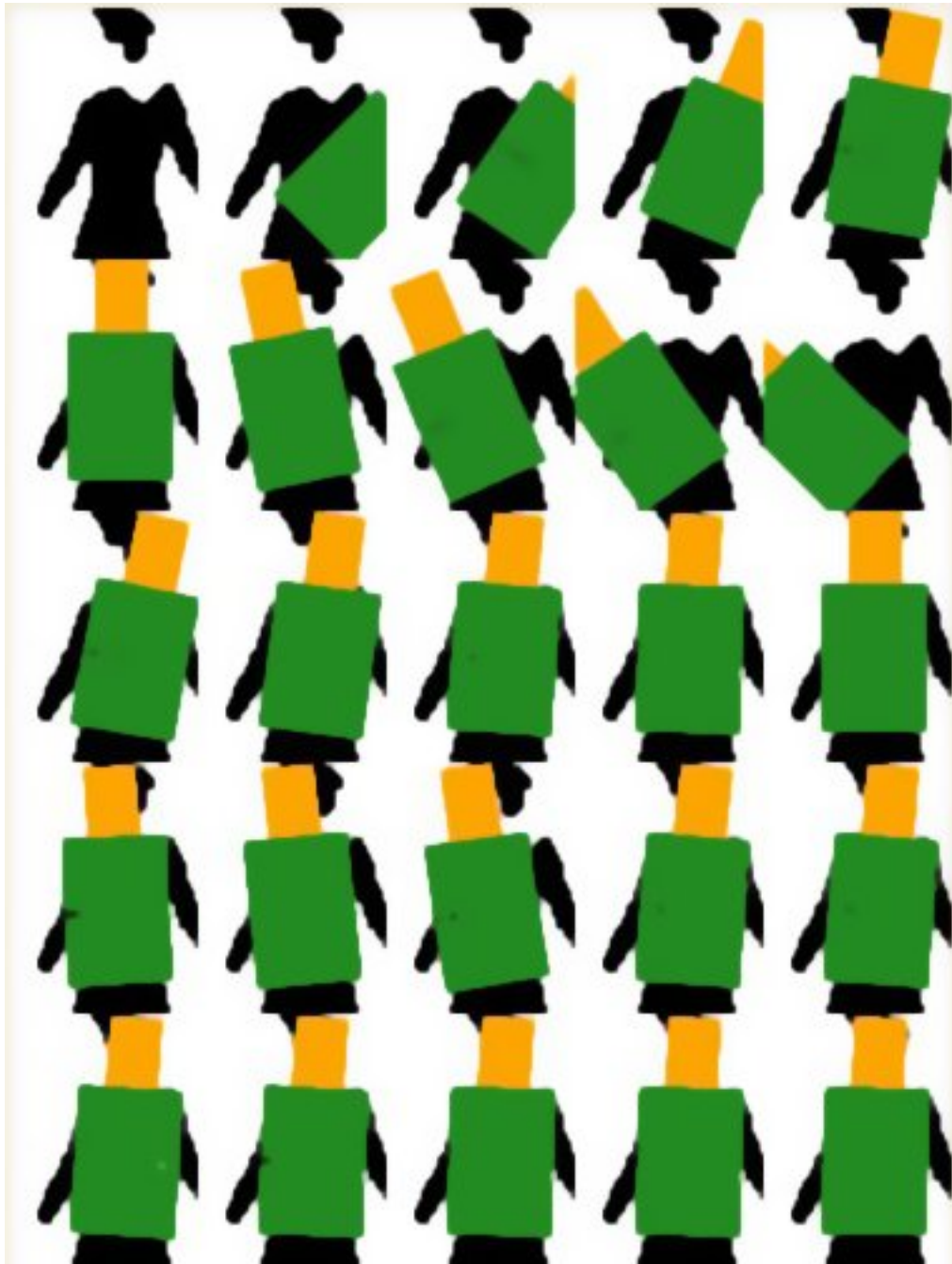


Figure 3.4: The iterative estimation of the torso angle.



Figure 3.5: The iterative estimation of the upper left leg angle.

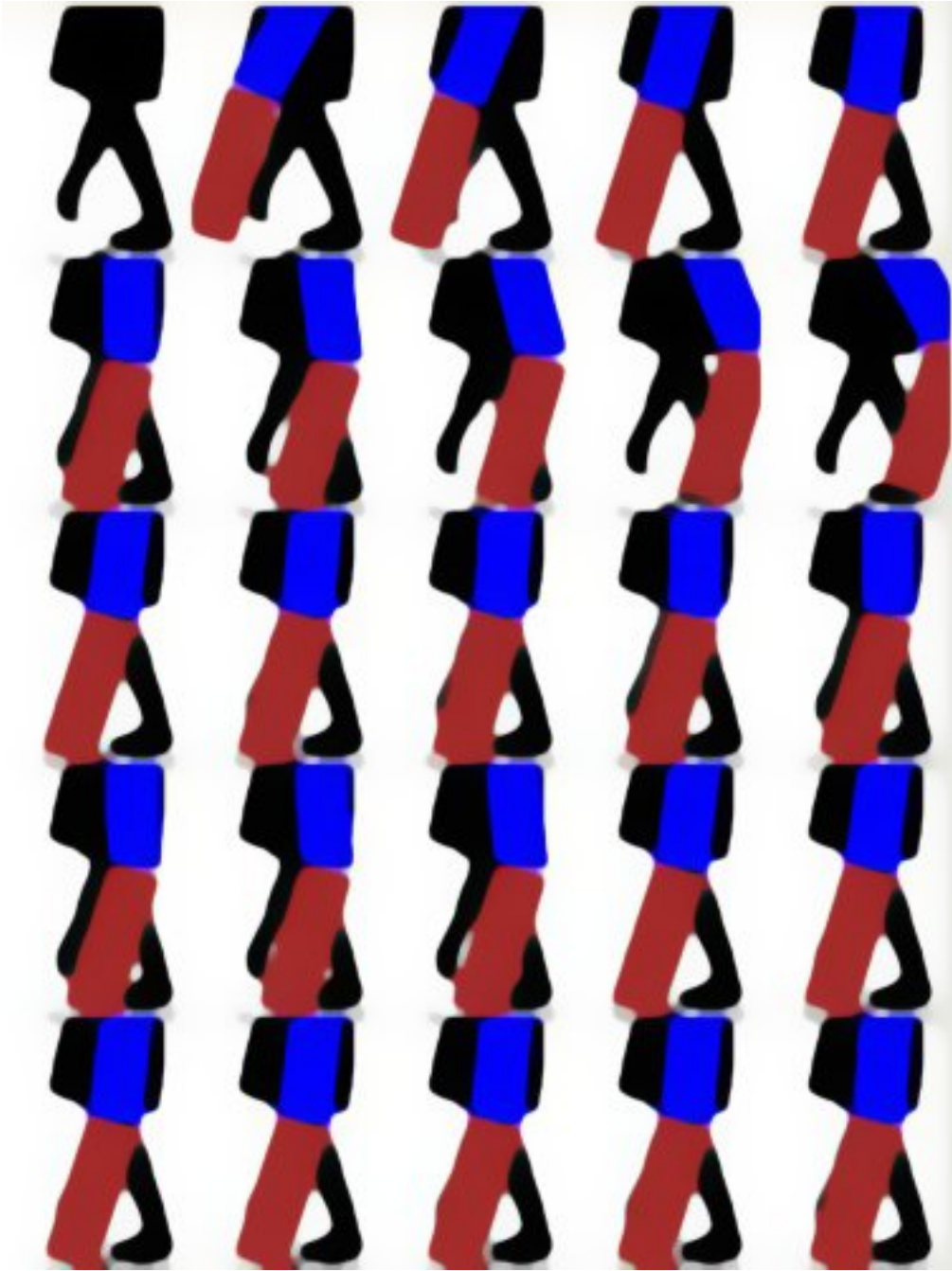


Figure 3.6: The iterative estimation of the upper right leg angle.

We divide the angle range specified by the lower and upper limits to a number of intervals and we measure how well the silhouette is covered by the segments for each interval. We find the division (angle) that covers the silhouette best. We narrow the search interval at each iteration by selecting the next search interval for the joint angle as the division that gives best covering result merged with its neighbor divisions since there is a possibility the best angle can be in these divisions. We recursively continue to narrow the search interval and when the search interval is smaller than a threshold value the algorithm returns the angle that corresponds to the best fitting. Examples of the application of this algorithm for the torso, the upper left leg, and the upper right leg are shown in Figures 3.4, 3.5, and 3.6, respectively.

#### 3.1.4.3 Contour Analyzing Method

This method is used to find how much the segments of the body (lower left leg, upper right leg, upper left arm, lower left arm, upper right arm, and lower right arm) rotated around the normal axis of the image plane. We use contour analysis to find angles. We do not use Algorithm 1 for the arm segments because of the high possibility of occlusion with other parts. For lower legs, we cannot use Algorithm 1 because we have to first find the upper legs. This is because we do not know the end points of the upper legs. Trying to find the upper leg angle by using Algorithm 1 is also not appropriate. Since the upper legs occluded by cloths or being adjacent to each other that makes it difficult to find the joint angles for them. For this reason, we use Algorithm 2 that is based on contour analysis on the lower legs. After finding the angle of the lower legs by using Algorithm 2 we find upper leg angles by using Algorithm 1. While finding the upper leg angle with Algorithm 1 we combine the upper leg and the lower leg into a single unit, as shown in Figures 3.5 and 3.6. For some segments, Algorithm 1 gives better results if the segment is combined with other segments. For example, while we are determining the torso angle, the torso and the head are also taken together as shown in Figure 3.4. The head helps to find the torso angle easily. By the same logic, since the lower leg angles are found with the contour analysis method, knowing the lower leg angle helps to find the upper leg angles easily as shown in

Figures 3.5 and 3.6.

The contours of the left arm, the right arm and the lower left leg and the lower right leg are extracted from the silhouette by traversing horizontal scan lines over the silhouette. For this purpose, a curve is extracted for each part. For the lower left leg and the lower right leg we compute the joint angle from the curves as shown in Algorithm 2.

```

S ← Silhouette;
LeftArmPoints ← GetLeftArmCurve(S );
RightArmPoints ← GetRightArmCurve(S );
LeftLowerKneePoints ← GetLeftLowerKneeCurve(S );
RightLowerKneePoints ← GetRightLowerKneeCurve(S );
/*LLA stands for LeftLowerArm */
/*LUA stands for LeftUpperArm */
/*RLA stands for RightLowerArm */
/*RUA stands for RightUpperArm */
/*LLK stands for LeftLowerKnee */
/*RLK stands for RightLowerKnee */
LeftArmAngles ← FindArmAngles(LeftArmPoints);
LLAAngle ← LeftArmAngles.Lower;
LUAAngle ← LeftArmAngles.Upper;
RightArmAngles ← FindArmAngles(RightArmPoints);
RLAAngle ← RightArmAngles.Lower;
RUAAngle ← RightArmAngles.Upper;
LLKAngle ← FindAngle(LeftLowerKneePoints);
RLKAngle ← FindAngle(RightLowerKneePoints);
LLKLineList ← KneeScanLinesLeft(S );
RLKLineList ← KneeScanLinesRight(S );
if
KneeCrossed(LeftLowerKneeLineList,RightLowerKneeLineList)
then
| Swap(LLKAngle,RLKAngle );
end
return LLAAngle, LUAAngle, RLAAngle, RUAAngle, LLKAngle,
RLKAngle

```

**Algorithm 2:** The algorithm to find the limb angles.



```

armPointList ← armpointlist;
lowerArmAngle ← 0;
upperArmAngle ← 0;
elbowCornerIndex ← FindElbowCorner(armPointList );
if elbowCornerIndex is -1 then
    | angle ← FindAngle(armPointList );
    | lowerArmAngle ← angle ;
    | upperArmAngle ← angle ;
else
    | startIndexUpper ← 0;
    | endIndexUpper ← elbowCornerIndex ;
    | startIndexLower ← elbowCornerIndex ;
    | endIndexLower ← ArmPointList.Length - 1;
    | UpperArmPoints ←
    | GetPoints(armPointList,startIndexUpper,endIndexUpper);
    | LowerArmPoints ←
    | GetPoints(armPointList,startIndexLower,endIndexLower);
    | lowerArmAngle ← FindAngle(LowerArmPoints);
    | upperArmAngle ← FindAngle(UpperArmPoints);
end
return lowerArmAngle,upperArmAngle

```

**Algorithm 3:** The algorithm to find the arm angle.

```

armPointList ← ArmPointList;
minimumCornerDistance ← 4;
minimumCornerAngleDifference ← 10;
listOfPossibleCorners ← CreateEmptyList();
for i ← minimumCornerDistance to armPointList.Length - 1 do
    startIndexUpper ← 0;
    endIndexUpper ← i;
    startIndexLower ← i;
    endIndexLower ← armPointList.Length - 1;
    upperArmPoints ←
    GetPoints(armPointList,startIndexUpper,endIndexUpper);
    lowerArmPoints ←
    GetPoints(armPointList,startIndexLower,endIndexLower);
    ArmAngles.Lower ← FindAngle(lowerArmPoints);
    ArmAngles.Upper ← FindAngle(upperArmPoints);
    cornerAngleDifference ←
    Absolute(ArmAngles.Upper-ArmAngles.Lower);
    if
    cornerAngleDifference ≥ minimumCornerAngleDifference
    then
        Add(listOfPossibleCorners,i,cornerAngleDifference);
    end
end
cornerIndex ← -1;
if listOfPossibleCorners.Length ≥ 0 then
    cornerIndex ←
    FindIndexOfMaxDifference(listOfPossibleCorners);
end
return cornerIndex

```

**Algorithm 4:** The algorithm to find the elbow corner.

For the arms, we have to find two angles: one for the lower arm and one for the upper arm. We extract the two angles from the arm curves by trying to find potential elbow corners on their point lists. We use Algorithm 4 to find the elbows. Algorithm 4 tries some points as if they are elbow corners and checks whether enough angle difference exists for the upper and lower segments. If there are more than one candidate point to be the elbow, Algorithm 4 selects the elbow point that gives the largest angle difference between the upper arm and the lower arm. If an elbow is found, the arm curve is divided into two pieces from the elbow and the corresponding angle is computed for each segment like in lower leg segments. If there is no elbow detected then the angle is computed from the whole arm curve and the lower and upper arm are assigned the same angle value that is computed from the whole arm curve.



Figure 3.7: The crossed left and right lower legs.

If the legs cross each other, as in Figure 3.7, we cannot determine which one is left and which one is right. We use Algorithm 5 to detect whether the lower legs cross each other. If Algorithm 5 decides that the legs cross each other, then we swap the left and right lower leg in 2D pose extraction process.

```

/*LLL stands for LeftLowerLeg */
/*RLL stands for RightLowerLeg */
LLLLineList ← LeftLowerLegLineList;
RLLLineList ← RightLowerLegLineList;
OneKneeStarted ← false;
TwoKneeStarted ← false;
TwoKneeCrossed ← true;
for i ← 0 to LLL.Length - 1 do
    if LLLLLineList[i] intersects RLLLineList[i] then
        if TwoKneeStarted then
            TwoKneeCrossed ← false;
            break;
        end
        OneKneeStarted ← true;
    else
        if OneKneeStarted then
            TwoKneeStarted ← true;
        else
            TwoKneeCrossed ← false;
            break;
        end
    end
end
if TwoKneeStarted is false then
    TwoKneeCrossed ← false;
end
return TwoKneeCrossed

```

**Algorithm 5:** The algorithm to detect whether the lower legs cross each other.

Algorithm 5 goes over the lower parts of the silhouette by following the scan lines and produces a line segment list for the parts inside the silhouette. Then it analyzes the list of lines to decide whether lower legs cross each other. If we observe that the line lists at the beginning of lower silhouette parts start with one line and becomes two lines for the remaining scan lines then we conclude that the legs cross each other. In such a case, we swap the angle values of the left lower leg with that of the right lower leg.

#### 3.1.4.4 Finding Foreshortening Angle of a Body Part

We can find how much a segment is foreshortened based on the orthographic projection. However, we cannot detect the direction of the foreshortening. We get foreshortening direction from the user. Algorithm 6 finds how much one or more body segments foreshortened. The algorithm does not care the direction of the foreshortening. The algorithm finds the relative ratio that is a measure of the foreshortening angle.

Algorithm 6 takes the fragment of the human silhouette that contains the body segment to be searched as input. The algorithm tries to find the best ratio that covers the segment being searched. The algorithm starts with an initial target ratio interval  $(0, 1)$ . Then, it iteratively narrows the target interval and finds the best ratio that covers the segment. At each iteration, ratio interval is divided to a number of intervals we specified and for each ratio value that corresponds to each division, we measure how well the silhouette is covered by the segment. We find the division (ratio) that covers the silhouette best. We recursively continue to narrow the search interval and when the search interval is smaller than a threshold ratio, the algorithm returns the ratio that maximizes the fitting. Examples of the application of this algorithm are shown in Figures 3.8 and 3.9.

```

S ← PartialSilhouetteContainingSegmentsToBeSearched;
beginPoint ← beginPoint;
endPoint ← beginPoint;
length ← DistanceOf(beginPoint, endPoint) ;
startRatio ← 0;
endRatio ← 1;
numberOfDivision ← 5;
minimumStepRatio ← 1/length;
searchRatioInterval ← (endRatio – startRatio);
StepRatio ← (searchRatioInterval/numberOfDivision);
RatioThatMaximizes ← (endRatio + startRatio)/2;
maximumSimilarity ← 0;
while searchRatioInterval > minimumStepRatio do
    ratioForBegin ← RatioThatMaximizes – searchRatioInterval/2;
    if ratioForBegin < startRatio then
        | ratioForBegin ← startRatio;
    end
    ratioForEnd ← RatioThatMaximizes + searchRatioInterval/2;
    if ratioForEnd > endRatio then
        | ratioForEnd ← endRatio;
    end
    ratioForBodyPart ← ratioForBegin;
    while ratioForBodyPart ≤ ratioForEnd do
        | currentModelPose ← DrawBodyPart();
        | similarityResult ←
        | MeasureSimilarity(S, currentModelPose);
        | if similarityResult > maximumSimilarity then
        | | maximumSimilarity ← similarityResult;
        | | RatioThatMaximizes ← ratioForBodyPart;
        | end
        | ratioForBodyPart ← ratioForBodyPart + StepRatio;
    end
    searchRatioInterval ←
    2 × (searchRatioInterval/numberOfDivision);
    StepRatio ← 2 × (StepRatio/numberOfDivision);
end
return RatioThatMaximizes

```

**Algorithm 6:** The algorithm to find the absolute foreshortening.



Figure 3.8: The iterative estimation of the foreshortening ratio of the torso.



Figure 3.9: The iterative estimation of the foreshortening ratio of the upper arm.

### 3.1.5 3D Pose Estimation

This stage estimates the 3D pose of the performer on the video. Given 2D joint coordinates and foreshortening direction of each body segment, 3D pose can be constructed under orthographic projection.

For the whole body, we take the orientation of the body as input from the user. The body can be in one of the six orientations: left, right, backward left, backward right, forward left, forward right. We also use horizontal foreshortening of the torso with the orientation of the body taken from the user to find how much the performer is rotated around the  $y$ -axis of the image. From the orientation, we also determine the left leg (arm) and the right leg (arm). Finally, we use the 3D pose found for each key frame to animate the character. We use linear interpolation to calculate the intermediate poses of the human body.



# Chapter 4

## Experimental Results

### 4.1 Visual Results

We tried our method on two videos that we got from public resources. The first one is a walking video [3] and the second one is a dancing video [4]. The walking sequence consists of 82 frames and it is illustrated in Figures 4.1. The dancing sequence consists of 200 frames and it is illustrated in Figures 4.2. The video frame, which our method is applied on, and the extracted silhouette from the video frame and the pose we constructed is depicted in each figure. We created an animation for each video by using key-framing technique. For dancing video, we applied our method to 33 key-frames. For the walking video, we applied our method to 17 key-frames. While selecting the key-frames, we preferred the ones with low occlusion in order to get better results. We used linear interpolation technique to create intermediate frames. In dancing and walking videos we needed user interaction in a few frames to find the 2D joint coordinates because of high occlusion.

## 4.2 Performance Analysis

We implemented the proposed system on a PC platform with Intel Celeron 2.70 GHz processor, 512 MB Main Memory and Intel(R) 82852/82855 GM/GME graphics card. We implemented the proposed system by using Microsoft C#.NET platform.

It takes nearly one second to find the 2D joint coordinates of a human body from a single video frame. The video frame size does not affect the performance significantly since we normalize the silhouette to a fixed size.

## 4.3 Effectiveness of the Proposed Framework

In previous works using the orthographic projection, the user specifies the joint coordinates for each video frame and the foreshortening direction of each segment. In our proposed framework we find the 2D joint coordinates from image automatically and only take the foreshortening direction and the body orientation. Our proposed framework only needs the user input that rarely changes during a motion, which requires minimal user intervention.

In the walking video we experimented, the orientation of the body does not change during the whole video (82 frames). In the dancing video we experimented, the orientation of the body changes 7 times along the 200 frames. For the interactions required to specify the foreshortening direction, 52 changes occurred along the 82 frames in the walking video, 44 changes occurred along the 200 frames in the dancing video. In the dancing video, 8 user inputs are needed, and in the walking video, 7 user inputs needed for the highly occluded parts in a few frames. 14 user inputs (coordinates of joints) are needed for each video frame in previous methods [16], which are determined automatically in our implementation.

For the walking video, if we use the approach presented in [16] we need  $17 \times 14 + 52 = 290$  user inputs, where 17 is the number of key frames used to prepare the animation, 14 is the number of joint coordinates specified for each frame,

52 is the number of user interactions needed for specifying the foreshortening directions. But with our proposed work  $52 + 1 + 7 = 60$  user inputs are needed. 52 is the number of user interactions needed for the foreshortening directions, 1 is the number of the user interactions needed for the orientation of the body and 7 is the number of user interactions needed for highly occluded parts.

For the dancing video, if we use the approach presented in [16], we need  $33 \times 14 + 44 = 506$  user inputs, where 33 is the number of frames used to prepare the animation, 14 is the number of user interactions necessary for specifying the joint coordinates and 44 is the number of user interactions needed for specifying the foreshortening directions. But with our proposed work  $44 + 7 + 8 = 59$  user inputs needed, where 44 is the number of user interactions needed for specifying the foreshortening directions, 7 the number of user interactions needed for specifying the orientation of the body and 7 the number of user interactions needed for specifying the joint angles of the highly occluded segments.

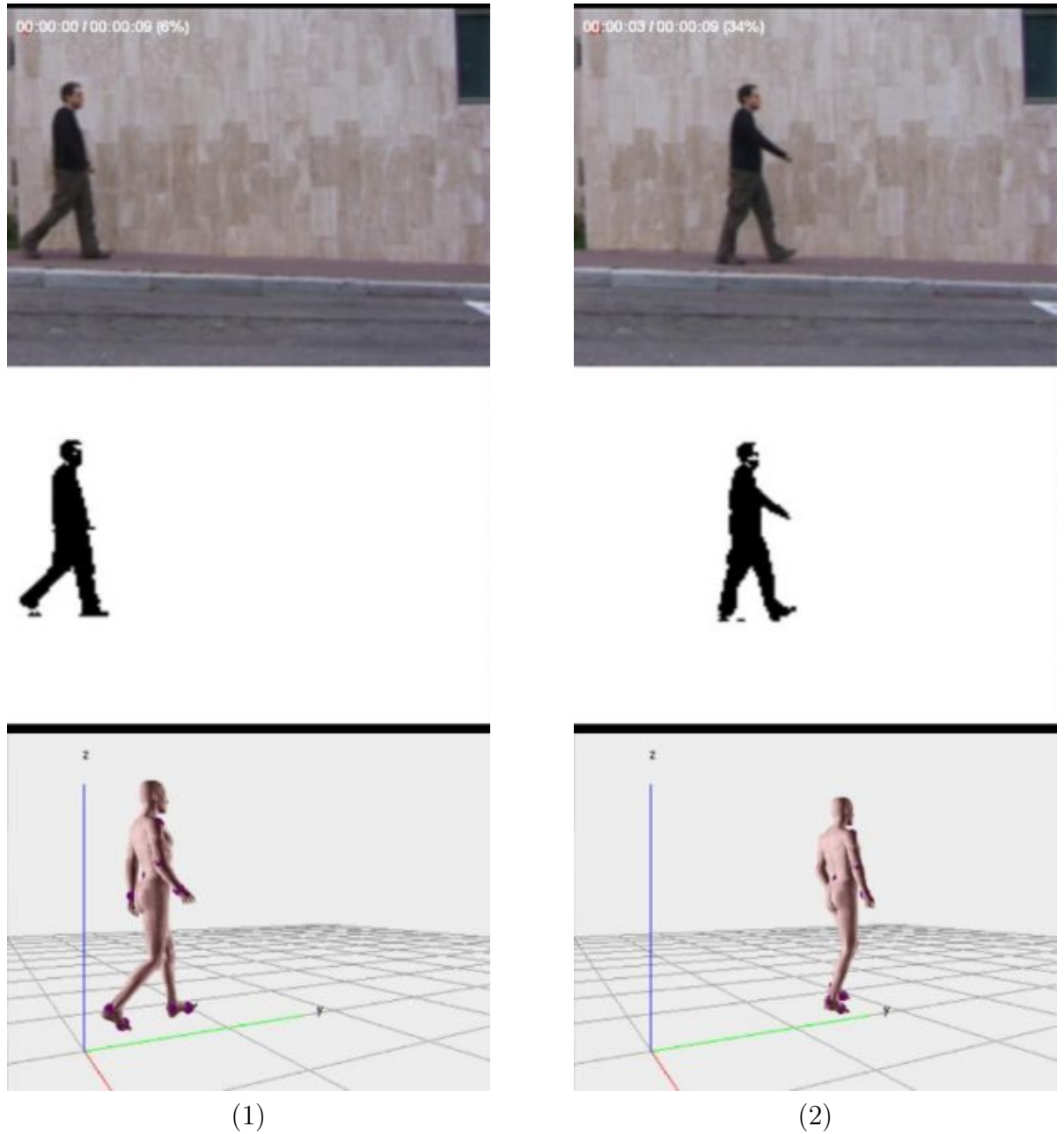
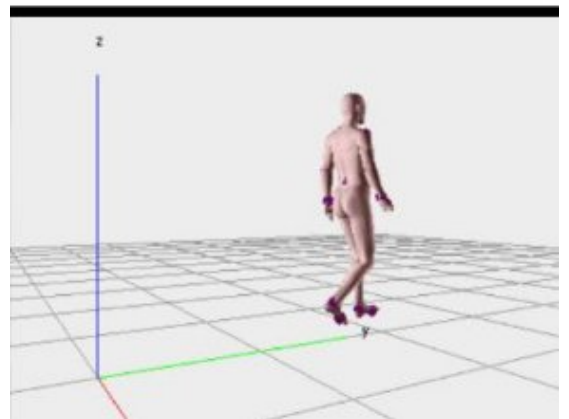
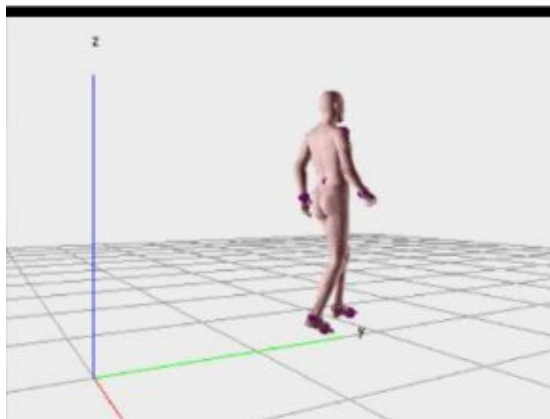
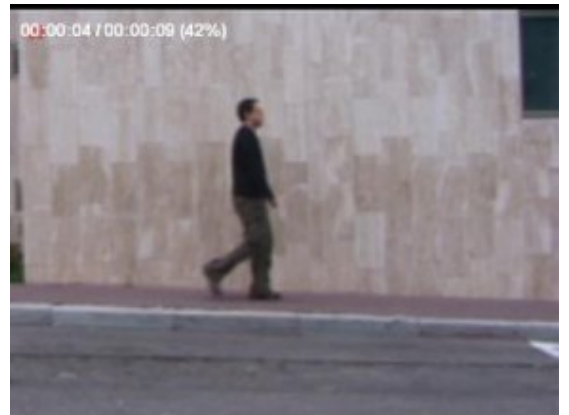


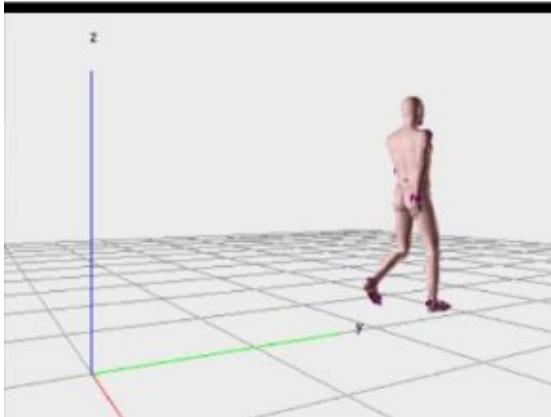
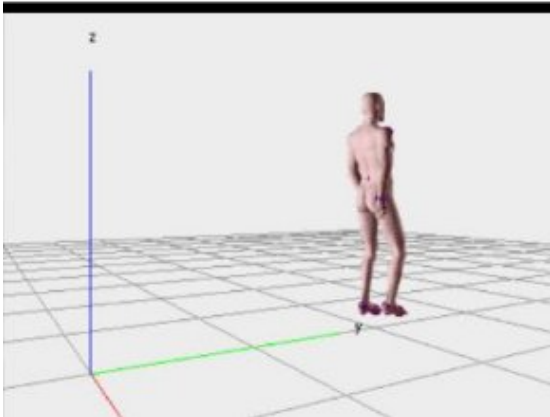
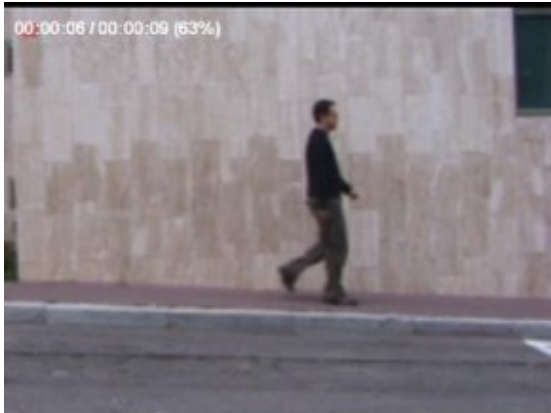
Figure 4.1: The results obtained by applying the proposed framework to the frames of a public walking video. The input frame (top), the extracted silhouette (middle), and the reconstructed 3D pose of the human in the video (bottom).



(3)

(4)

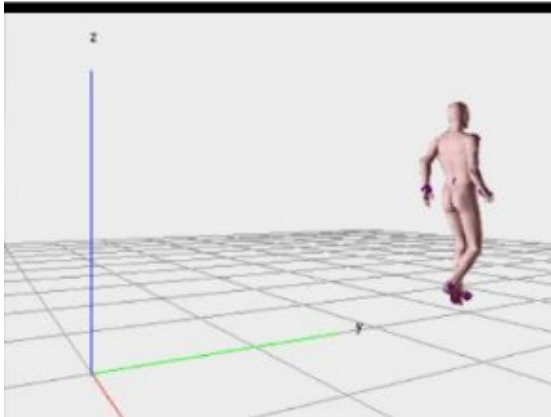
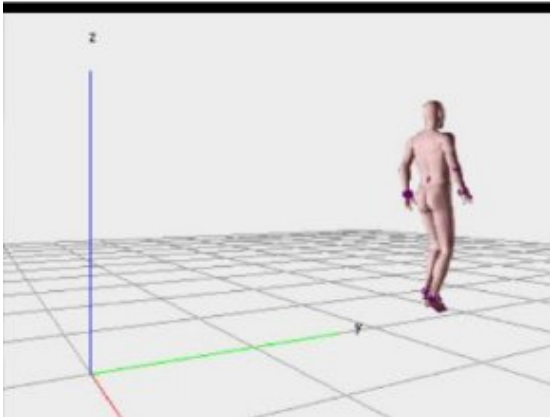
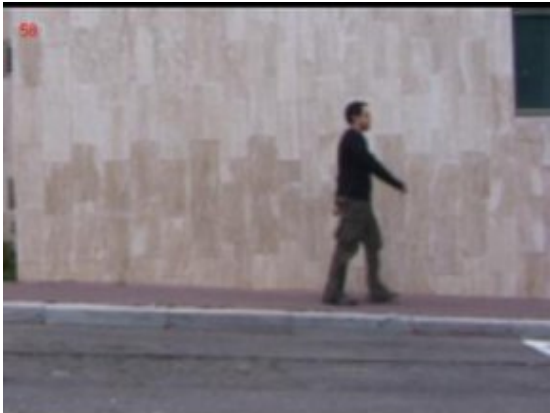
Figure 4.1: (continued).



(5)

(6)

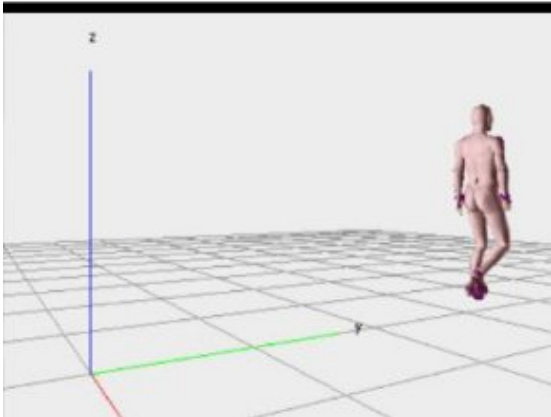
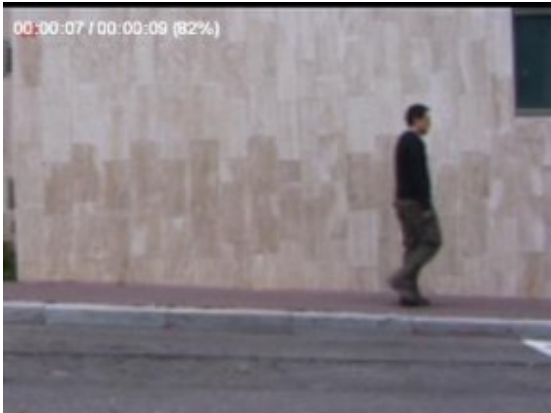
Figure 4.1: (continued).



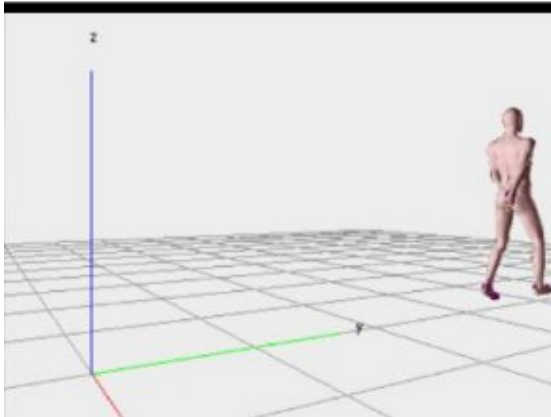
(7)

(8)

Figure 4.1: (continued).



(9)



(10)

Figure 4.1: (continued).



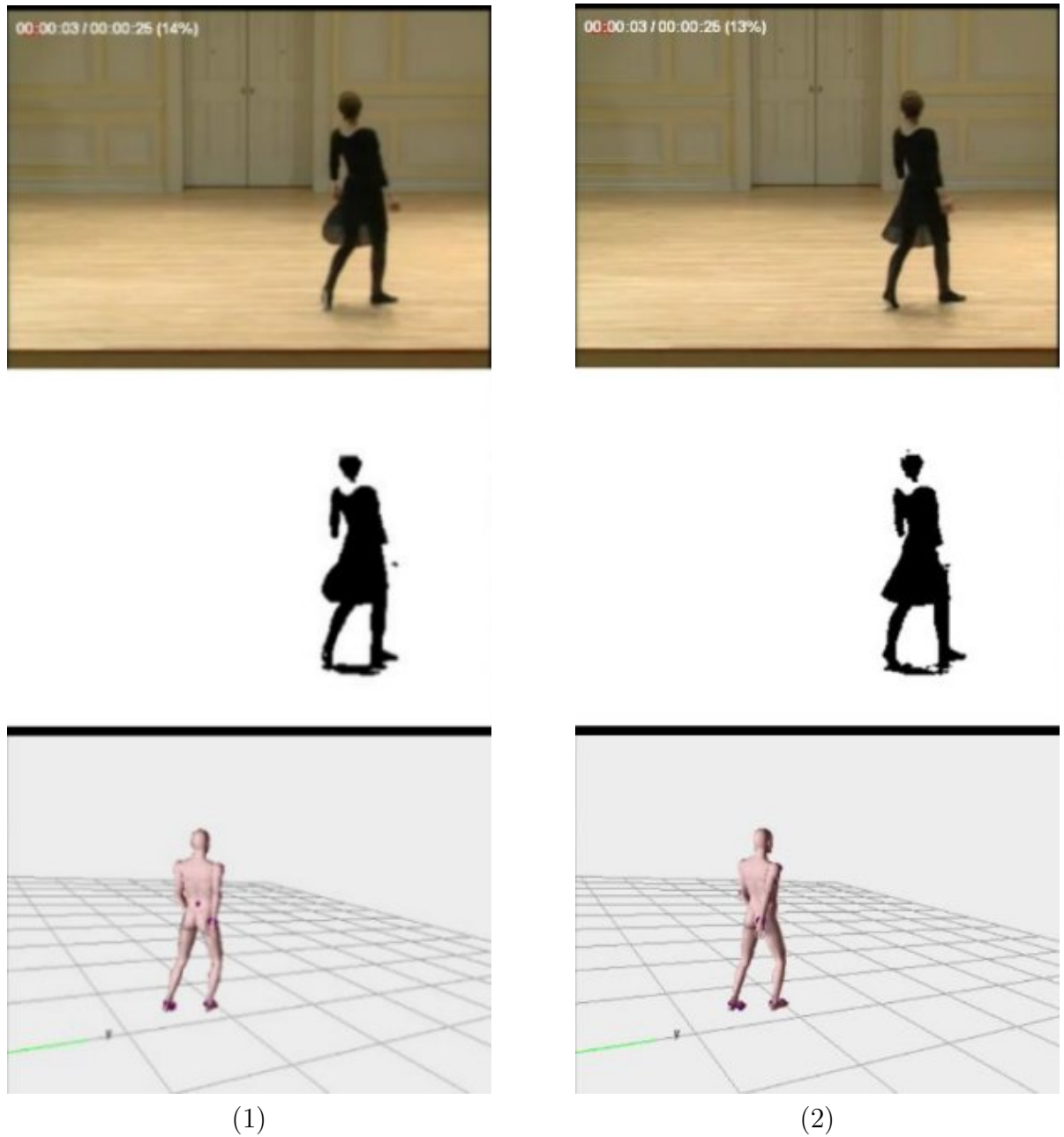
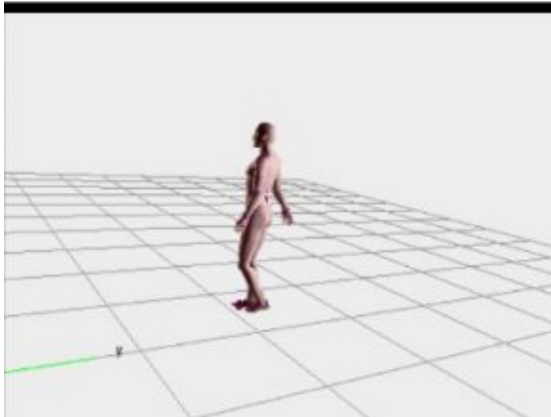
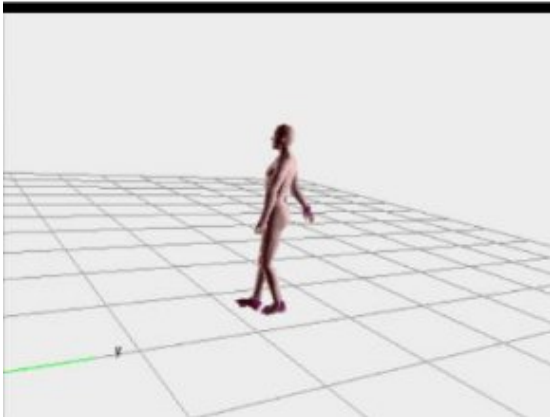


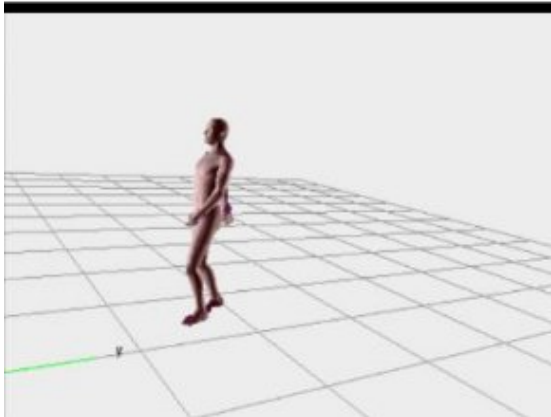
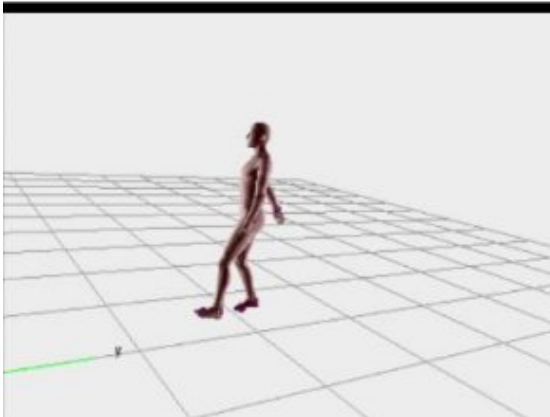
Figure 4.2: The results obtained by applying the proposed framework to the frames of a public dancing video. The input frame (top), the extracted silhouette (middle), and the reconstructed 3D pose of the human in the video (bottom).



(3)

(4)

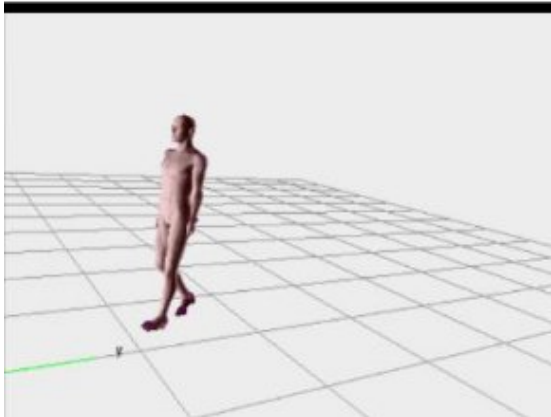
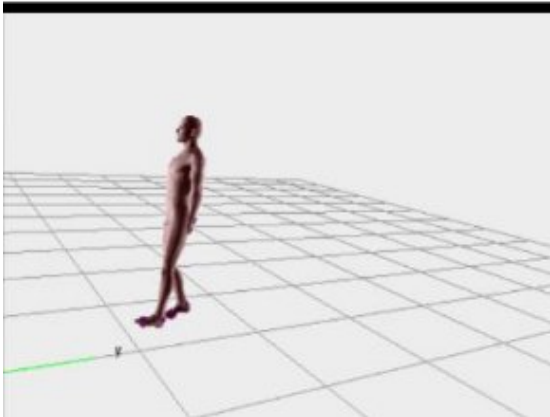
Figure 4.2: (continued).



(5)

(6)

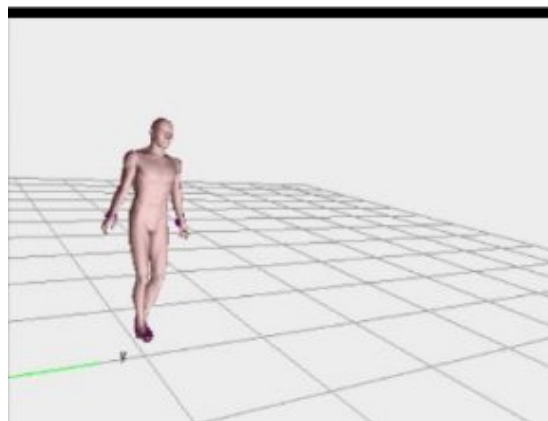
Figure 4.2: (continued).



(7)

(8)

Figure 4.2: (continued).



(9)

Figure 4.2: (continued).

# Chapter 5

## Conclusions and Future Work

This thesis proposes a framework for constructing the 3D human pose from a video sequence obtained from a single view. The proposed framework does not require camera calibration and requires minimal user intervention. Videos taken from public resources can be processed by the proposed framework. The framework assumes the input video has a static background and the video has no significant perspective effects.

We model the human body as an assembly of cylinders and we assume that the lengths of these cylinders are known at least relative to one another.

The proposed framework uses orthographic projection. By considering the foreshortening of the segments, the 3D pose of the human in the video can be reconstructed under orthographic projection. However, the method proposed in [16] requires intensive user interaction; the user has to specify the joint coordinates on the images and the foreshortening direction for each segment. We calculate the joint coordinates automatically from the video images and succeeded to reduce the user interaction to 12-21% during the pose construction. By using the 3D poses constructed, we produced animations of the human walking and dancing.

An animator has to control both the appearance and the movement of the characters. Since there are many degrees of freedom to be controlled, controlling

the movement of characters is a difficult task that requires skill and labor. For this reason, the animators usually begin their work by sketching the coarse version of the movements on key poses. They rework and refine the key poses to produce the final animation [6]. Our proposed framework can help the professional animators an initial version of the motion that can be refined.

One of the apparent results of our human pose estimation framework is to enable non-skilled computer users to use computer animation. The main advantage of our approach over the other approaches is to be able to create the 3D pose that can be easily mapped onto different characters, or modified to fit the needs of a specific animation. Since our approach requires less user interaction and has less constraints, it can be used to construct motion libraries from public resources.

# Bibliography

- [1] Overview of the MPEG-4 Standard, Moving Picture Experts Group (MPEG). A Working Group of ISO/IEC, <http://www.chiariglione.org/mpeg/standards/mpeg-4/mpeg-4.htm/>.
- [2] H.Anim: Specification for a Standard VRML Humanoid version 1.1., Humanoid Animation Working Group, Web3D Consortium, <http://www.hanim.org/Specifications/H-Anim1.1/>.
- [3] The Walking Video, <http://www.wisdom.weizmann.ac.il/~vision/SpaceTimeActions.html>.
- [4] The Dancing Video, An American Ballroom Companion, <http://rs6.loc.gov/ammem/dihtml/divideos.html>.
- [5] S. Bryson. Virtual reality hardware. *ACM Computer Graphics SIGGRAPH'93 Course*, pages 1.3.16–1.3.24, 1993.
- [6] J. Davis, M. Agrawala, E. Chuang, Z. Popović, and D. Salesin. A sketching interface for articulated figure animation. *In Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 320–328, July 2003.
- [7] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59(2):167–181, September 2004.
- [8] M. Gleicher and N. Ferrier. Evaluating video-based motion capture. *In Proceedings of Computer Animation 2002*, pages 75–80, June 2002.



- [9] C.-L. Huang and C.-Y. Chung. A real-time model-based human motion tracking and analysis for human-computer interface systems. *EURASIP Journal on Applied Signal Processing*, 11:1648–1662, 2004.
- [10] G. Johansson. Visual motion perception. *Scientific American*, 232(6):75–80, 85–88, June 1975.
- [11] S. Kiss. Computer Animation for Articulated 3D Characters. Technical Report 2002-45, CTIT Technical Report Series, November 2002.
- [12] V. Mamania, A. Shaji, and S. Chandran. Markerless motion capture from monocular videos. In *Proceedings of Indian Conference on Computer Vision, Graphics and Image Processing (ICVGIP'2004)*, pages 126–132, 2004.
- [13] A. Memişoğlu. Human motion control using inverse kinematics. Master's thesis, Bilkent University, Turkey, 2003.
- [14] T. B. Moeslund and E. Granum. A survey of computer vision-based human motion capture. *Computer Vision and Image Understanding*, 81(3):231–268, 2001.
- [15] F. Perales. Human motion analysis and synthesis using computer vision and graphics techniques: state of art and applications. In *5th World Multi-Conference on Systemics, Cybernetics and Informatics*, 2001.
- [16] C. J. Taylor. Reconstruction of articulated objects from point correspondences in a single uncalibrated image. *Computer Vision and Image Understanding*, 80(3):349–363.
- [17] D. Thalmann. Physical, behavioral, and sensor-based animation. *In Proc. of Graphicon'96*, pages 214–221, 1996.
- [18] N. M. Thalmann and D. Thalmann. Computer animation. *ACM Computing Surveys*, 28(1):161–163, 1996.
- [19] H. Zhou and H. Hu. A survey - human movement tracking and stroke rehabilitation. Technical Report 1744 - 8050, CSM-420 Department of Computer Sciences, University of Essex, United Kingdom, December 1996.

- [20] H. Zhou and H. Hu. A survey - human movement tracking and stroke rehabilitation. Technical Report TR97-045, 16, 1997.