GENERATION OF HUMAN BODY MODELS

Lei He

A thesis submitted in partial fulfillment of the
requirements for the degree of
Master of Science in Computer Science

The University of Auckland
April 2005

Abstract

The goal of the research presented in this thesis is to design an integrated system for the automatic acquisition of a human body model, using input images from mutually orthogonal views. The model consists of two sets of free-form surface patches: the torso and its arms. We determine a neck joint on the torso and six joint positions on the arms (i.e., one location for each shoulder, elbow and wrist).

A conventional three-dimensional reconstruction technique called "shape from silhouettes" is applied in the project. However, one of the challenges of the project is to automatically separate the body segments during the reconstruction of the model from occluded silhouettes. The system suggests a body part localization procedure to determine the joint positions through the analysis of a feature view of body poses. Furthermore, this virtual human model is capable of simulating the motion of a real person using a-priori knowledge of the body. The human model will be specified using Virtual Reality Modelling Language (VRML) and visualized and rendered in a scene graph system, OpenSG.

Acknowledgments

I would like to give my thanks to all of those people who supported me through my research.

I wish to thank Alistair Niall for his technical support. Especially, I wish to thank my supervisors: Dr. Bodo Rosenhahn and Prof. Reinhard Klette for their guidance, encouragement and patience during the development of this thesis.

Contents

Chapter 1

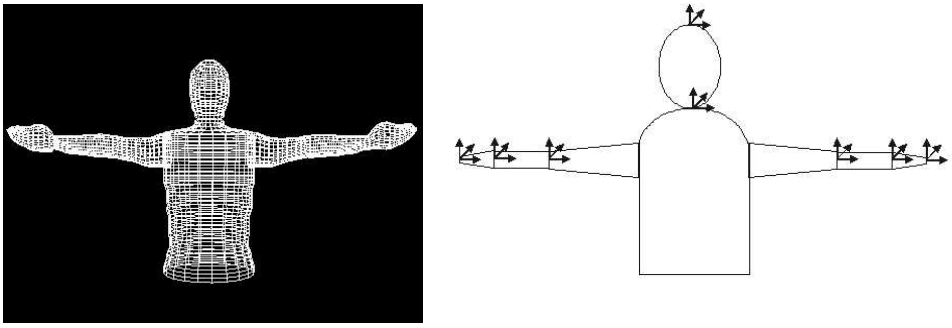
Introduction

1.1 Human body models

Human motion modelling plays an increasingly important role in medical applications, surveillance systems, avatar animation for movies and computer games. It requires the identification of a human body and estimation of its shape and motion parameters. We define the human motion modelling problem to be a combination of two major components, model acquisition and pose estimation. Model acquisition aims to reconstruct the parameters of a human body model that correspond to the specific shape and size of a given human subject. Pose estimation aims to configure a model such that it accurately reflects the position and the configuration of the human subject during a motion capture process.

A large variety of two-dimensional (2D) and three-dimensional (3D) human models has been proposed. Different applications require different levels of representation details. For surveillance purposes, simple low-level 2D image blob

models [18] have proved their effectiveness for approximating 2D human tracking. Other 2D models use planar articulated representations [8], where the model is built in terms of limbs, and represented as intensity patches and joints which are modelled as planar articulations. Such models have been employed both for body part labeling and tracking [17]. However, despite their effectiveness, they fundamentally represent only 2D information and cannot represent 3D constraints. Research in recent years on human motion estimation has used 3D volumetric and kinematic models. Many algorithms have also proved fairly effective [5, 25], but they often use simplified models of the human body built from ellipsoidal, cylindrical or skeleton parts, and do not use a realistic surface model. Actually, realistic 3D body models are possible and may be effective in controlled scenarios, especially when the human subject wears tight fitting clothing. Such models use free-form surface patches as a means of marker-free human motion tracking (e.g., see, [33]) and have lead to robust results.



(a) Grid visualization of a human model.

(b) Coordinate systems of a human model.

Figure 1.1: A human model and its design.

Being motivated by the design of a robust algorithm for 2D-3D pose estimation of human beings in [31, 33], we use free-form surface meshes to build a realistic human body model. This allows to incorporate a diversity of constraints which configure the model, and in consequence it supports a more stable tracking performance. Human body models in this thesis are assumed to be rigid objects. Furthermore, both the motions of the artificial body model and the real human body can be described by hierarchical kinematic chains. This scheme implies that the transformation of these joints that are lower in the hierarchy, involves all transformations of the preceding joints too. Therefore, the body model is designed to consist of a hierarchic arrangement of eight body segments: head, torso, upper arm, lower arm, and so forth. Body segments are defined by their local coordinates, and joints (neck, shoulder, elbow and wrist) are added in between. Figure 1.1 shows an example of such a model and its coordinate systems. A joint here is represented by a joint that has three degrees of freedom (DOF). Rigid transformations at each joint define a specific body pose for the model. These transformations are constrained to simulate the actual motions of the body.

1.2 Related work

This section is restricted to the research involving realistic human body modelling and body part separation. Techniques for visual object reconstruction can be broadly divided into active methods, in which light patterns are projected into a scene to provide visual features for matching, and passive methods

that rely on matching visual cues between images [36]. Active techniques often use a range scanner for model acquisition, which employs a time-of-flight approach. A focused laser pulse is emitted from a transmitter with a constant speed. It is reflected by the object surface and detected by a receiver. The time between transmitting and receiving gives the distance traveled. The laser device scans through the entire surface and provides a depth map of the object. This approach acquires highly accurate geometric data of the shape. However, it usually outputs point cloud data, which are not suitable for animation. To overcome this issue, [27] developed techniques for automatically creating and animating models obtained from scanned data. [27] extracts key landmarks and fits them to an underlying skeleton. Articulated models are generated without the interaction of the users. One of the disadvantages of active methods is that it requires a large amount of time and expertise to use it. Moreover, such systems are prohibitively expensive and need a special environmental setting.

Compared to active sensing techniques, passive scene reconstruction from images is a low-cost technique. It enables greater flexibility in scene capture and relies on matching visual cues such as features, surface appearance, shading and silhouettes. Therefore passive methods have received considerable interest in the past few years, and many approaches have been proposed.

Kakadiaris and Metaxas have developed a system for 3D human body model acquisition in [19] using three cameras placed in a mutually orthogonal configuration. A subject is requested to perform a set of movements according to

a protocol that reveals the structure of the human body. The body parts are identified and reconstructed incrementally from 2D deformable contours.

Hiltion et al. [14] proposed an approach for modelling the human body from four views. This approach uses extrema for helping to find feature points. It is simple and efficient. However, it is not reliable for identifying neck joints and it does not provide a solution to find elbow and wrist joints. Furthermore, it is not a seamless model for animations. W. Lee et al. [23] builds seamless human models having similar body structure as defined in [14]. Different methods are employed to head and body reconstruction. The approach ensures robust and efficient results in face modelling, and a realistic appearance of the whole body. However, it is not a fully automatic approach and cannot detect the joint positions, which have to be manually given by users.

1.3 Purpose of this thesis

This work is part of a human motion analysis project based on captured sequences, as presented in [31, 32, 33]. For a detailed study of human motions, the project requires an automatic model generation system, so that pose recovery can be evaluated for different persons (e.g., male or female, small or tall people).

Our goal is to provide an integrated framework for the automatic generation of

human models that consist of free-form surface patches and body segments connected by joints. The main difficulties in developing algorithms for modelling stem from occlusions among body parts. This thesis solves that problem and acquires a concise 3D model of a human body using vision sensors. Furthermore, as a first stage towards human motion modelling and pose estimation, an application for simple joint tracking is presented.

1.4 Organization of this thesis

Chapter 2 firstly exploits the mathematical fundamentals used in this thesis. Human motion modelling is an extended study of this project. This chapter also provides a general introduction to rigid body motions. The discussed concepts and notions will be used in later chapters.

Chapter 3 gives the system overview. It includes the lab setup for model acquisition, system overview and the software requirements for running the program produced in our work.

Chapter 4, Chapter 5 and Chapter 6 describe three components for modelling a human body. They combine various image processing and 3D reconstruction techniques. Chapter 4 deals with segmenting the region of interest by means of background subtraction [15]. A computational model is used to distinguish the object and background. Morphologic operators [22] are also introduced.

Chapter 5 discusses a feasible way to find the positions of joints. It is a combination of skeletonization [4, 20], chain coding [11] and corner detection [9]. Refinements are also discussed. Chapter 6 discusses the 3D reconstruction of human bodies. It discusses different strategies for 3D reconstruction of torso and arms. Apart from 3D modelling, camera calibration [37] is also involved.

Chapter 7 demonstrates some experimental results from the implementation of the ideas described in this thesis. In particular, we illustrate animation sequences showing motion tracking using the concepts described in Chapter 2, and a-priori knowledge of body parameters obtained from considerations discussed in Chapter 5 and Chapter 6.

Chapter 8 summaries this thesis. It also discusses improvements for future work and possible extensions of this study.

Chapter 2

Mathematical basics and rigid body motions

This chapter contains two parts. The first part provides a general introduction to the mathematical fundamentals used in this thesis. The second part deals with the formulation of *rigid body motions*. Modelling of human motions is an extended study of this project. It is introduced in Chapter 7.

2.1 Basics

2.1.1 Representations for points, lines and planes

A point, representing a position in the 2D or 3D space, can simply be expressed by a two-dimensional or three-dimensional vector. For our purpose, homogeneous coordinates [26] are used. For example, 2D points are represented as 3D vectors $(x, y, 1)^T$, and 3D points as 4D vectors $(x, y, z, 1)^T$.

A line l can be represented by a combination of two points in 2D or 3D space, denoted as $\overline{P_1P_2}$. In parametric representation, we specify a line L as a set of points P_l with

$$L = \{P_l | P_l = P_1 + u(P_2 - P_1), u \in \mathbb{R}\}, \tag{2.1.1}$$

where u is a positioning factor with respect to P_1.

Implicitly, a line can be specified by its direction n and a moment m. We denote that by $L = (n, m)$. The direction n is defined as a unit direction,

$$n = \frac{P_2 - P_1}{||P_2 - P_1||}. \tag{2.1.2}$$

The moment m is defined as the cross product of a point on the line and direction n,

$$m = P_1 \times n. \tag{2.1.3}$$

It is noted that m is independent of the point position [31]. A point P_l is incident with the line $L = (n, m)$ iff

$$P_l \times n - m = 0. \tag{2.1.4}$$

A plane can be specified by three points P_1, P_2, P_3 on that plane. Its parametric representation as follows:

$$P_p = P_1 + u_1(P_2 - P_1) + u_2(P_3 - P_1), u_1, u_2 \in \mathbb{R}. \tag{2.1.5}$$

Implicitly, the equation of a plane can be represented by a unit normal vector $n = (a, b, c)^T$ of the plane and the Hessian distance d from the origin to the

plane (Figure 2.1). Incidence of a point P_p on the plane is given as

$$(n \cdot P_p) - d = 0, \tag{2.1.6}$$

where, $n = (a, b, c)^T$, $||n|| = 1$ and $d \in \mathbb{R}$. If a plane is denoted by a vector of its parameters, $(a, b, c, d)^T$, the general form for an equation of a plane is

$$ax + by + cz + d = 0. \tag{2.1.7}$$

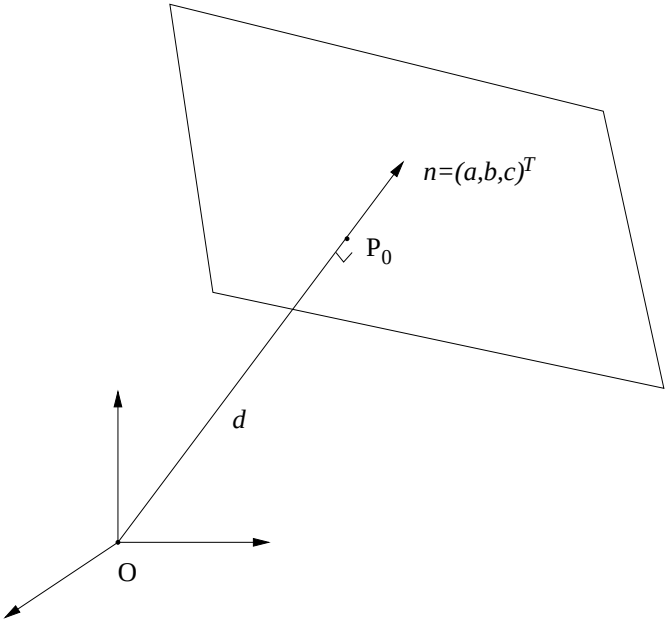


Figure 2.1: A plane can be represented by a normal vector n of the plane and the Hessian distance d from origin to the plane.

2.1.2 Line-line intersection

We first consider the situation for the 2D case. Two lines $L_1 = \overline{P_1 P_2}$, and $L_2 = \overline{P_3 P_4}$, intersect at a point if $P_a = P_b$, for $u_a, u_b \in \mathbb{R}$

$$P_a = P_1 + u_a(P_2 - P_1), \tag{2.1.8}$$

$$P_b = P_3 + u_b(P_4 - P_3). \tag{2.1.9}$$

Having $P_a = P_b$ and $P_i = (x_i, y_i)$, this leads to the following two equations in unknown factors of u_a and u_b:

$$x_1 + u_a(x_2 - x_1) = x_3 + u_b(x_4 - x_3), \tag{2.1.10}$$

$$y_1 + u_a(y_2 - y_1) = y_3 + u_b(y_4 - y_3). \tag{2.1.11}$$

Solving gives the following expressions for u_a and u_b:

$$u_a = \frac{(x_4 - x_3)(y_1 - y_3) - (y_4 - y_3)(x_1 - x_3)}{(y_4 - y_3)(x_2 - x_1) - (x_4 - x_3)(y_2 - y_1)}, \tag{2.1.12}$$

$$u_b = \frac{(x_2 - x_1)(y_1 - y_3) - (y_2 - y_1)(x_1 - x_3)}{(y_4 - y_3)(x_2 - x_1) - (x_4 - x_3)(y_2 - y_1)}. \tag{2.1.13}$$

Substituting either of these into the corresponding equation, Equation (2.1.8) or Equation (2.1.9), gives the intersection point.

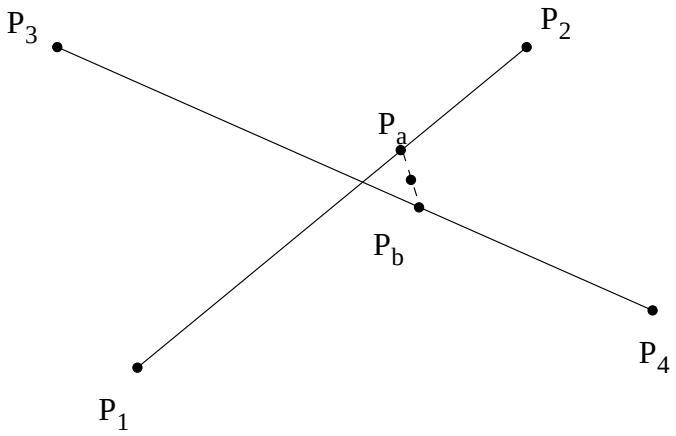


Figure 2.2: Line-line intersection in 3D space.

In the 3D case, the intersection of the two lines may not necessarily exist, so that we have to find the closest pair of points, one per line, that approximate

the intersection. Thus the point being closest to two 3D lines can be found by minimizing the Cartesian distance between the points of these lines (Figure 2.2). That is minimizing the following:

$$||P_a - P_b||^2 = ||P_1 + u_a(P_2 - P_1) - P_3 - u_b(P_4 - P_3)||^2. \tag{2.1.14}$$

The unknown factors can take the form of a 2×1 vector with the elements u_a and u_b. After partial derivatives by desired factors are set equal to zero, the solution of a closest pair is derived as follows:

$$\begin{pmatrix} u_a \\ u_b \end{pmatrix} = \begin{pmatrix} \Delta_{12}^T\Delta_{12} & \Delta_{12}^T\Delta_{34} \\ \Delta_{12}^T\Delta_{34} & \Delta_{34}^T\Delta_{34} \end{pmatrix}^{-1} \begin{pmatrix} -\Delta_{31}^T\Delta_{12} \\ -\Delta_{31}^T\Delta_{34} \end{pmatrix} \tag{2.1.15}$$

where $\Delta_{ji} = P_i - P_j$. Finally, the middle point of the line segment between P_a and P_b is regarded as the intersection.

2.1.3 Line-plane intersection

Given a plane $(a, b, c, d)^T$ and a line through $X_1(x_1, y_1, z_1)$ and $X_2(x_2, y_2, z_2)$, the combination with Equation (2.1.1) and Equation (2.1.7) gives

$$a(x_1 + u(x_2 - x_1)) + b(y_1 + u(y_2 - y_1) + c(z_1 + u(z_2 - z_1)) + d = 0. \tag{2.1.16}$$

Solving for u gives

$$u = \frac{ax_1 + by_1 + cz_1 + d}{a(x_1 - x_2) + b(y_1 - y_2) + c(z_1 - z_2)}. \tag{2.1.17}$$

If the denominator is zero then the line is either parallel to the plane, and there are no solutions, or the line is on the plane, in which case there is an infinite number of solutions. If the denominator is nonzero, back-substituting u to Equation (2.1.1) gives the intersection point.

2.2 Rotations

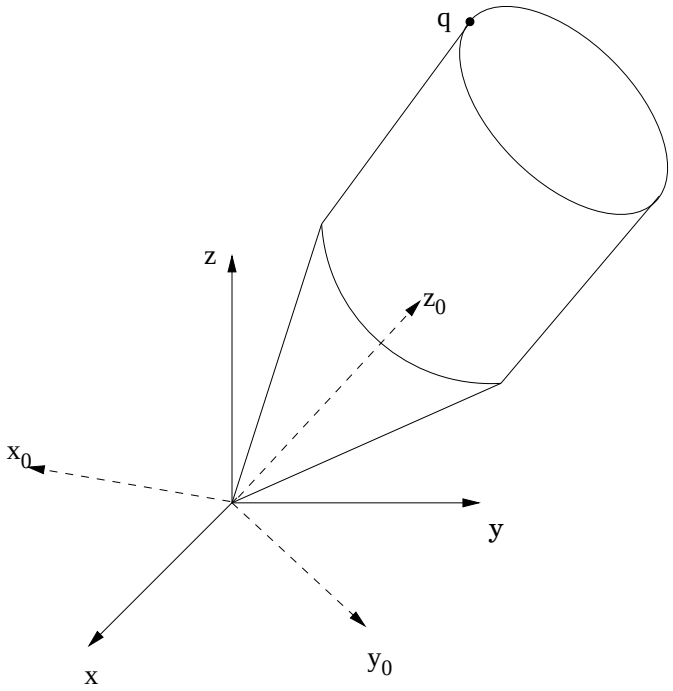


Figure 2.3: Rotation of a rigid object about a point. The solid coordinate frame of an inertial frame and the dashed coordinate frame is attached to the rotating rigid body.

A 3×3 rotation matrix R can be used to describe the orientation of an object by giving the relative orientation between a coordinate frame attached to the object and a fixed or inertial coordinate frame (see Figure 2.3, the dashed coordinate frame $x_0 y_0 z_0$ is attached to the rotating rigid body, and the coordinate frame xyz remains fixed). A rotation matrix R for a right-handed coordinate frame,

$$R = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix},$$

has the following key properties,

1. $RR^T = I$;

2. $det(R) = 1$.

Euler angles α, β, γ are often used in classical mechanics to represent a rotation by using these three parameters [26]. There are several conventions for Euler angles, depending on the axes about which the rotations are carried out. A way to describe the orientation of a coordinate frame B relative to another coordinate frame A is as follows:

1. Start with frame B coincident with frame A;

2. Rotate frame B about the z-axis of frame B by an angle α;

3. Rotate about the new y-axis of frame B by an angle β;

4. Rotate the z-axis of frame B by an angle γ.

This yields a net orientation $R(\alpha, \beta, \gamma)$ with the triple of Euler angles (α, β, γ) representing the rotation. We define the elementary rotations about the x, y and z axes as follows,

$$R_x(\gamma) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & cos(\gamma) & -sin(\gamma) \\ 0 & sin(\gamma) & cos(\gamma) \end{pmatrix};$$

$$R_x(\gamma) = \begin{pmatrix} cos(\beta) & 0 & sin(\beta) \\ 0 & 1 & 0 \\ -sin(\beta) & 0 & cos(\beta) \end{pmatrix};$$

$$R_x(\alpha) = \begin{pmatrix} cos(\alpha) & -sin(\alpha) & 0 \\ sin(\alpha) & cos(\alpha) & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

Concatenation of these three matrices leads to the general rotation which defines the final orientation of frame B relative to frame A,

$$R = R_z(\alpha)R_y(\beta)R_x(\gamma). \tag{2.2.1}$$

2.3 Exponential mapping in rigid motions

In [31] Lie groups and Lie algebras are used to model rigid motions. We first begin with $SO(3)$, which denotes the group of rotational motions in $\mathbb{R}^3$. Let $\omega \in \mathbb{R}^3$ be a unit vector that specifies the direction of rotation and let $\theta \in \mathbb{R}$ be the angle of rotation in radians. According to [26], every rotation of an object corresponds to some rotation matrix $R \in SO(3)$. This means that R should be an orthogonal matrix with determinant 1. $SO(3)$ is identified with the family of all of these 3×3 matrices. The notation SO is an abbreviation for "special, orthogonal". We write R as a function of ω and θ, $R(\omega, \theta)$, if we rotate about the axis ω at unit velocity by an amount θ.

For a one-parameter Lie group given in matrix representation, the tangent space defines its Lie algebra $so(3)$ (see [26]) as follows:

$$so(3) = \{\hat{\omega} \in \mathbb{R}^{3\times 3} | \hat{\omega} = -\hat{\omega}^T\}. \tag{2.3.1}$$

The matrix $\hat{\omega}$ is a skew-symmetric matrix,

$$\hat{\omega} = \begin{pmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{pmatrix}. \tag{2.3.2}$$

The space of all 3×3 skew-symmetric matrices is denoted by $so(3)$. As exponentials of skew matrices are orthogonal, Lie group $SO(3)$ and Lie algebra $so(3)$ are connected via the exponential mapping function $\exp : so(3) \to SO(3)$. For any $\hat{\omega} \in so(3)$, the *Rodrigues' formula* [26] gives an efficient method for computing the exponentials,

$$R(\omega, \theta) = \exp(\hat{\omega}\theta) = I + \hat{\omega}\sin(\theta) + \hat{\omega}^2(1 - \cos(\theta)) \tag{2.3.3}$$

where $\hat{\omega}$ is a skew-symmetric matrix, for $\omega = (\omega_1, \omega_2, \omega_3)^T$ with $||\omega|| = 1$.

Geometrically, the skew-symmetric matrix corresponds to an axis of rotation, and the exponential map generates the rotation corresponding to rotation about the axis ω by a specified amount θ.

An extension from $SO(3)$ is the group of rigid body motions denoted by $SE(3)$ (special Euclidean group), which consists of a rotation matrix R and a translation vector T. Homogeneous coordinates are common in representing rigid motions. Every 3D rigid body motion (RBM) can be represented by a 4×4 matrix

$$M = \begin{pmatrix} R_{3\times3} & T_{1\times3} \\ 0_{3\times1} & 1 \end{pmatrix}, \text{ with } R \in SO(3) \text{ and } T \in \mathbb{R}^3. \tag{2.3.4}$$

The corresponding Lie algebra to $SE(3)$ is

$$se(3) = \{(v, \hat{\omega}) : v \in \mathbb{R}^3, \hat{\omega} \in so(3)\}. \tag{2.3.5}$$

An element $\xi = (v, \hat{\omega}) \in se(3)$ is called a twist, and its matrix representation takes the form

$$\hat{\xi} = \begin{pmatrix} \hat{\omega} & v \\ 0_{3\times 1} & 0 \end{pmatrix}. \tag{2.3.6}$$

$\hat{\xi}$ is referred to as an infinitesimal generator of the Euclidean group.

Analogous to $SO(3)$, the exponential map $\exp : se(3) \to SE(3)$ leads to a mapping from $se(3)$ to $SE(3)$ [26], which can be computed by evaluating

$$\exp(\hat{\xi}\theta) = \begin{pmatrix} \exp(\hat{\omega}\theta) & (I - \exp(\hat{\omega}\theta)(\omega \times v) + \omega\omega^T v\theta) \\ 0 & 1 \end{pmatrix}. \tag{2.3.7}$$

2.4 Screw motion

As mentioned previously, a rigid body motion corresponds to an Euclidean transformation group $SE(3)$. The exponent $\exp(\theta\hat{\xi})$ leads to a rigid motion and corresponds to a screw motion. A screw motion is a specific class of rigid body motion that consists of a rotation about a straight line combined with a translation parallel to that line. It is defined by an axis ω in space through an angle of θ, combined with a translation along the same axis by an amount d. Screw motions are not only special cases of rigid motions. Indeed, Chasles has proved in 1802 that the reverse is true: Every rigid body motion can be expressed as a screw motion. Figure 2.4 illustrates the principle of screw motion.

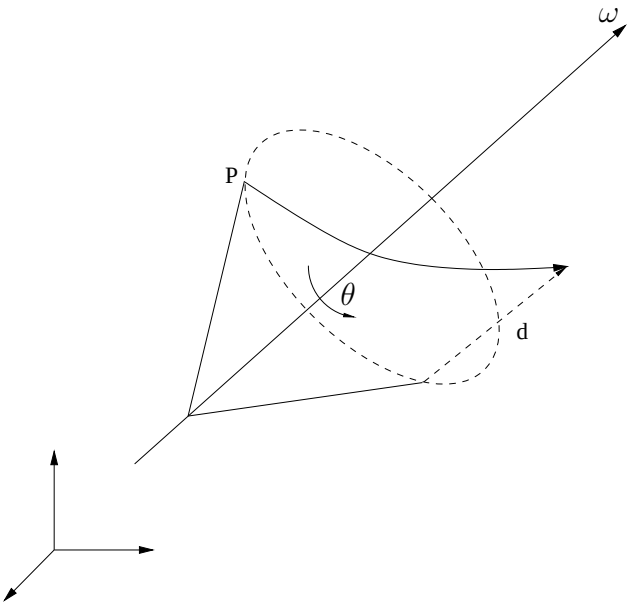


Figure 2.4: A screw motion along axis ω.

The *pitch* of the screw is defined to be the ratio of translation to rotation during motion, $h := \frac{d}{\theta}$ $(d, \theta \in \mathbb{R}, \theta \neq 0)$. There exist several special cases of screw motion. If $h \to \infty$ then the screw motion corresponds to a pure translation along the axis of the screw, called an *infinite pitch screw*. A *zero pitch screw* is a screw motion for which the pitch is h=0, corresponding to a pure rotation about an axis.

Joints on a human body are revolute joints. They can be modelled by zero pitch screws, which is a general rotation. We describe a general rotation as a rotation of a point around an arbitrary line l, usually not passing through the origin in space. To model a general rotation of a point, the general idea is to translate the point by the distance vector between a line and the origin, such to perform a rotation, and to translate the transformed point back. If T is a matrix denoting

the translation, and R is the rotation given by Equation (2.3.7), the general rotation has the form

$$M = TRT^{-1}. \tag{2.4.1}$$

In human body pose estimation algorithms [31, 33], the preferred method of representing the RBM matrix is as merely an exponential of a twist with zero pitch, since the exponential form enables us to linearize the rigid motion with respect to θ, and it is more easy to calculate its derivative this way.

Chapter 3

System design and overview

3.1 Configuration of stereo system

We assume that a human subject stands in front of a monochromatic background that is used to distinguish the object from the environment. At least two perpendicular views of the object should be captured by CCD cameras. A blue curtain is chosen as the background.

Two cameras are used during the experiments (but our approach can easily be extended to more cameras). The subject stands still with arms stretching out, while images are captured by the two cameras from perpendicular positions (see Figure 3.1). To create the lab setup for these experiments, the following minimal equipment is needed: a blue curtain, two frame grabbers, a computer, two CCD cameras, and two tripods.

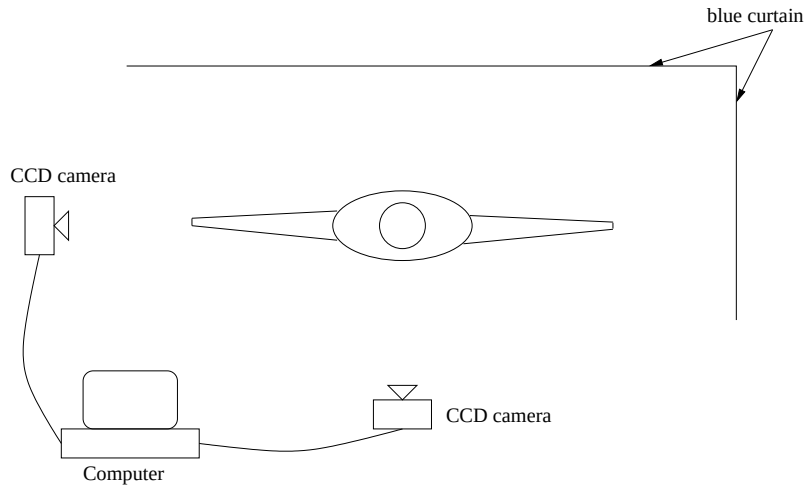


Figure 3.1: The lab setup scene.

3.2 General design and requirement

3.2.1 System overview

One of our goals is the design of a robust and flexible automatic 3D reconstruction system for a human model. This means that the system should be running without too many manually interventions. It also implies that the system should be able to modify every parameter "on the fly" (without restarting any process).

Figure 3.3 illustrates the interface of the application. The implementation of the system involves seven functional modules correspondingly shown by seven buttons on the toolbar of the interface. Figure 3.2 describes the flow chart of running the system.

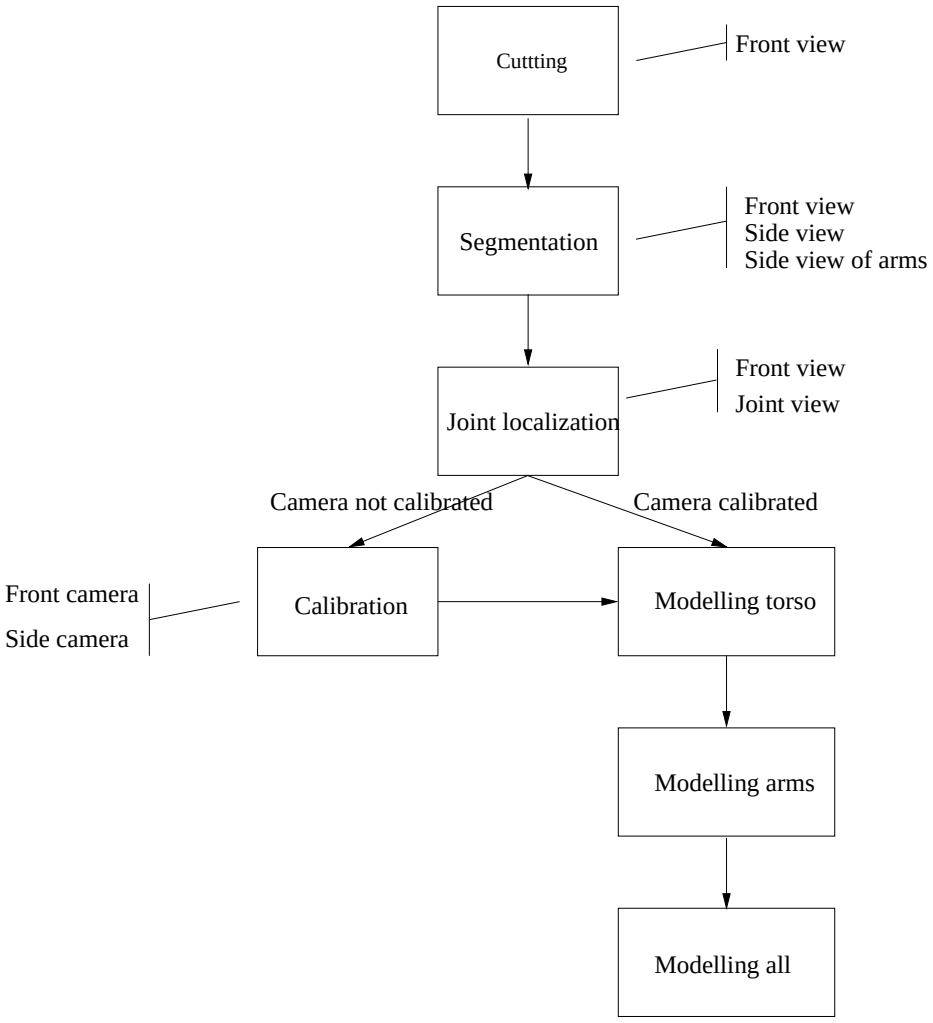


Figure 3.2: Flow chart of the system.

"Segmentation" extracts the silhouette of an object against a background. This module takes a colour image as input and outputs a binary image. It will be applied on three different views. They are the front view, the side view, and the side view of the arms. The related algorithms are described in Chapter 4. The second button "cutting" in Figure 3.3 is an additional function for our convenience, as we currently model only the upper part of the human body. A "cutting" function is used to define a horizontal line below which the body

parts are invisible. This module is only necessary for the front view. The picture in Figure 3.3 shows an example. The cutting line was determined by simply clicking on the front view at a desired position.

The module of "joint localization" is responsible for finding joint positions such as the shoulder, elbow and wrist on the body. The related algorithms are described in detail in Chapter 5.

We build separately the 3D models of body torso and arms separately. The buttons "modelling torso" and "modelling arms" correspond to these two modules, respectively. The whole model of the body will be integrated at the end (button "modelling all"). "Camera calibration" is a pre-setting step that calibrates the cameras. The stereo system will be calibrated using Tsai's method [37]. This module takes two views of a calibration cube and outputs two lists of camera parameters, respectively. The related algorithms of 3D reconstruction, and also camera calibration, are described in Chapter 6.

3.2.2 Libraries and programming language

Considering the above system design, we need to choose suitable graphics libraries for visualization, and also a suitable programming language.

A proper 3D graphics library is helpful in rendering the model and integrating

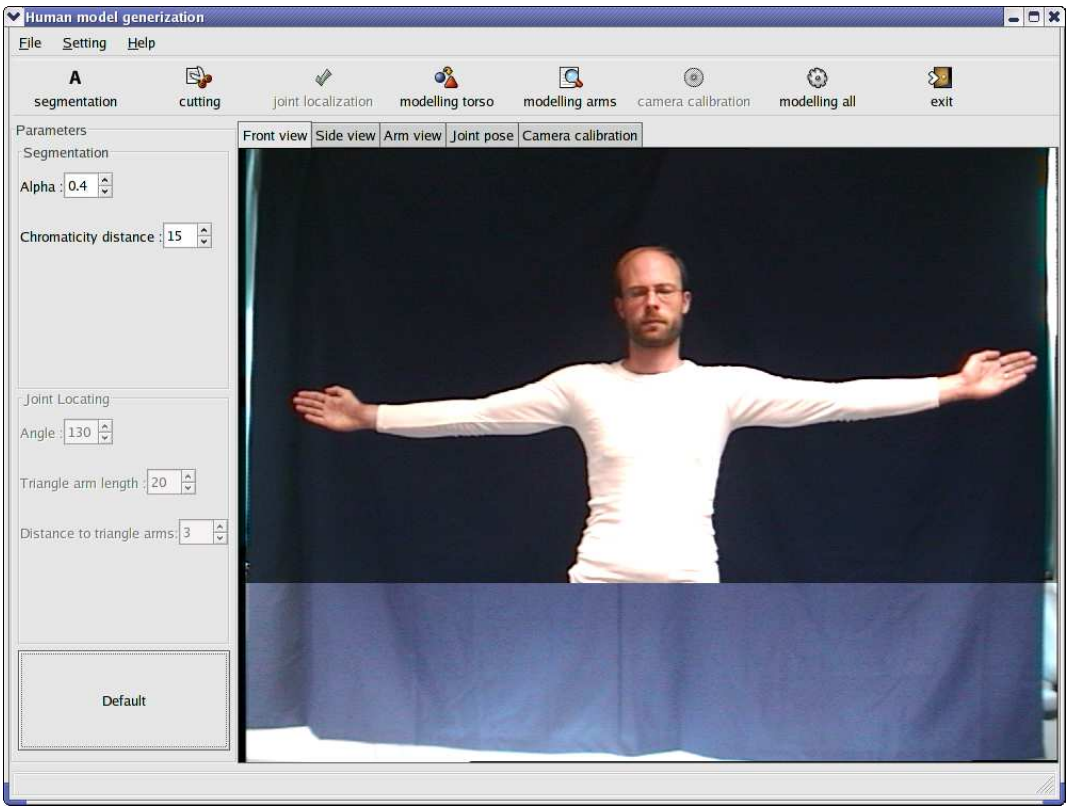


Figure 3.3: The interface.

our system into a 3D scene. The choice of the 3D graphics library and programming language is relatively simple, considering what is available and what our needs are. We need a complete and fast library. The only choice at the moment is OpenGL [38] maintained by Silicon Graphics. OpenGL is available on almost every possible platform, from the personal computers to the multi-processor machines. Many graphic cards are designed for OpenGL and implement efficiently the rendering primitives.

The choice of a suitable programming language is also obvious. It should combine qualities such as powerful mechanisms, good performance, efficiency and

of course the support of OpenGL. The two best candidates that also have an OpenGL binding are `C/C++` and `Java`. Although these two languages are very comparable in features, the decisive point may be the speed of execution as we need to handle a sequence of images. Furthermore, the system is designed running under Linux. It is not necessary to consider too much about the portability. `C/C++` [16, 28], the most popular language under the Linux environment, is the language selected for use.

VRML [30] is an ISO standard of a file format for describing interactive 3D objects and worlds. It is a powerful scene graph language. We adopt VRML representation to define the generic human model as it can be specified with multiple levels-of-detail, and it can achieve both an efficient and realistic visual appearance. OpenSG [29], developed and maintained by the OpenSG Symposium, is a portable scene graph system to create real time graphics programs (e.g., for virtual reality applications). In the system, OpenSG is used as VRML parser and rendering tool.

Finally, it is necessary to take a graphical user interface (GUI) into account. A GUI serves as a front end to the system. It has the responsibility of displaying information in a clear and concise manner to the user, and being able to handle user input. For GUI design, `GTK` and `Qt` are two useful GUI toolkits under Linux. Again, without taking portability into account, they are compatible. The choice of `GTK` [34] is subjective.

In summary, the softwares or tools for running the system are listed here: The system is developed under a Linux Redhat 9.0 environment. OpenGL is used as the graphic API. The application is written in `C/C++`. Human models are specified in the VRML format. A scene graph system OpenSG helps to parse and visualize VRML files. `GTK` is our GUI development toolkit.

Chapter 4

Segmentation

Silhouette extraction or segmentation is the process of extracting a region of interest from an image. The accuracy and efficiency of contour segmentation is clearly very crucial to our task. This can be achieved through a background subtraction technique together with a series of morphologic operations.

The idea of background subtraction is to subtract the current image from a reference image. T. Horprasert et al. proposed an algorithm in [15] to extract objects against a static background. This algorithm deals with extracting objects from a video sequence based on a statistical approach. The reference image is acquired from a static background during a period of time. Our system follows the general procedure defined in [15] however we use a particular lab configuration described in Section 3.1, and the reference image is acquired from only one background image rather than a series of images. Therefore, we slightly modify the algorithm.

4.1 Computational colour model

Let the i^{th} pixel of the background have an RGB value E_i, which is also called the expected colour. Let the same pixel of the foreground have an RBG colour F_i, which is also called the observed colour. The colour values E_i and F_i are denoted as

$$E_i = [E_R(i), E_G(i), E_B(i)]$$

$$F_i = [F_R(i), F_G(i), F_B(i)]$$

where $E_R(i), E_G(i)$ and $E_B(i)$ are the red, green and blue values of the i^{th} pixel in the background image; $F_R(i), F_G(i)$ and $F_B(i)$ are the red, green and blue values of the i^{th} pixel in the foreground image.

The distortion between E_i and F_i can be decomposed into two components: brightness distortion and chromaticity distortion. Figure 4.1 illustrates this colour model, the line OE_i passing through the origin and the point E_i is called the *expected chromaticity line*. The brightness distortion α is a scalar value that brings the observed colour close to the expected line. The chromaticity distortion CD is defined as the orthogonal distance between the observed colour and the expected chromaticity line. The brightness distortion α_i is 1 if the brightness of the given pixel in the foreground image is the same as in the background image, α_i less than 1 if it is darker, and greater than 1 if it becomes brighter than the expected brightness. The brightness distortion can be obtained by minimizing the distance between the foreground colour and the expected chromaticity

line, which is the chromaticity distortion. For

$$\alpha_i = \frac{F_R(i)E_R(i) + F_G(i)E_G(i) + F_B(i)E_B(i)}{E_R(i)^2 + E_G(i)^2 + E_B(i)^2},$$

we get

$$\begin{aligned} CD_i &= ||F_i - \alpha_i E_i|| \\ &= \sqrt{(F_R(i) - \alpha_i E_R(i))^2 + (F_G(i) - \alpha_i E_G(i))^2 + (F_B(i) - \alpha_i E_B(i))^2}. \end{aligned}$$

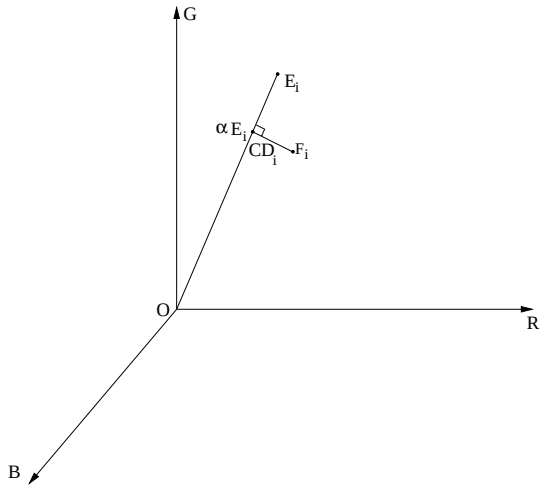


Figure 4.1: Color model in RGB space.

4.2 Subtraction operation

During the subtraction operation, the difference between the background image and the foreground image is evaluated. The difference is decomposed into brightness and the chromaticity components. The thresholding should be operated over these two components. Applying two suitable thresholds on the

brightness distortion α, T_α, and chromaticity distortion CD_i, T_{CD}, for pixel i, it segments the foreground image into a binary image. Based on this definition, a pixel is classified into two categories, either background or foreground (object): a pixel is a background pixel only if it has both brightness and chromaticity similar to those of the same pixel in the background image, otherwise it is an object pixel. This decision procedure is summarized in Figure 4.2.

```
PROCEDURE background subtraction
1.  Input: background image E and foreground image F,
           brightness distortion threshold T_α,
           chromaticity distortion threshold T_CD;
2.  for each pixel i in F and the same pixel in E
3.     compute brightness α_i;
4.     compute chromaticity CD_i;
5.     if |α_i − 1| > T_α or CD_i > T_CD;
6.        pixel i set to be object
7.     else
8.        pixel i set to be background
ENDPROCEDURE
```

Figure 4.2: Procedure for background subtraction.

4.3 Morphologic operations

Morphology [21] is the study of form and patterns (i.e, of geometric properties of binary images). The two basic morphologic operations are dilation and erosion, where dilation causes an enlargement of objects, and erosion causes an enlargement of the background. Opening and closing operations are two additional morphologic operations that are derived from the fundamental operations of

dilation and erosion. They allow to derive shape information, and can be used to decompose objects into parts.

Morphologic operations usually take two data types as an input. One is the binary image, and the other is a structuring element that determines the effects of the operator on the image. A structuring element is also known as a kernel, which consists of a pattern specified as the coordinates of a number of discrete points relative to some origin. Normally Cartesian coordinates are used, so that the elements can be represented as a small image on a rectangular grid in a convenient way. A commonly used example is depicted in Figure 4.3 as a 3×3 square, with the origin at its center. Its corresponding set of coordinate points (from top to bottom and from right to left) are defined as $\{(-1,-1),(0,-1),(1,-1),(-1,0),(0,0),(1,0),(-1,1),(0,1),(1,1)\}$.

1	1	1
1	* 1	1
1	1	1

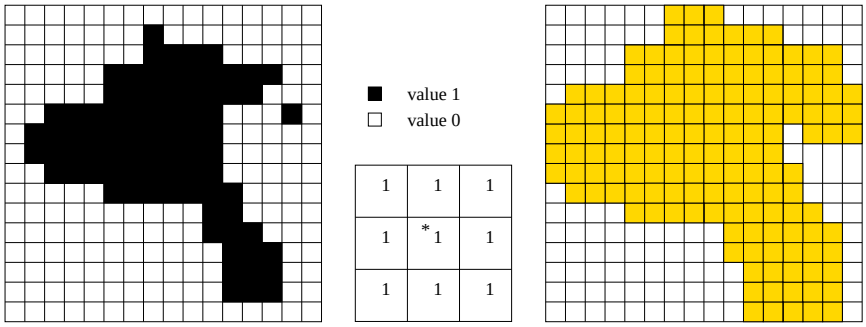
Figure 4.3: A 3×3 square structuring element with origin at *.

4.3.1 Dilation

Dilation is an operation that expands an object in some way, thus potentially filling in small holes and connecting disjoint objects. Figure 4.4 shows an example of dilation using a 3×3 structuring element.

Suppose that X is the set of Euclidean coordinates corresponding to an input binary image, and that B is the set of coordinates for the structuring element. Let B_x denote the translation of B so that its origin is at x. Then the dilation of X by B is simply the set of all points x such that the intersection of B_x with X is non-empty. The dilation operation is also written as

$$X \oplus B = \{x : B_x \bigcap X \neq \emptyset\}. \tag{4.3.1}$$



(a) X (b) B (c) $X \oplus B$

Figure 4.4: An example of dilation.

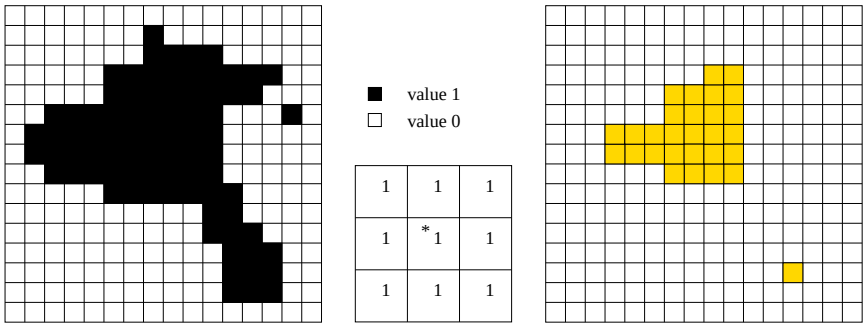
4.3.2 Erosion

Erosion is the dual operation of dilation. Erosion shrinks an object by etching away (eroding) their boundaries. Figure 4.5 shows an example of erosion using a 3×3 structuring element.

Suppose that X is the set of Euclidean coordinates corresponding to an input

binary image, and that B is the set of coordinates for the structuring element. Let B_x denote the translation of B so that its origin is at x. The erosion of X by B is simply the set of all points x such that B_x is a subset of X. The dilation operation is also written as

$$X \ominus B = \{x : B_x \subset X\}. \tag{4.3.2}$$



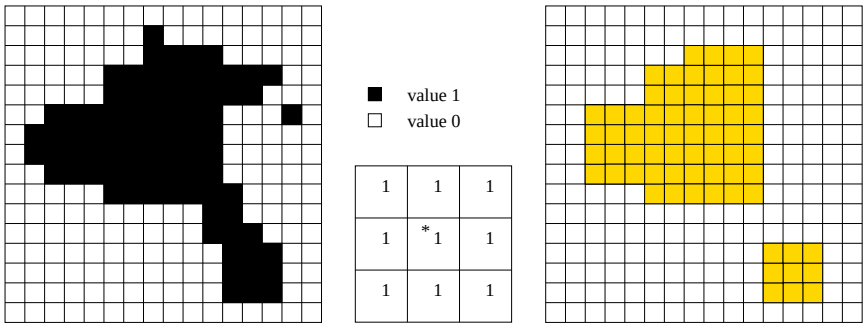
(a) X (b) B (c) $X \ominus B$

Figure 4.5: An example of erosion.

4.3.3 Opening

An opening operation is defined as an erosion followed by a dilation, using the same structuring element for both operations. Figure 4.6 shows an example of opening using a 3×3 structuring element. The opening operation over an image X by the structuring element B can be written as

$$X \circ B = (X \ominus B) \oplus B. \tag{4.3.3}$$



(a) X (b) B (c) $X \circ B$

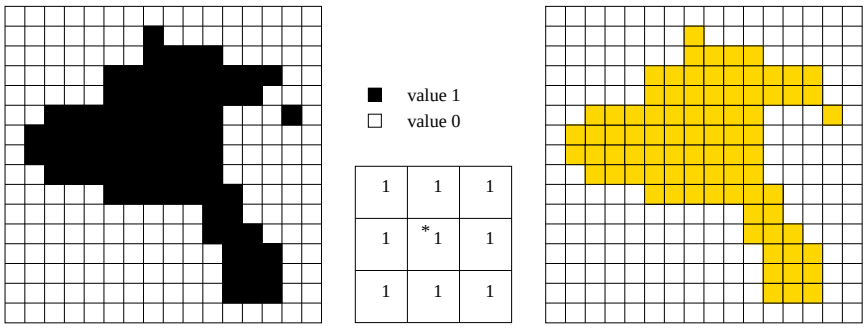
Figure 4.6: An example of opening.

4.3.4 Closing

A closing operation is simply an opening performed in reverse. It is defined as a dilation followed by an erosion, using the same structuring element for both operations. Closing is similar in some ways to dilation but it is less destructive on the original boundary shape. An example is shown in Figure 4.7. The closing operation over an image X by the structuring element B can be written as

$$X \bullet B = (X \oplus B) \ominus B. \tag{4.3.4}$$

In practice, we usually obtain a binary image that contains noise after segmentation (Figure 4.8(a)). Morphologic operations are used to remove noise and to fill in gaps on edges. Figure 4.8(b) shows a result using 3×3 operators.



(a) X (b) B (c) $X \bullet B$

Figure 4.7: An example of closing.

(a) Before morphological processing. (b) After morphological processing.

Figure 4.8: Morphological processing after segmentation.

Chapter 5

Joint localization

The main idea of joint localization is the extraction of the body parameters such as the length of the arms, hands, and so forth, from a feature posture, and consequently being able to estimate the position of joints on the occluding silhouettes. The proposed approach involves a variety of image processing techniques such as skeletonization [4, 20], chain coding [11], and corner detection [9]. The first three sections will discuss these fundamental techniques. The joint localization algorithm and its refinements will be addressed in the last two sections.

5.1 Skeletonization

5.1.1 Basics

The aim towards skeletonization is to determine the position of the joints. *Skeletonization* is a process of reducing foreground regions in a binary image to a

skeletal remnant that largely preserves the extent and connectivity of the original region while throwing away most of the original foreground pixels.

A skeleton is a lower dimensional shape description of an object that represents the topological structure of the object. It can be computed from an underlying silhouette. Four pairs of images [1] in Figure 5.3 show the extracted skeletons of their shapes. A skeleton should be able to fulfill these requirements: (1) Similar topology. It should retain the original shape. (2) Centred. It should be in the middle of the object to preserve the shape. (3) Affine-invariant. It should be invariant under affine transformations.

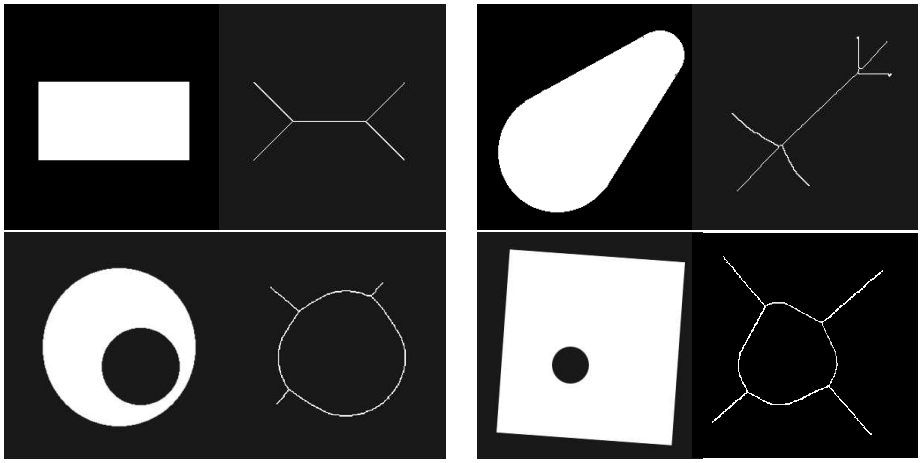


Figure 5.1: The example skeletons of 2D shapes.

There are two categories [20] of skeletonization methods. One category is based on thinning approaches. Thinning is a one-way simple deformation. More specifically, it is a deformation of an image that changes the value of simple pixels while preserving the adjacency relations between the connected components of object pixels and background pixels. It is normally implemented by

an iterative process of transforming the object pixels into background pixels without destroying the topology of the image. The notion of a simple pixel is defined as follows [20].

Definition 5.1 A simple pixel is *a single element p of a digital image with value I(p) that can change this value I(p) without destroying the topology of the image.*

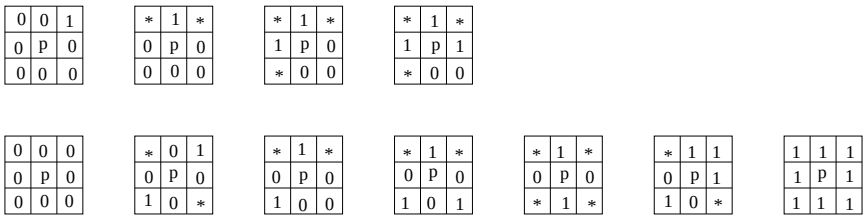


Figure 5.2: Examples of simple pixels (upper row) and non-simple pixels(lower row); the p and *s can be either 0 or 1.

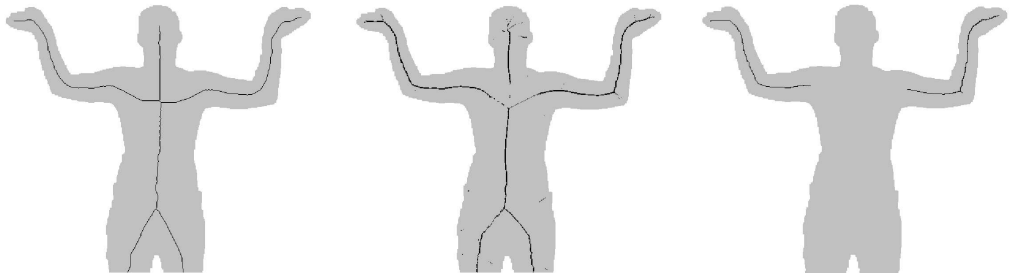
Some examples of simple and non-simple pixels are shown Figure 5.2 [22]. For example, [20] gives criteria for identifying simple pixels using an algorithm called an "iterative thinning algorithm". The skeleton obtained from thinning is a topologically equivalent image, however it is not a subset of the medial axis of the original shape.

Another category is based on distance transforms [35]. Brute force distance calculations are very expensive, since for each pixel of the region the distance to the nearest boundary point has to be evaluated for all the boundary points. An

approximate technique has therefore been developed, known as a distance transform, which tries to estimate the Euclidean distance in a reasonable time. Its main idea is to replace the global distance computation by a local propagation of distance in a small neighbourhood. This approach always requires several passes through the image data. One efficient approach, in terms of precision and cost calculation, is the Chamfer distance transform [4]. The skeletons obtained by those methods are on the medial axis of the shape. However it is not topology-preserving. A comparison of two categories of skeletonization methods is made in Figure 5.3(a) and Figure 5.3(b). The skeletons are extracted from the same underlying silhouette by using an iterative thinning algorithm and Chamfer distance transformation, respectively.

The first leads to a well connected, but not a centred result. Furthermore, we are interested in detecting corners of skeletons, but the resultant curve is very smooth, which makes it hard to detect, for example, the position of the elbow joint.

The second category of methods allows centred skeleton, but violates the connectivity constraint. For our specific task, we decided to work with the approach based on Chamfer distance transform since we want to extract the positions of joints that are always on the medial axis of the shape. We will introduce this approach in the second part of this section. The drawback of this approach must be avoided by applying an additional process that is to be discussed in the next section.



(a) Iterative thinning.

(b) Chamfer distance transform.

(c) The skeleton of arms extracted by the proposed approach.

Figure 5.3: A comparison between the different skeleton extraction methods and the proposed method.

5.1.2 Chamfer distance transform

For a given digitized object O, the distance to the background d_b of a point P belonging to O is defined as [6]

$$\forall(P \in O, Q \in O), d_b = min[d(P, Q)].$$

The Chamfer distance transforms is derived under the assumption that the distance can be computed from the values at the neighbouring position plus a mask constant. A (3,4)-Chamfer distance map can be produced in two raster scans, a forward scan and a backward scan, over the image using the mask in Figure 5.4. In the forward scan, the mask starts in the upper left corner of the image, moves from left to right and from top to bottom by using the upper triangle of the mask. When an object pixel is reached by the forward run, its lowest discrete distance d_f is computed as

$$d_f =^{min}_{\{N(P)\}} [d_f[N(P)] + d(P, N(P))]. \tag{5.1.1}$$

There, $N(P)$ denotes a neighbour of P defined in its mask. This value is then assigned to P. The same process is applied in the backward scan. It starts at the bottom-right corner, moves from right to left and from bottom to top by using the lower triangle as its mask. When an object pixel is reached by the backward run, it is assigned the value

$$d_b(P) = min[d_f(P), \overset{min}{_{\{N(P)\}}} [d_f[N(P)] + d(P, N(P))]]. \tag{5.1.2}$$

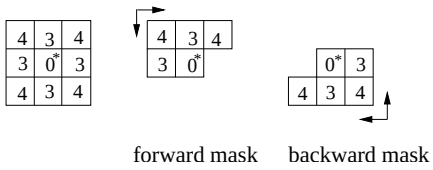


Figure 5.4: Masks used by (3,4)-Chamfer algorithm.

The skeleton can then be defined as the set of pixels S whose distance d_b is a local maximum with regard to neighbouring pixels,

$$P \in S \Leftrightarrow \forall N \in \{N(P)\}, d_b(P) \geq d_b(N(P)).$$

This simple definition of a skeleton does not preserve connectivity but it represents the medial axis at reasonable quality, as an example shows in Figure 5.3(b). The connectivity problem can be solved by a combination of Chamfer distance transform and chain coding, which will be introduced in the next section. Figure 5.3(c) shows the final result using the proposed method. It is noted that we are only interested on the arm skeletons.

5.2 Chain coding

5.2.1 Basics

A successful chain coding operation is a crucial step to stabilizing the application. Chain coding is a fundamental image processing technique which deals with a region or object representation in a compact way. Chain code representation is based on work of Freeman [11]. Standard chain coding will be applied over the silhouette to get a chained boundary of the shape, while a modified chain coding approach will be applied to the skeletons of the arms. The former provides input for corner detection (Section 5.3), and the later solves the disconnectivity problem of skeletonization (Section 5.1).

Figure 5.2.2 shows an example of the *directional encoding* of a curve. The directional codes are usually called *chain codes*. A *chain* is an ordered finite sequence of code numbers. The chain coding function takes a binary image as an input and returns a linked list of border pixels with their direction. Essentially, it starts with a point that is believed to be on the boundary, normally the upper left corner of the shape, and follows the contour in a counter-clockwise manner. Once a neighbouring pixel is found, its position is noted and then searches clockwise, starting 135 degrees counter-clockwise of the current direction. When it finds a pixel, it records the direction to that pixel in the location of the previous pixel in a labeled array. Once the starting pixel is found, we iterate over the labeled pixels to build a linked list, which is then returned. There

are eight possible directions for a link between a point and its neighbours. In Figure 5.2.2, these eight directions are numbered 0 through 7 in a counter-clockwise sense. Each of these can be considered to be an angular direction, in multiples of 45 degrees, which are considered as possible moves from one pixel to the next. The absolute coordinates (x, y) of the first boundary pixel, together with the chain codes, represent a complete description of a discrete boundary of a region.

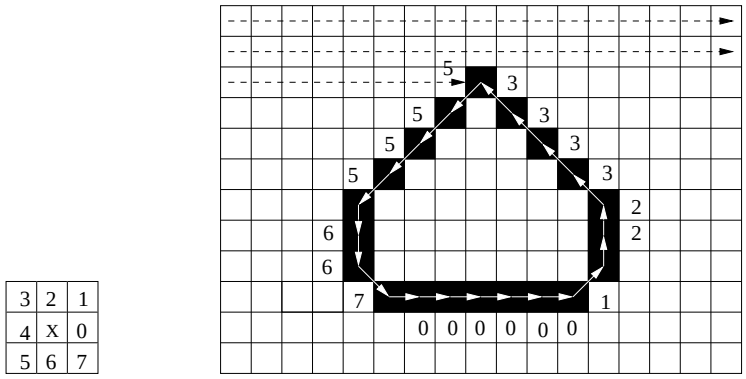


Figure 5.5: Chain code and one example of a simple shape.

5.2.2 Modified chain code

Chain coding for a body silhouette can be done by applying the conventional chain coding technique. However, the resultant skeletons from the (3,4)-Chamfer distance transform (as mentioned previously) are disconnected arcs. The problem inherent in the chain code definition must be solved before embarking on further processing. The solution here is to apply a horizontal and vertical scan wherever a pixel is found to be in a position of a gap. It defines the nearest pixel

as the next pixel, and then continues with chain coding. This solution relies on knowledge of the trend of the skeletons because we design a particular posture of the arms. The general trend of skeletons in the right arm goes from top-right to bottom-left; whereas it goes from bottom-right to top-left in the left arm.

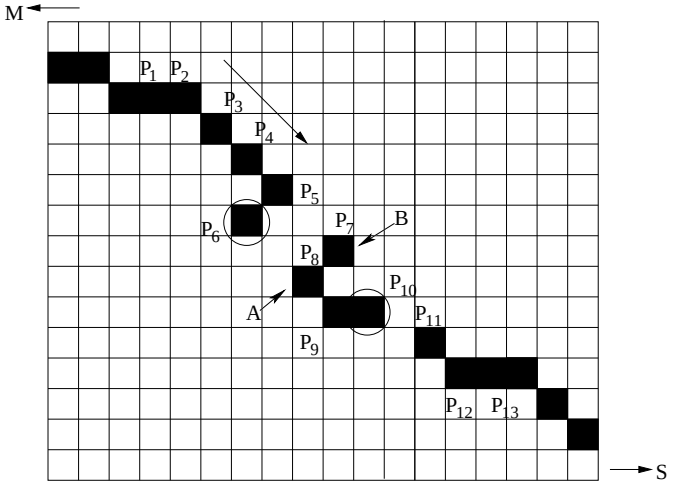


Figure 5.6: Detect possible gap positions in modified chain coding.

Two end points of the skeleton on the right arm are assumed to be known. The one end, the rightmost point $M(x_r, y_r)$, can be found by a simple raster scan. The other end, which is the right shoulder joint $S(x_s, y_s)$, can also be known from the joint localization procedure which will be introduced later in Section 5.4. The skeleton of interest runs through these two points. A pixel P_i in the right arm, that is in the position of a *possible gap* is defined as:

- Every 8-neighbour pixel on the right is either a non-skeleton pixel or a skeleton pixel previous to P_i on the chain.
- Every 8-neighbour pixel below is either a non-skeleton pixel or a skeleton

pixel previous to P_i on the chain.

An example is illustrated in Figure 5.6. Starting with pixel P_1, we arrive at pixel P_6, which is an end pixel and a position of a possible gap, as its subsegment neighbourhood pixels are non-skeleton pixels but its north-east neighbour is an "already chained" pixel (P_5). A gap is also assumed at P_{10}, as all its neighbouring pixels, except the west neighbour, are non-skeleton pixels.

If a pixel P_i assumed to be in the position of a possible gap the nearest pixel P_{i+1} (i.e., next to P_i) on the chain, can be found using the following steps:

- Search the rightmost pixel $A(x_a, y_a)$ in columns between $x_{i+1} \sim x_s$, and in rows between $y_{i+1} \sim y_s$;
- Search the topmost pixel $B(x_b, y_b)$ in rows between $y_{i+1} \sim y_a$ and in columns between $x_{i+1} \sim x_s$;
- If B is found, return B as pixel P_{i+1} next to P_i.
- If there is no such pixel found then return S as the end point.

In the example of Figure 5.6, once pixel P_6 is identified as a possible gap, A can be found in a horizontal scan, and then B can also be found in a vertical scan, defining the pixel P_{i+1}.

Similar processing is applied to the skeleton in the left arm. However, in contrast to the right arm, the skeleton of interest goes from the left shoulder joint to the

leftmost point. Therefore, searching is from bottom-right to top-left. In the modified chain code algorithm, the relationship between two adjacent pixels on the chain cannot always be presented with a direction code. This happens when a pixel is in the position of a possible gap. It is necessary to note the coordinate whenever a next pixel is found. Hence the output of the modified chain code is a set of chained points (i.e. a list of coordinates).

5.3 Corner detection

To locate the position of joints such as the shoulder joint, elbow joint, and so forth, it is necessary to extract corners from silhouettes or skeletons. A corner detection method is proposed by Chetverikov in [9]. Our application follows the general procedure defined in [9], but we do some slight modifications since we have more constraints to be considered.

The proposed two-pass algorithm in [9] defines a corner in a simple and intuitively appealing way, as a location where a triangle of specified size and opening angle can be inscribed into the curve. The input of a corner detector should be a chain-coded curve, which is converted into a connected sequence of 2D points $P_i = (x_i, y_i), i = 1, 2, ..., n$. For each point P on a curve, the corner detector tries to inscribe a triangle (P^-, P, P^+) along the curve, where P^- is the backward point and P^+ the forward point. In the second pass, similar to edge detection, we find the local minima angle of these inscribed triangles. One example is shown in Figure 5.7. Assume that P_1, P_2 and P_3 are three points in a sequence,

triangles are inscribed at the three points along the edges of the shape. Angles α_1, α_2 and α_3 are opening angles of these triangles. Since α_2 appears to be a local minima, P_2 is regarded as a corner.

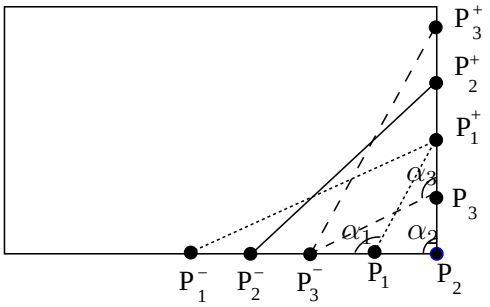


Figure 5.7: An example for the corner detection algorithm.

The notion of a corner is subjective. The corner here is defined as an obvious corner such as an armpit. The small corners caused by the uneven dressing or the quality of the silhouette are neglected. This definition requires long triangle arms: $L^-_{arm} = ||P^-P||$, or $L^+_{arm} = ||PP^+||$. However, the problem in working with a long triangle arm is that we also detect " false triangles". As shown on the right of Figure 5.8, the inscribed triangle cannot represent the corner at that point. To solve this problem, we examine every point P_m between the points P^+ and P^-. If the distance d_m from P_m to backward arm $\overline{PP^-}$ or forward arm $\overline{PP^+}$ is longer than a threshold T_d, $d_m > T_d$, then we do not inscribe a triangle to that point. The opening angle of that point can be set with a large value. The corner detection algorithm is stated formally below.

1. `For each pixel` P `on the chained object contour, construct such a`

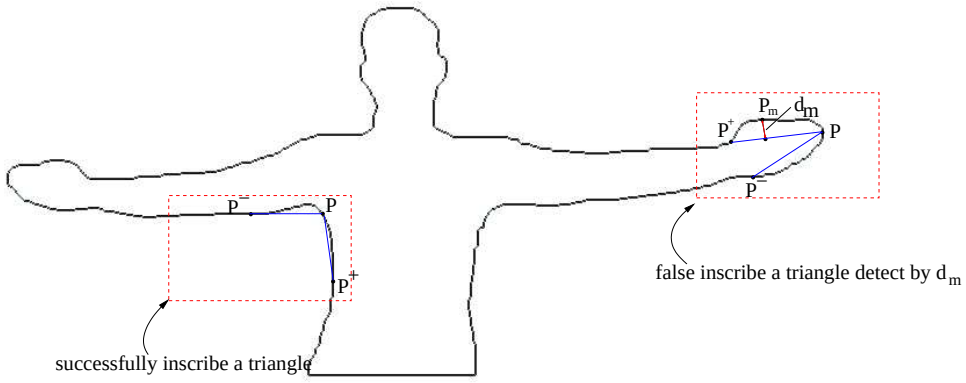


Figure 5.8: The examples show a successful inscribing triangle (left) and a false inscribing triangle (right).

triangle (P^+, P, P^-) as constrained by a set of rules as follow,

(a) Triangle arms $L^-_{arm} \geq T_{arm}$ and $L^+_{arm} \geq T_{arm}$, where T_{arm} is a pre-defined threshold of the length of triangle arm;

(b) For each pixel P_m on the chain between P^+P and PP^-, if d_m is the distance from P_m to the forward arm $\overline{P^+P}$ or the backward arm $\overline{PP^-}$, then $d_m < T_d$, where T_d is a pre-defined threshold of distance value.

2. If the triangle (P^+, P, P^-) is successfully inscribed then calculate its opening angle and assign this value to that pixel;

3. For each pixel P on the chained object contour, if it is satisfied by:

(a) The pixel P is assigned with an angle;

(b) `The assigned angle is a local minima;`

(c) `The assigned angle is less than a pre-defined angle threshold` T_{angle}.

`Then this pixel is at corner position, otherwise it is not.`

5.4 Initial estimation of joint locations

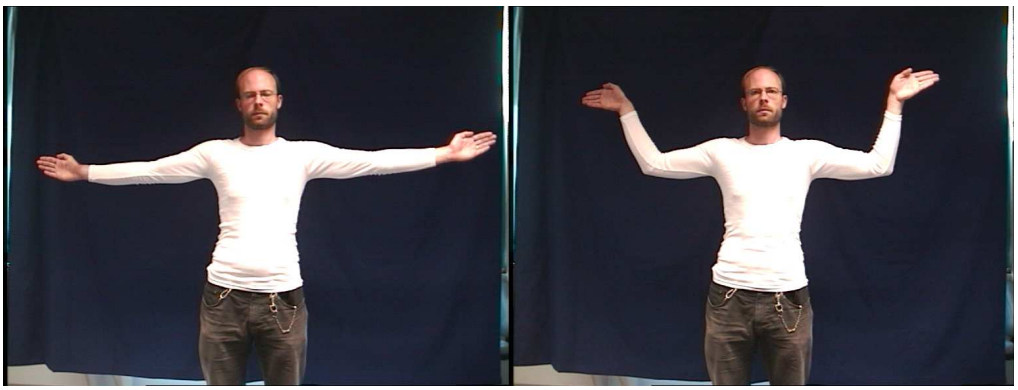


Figure 5.9: The input images for joint localization. Left: a front view. Right: a joint view.

A particular pose which is called *joint view* (as shown on the right of Figure 5.9) is used to locate the positions of the joints. We begin the joint localization process with shoulder joints and neck joint extraction directly from the front view (the left of Figure 5.9). Shoulder joints are located by searching armpits D_1 and D_2 (see Figure 5.10), which are the two lowermost corners over the silhouette that are not located on the bottom line. The position of the neck joint can be found when walking along the boundary of the silhouette from an upper shoulder point towards the head. It is common sense that the slice where a neck joint is located, is always the narrowest x-slice of the silhouette.

Therefore, when we walk along the boundary from the upper shoulder point U_1, the x value will increase. We stop at a point whose x value does not increase. It can be asserted that this point N_1 is located in the slice of the neck joint. Likewise we have N_2 from another side. So far, two possible slices are obtained from both sides. The narrowest one gives the neck joint. Next, to gain the length of the hand, upper arm, and so forth, we use a special reference frame (joint view). The aim at this stage is to find the positions of the elbow joints and wrist joints. We first perform a skeletonization operation (Section 5.1) on the joint view. Second, we apply a corner detection operation (Section 5.2 and 5.3) on the skeleton. This gives us the positions of the elbow joints and wrist joints. Lengths of body parts therefore can be calculated. With these parameters it is possible to estimate the joints within the occluding front view. The joint localization algorithm is summarized in four steps as follows:

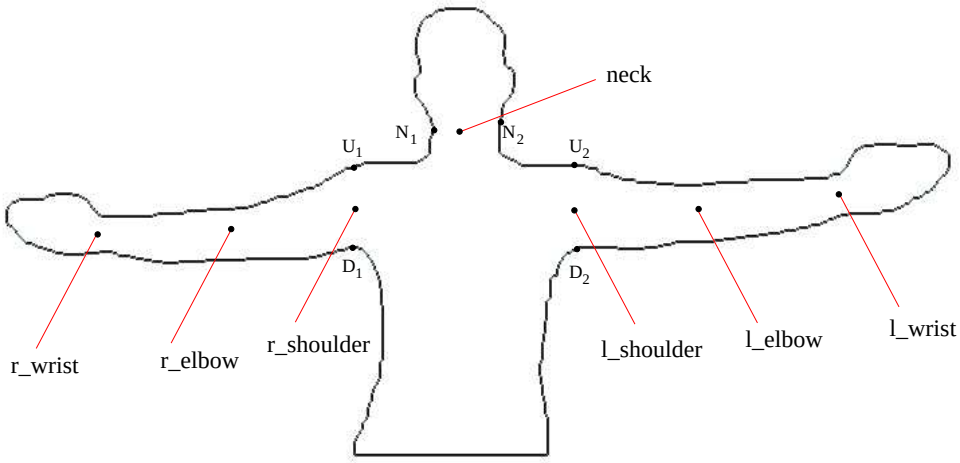


Figure 5.10: Joints and feature points.

Step 1: Estimate the shoulder joints in the front view.

(a) Extract the chained boundary from the front view silhouette;

(b) Apply corner detection on the chained boundary, find the armpits D_1 and D_2.

(c) Locate the shoulder joints by scanning up from the armpits.

Step 2: Estimate the neck joint in the front view.

(a) Search for the upper shoulder points U_1 and U_2, from the armpits D_1 and D_2;

(b) Walk along the boundary from U_1 and find the narrowest point N_1;

(c) Walk along the boundary from U_2 and find the narrowest point N_2;

(d) Compare the widths of the horizontal slices through N_1 and N_2, the neck joint is located at a slice has small value.

Step 3: Estimate the elbow joints and wrist joints in the joint view.

(a) Extract the skeleton in the joint view;

(b) Locate the shoulder joint in the joint view, extract the chained skeleton of the arm;

(c) Apply corner detection on the chained skeleton of the arms, estimate the elbow joint and wrist joint in the joint view;

(d) Calculate the lengths of the upper arm L_{up} and lower arm L_{low} from the positions of the shoulder joints, elbow joints and wrist joints in the joint view;

Step 4: Estimate elbow joints and wrist joints in the front view.

(a) Locate the elbow joints in the front view with the length of the upper arm L_{up}, starting from the shoulder joint;

(b) Locate the wrist joints in the front view with the length of the lower arm L_{low}, starting from the elbow joint.

A corner detection operator normally outputs a list of corner points. We need to classify these corners as belonging to a particular joint or feature point. Firstly, it is easy to determine a corner that belongs to the left or right arm with the knowledge of relative position between the centroid of the silhouette and corners. Secondly, further consideration is needed according to which corner we want to detect. In step one, a corner detector is applied over the silhouette and the armpits can be regarded as the lowest corners next to the bottom line. In step three, corner detectors are applied to the skeletons of the two arms respectively. The wrist is the uppermost corner while the elbow is the lowermost corner. The body part cutting in Figure 5.11 presents the result of joint localization.

5.5 Refinements on joint localization

We have addressed the main strategy of joint localization. In practice, it is observed that the elbow joint and wrist joint are often not very accurate. An

Figure 5.11: The result of joint localization.

additional mechanism for estimating "anatomic better fitting" joint locations is needed.

The refinement for the position of the wrist joint is straightforward. A general knowledge of location around the wrist is available from the initial estimation. We measure the heights of the y-slices within the wrist area. The slice at the wrist joint has the lowest value. The left wrist slice is the leftmost one having a lowest value while the right wrist slice is the rightmost one.

As to elbow joint localization, an arm anatomy view in Figure 5.12 [24] is helpful for understanding its refinement. From the previous section, we regard the elbow joint as located on the medial axis at E'. Actually, from the anatomical point of view, it is at E, one end of the humerus. We visualize the error of locating the elbow joint at E' by an experiment that overlays the model on images (the framework of the experiment will be described in Chapter 7). It is shown in the middle of Figure 5.14.

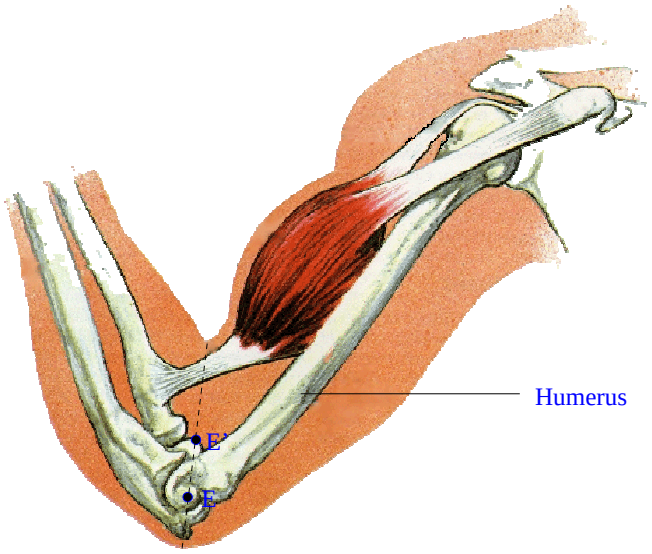


Figure 5.12: Anatomy of an arm [24].

To estimate the position of the elbow joint, since the knowledge of E' and the shoulder joint S (Figure 5.13) is available, we extend the line $\overline{E'S}$ to reach the boundary at B. It is easy to compute the middle point C of line segment $\overline{E'B}$. We use the distance of $||CS||$ to approximate the length of the upper arm L_{up} and locate the elbow joint in the front view. The right of Figure 5.14 shows this approximation, which has a better result in comparison to the initial joint estimation.

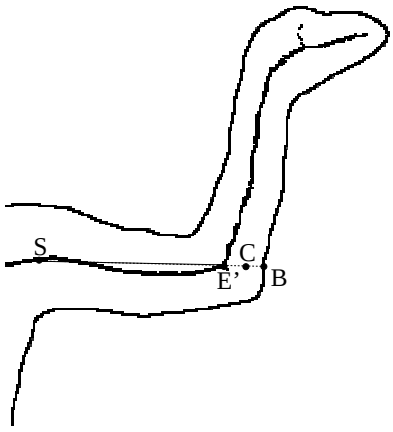


Figure 5.13: Extend the predetermined elbow joint from E' to C.

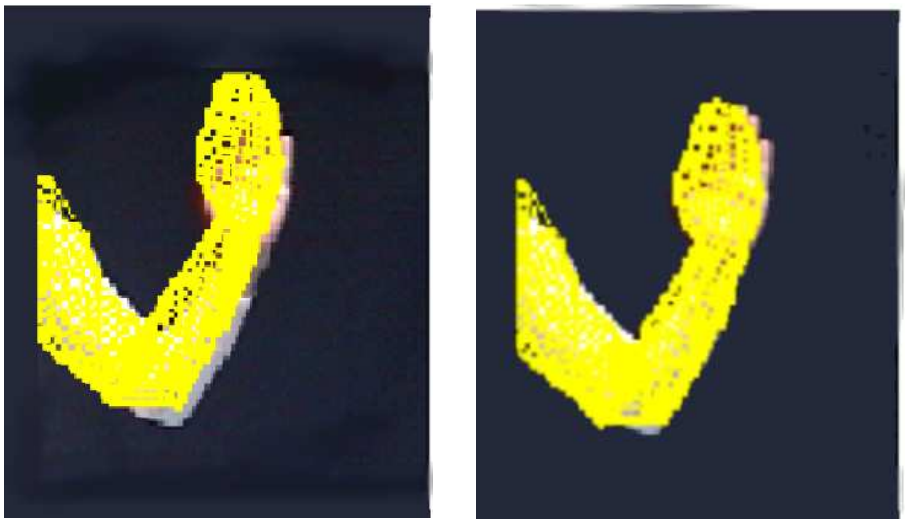


Figure 5.14: A comparison of using different methods to estimate the elbow joint. Left: overlaid with the initial estimated model. Right: overlaid with the refined model.

Chapter 6

3D Reconstruction of a human body

Reconstruction of an object surface is a well known discipline in computer vision [22]. The task is to build surface meshes for a human being from two mutually orthogonal views by using the "shape-from-silhouettes" [22] technique.

6.1 Basics

There are two main approaches for 3D reconstruction using shape from silhouettes. One is a volume carving approach [22] and the other a surface-based approach [39]. The volume carving approach calculates projection rays of all the points on the occluding contours and cuts the predetermined volume of interest into object and non-object. It is a simple, fast and robust method. However, it identifies every voxel in the volume of interest, and does not give an explicit way to define points on surface meshes.

In comparison to volume carving, the surface based approach is a different type of mathematical approach for surface modelling. It uses B-spline surface patches as a mathematical model of 3D surface reconstruction from epipolar geometry, which deals with the correspondence problem between two images. A point on the occluding contour in one image is matched to a point on a contour in the other image by searching along epipolar lines. The surface-based approach produces a better result than the volume carving approach for an object having a smooth and continuous surface, for instance, a human body. This section will discuss fundamentals of the surface-based approach. It first includes the geometry of a camera system as well as epipolar theory, and continues with the introduction to the proposed mathematical model.

6.1.1 Perspective camera model and camera calibration

3D scenes can be projected on 2D images by perspective transformations. A common way to model perspective cameras is to use a pinhole camera model [22]. An ideal pinhole model, shown in Figure 6.1, is the simplest geometric camera model. It consists of the image plane and the optical center O, located at distance f, the focal length of the optical system. The optical axis is the line perpendicular to the image plane which crosses O.

The relationship between a 3D point $P(X, Y, Z, 1)^T$ and its projected point $p(x, y, 1)^T$ on the image plane can be written as

$$x = -\frac{fX}{Z} \quad \text{and} \quad y = -\frac{fY}{Z}. \tag{6.1.1}$$

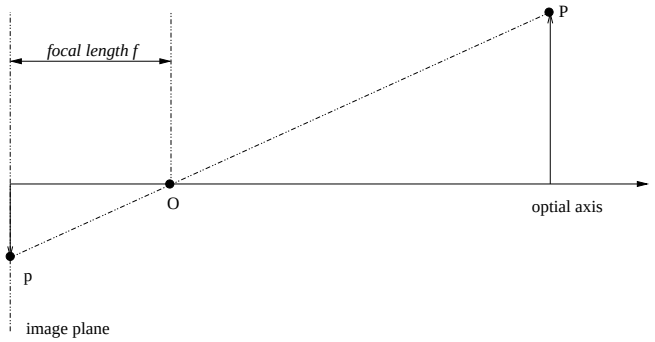


Figure 6.1: The pinhole camera model.

A linear mapping can be used to represent the projection from 3D to 2D in homogeneous coordinates. In general, this mapping can be defined by a 3×4 matrix P, called the projection matrix. The projection matrix P can be obtained by combining a series of coordinate transformations and a projection into the camera coordinate system. It involves the following steps:

1. **Rigid body motion.** A transformation from the 3D world coordinate system to the camera coordinate system;

2. **Central Projection.** A projection of the camera coordinates onto the image plane;

3. **Pixel transformation.** A transformation of the undistorted coordinates into the distorted coordinates, and a shift of centred distorted coordinates into non-centred image coordinate having the origin at the upper-left corner.

We further assume that the projection is a linear projection, which means that

there is no lens distortion. The camera projection matrix is as follows,

$$P = \underbrace{\begin{pmatrix} -fk_{cx} & 0 & x_c \\ 0 & -fk_{cy} & y_c \\ 0 & 0 & 1 \end{pmatrix}}_{K} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \underbrace{\begin{pmatrix} R_{3\times3} & T_{1\times3} \\ 0_{3\times1} & 1 \end{pmatrix}}_{M}, \qquad (6.1.2)$$

where f is the focal length, (x_c, y_c) are the coordinates of the image center (in pixels), and the scaling factors k_{cx} and k_{cy} are expressed in relative units (pixel/meter) and inversely proportional to the horizontal and vertical size of the pixel.

The rotation matrix R and translation vector T describe the orientation and position of the camera with respect to the world coordinate system. They are components of the matrix M and are called *extrinsic* camera parameters. Meanwhile, the *intrinsic* camera parameters f, k_{cx}, k_{cy}, x_c and y_c, which define the matrix K, specify the optics and the physics of the camera.

Camera calibration is the process of estimating the extrinsic and the intrinsic parameters of a camera. Various camera calibration methods exist [22]. In 1986, R. Y. Tsai suggested a method today well known as *Tsai's calibration* [37]. Because it works accurately and there is a fully developed free software available on the Internet [2], Tsai's method is widely used. It has two variants, one for coplanar calibration marks and one for non-coplanar calibration marks. Non-coplanar calibration requires at least seven accurately measured points which are given in arbitrary but known geometric configuration. Fully optimized calibration requires at least eleven points.

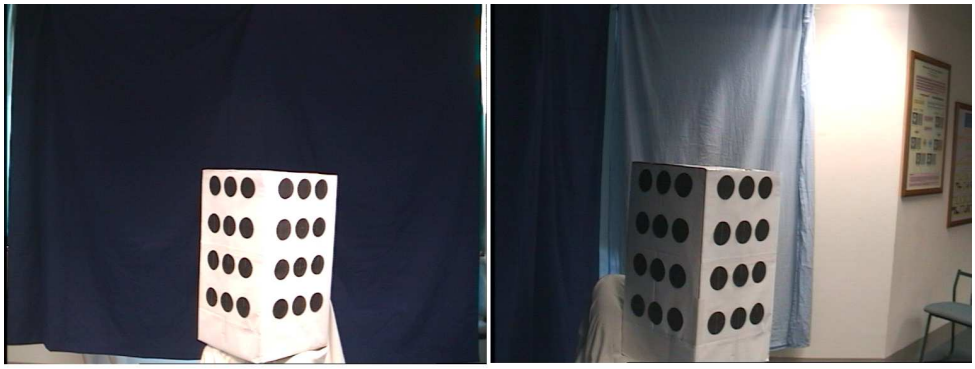


Figure 6.2: Two views of a calibration cube from both cameras; the left one is from the front camera, and the right one from the side camera.

To calibrate two nearly perpendicular positioned cameras, calibration marks are placed on three faces of a cube as shown in Figure 6.2, so that each camera is capable of seeing two faces. In our set-up, the calibration cube contains thirty-six marks with twelve on each face.

6.1.2 Epipolar geometry

The epipolar geometry [22] is the intrinsic projective geometry between two views. It is independent of scene structure and only depends on the camera parameters. The epipolar geometric constraint is useful for correspondence search. Assume we start a correspondence analysis process with a point in one image plane, say x_1. A search for the corresponding point x_2 in the second image plane can be simplified significantly by utilizing the epipolar geometry underlying the binocular image acquisition. Figure 6.3 visualizes this geometric

relation of a binocular system. An *epipolar plane* is defined by a point X in 3D space and both optical centers O_1 and O_2. The *base line* is incident with O_1 and O_2. If both image planes are not coplanar, the base line intersects each plane in one point. These two intersection points are called *epipoles*. The epipole e_1 is the projection point of O_2 in Camera 1, and e_2 is the projection point of O_1 in Camera 2. An *epipolar line* is the intersection of an epipolar plane with an image plane. An epipolar line on an image plane with respect to X is incident with the epipole and its projection point. This allows the following definition of the epipolar constraint: A point x_1 in one image can only correspond to such a point x_2 in the second image which lies on the corresponding epipolar line in the second image which is uniquely defined by x_1.

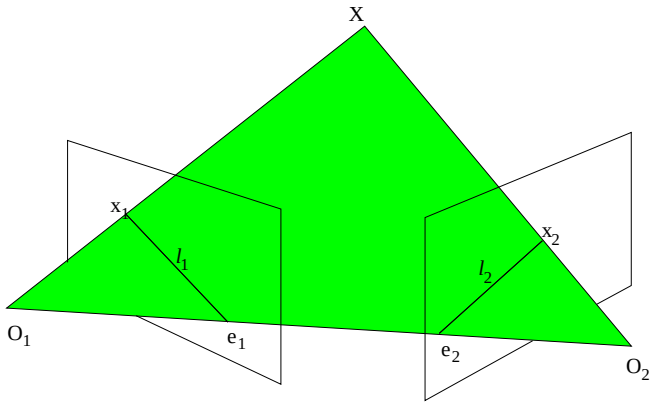


Figure 6.3: Epipolar geometry.

In the example of Figure 6.3, the epipolar constraint allows to define the epipolar line l_2 as a function of x_1, $l_2 = Fx_1$, where F is the *fundamental matrix*. As x_2 belongs to l_2, we have $x_2^T l_2 = 0$. The relationship between the corresponding

image points x_1 and x_2 satisfies, $x_2^T F x_1 = 0$. Actually, the fundamental matrix F is the algebraic representation of epipolar geometry and encapsulates this intrinsic projective geometry. It can be computed from either the knowledge of the internal camera parameters and relative camera pose, or from at least eight point correspondences [12]. In this thesis, we investigate the computation of a fundamental matrix only from camera properties.

Suppose the projection matrices of two cameras are known, P_1, for the front camera, and P_2, for the side camera. The relation between X and x_1, x_2 are given by

$$x_1 = P_1 X, \tag{6.1.3}$$

and

$$x_2 = P_2 X. \tag{6.1.4}$$

Therefore, the ray back-projection from x_1 is obtained by solving $P_1 X = x_1$. The solution is given as

$$X = P_1^+ x_1, \tag{6.1.5}$$

where P_1^+ is the pseudo-inverse of P_1. It can be computed by

$$P_1^+ = P_1^T (P_1 P_1^T)^{-1}. \tag{6.1.6}$$

For an epipole e_2, we have, $e_2 = P_2 O_1$. The epipolar line is the line joining two projection points of e_2 and x_2. It can be represented as the cross product of these two points $l_2 = e_2 \times x_2$. Together with Equation(6.1.4), l_2 is derived as

$$l_2 = (P_2 O_1) \times (P_2 X). \tag{6.1.7}$$

Combining with Equation (6.1.5), we get

$$l_2 = (P_2O_1) \times (P_2P_1^+x_1) = Fx_1. \tag{6.1.8}$$

We then have the fundamental matrix

$$F = (P_2O_1) \times (P_2P_1^+). \tag{6.1.9}$$

6.1.3 B-spline interpolation

Free-form surfaces represented as B-splines are suitable geometric models for representing complex 3D objects such as a human body. B-spline curves [3, 13] are piecewise polynomial functions that can provide approximation of shapes using a sequence of a small number of *control points*. These control points indicate the general shape of a curve while B-spline curves result in smoothing of coarsely digitized contours. Figure 6.4 shows an example of a B-spline curve.

A B-spline curve can be expressed as a linear combination of basis functions,

$$P(t) = \sum_{i=0}^{n} P_i N_{i,k}(t), \tag{6.1.10}$$

where P_i $(i = 0, 1, \cdots, n)$ are the vertices of the control polygon, and $N_{i,k}(t)$ are B-spline basis functions of order k. The order k determines the number of control points that have an influence on the points of the curve. The curve is then C^{k-2} continuous. For the cubic B-spline that we use in the project, we have C^2 continuity for $k = 4$.

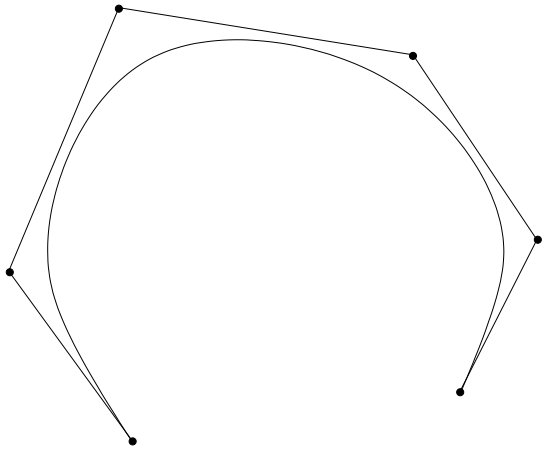


Figure 6.4: An example of a B-spline curve defined by control points.

The de Boor algorithm [10] provides a fast and numerically stable algorithm for finding a point on a B-spline curve at the interval defined by a *knot vector* $T = (t_0, t_1, t_2, \cdots, t_{n+k})$, with $t_0 \leq t_1 \leq \cdots \leq t_{n+k}$. It defines the B-spline basis function recursively as follows:

$$N_{i,1}(t) = \begin{cases} 1 & t_i \leq x \leq t_{i+1} \\ 0 & otherwise \end{cases} \tag{6.1.11}$$

$$N_{i,k}(t) = \frac{t - t_i}{t_{i+k-1} - t_i} N_{i,k-1}(t) + \frac{t_{i+k} - t}{t_{i+k} - t_{i+1}} N_{i+1,k-1}(t). \tag{6.1.12}$$

In terms of geometry, the de Boor algorithm actually is a corner-cutting process. That is, corner $P_i^{[r-1]}$ is cut by line segment $P_i^{[r]} P_{i+1}^{[r]}$. Figure 6.5 illustrates that the cutting begins at polygon $P_{j-k+1} P_{j-k+2} \cdots P_j$. After k-1 iterations, we reach the point P_j^{r-1} which is on the curve.

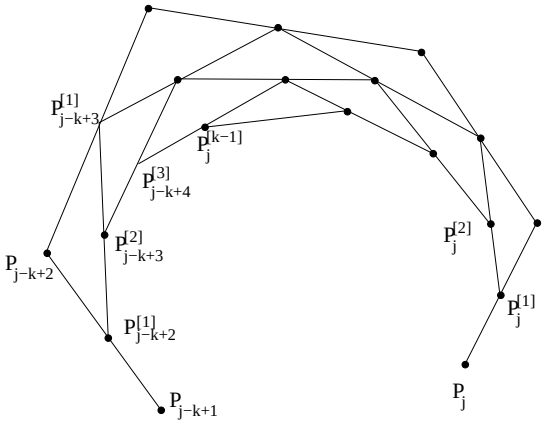


Figure 6.5: Corner cutting processing of the de Boor algorithm.

6.2 Modelling a human torso

6.2.1 Geometric reconstruction

Figure 6.6: The input images for reconstructing the torso. Left: the front view. Right: the side view.

The body torso and the arms are treated as two different sets of free-form surface patches. The surface patches of the body torso are sliced horizontally

whereas the surface patches of the arms are sliced vertically. Once body segments are separated by the joint localization procedure (Chapter 5), it allows to reconstruct the torso and arms separately. The reconstruction process takes two perpendicular views (Figure 6.6), the body parameters together with camera parameters as input. Since then each slice has four control points to fit a B-spline curve. The principle of reconstruction for a human torso is illustrated in Figure 6.7. The main idea is to find the four control points C_1, C_2, C_3 and C_4 for each slice and, then to interpolate them as a B-spline curve. It assumes that a horizontal line in a silhouette of the front view corresponds to a slice in 3D space. Therefore we can start with one slice of the front view, and use its edge points P_{11} and P_{21} as the first two reference points. Multiplying them with the fundamental matrix (Equation (6.1.9)) to the second camera gives the epipolar lines (Equation (6.1.8)). Their correspondences P_{41} and P_{42} in the side view are the intersection points of the epipolar line with one downline of the second silhouette. In the case of Figure 6.7, it is on the right. Hence the control points C_2 and C_3 are obtained. On the other hand, to find the other two control points C_1 and C_4, we start from P_{31} and P_{32}, which are the points located in the same row of P_{31} and P_{41} but on the opposite side (the left side) of the side view. The algorithm for modelling a torso is summarized below.

1. For each horizontal slice in the front view, find its edge points P_{11} and P_{21};

2. Calculate the epipolar lines to the side view for P_{11} and P_{21}, and search for their corresponding points, P_{41} and P_{42};

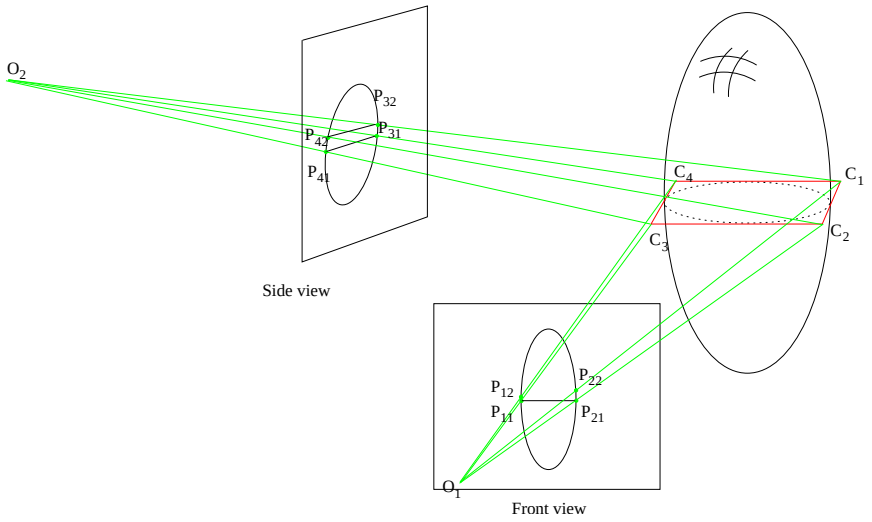


Figure 6.7: Modelling the torso.

3. A horizontal scan from P_{41} and P_{42} in the side view locates the edge points P_{31} and P_{32} in the other side;

4. Calculate the epipolar lines to the front view for P_{31} and P_{32}, and search for their corresponding points P_{12} and P_{22};

5. Estimate four control points from corresponding pairs in 3D space:

   (a) Projection rays from P_{22} and P_{23} intersect at C_1;

   (b) Projection rays from P_{21} and P_{42} intersect at C_2;

   (c) Projection rays from P_{11} and P_{41} intersect at C_3;

   (d) Projection rays from P_{12} and P_{31} intersect at C_4.

6. Interpolate as a B-spline curve from the four control points.

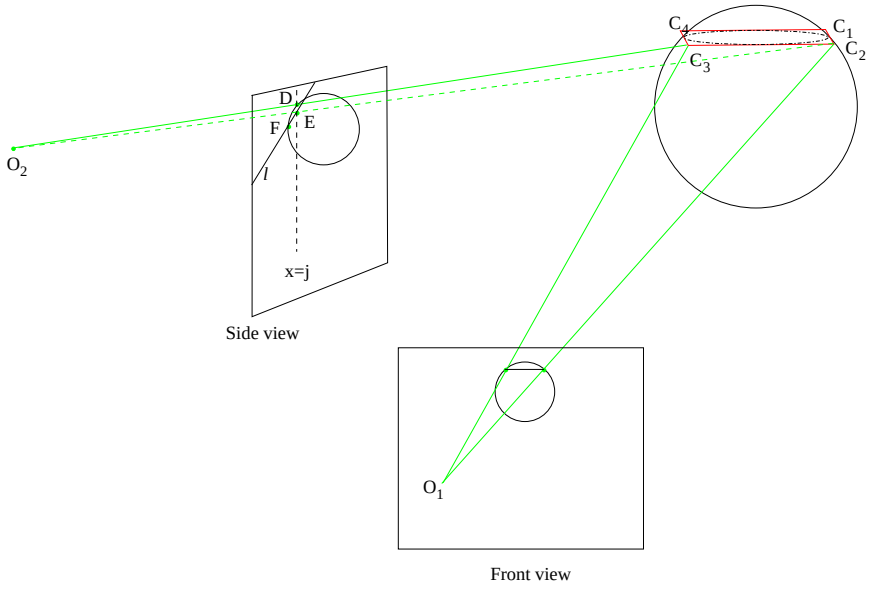


Figure 6.8: Error in modelling when the object is close to the top of the image.

6.2.2 Correction function

It is observed that searching for a corresponding point in such a way by the intersection of epipolar lines and silhouettes, will cause errors when the control point is invisible in one of the views. This is shown in Figure 6.8. The control point C_2 is invisible in the side view. Its projection point E is inside the silhouette. However, the corresponding point obtained from this approach is F which is the intersection of the epipolar line l and the silhouette contour. In particular, this error will be obvious when the object of interest lies in the area far from the image center. In order to reduce such an error, we propose a correction function. With respect to the side view, C_3 is on the same side as C_2 and always visible. It assumes that their projection points D and E are on the same column j in the image. Therefore the corresponding point of C_2 in

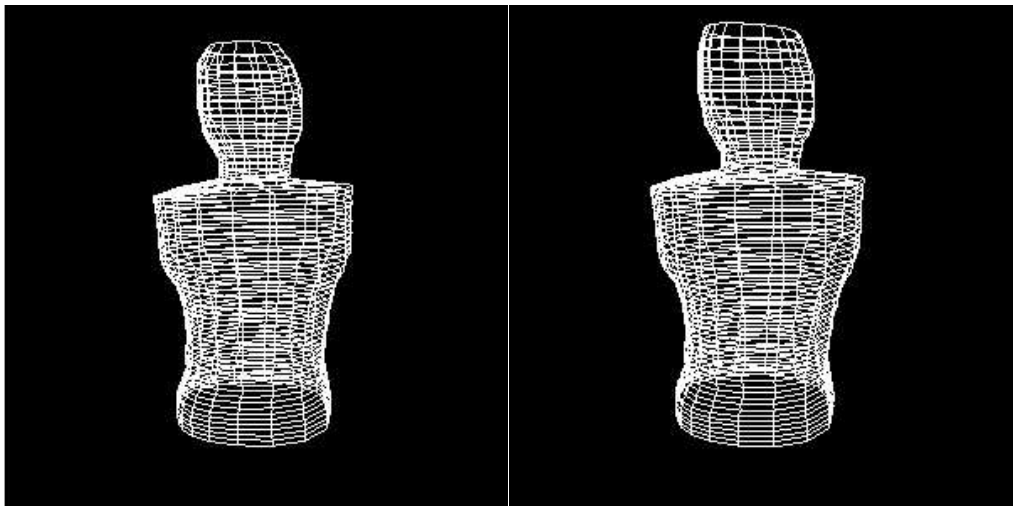


Figure 6.9: Visualization of the two different methods. Left: surface model of the torso with the correction function. Right: surface model of torso without the correction function.

the side view is adapted to the intersection of l and vertical line $x = j$. This correction function is also applied to the other three control point pairs: C_3 and C_4, C_4 and C_1, C_1 and C_2. Figure 6.9 visualizes the error caused in the normal method by comparing it with the method using the correction function.

6.3 Modelling arms

In comparison to the 3D reconstruction of the torso, we use a different scheme to build the model of the arms, as they are sliced vertically instead of horizontally. Arms can also be modelled from two nearly perpendicular views, which are the front view and the side view of arms. However, both views are acquired from the same camera instead of two cameras. This is shown in Figure 6.10. The front view is the same image as in the torso reconstruction (on the left in Figure 6.10), and the side view of arms is obtained by turning over the arms at

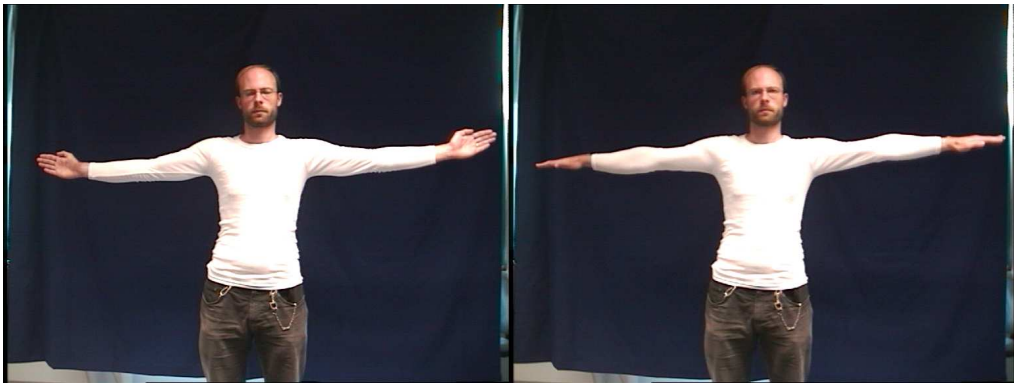


Figure 6.10: The input images for reconstructing the arms. Left: the front view. Right: the side view of the arms.

ninety degrees as shown on the right of Figure 6.10.

6.3.1 Aligning the arms

Epipolar theory cannot be applied to the correspondence analysis for arm reconstruction since only one camera is used. We establish the corresponding pairs by simply matching vertical slices between two views (see Figure 6.11). In practice, the corresponding slices may not lie in the same column of the two images when the human subject turns over the arms. This problem can easily be solved by aligning the arm in both views at the fingertips. We regard the front view as a reference view, and the corresponding slice l_2 in the side view with respect to slice l_1 in front view is given by

$$l_2 = w_{a1} + (l_1 - w_1) \tag{6.3.1}$$

for the right arm, and

$$l_2 = w_{a2} - (w_2 - l_1) \tag{6.3.2}$$

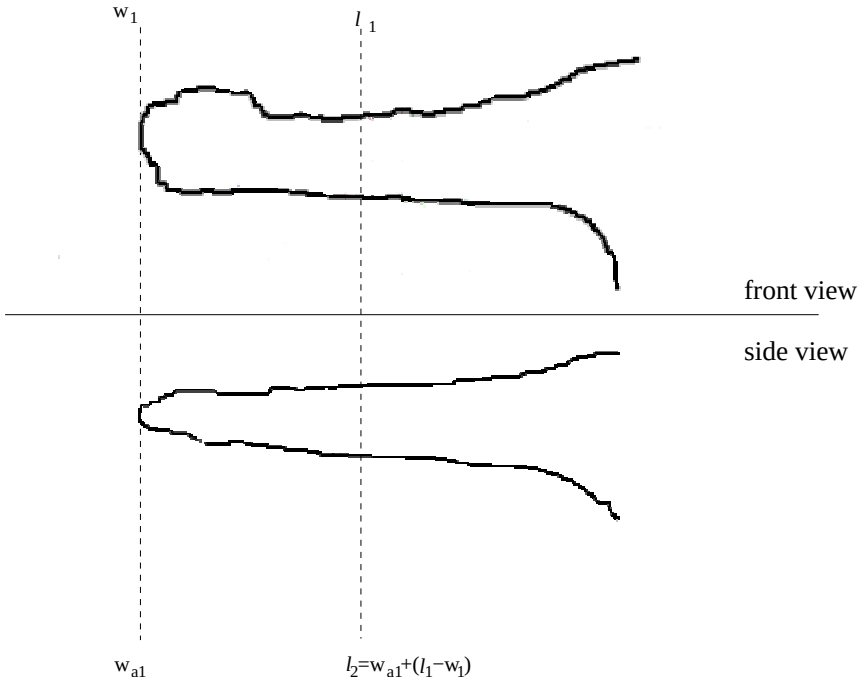


Figure 6.11: Two views of the right arms are aligned at the fingertip.

for the left arm. There, l_1 is the slice column in the front view, l_2 is the slice column in the side view corresponding to l_1, w_1 is the right fingertip column in the front view, w_2 is the left fingertip column in the front view, w_{a1} is the right fingertip column in the side view, and w_{a2} is the left fingertip column in the side view.

6.3.2 Mid-plane geometry

We define the mid-plane as a plane going through the middle of the torso (see Figure 6.12). It assumes that this mid-plane also goes through the middle of the arms and all the joints of the arms are laid on the plane. As we have already

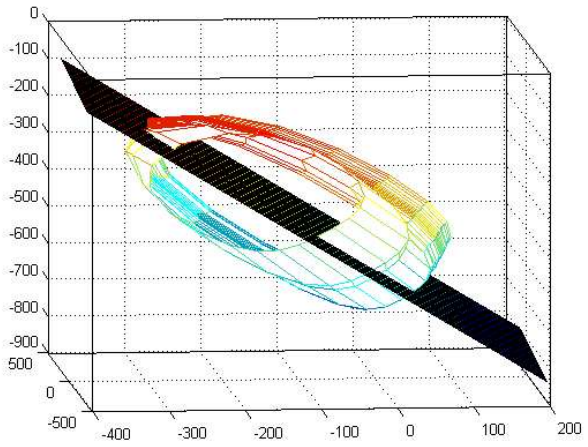


Figure 6.12: A mid-plane goes through the middle of the torso.

reconstructed the torso, it is possible to calculate the equation of the mid-plane from geometric data of a slice in the torso. Normally we choose a closest slice to the shoulder joint. Figure 6.13 illustrates the principle of the calculation. The points on the slice plane S are given, therefore vectors v_1 and v_2 are known. The normal of the slice plane n' is given by

$$n' = v_1 \times v_2. \tag{6.3.3}$$

Thus, the normal of the mid-plane n is

$$n = n' \times v_1. \tag{6.3.4}$$

Having the normal of a plane means that its parameters a, b, c are known. Together with any point on the mid-plane π_0, the parameter d of π_0 can be computed from Equation (2.1.7).

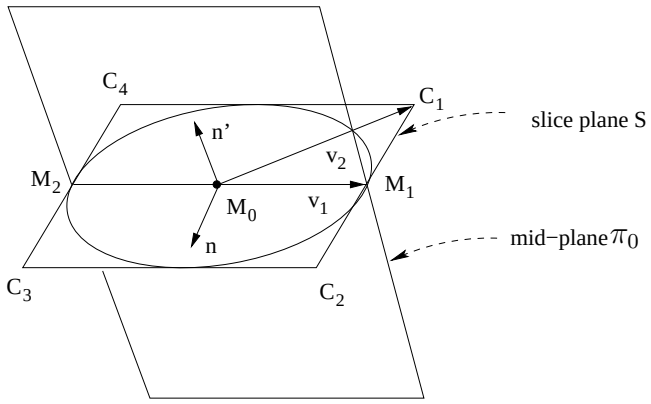
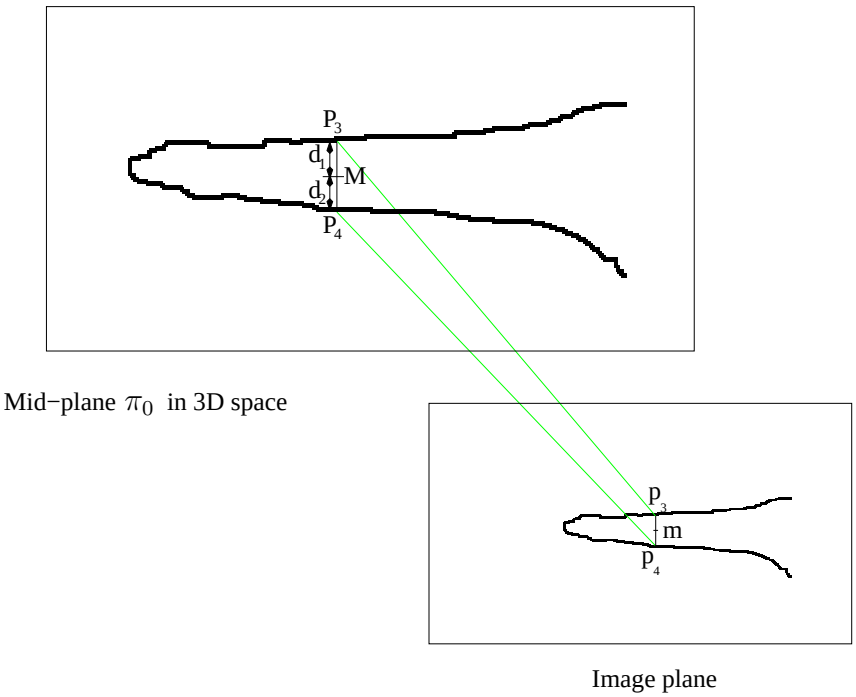


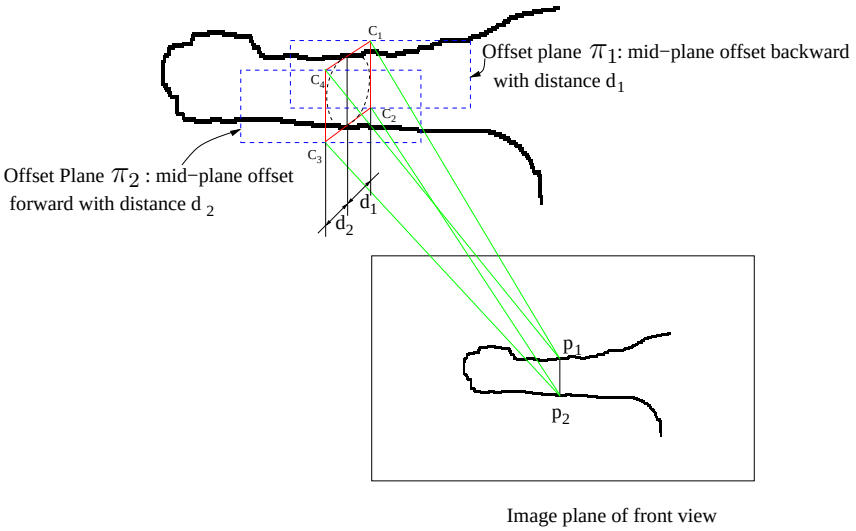
Figure 6.13: Mid-plane π_0 is calculated from geometric data of a slice plane S.

6.3.3 Geometric Reconstruction

Figure 6.14 illustrates the computation of the 3D control points for an arm slice. We offset the mid-plane backward or forward; then we reconstruct the rays from the edge points of the slice. The intersections of these rays with the planes give desired control points. The backward or forward offset distance for each slice can be computed from the side view. This is shown in Figure 6.14(a). In the side view, we have two edge points of a slice, p_3 and p_4 and their mid-point m in the 2D image. Back projecting them onto the mid-plane gives their 3D positions P_3, P_4 and M. Spatial distances from P_3 and P_4 to M, d_1 and d_2, are the offset distances. Since the offset planes π_1 and π_2 have the same normal as mid-plane $\pi_0 = (a_0, b_0, c_0, d_0)^T$, the equations of the offset planes can be derived as $\pi_1 = (a_0, b_0, c_0, d_1)^T$ and $\pi_2 = (a_0, b_0, c_0, d_2)^T$. Result surface meshes are visualized in Figure 6.15. We summarize the algorithm for modelling arms below.



(a) Offset distances are calculated when the arms of the side view are projected onto the mid-plane.



(b) Control points are estimated by projecting the arms of the front view to the offset mid-planes.

Figure 6.14: Modelling the arms.

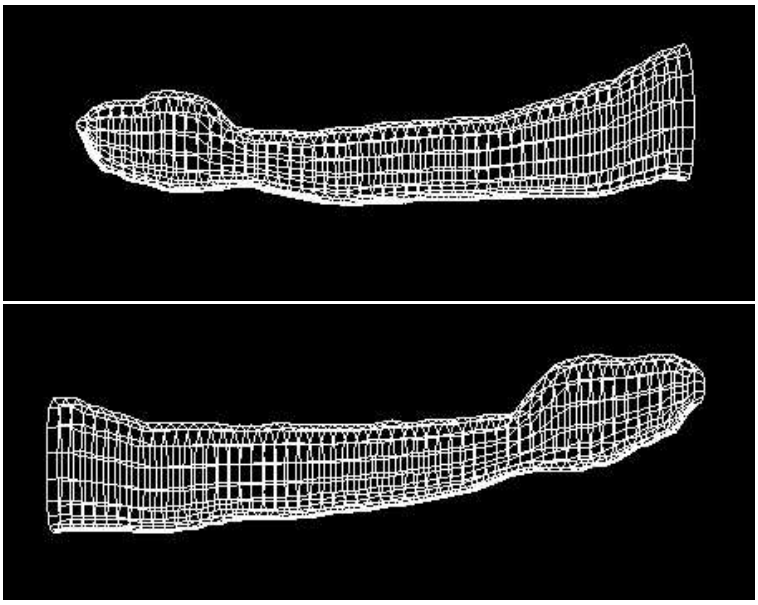


Figure 6.15: Grid visualization of the results of modelling arms, from up to bottom: the right arm and the left arm.

1. For each vertical slice in the front view of the arms, find the edge points p_1 and p_2;

2. Align the side view of the arms with the front view at the fingertips, find the corresponding points p_3 and p_4 in the side view;

3. Reconstruct the projection rays from p_3, p_4 and their middle point m in the side view; estimate their 3D positions P_3, P_4 and M by intersecting these rays with the mid-plane respectively;

4. Calculate the distance $d_1 = ||MP_3||$ and $d_2 = ||MP_4||$, offset the mid-plane forward with distance d_1 as plane π_1 and backward with distance d_2 as plane π_2;

5. Reconstruct the projection rays from p_1 and p_2 in the front view; estimate the control points of the slice as rays intersecting with the offset plane π_1 and π_2:

   (a) Projection ray from p_1 intersects plane π_1 at C_1;

   (b) Projection ray from p_2 intersects plane π_1 at C_2;

   (c) Projection ray from p_1 intersects plane π_2 at C_3;

   (d) Projection ray from p_2 intersects plane π_2 at C_4.

6. Interpolate the four control points as a B-spline curve.

6.4 Integration

The joint localization procedure allows the classification of 3D data into different data sets of body segments. However, all the 3D points computed so far are defined in the world coordinates. As the body segments are required to be defined by their own local coordinate systems, we need to transform the world coordinates to the local coordinates. Furthermore we also need to investigate the translations between neighbouring segments so that we can store the data sets into a VRML file.

Transforming a coordinate system from one to another requires the following information: a normalized rotation matrix (see Section 2.2) defining the orientation of the new coordinate system, and the translation between both systems. We assume that the local coordinate systems have exactly the same orientation

as the mid-plane π_0. The orientation R of the mid-plane with respect to the world coordinate system can be decomposed into three components, vector u which is the orientation of axis x, vector v which is the orientation of axis y, and vector w which is the orientation of axis z. Referring to Figure 6.12, the direction of axis x, y and z is the same as vector v_1, n' and n, respectively. Thus, vectors u, v and w are equal to the normalized vectors of v_1, n' and n, respectively.

The translation from the world coordinate system to the local coordinate system is actually revealed by its origin with respect to the world coordinate system. We also assume that the origins of local coordinate systems are all placed on the mid-plane π_0. They generally sit at joints or extrema points. The arrangement of these origins is depicted in Table 6.1 and also visualized in Figure 1.1(b). In a 2D image, they are already known during the joint localization processing. Hence, the positions of these origins with respect to the world coordinate system can be estimated by reconstructing the rays from the image and intersecting them with the mid-plane.

Given the orientation R and origin O' with respect to the world coordinate system, the formula of transforming a point P in world coordinate system to a point P' in the local coordinate system is as follows:

$$P' = P \cdot R - O' \cdot R. \tag{6.4.1}$$

There, O' is the coordinate of an origin with respect to world coordinate system,

Body segments	Positions of origins
head	top tip
torso	neck joint
left hand	left finger tip
left lower arm	left wrist joint
left upper arm	left elbow joint
right hand	right finger tip
right lower arm	right wrist joint
right upper arm	right elbow joint

Table 6.1: The arrangements for the local coordinate systems.

and $\cdot$ is the dot (inner) product operator. The offsets or translations between neighbouring body segments are determined by the distances of their origins. We have assumed that all the joints and extrema points are located in the same plane, it follows that offsets in z-direction are always zero.

6.5 Texture mapping

Texture mapping is useful to get a more realistic colourful model. This section presents an approach of texture mapping using the surface patch of the head as an example. This is because the surface patch of the head carries more information than other patches.

The main idea of texture mapping is to get an image by combining two orthogonal views in a proper way to get the highest resolution for most detailed parts. We first map two views onto the model. Since the stereo system is calibrated, it is possible to project the 3D points onto the corresponding views and get their

texture coordinates. In the next step, we flatten the textured 3D models and generate two texture images. As each surface patch is defined in grid structure, flattening grid points onto a 2D image at any resolution is straightforward. For example, to flatten an $n \times m$ surface patch onto an image, we divide this image into an $n \times m$ grid as well. Each grid of the image is associated with a grid on the 3D model. It can therefore be assigned with the same texture coordinates. This produces two texture images for both the front view and side view (see Figure 6.16). One of the advantages of this approach is that the two generated images are matched pixel by pixel. This enables us to gain a fused texture image with the highest resolution. Burt *et al.* [7] proposed a multi-resolution method for removing boundaries between different image sources (see the first of Figure 6.17). [7] uses a weighted average splining technique. For the sake of simplicity, we adapt it to a linear weighted function in our work.

Figure 6.16: Texture maps of two views of a head, from top to bottom: front view and side view.

Suppose that one image, $Fl(i)$, is on the left and the other, $Fr(i)$ is on the right, and that the images are to be combined at point i. Let $Hl(i)$ be a weighting function which decreases monotonically from left to right, and let $Hr(i) = 1 - Hl(i)$. Then the combined image $F(i)$ is given by

$$F(i) = Hl(i)Fl(i) + Hr(i)Fr(i). \tag{6.5.1}$$

This function is applied to a predetermined transition zone over which Hl ranges from 1 to 0. There are two transition zones defined in the upper part of Figure 6.17. When it is applied with the multi-resolution method, the edges in transition zones are no longer visible (the lower part of Figure 6.17).

Figure 6.17: Two views are connected before (top) and after (bottom) the multi-resolution method is applied.

Chapter 7

Experimental results

This chapter demonstrates some experimental results for our system. The experiments are implemented on a machine that has a 2.8GHz Pentium IV CPU and 1GB RAM running under a Linux Redhat 9.0 environment. We used EVI-D31 CCD cameras. The first experiment demonstrates different test cases and evaluates the performance of our system. The second experiment proposes an application to track the joint movements in 3D space. It uses the knowledge of body parameters gained in the model generation system. The 3D model is overlaid on the image sequence in order to visualize the errors between subject and generic model. This is also a first stage of human motion tracking. We apply the notions of human motion modelling which have been introduced in Chapter 2.

7.1 Experiment I – Different test cases

We test our algorithm on different test subjects. Figure 7.1 shows four test subjects. The experiment showed that we are able to generate different models

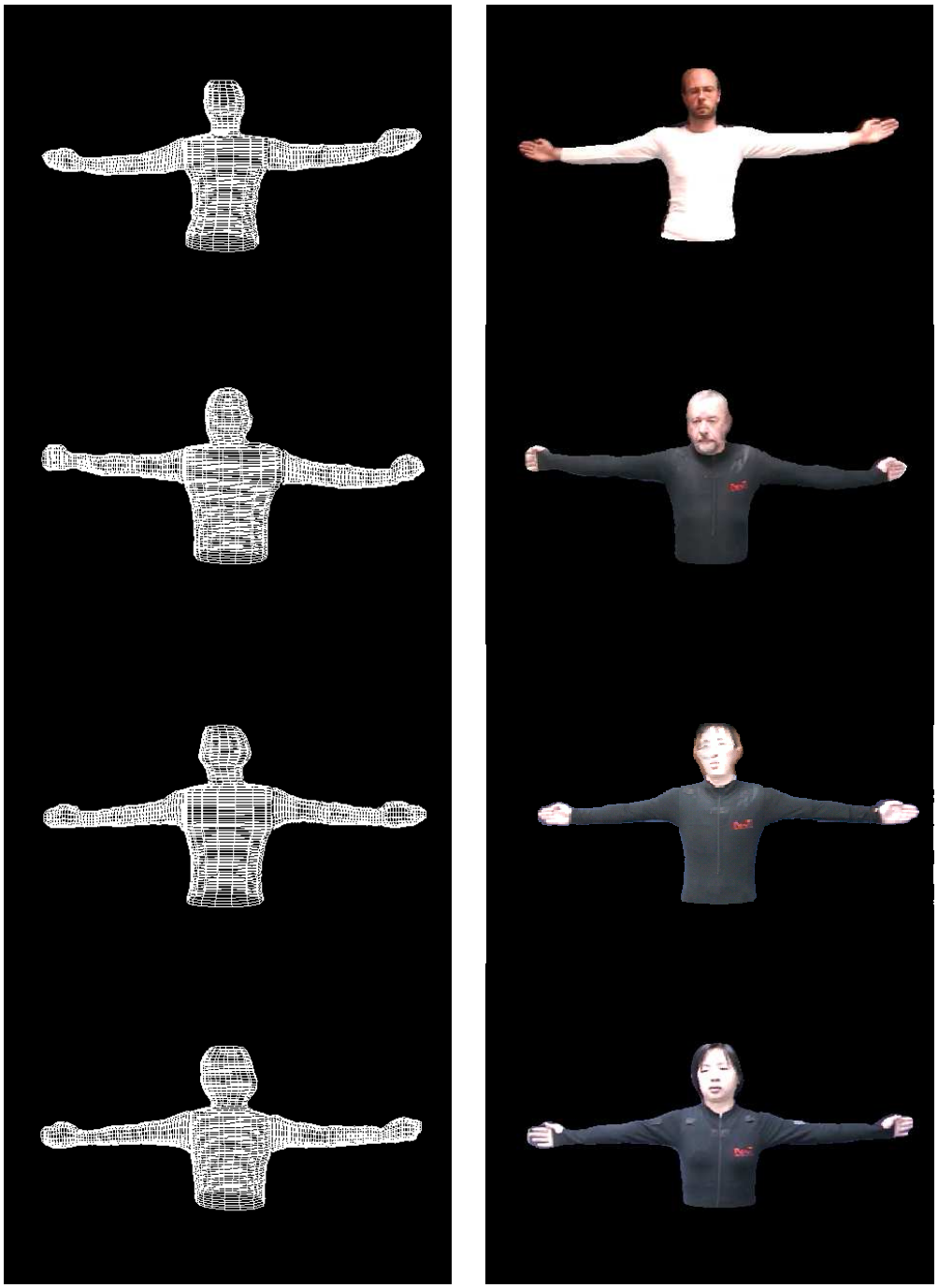


Figure 7.1: 3D models of different test subjects. Left: grid models. Right: texture-mapped models. From top to bottom, subject "Bodo", "Reinhard", "An" and "Lei".

with fine-efficiency. On the other hand, for our tests we asked the subjects to perform a particular pose. In our practice, we found that not everyone will follow a briefly described pose in the same way. This is shown in Figure 7.2. Poses have differences. Bodo (upper left), presents a pose with bended wrist; whereas Reinhard the (upper right) presents a non-flexible wrist configuration. However, our system proved to be very robust in joint localization and can be used under such differing situations.

Figure 7.2: Joint poses for different subjects.

The data in Table 7.1 show differences in lengths of body parts between subjects and our generic model. For a subject, we manually measure the lengths of body parts and compare these data with the geometric data calculated from the

3D model. It shows that the errors vary by about 1-2cm, which is satisfactory for our purpose. It is suspected that the potential source of errors are from the following aspects. (1) Segmentation. Although our background subtraction algorithm is robust, we cannot completely remove all the shadows. This particularly effects finding the wrist joint. As we regard the narrowest position on the silhouette of arm as the wrist joint, the shadow of hand always enlarge the shape of the hand. This is the reason why the lengths of the hands are always bigger than expected. Furthermore, due to the restriction of our lab facilities, we do not have a monochromatic blue background. Therefore a global threshold does not guarantee that every pixel can be accurately distinguished into background and object. (2) Camera calibration. Errors of calibration normally come from these points, the precision of the calibration planes; the measurement accuracy of the calibration patches. In practice, we also consider that the error may be due to the shape of the calibration object. We designed such an object having a long shape so that it can cover a subject in height since the system is used for a standing person. However it cannot cover the subject in width as the subject is stretching the arms. Therefore the calibration error will propagate towards the edge of the image horizontally. (3) The manually measurement of subjects.

Having the human models, it is capable of simulating the motions of a subject in a virtual environment. The image sequence on the top of Figure 7.3 shows the motions of a subject. The pose results from the sequence are modelled by a

Subject		Measurement	Generic model	Error
Bodo	head	25.6	25.7	0.1
	lower arm	25.0	27.0	2.0
	hand	20.0	19.0	1.0
	lower arm	25.0	27.0	2.0
	upper arm	25.0	27.0	2.0
	width	185.2	187.3	2.1
Reinhard	head	23.5	23.7	0.2
	hand	18.0	19.0	1.0
	lower arm	27.0	27.0	0.0
	upper arm	27.8	27.0	0.8
	width	179.0	181.0	2.0
An	head	24.2	24.1	0.1
	hand	18.0	19.0	1.0
	lower arm	21.0	20.2	0.8
	upper arm	21.1	22.0	0.9
	width	166.1	168.0	1.9
Lei	head	22.6	22.7	0.1
	hand	17.0	18.0	1.0
	lower arm	20.2	19.1	1.1
	upper arm	21.3	21.0	0.3
	width	151.1	152	0.8

Table 7.1: Comparison of body parts in lengths between measurement and generic model (unit:cm).

5DOF kinematic chain for each arm. There are three joints on the shoulder (up-and-down, backward-and-forward, twist) one joint on elbow (up-and-down) and one joint on the wrist (up-and-down). The estimated joint angles are provided by the human motion analysis project [32, 33]. We impose this data to the generic models and drive them to perform the same movements. The bottom four rows in Figure 7.3 shows three poses of the subjects simulating the motions of a real person.

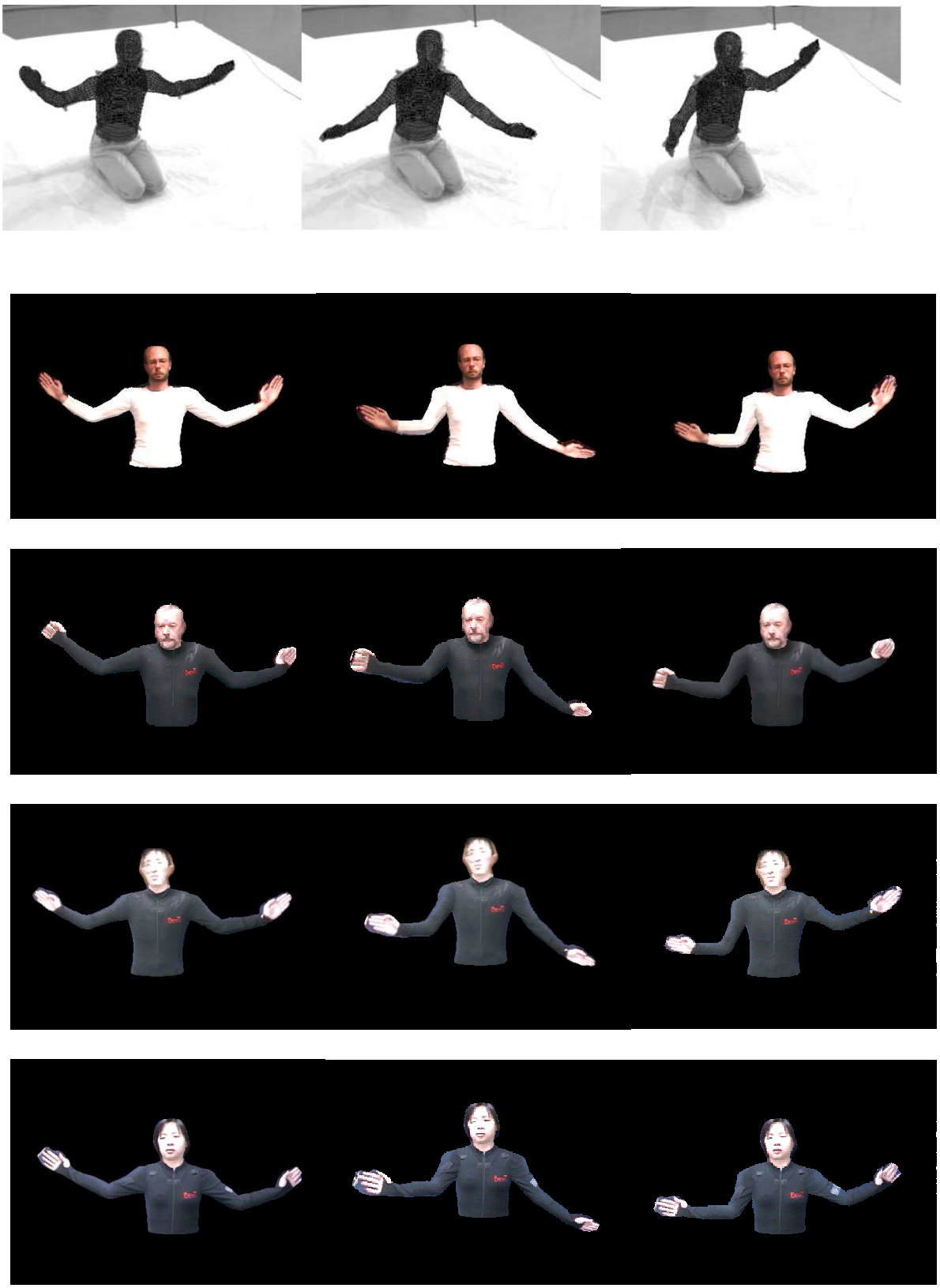


Figure 7.3: The captured motions (top row) and different subjects perform the same motions (bottom four rows).

7.2 Experiment II–Joint movement estimation

7.2.1 Framework of the experimental application

The application takes a video sequence as input. The human motions captured have six DOFs, one DOF on each wrist, elbow, shoulder. This implies that the arms are moving up and down within the mid-plane. It assumes that the right arm moves only on the right side of the body, and the left arm moves only on the left. The same segmentation and skeleton extraction algorithms stated in Chapter 4 and Chapter 5 are applied on the image frames. At this stage, human body parameters (i.e., the lengths of the body segments) are already computed during the model reconstruction. It allows finding the joints with these parameters. The next step is to calculate the joint angles from the image data.

Joint recovery

For the sake of simplicity, we assume that the human subject is standing at the same place as in the model reconstruction experiment. The position of the shoulder joints are considered the same as the front view. As the length of the upper arm L_{up} and length of the lower arm L_{low} have already been given from model reconstruction, joints are expected to be recovered by measuring the lengths from the skeleton of the arms. However, as mentioned in Chapter 5, the skeletonization algorithm does not preserve the connectivity and the poses of the arms are arbitrary. The modified chain code algorithm cannot be applied here. Alternatively, the elbow joint can be reached by a raster scan for a skeleton

point starting from the shoulder joint with distance L_{up} in arm space, while the wrist joint can be reached by a similar scan starting from the elbow joint with distance L_{low}. Note that in searching for the wrist joint and starting from the elbow, it probably reaches two candidate positions. One is at the upper arm and the other is at the lower arm. They are easy to identify by comparing the distances to the shoulder joint. The wrist is located in the position with longer distance.

Calculation of joint angles

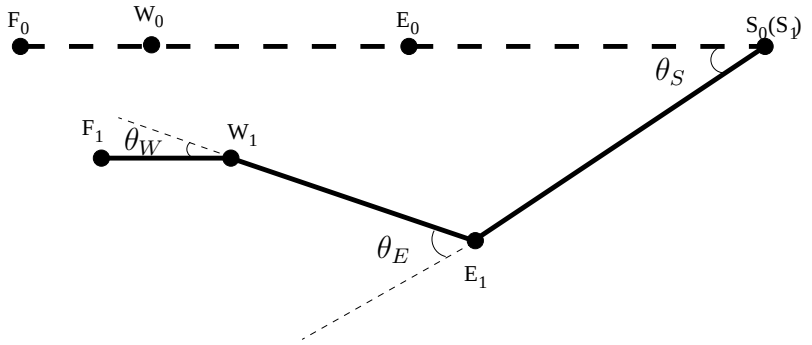


Figure 7.4: Joint angles

The angle of the shoulder joint is calculated from the comparison between the initial pose and the current pose of the upper arm. As shown in Figure 7.4, the initial elbow joint position E_0 and shoulder joint position S_0 are given from the front view pose (initial view), the elbow joint position E_1 and wrist joint position W_1 are given from the current pose. The shoulder joint angle is

$$\theta_S = \angle(\overrightarrow{E_0S_0}, \overrightarrow{S_1E_1}).$$

There, $\angle(\overrightarrow{E_0S_0}, \overrightarrow{S_1E_1})$ denotes the angle between the vectors $\overrightarrow{E_0S_0}$ and $\overrightarrow{S_1E_1}$.

Similarly, the angle of the elbow joint is estimated by the angle between the vectors $\overrightarrow{S_1E_1}$ and $\overrightarrow{E_1W_1}$:

$$\theta_E = \angle(\overrightarrow{S_1E_1}, \overrightarrow{E_1W_1}).$$

The angle of the wrist joint is estimated by the position of the fingertip F_1 together with the position of the wrist joint W_1 and the elbow joint E_1:

$$\theta_W = \angle(\overrightarrow{E_1W_1}, \overrightarrow{W_1F_1}).$$

However, normally the skeleton does not reach the finger tip. A fingertip finding algorithm is performed in two steps. First of all, the hand is segmented by classifying the object into hand pixels and non-hand pixels. We then regard the fingertip as a hand pixel having the longest distance to wrist joint W_1.

Step 1: Hand segmentation. Given the position of the wrist joint W_1 and elbow joint E_1, hand pixels can be segmented by drawing a cutting line l through W_1 and orthogonal to the line $\overline{W_1E_1}$. Since E_1 is known, it is easy to determine that the hand pixels are located on the side opposite to E_1 of l. However, in a case as similar to that shown in Figure 7.5, some non-hand pixels may also be included. A verification function is therefore introduced. For each candidate pixel located on the hand side of line l draw a line from that point to the wrist joint. If the line goes through the background space, it is not a hand pixel, and vice versa. In the example, as line $\overline{PW_1}$ passes through the background, P is not a hand pixel.

Step2: Searching for finger tips. A straightforward way to find a finger tip

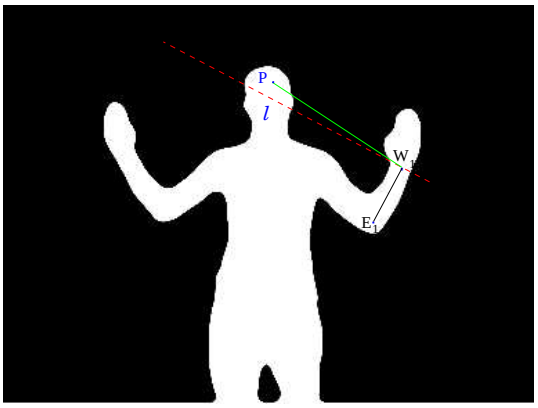


Figure 7.5: Hand segmented by cutting line l.

is to calculate the distance to the wrist joint W_1 from each hand pixel. The pixel having the longest distance value is regarded as the finger tip.

7.2.2 Joint transformed model

The human model generalized by the system has three DOFs at each joint. However, the human anatomy allows for much more DOFs than that. A human being is not only able to move the arms backwards/forwards or up/down, but also able to move the shoulder simultaneously. An example during an image sequence is shown in Figure 7.6. A person is moving the arms down, and, as shown in the left image, the shoulders are moving downwards as well. This additional DOF can be modelled by a linear joint transformation. The amount of joint transformation is steered through the angle of the shoulder joint. The pose result for the non-transformed model is shown in the right image and the joint-transformed model is shown in the middle image. This shows that the matching result of the joint transformed model is much better. It is observed that, if the shoulder is moving, muscles are tense and the skin is morphed. The

amount of surface deformation is also dependent on the joint angle. Research related to this topic is reported in [32]. Due to time constraints, we did not implement the surface morphing but concentrated on the joint transformation which is sufficient to extend the experiment in this way.

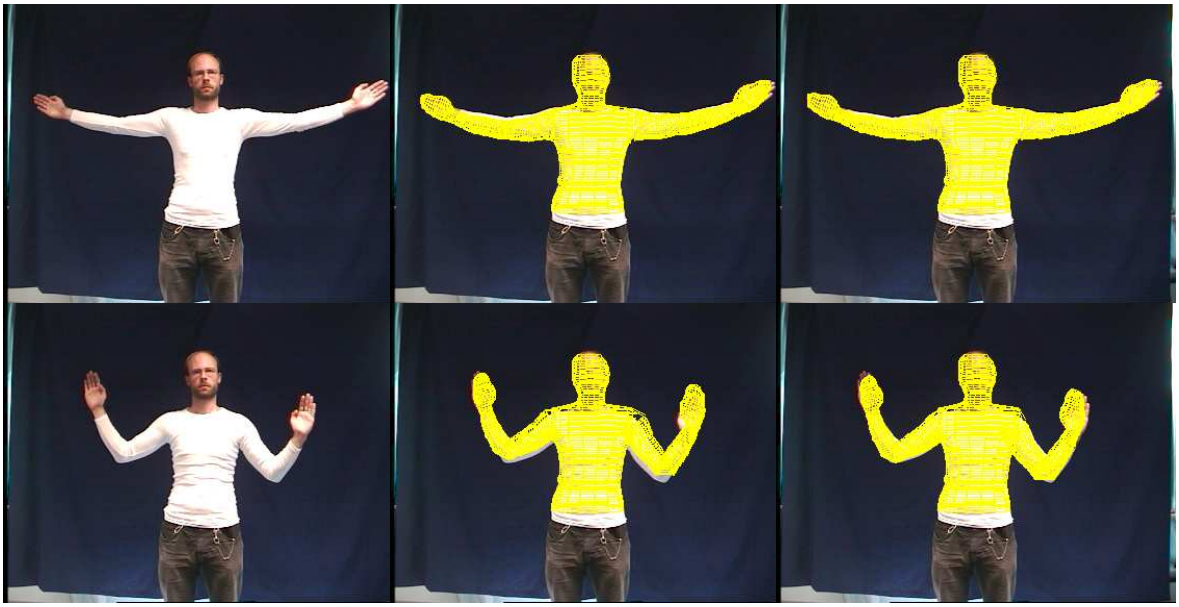


Figure 7.6: Comparison of pose results for a joint transformed model and a non-transformed model in the arms, from left to right: the original model, the non-transformed model and the transformed-model.

7.2.3 Tracking the joints

This section presents experiments is to study the performance of the joint tracking system. The algorithm is tested on a monocular view (front view) video sequence at a resolution of 384×286 pixels. In the first experiment, a tracked sequence contained 85 frames is captured. Figure 7.7 shows six frames of a tracked sequence. In this sequence we use just three joints on each arm. All joints are up-and-down joints. The first of the pair is the original image. The

second is the 3D model overlaid to the image. As can be seen, the pose overlaid with image data appears to be good. The diagram in Figure 7.8 shows the estimated angles of the joints during the image sequence. Though we do not have any ground truth, the angles match with the real motion and the curves are relatively smooth indicating a reasonable stable algorithm. Another sequence shown in Figure 7.10 indicates that we are able to track the movement of head as well. In this experiment, the subject's head moves as the arms are moving. Extracting the skeletons for each frame allows us to evaluate the displacement of the neck joint. We are therefore able to approximate the angle of neck joint that defines the configuration of the head.

7.3 Experiment III–A special test

In this experiment, we rebuild a model using the parameters from other subject. We would like to visualize the errors caused. In the example, Lei's data are imposed onto Bodo's model. Therefore a new model is generated by a body cut different to the one shown in the previous section. We perform the tracking by overlaying the new 3D model using the joint angles extracted from Section 7.1 to the image. The new sequence gained is shown in Figure 7.10. It can be seen that tracking in this case has poor results in comparison to the experiment shown in Figure 7.7. Our algorithm for estimating joint positions and 3D reconstruction can be used to produce reasonably accurate human body models.

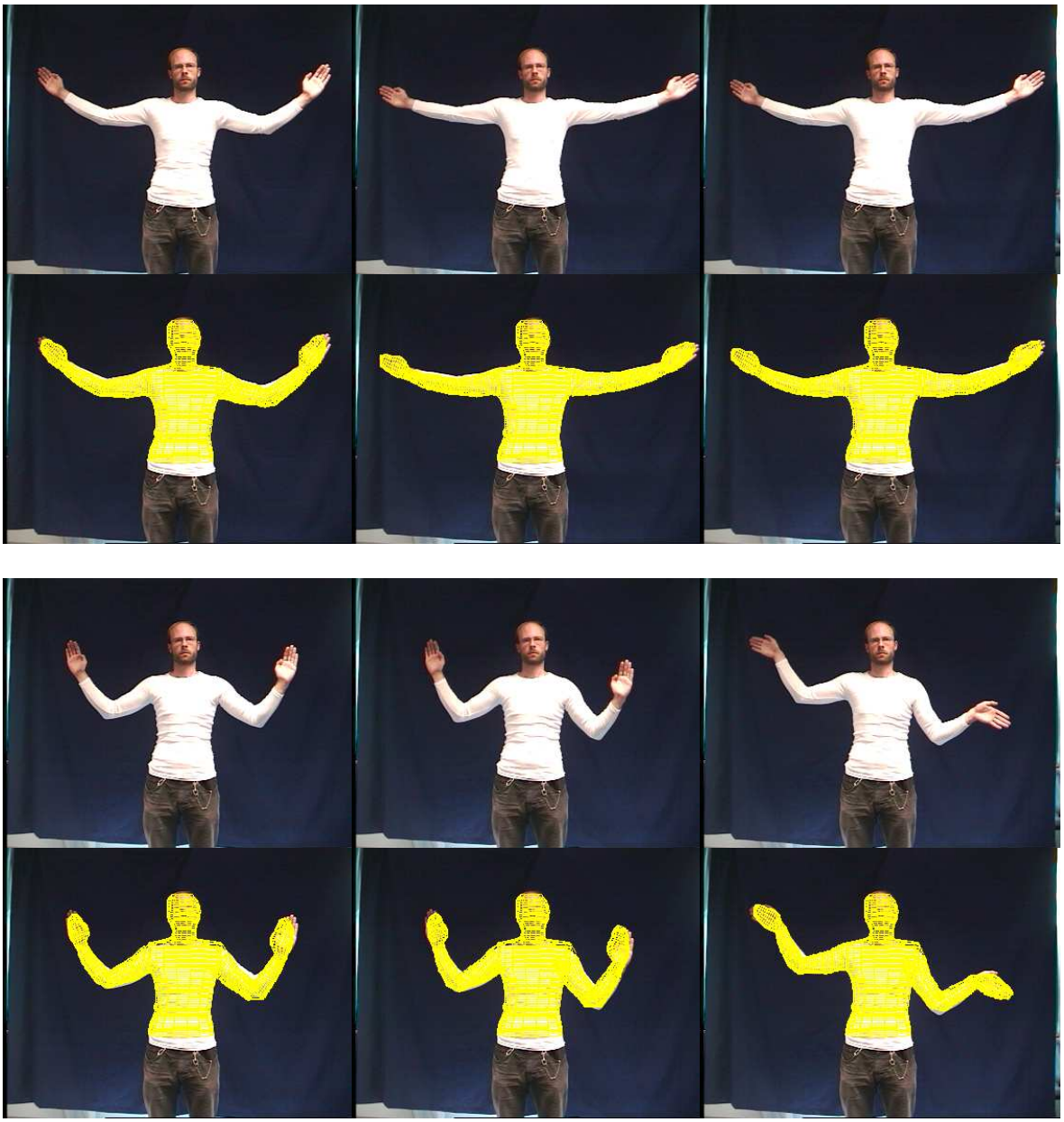
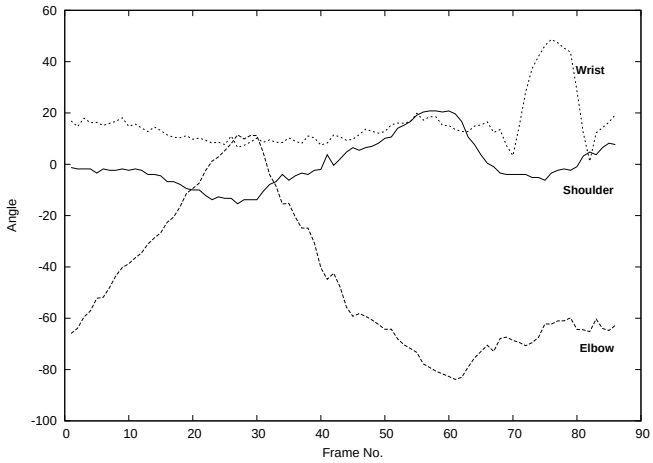
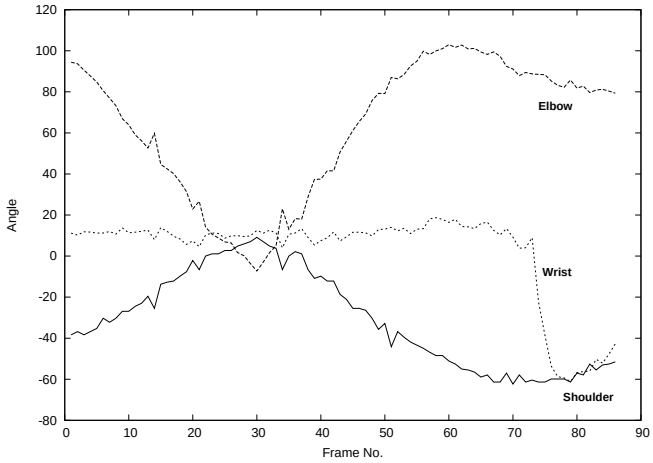


Figure 7.7: Results with 3DOF kinematic chains.



(a) Right



(b) Left

Figure 7.8: Joint angles of arms.

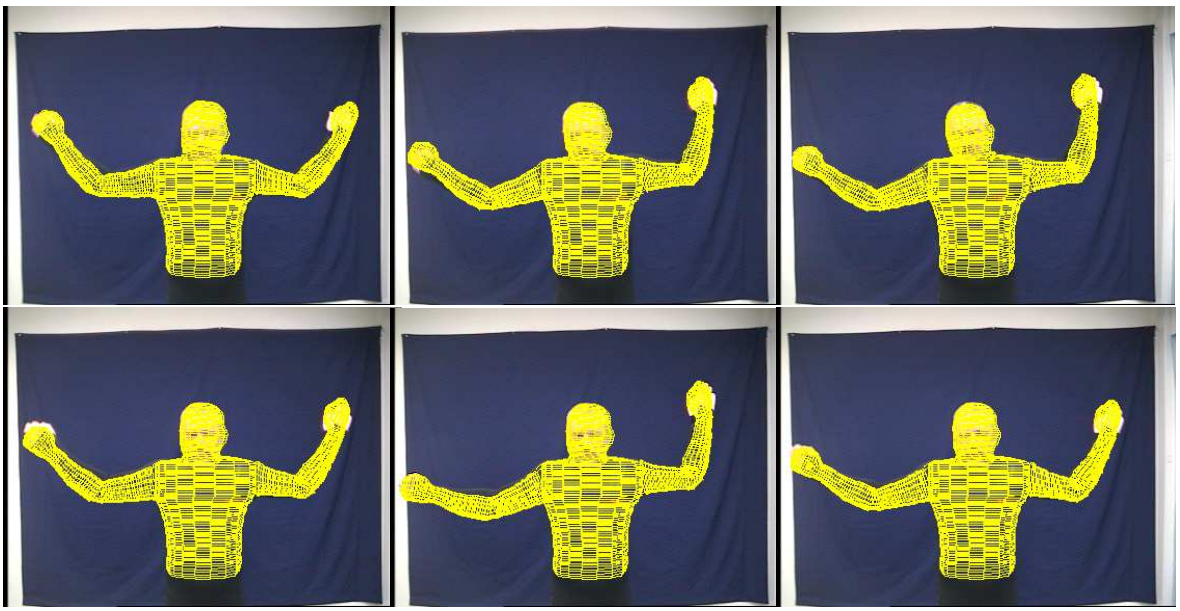


Figure 7.9: A tracking sequence for the movements of head and arms.

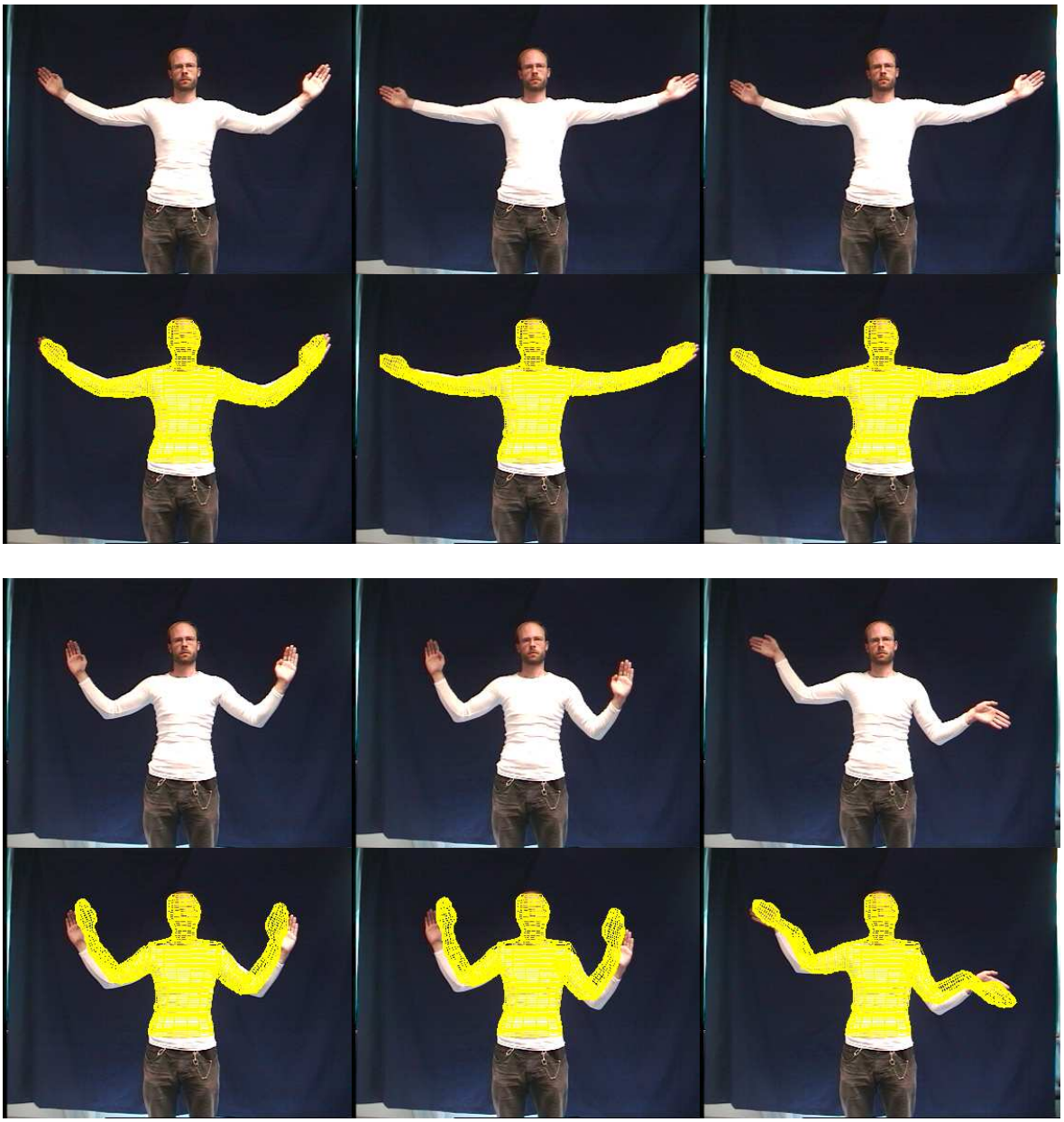


Figure 7.10: Tracking sequence with different object model.

Chapter 8

Conclusions and possible extensions

This thesis presents a system for the automatic generation of a 3D surface model for a human being using two calibrated views. We summarize the work in this thesis:

Firstly, this work utilizes many fundamental techniques in the fields of image processing and computer graphics. We exploited the basics of background subtraction (Section 4.2), morphologic operations (Section 4.3), skeletonization (Section 5.1), chain coding (Section 5.2), corner detection (Section 5.3), B-spline curves (Section 6.1.3), epipolar geometry (Section 6.1.2) and texture fusion (Section 6.5).

Secondly, we adapted fundamentals to suit our task. This thesis proposes a reconstruction algorithm for the human body. One core of the reconstruction algorithm is joint extraction. Another is the reconstruction algorithm which uses

a "shape-from-silhouettes" technique. Joints are recovered from the occluded silhouette by making use of a particular posture. This work also involved the development of a GUI for friendly interaction of threshold parameters.

Thirdly, we investigate the human motion models coupled with kinematic chains. We performed several experiments with different subjects. In one experiment, we drive the generic model to simulate the motions of a real person. In another experiment, the generic model is capable of tracking the movements of joints.

The results demonstrate feasibility of the reconstruction. However, further improvement will provide potentials for the rapid generation of more realistic models. First of all, for joint localization, it will be desirable to find a method that can provide optimal skeletons in a reasonable time. The desired skeleton should be centred, connected and without unexpected artifacts.

For 3D reconstruction, the current system uses B-spline curves obtained from two views to approximate the shape of the subject. It appears suitable for the reconstruction of a torso and arms, but it is not well enough for the reconstruction of a more complex shape such as a head. A necessary and possible solution is to use multiple cameras. This will significantly improve the accuracy. In terms of accuracy, according to what we have analyzed for the potential source of errors in Chapter 7, employing a monochromatic blue background and a concise, bigger calibration object is suggested for further improvement.

As mentioned previously, the purpose of this work is to provide a generic model for a wider human motion analysis project at the University of Auckland. This project deals with the human pose estimation problem and aims for natural looking human motion. Consequently an interesting extension of our study goes towards building "more realistic" human models. Namely, deformations of a human during its motions can be realistically modelled. The hypothesis is that the more accurate the human being is modelled, the more accurate pose results will be. The rigid body assumption for body segments in this work currently is not consistent with the rules of human motions. Actually, human motions are caused by the movement of muscles together with the bones. Apparently, the muscles are tense and the skin is morphed. In Chapter 7, the investigation of joint movements is performed for a very simple situation, and this is just a first step towards this topic. Further study is desirable for producing physically realistic forms of motions.

Bibliography

[1] Skeletonization/medial axis transform. Available at http://www.cee.hw.ac.uk/hipr/html/skeleton.html, last visited at February 26, 2005.

[2] Tsai camera calibration software. Available at http://almond.srv.cs.cmu.eds/afs/cs/usr/wgw/www/TsaiCode.html, last visited at February 26, 2005.

[3] R. Bartels, J. Beatty, and B. Barsky. *An Introduction to Splines for Use in Computer Graphics and Geometric Modeling*. Morgan Kaufman, San Mateo, 1987.

[4] G. Borgefores. Distance transform in arbitrary dimensions. *Computer Vision, Graphics and Image Processing*, 27:321–345, 1984.

[5] C. Bregler and J. Malik. Tracking people with twists and exponential maps. *IEEE Int. Conf. on Computer Vision*, I:8–15, 1998.

[6] J.B. Brzoska, B. Lesaffre, B. Coléou, K. Xu, and R.A. Pieritz. Computation of 3D curvatures on a wet snow sample. *The European Physical Journal*, 7:45–57, 1999.

[7] P.J. Burt and E.H. Andelson. A multiresolution spline with application to image mosaics. *ACM Trans. on Graphics*, II(4):217–236, 1983.

[8] T. Cham and J. Rehg. A multiple hypothesis approach to figure tracking. *IEEE Int. Conf. Computer Vision Pattern Recognition*, II:239–245, 1999.

[9] D. Chetverikov. A simple and efficient algorithm for detection of high curvature points in planar curves. In N. Petkov and M. A. Westenberg, editors, *Computer Analysis of Image and Patterns*, volume 2756 of *LNCS*, pages 746–753. Springer, 2003.

[10] C. de Boor. *A practical guide to splines*. Springer, New York, 2001.

[11] H. Freeman. Boundary encoding and processing. In *B. S. Lipkin and A. Rosenfeld, editors, Picture Processing and Psychopictorics*, pages 241–266. Academic Press: New York, 1970.

[12] R. Hartley and A. Zisserman. *Multiple view geometry in computer vision*. Cambridge University Press, New York, 2000.

[13] D. Hearn and M.P. Baker. *Computer graphics, C version*. Prentice Hall, Upper Saddle River, N.J., 1997.

[14] A. Hiltion, D. Beresford, T. Gentils, R. Smith, and W. Sun. Virtual people: capturing human models to populate virtual worlds. In *Proc. Computer Animation 1999*, pages 174–185, 1999.

[15] T. Horprasert, D. Harwood, and L.S. Davis. A statistical approach for real-time robust background subtraction and shadow detection. *Proc.*

IEEE ICCV'99 FRAME-RATE Workshop, Kerkyra, Greece. Available at http://www.vast.uccs.edu/∼tboult/FRAME/Horprasert/ Horprasert-FRAME99.pdf.

[16] I. Pohl. *C++ for C Programmers, Third Edition*. Addison-Wesley, Massachusetts, 1999.

[17] S. Ioffe and D. Forsyth. Human tracking with mixtures of trees. *IEEE Int. Conf. on Computer Vision*, I:690–695, 2001.

[18] M. Isard and J. MacCorrmic. BraMBLe: A Bayesian multiple-blob tracker. *IEEE Int. Conf. on Computer Vision*, II:34–41, 2001.

[19] I. Kakadiaris and D. Metaxas. Three-dimensional human body model acquisition from multiple views. *Int. J. of Computer Vision*, 30(3):191–218, 1998.

[20] G. Klette. A comparative discussion of distance transformations and simple deformations in digital image processing. *Machine Graphics & Vision*, 12(2):235–356, 2003.

[21] R. Klette and A. Rosenfeld. *Digital Geometry–Geometric Methods for Digital Picture Analysis*. Morgan Kaufmann, 2004.

[22] R. Klette, K. Schlüns, and A. Koschan. *Computer Vision - Three Dimensional Data from Images*. Springer, Singapore, 1998.

[23] W. Lee, J. Gu, and N. Magnenat-Thalmann. Generating animatable 3D virtual humans from photographs. In *EUROGRAPHICS 2000*, volume 19, pages 1–10, 2000.

[24] R.L. Memmler and D.L. Wood. *Structure and Function of the human body.* J.B. Lippincott Company, 1987.

[25] I. Mikic, M. Trivedi, E. Hunter, and P. Cosman. Human body model acquisition and tracking using voxel data. *Int. J. of Computer Vision*, 53(3):199–233, 2003.

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

[27] J. F. Oliveira, D. Zhang, B. Spanlang, and B. F. Buxton. Animating scanned human models. In *Journal of Winter School of Computer Graphics*, volume 11, pages 362–369, 2003.

[28] R. House. *Beginning with C: An Introduction to Professional Programming.* Nelson Thomson Learning, South Melbourne, 1994.

[29] D. Reiners, G. Voβ, and J. Behr. OpenSG - Basic Concepts. In *Proc. OpenSG Symposium*, Darmstadt, Germany, January 2002.

[30] C. Rikk, B. Gavin, and M. Chris. *Virtual Reality Modeling Language,(VRML97).* ISO/IEC 14772-1:1997, The VRML Consortium Incorporated, 1997.

[31] B. Rosenhahn. *Pose Estimation Revisited*. Technical report 0308, Inst. f. Informatik u. Prakt. Math. der Christian-Albrechts-Universität zu Kiel, 2003. Available at http://www.ks.informatik.uni-kiel.de/.

[32] B. Rosenhahn and R. Klette. Geometric algebra for pose estimation and surface morphing in human motion estimation. In R. Klette and J. Zunic, editors, *Tenth Int. Workshop Combinatorial Image Analysis, Auckland, New Zealand*, volume 3322 of *LNCS*, pages 583–596. Springer-Verlag, Berlin Heidelberg, 2004.

[33] B. Rosenhahn, R. Klette, and G. Sommer. Silhouette based human motion estimation. In C.E. Rasmussen, H.H. Bülthoff, M.A. Giese, and B. Schölkopf, editors, *Pattern Recognition 2004, 26th DAGM-symposium*, volume 3175 of *LNCS*, pages 294–301. Springer-Verlag, Berlin Heidelberg, 2004.

[34] S. Logan. *Gtk+ programming in C*. Prentice Hall PTR, Upper Saddle River, N.J., c2002.

[35] M. Sramek. Distance fields in visualization and graphics. In Ivan Viola and Thomas Theussl, editors, *Proc. of CESCG'2002*, pages 13–16, Slovak Republic, April 2002. Budmerice.

[36] R.J. Starck. *Human modelling from multiple views*. PhD thesis, University of Surrey, 2003.

[37] R.Y. Tsai. A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses. *IEEE J.Robotics and Automation*, pages 323–344, August 1997.

[38] M. Woo. *OpenGL programming guide: the official guide to learning OpenGL, version 1.2*. Addison-Wesley, 1999.

[39] C. S. Zhao and Roger Mohr. Global three-dimensional surface reconstruction from occluding contours. *Computer Vision and Image Understanding*, 64(I):62–96, 1996.