Visual Analysis of High DOF Articulated Objects with Application to Hand Tracking

James M. Rehg

April 1995

CMU-CS-95-138

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Submitted to the Dept. of Electrical and Computer Engineering in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Committee: Dr. Takeo Kanade, Chair
Dr. José Moura
Dr. Katsushi Ikeuchi, SCS
Dr. Andrew Witkin, SCS

This research was conducted at the Robotics Institute, Carnegie Mellon University, and partially supported by the NASA George Marshall Space Flight Center (GMSFC), Huntsville, Alabama 35812 through the Graduate Student Researchers Program (GSRP), Grant No. NGT-50559. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of NASA or the U.S. government.

Keywords: Model-Based Visual Tracking, Articulated and Nonrigid Object Motion, Occlusions, Human Motion Analysis, Human-Computer Interaction, Gesture Recognition

Abstract

Measurement of human hand and body motion is an important task for applications ranging from athletic performance analysis to advanced user-interfaces. Commercial human motion sensors are invasive, requiring the user to wear gloves or targets. This thesis addresses noninvasive real-time 3D tracking of human motion using sequences of ordinary video images. In contrast to other sensors, video cameras are passive and inobtrusive, and can easily be added to existing work environments. Other computer vision systems have demonstrated real-time tracking of a single rigid object in six degrees-of-freedom (DOFs). Articulated objects like the hand present three challenges to existing rigid-body tracking algorithms: a large number of DOFs (27 for the hand), nonlinear kinematic constraints, and complex self-occlusion effects. This thesis presents a novel tracking framework for articulated objects that uses explicit kinematic models to overcome these obstacles.

Kinematic models play two main roles in this work: they provide geometric constraints on image features and predict self-occlusions. A kinematic model for hand tracking gives the 3D positions of the fingers as a function of the hand state, which consists of the pose of the palm and the finger joint angles. Image features for the hand consist of lines and points which are obtained by projecting finger phalanges and tips into the image plane. The kinematic model provides a geometric constraint on the image plane positions of hand features as a function of the hand state. Tracking proceeds by registering the projection of the hand model with measured image features at a high frame rate.

Self occclusions are modeled by arranging the image features in overlapping layers, ordered by their visibility to the camera. The layered representation is generated automatically by the kinematic model and used to constrain registration. This framework was implemented in a hand tracking system called *DigitEyes* and tested in two sets of experiments. First, a hand was tracked in real-time using two cameras and a 27 DOF model, and using a single camera in a 3D mouse user-interface trial. Second, the occlusion handling framework was tested off-line on a motion sequence with significant self-occlusion. These results illustrate the effectiveness of explicit kinematic models in 3D tracking and analysis of self-occluding motion.

Dedicated to Jim and Marci

Acknowledgments

I wish to thank my advisor, Dr. Takeo Kanade, for his support and technical advice during my graduate studies. My years in the Vision and Autonomous Systems Center (VASC) were extremely enjoyable, and I'm grateful to have had the opportunity to complete my thesis in such a marvelous environment.

I was fortunate to interact closely with my committee members, Dr. Katsushi Ikeuchi, Dr. José Moura, and Dr. Andrew Witkin at different stages of my thesis work. I am grateful to them for their helpful comments and insight. I am especially grateful to Andy for introducing me to deformable models and for making the resources of the graphics lab available for my use.

I want to thank Dr. Ingemar Cox for an enjoyable and productive internship at the NEC Research Institute in Princeton, NJ during the summer of 1991.

I would like to thank my parents, Jim and Marci, for the encouragement and effort that made this dissertation possible. I want to thank my wife, Dorothy, for her constant support and valuable technical insight.

I am grateful to Alberto Elfes for introducing me to computer vision and robotics, and to Radu Jasinschi for his ideas and friendship. I have enjoyed many interesting conversations with members of the VASC group and ECE department over the years. In particular, I want to thank Sandra Ramos-Thuel, Omead Amidi, Heung-Yeung Shum, Luc Robert, and Fabio Cozman for their time.

Thanks to Omead Amidi and Yuji Mesaki for their help with the hardware environment, and to Dr. David Sturman for kindly providing the postscript file for Fig. 2.4.

Jim Rehg
April 7, 1995

Contents

List of Figures

Chapter 1

Introduction

Tracking the motion of hands and bodies in three dimensions (3D) is an important task for applications in computer graphics, athletic performance analysis, and user-interfaces. Commercial human motion sensors are invasive, requiring the user to wear gloves or targets [74, 37]. For example, current motion capture systems work by recording the 3D trajectories of magnetic trackers or optical targets attached to the user's hands and limbs. These trajectories are used in computer graphics applications to imbue animated characters with realistic motion [54]. In other examples, various glove-based sensors for palm and finger motion have been used to interpret sign language [17] and control 3D CAD models [9]. In all of these cases, the usefulness and convenience of the sensor is limited by the need to wear clumsy, bulky devices, often tethered to an external computer.

This thesis addresses the noninvasive real-time tracking of human motion using sequences of ordinary video images. In contrast to other sensors, video cameras are passive and inobtrusive, and can easily be added to existing work environments. Other computer vision systems have demonstrated real-time tracking of a single rigid object in six degrees of freedom (DOFs) [20, 35]. Articulated objects like human figures and hands present three difficulties for these existing algorithms: the large number of DOFs required to describe their motion, nonlinearities in the mapping from the DOFs to the image motion, and the presence of complex occlusion effects, when one part of the body blocks the camera's view of another. This thesis explores the use of explicit kinematic models in a local tracking approach to overcome these difficulties. It describes a tracking framework for general articulated objects and presents experimental results for 3D hand tracking from natural image sequences.

1.1 Tracking with Kinematic Models

The kinematics of an articulated object provide the most fundamental constraint on its motion. Chapter 2 presents a general model-based framework for tracking with kinematic constraints; this section outlines its application to hand tracking. In the case of the hand, motion of the fingers and palm in 3D is constrained by the skeleton. The relationship between these skeletal constraints and a hand image is illustrated in Fig. 1.1(a). The black overlay shows the projection of a 3D kinematic hand model, illustrated in Fig. 1.1(b), into the image plane. The finger phalanges (links) are drawn as a set of black "T" shapes, connected together at the knuckles. Each phalange is represented by a cylinder, and each T shows the radius and axis of the cylinder's projection into the image. When the model has been *registered* to the image correctly, as in the figure, the projected cylinders are aligned with the fingers.

Local tracking consists of a series of registration problems in which the configuration of the 3D hand model is adjusted so that its projection is aligned with the current image. At the start of tracking, the image and the model are registered. For each subsequent image in the motion sequence, small corrections are made to the state of the hand that minimize the registration error. The state vector for the hand contains the pose of the palm and the finger joint angles. The registration error is described by a *residual* *function*, which is minimized by the state correction in each frame. This thesis explores two types of residual functions: *Sum of Squared Differences* (SSD) and *geometric feature* residuals. The SSD residual measures the intensity differences between the image and a template model for each body in the articulated object. A collection of templates can represent a wide variety of link shapes. Furthermore, since templates explicitly describe the region each link occupies in the image, they are useful in tracking self-occluding objects, as Chpt. 3 describes.

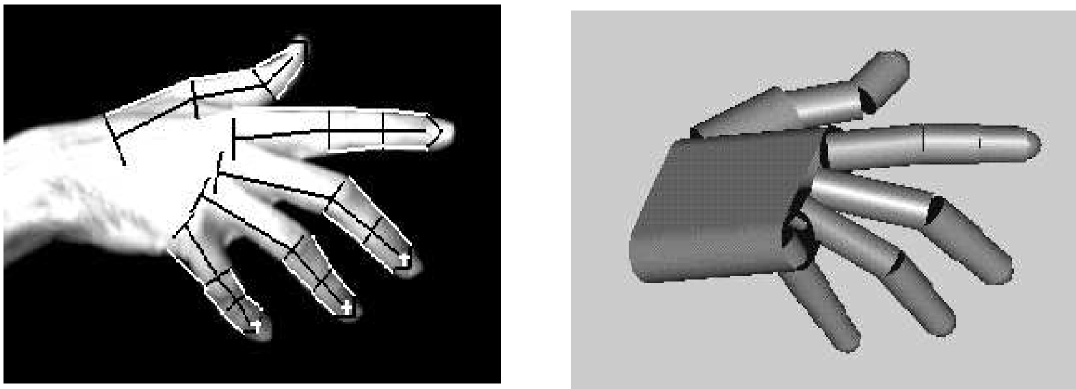


Figure 1.1: (a) Hand image with projection of 3D kinematic model overlaid in black and detected line and point features shown in white, and (b) 3D view of the hand model which is registered to the image in (a).

Images of hands and bodies can also be described by a collection of line and point features, as the "image skeleton" shown in Fig. 1.1 illustrates. In this example, pairs of lines and point features, drawn in white, mark the edges of the finger phalanges and the finger tip centers. The geometric feature residual used in this case measures the distance between the projected 3D model (the black overlay) and the measured line and point features (the white overlay.) This feature residual approximates the SSD residual for roughly cylindrical objects like finger phalanges and limbs. A simple, efficient algorithm for detecting the geometric features is described in Chpt. 4. It forms the basis for the real-time tracking experiments described there.

The residual error for each image is minimized using a gradient-based approach. The kinematic Jacobian for the articulated object is a key component of the residual gradient. It plays a role in articulated object tracking that is similar to its use in robot control. This duality is exploited in Sec. 2.5.4 in the study of kinematic singularities, which arise when certain states have no instantaneous effect on the image features. The geometric feature residual can be used to identify these singular cases, because it provides a closed-form expression for registration error as a function of the state. A standard technique for stabilizing rigid body trackers is shown to be effective in dealing with these singularities.

1.2 Tracking Self-Occluding Objects with Layered Templates

When the motion of an object like the hand is sampled at a high frame rate, the occlusion relations between its bodies hardly ever change. When they do, the change can be predicted from the kinematic model. This observation is exploited in Chpt. 3 to remove the estimation of occlusion from the tracking problem, leaving only the registration of partly occluded templates. The result is a layered representation of self-occlusion that is dynamically updated by the kinematic model. A set of rules for hand template ordering are developed through an analysis of planar kinematic chains.

The registration framework from Chpt. 2 is extended to the overlapping template case through the introduction of window functions that mask off the contributions of occluded templates. The presence of window functions complicates the derivation of the residual Jacobian. However, the structure of the layered templates can be expressed in a window tree, and analyzed to

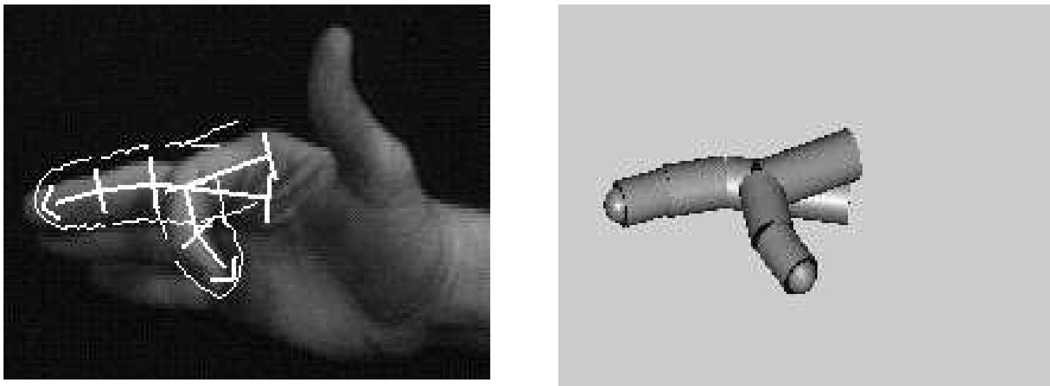


Figure 1.2: Two finger self-occlusion experiment from Chpt. 4. (a) Hand image with model overlays, (b) model state estimated from image (a).

obtain a simple procedure for Jacobian computation.

The existence properties of the local occlusion invariant that makes a layered representation possible is analyzed in Sec. 3.2 for arbitrary systems of rigid bodies. This analysis introduces two new occlusion representations, the occlusion graph and occlusion meta-graph, and describes their relationship to hand tracking. The resulting framework is an extension of the classical template registration paradigm to the case of self-occluding motion. Figure 1.2 illustrates experimental tracking results for self-occluding motion.

1.3 Hand Tracking Experiments

A real-time tracking system called *DigitEyes* was implemented and used to test the kinematic tracking framework from Chpt. 2 on real images of unmarked hands. In this system, a single board image processor was used to eliminate the image transfer bottleneck, and line and point features were used to simplify the image processing task. Chapter 4 presents real-time tracking results using a 27 DOF hand model.

The *DigitEyes* system was also applied to a 3D mouse user-interface ap-

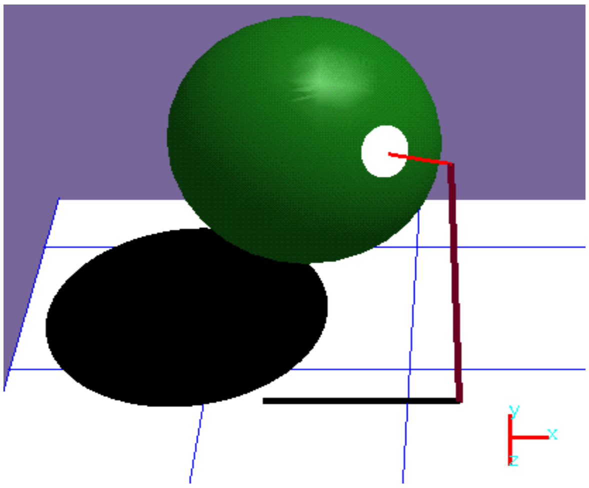


Figure 1.3: A sample graphical environment for a 3D mouse. The 3D cursor is at the tip of the "mouse pole", which sits atop the ground plane.

plication, to test its practical usefulness as an input device. The resulting non-invasive interface gives the user control over a 3D cursor in a graphical environment, using images from a single calibrated camera. Figure 1.3 shows sample output from the interface.

In the final experiment, described in Sec. 4.4, an off-line version of the *DigitEyes* system was used to test the self-occlusion framework of Chpt. 3. A 75 frame image sequence of two fingers undergoing significant self-occlusion was successfully tracked.

1.4 Contributions

This dissertation makes five main contributions:

1. Analysis of the application of kinematic models to visual tracking of articulated objects, addressing Jacobian singularities and sensitivity, as well as techniques for efficient Jacobian computation.

2. The first experimental demonstration of real-time tracking (at speeds of up to 10 Hz) of a high-DOF articulated object (a 27 DOF hand model), using both monocular and stereo image sequences of unadorned, unmarked hands [46, 48].

3. Application of the *DigitEyes* sensor to the 3D mouse user-interface problem, demonstrating the feasibility of 3D human sensing at reasonable accuracy levels using currently-available hardware [47].

4. The identification of a local ordering invariant for self-occluding objects, an analysis of its existence conditions, and the design of a tracking algorithm for self-occluding motion [49].

5. The first experimental demonstration of nontrivial 3D articulated object tracking in the presence of self-occlusion [50].

These results extend previous techniques in computer vision for rigid body tracking and demonstrate the feasibility of vision-based 3D human motion sensing.

Chapter 2

Tracking with Kinematic Models

The motion of an articulated object like the hand is determined by its skeleton. A camera can only observe the skeleton indirectly, however, through its effect on the skin. Skin and clothing deform during hand and body motion, producing *nonrigid* effects in an image sequence. The magnitude of these nonrigid components is small, however, compared to the effects of rigid, articulated body motion. This dissertation treats nonrigidity as unmodeled noise in the measurements of rigid, articulated objects. Experimental hand tracking results, presented in Chpt. 4, demonstrate the efficacy of this assumption. They are corroborated by experimental results for body tracking [23, 29], which make a similar assumption.

2.1 The Role of Kinematics in Visual Tracking

The use of kinematic models is vital for 3D tracking. As an example, consider the problem of estimating the pose of the first finger in the image of Fig. 1.2. The true finger pose and its projection into the image are shown with a line drawing in Fig. 2.1 (a). The line drawing is a useful abstraction of the geometric information contained in the image. For simplicity, assume that

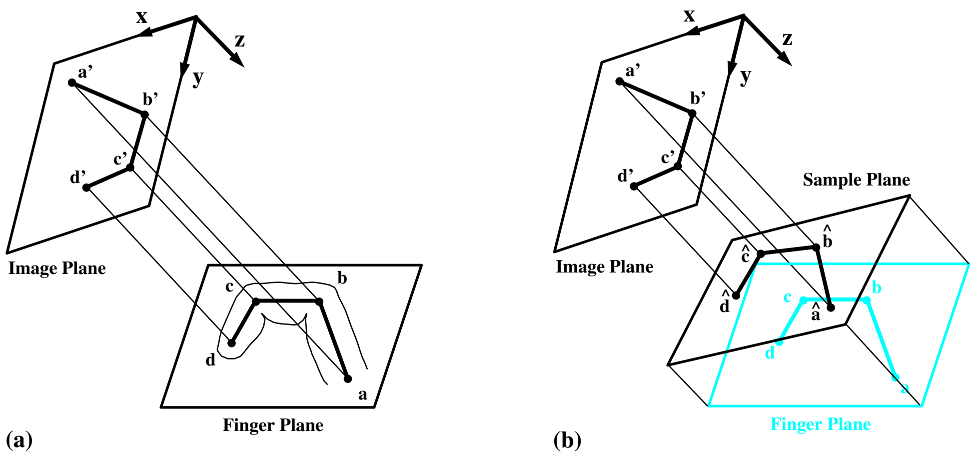


Figure 2.1: (a) Stick drawing of image features and model for the first finger in Fig. 1.2 and (b) two models with different kinematics that produce the same image.

the finger lies in a plane in space, and the camera model is orthographic.[1] From the geometry of figure (a), it is clearly impossible to determine the 3D pose of links $\overline{\mathbf{ab}}$, $\overline{\mathbf{bc}}$, and $\overline{\mathbf{cd}}$ from the image points $\{\mathbf{a}', \mathbf{b}', \mathbf{c}', \mathbf{d}'\}$ without a kinematic model. In fact, for any sample plane in 3D there exists a finger configuration that produces the given image. Fig. 2.1 (b) gives one example. A unique solution is possible only when the link lengths are known. Only in this case is the orientation of a link along the camera axis determined by its projection in the image.

The example in Fig. 2.1 also illustrates the difference between errors in registration and errors in 3D pose (state) estimates. Registration refers to the alignment between an image and the image plane projection of a 3D model. As Fig. 2.1 (b) illustrates, it is easy to achieve zero registration error without a kinematic model for any sample plane position, by aligning the projections of $\{\hat{\mathbf{a}}, \hat{\mathbf{b}}, \hat{\mathbf{c}}, \hat{\mathbf{d}}\}$ with $\{\mathbf{a}', \mathbf{b}', \mathbf{c}', \mathbf{d}'\}$. The corresponding pose error can be arbitrarily large, however, as the sample plane rotates away from the

[1]In orthographic projection, all rays from the scene to the camera are parallel.

true finger plane. Now suppose that a kinematic model is available, as in Fig. 2.1 (a), but that the model itself has some error. When the model errors are small, the pose error will also be small. The registration error will be nonzero in this case, as no configuration of the incorrect model will match the image exactly. A kinematic model makes it possible to extrapolate image registration into three dimensions. The quality of this extrapolation depends on the accuracy of the model.

There are two other sources of 3D pose information besides a kinematic model: shading and stereo. The shading in an image of the hand varies with its spatial orientation. These intensity changes carry information about the 3D pose of the palm and fingers. Shading cues are an important component of human perception, but exploiting them in a vision algorithm is known to be extremely challenging. In hand images, shadows and lighting variations make it difficult to interpret intensity changes correctly. As a result, it is unlikely that the accuracy of pose estimation due to shading alone would exceed that available from the kinematics.

Stereo is the second alternative approach to pose estimation, for links that are visible in two or more camera images. In stereo, triangulation with corresponding pairs of image points, such as $\{\mathbf{a}_1, \mathbf{a}_2\}$ in Fig. 2.2, produce 3D estimates of $\{\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}\}$. Stereo is inadequate by itself, however, when a link is *not* visible in both views due to occlusion, a common occurrence in practice. But suppose that a kinematic model is available in addition to stereo. In this case, localizing three of the points by stereo determines the plane of the finger, and the position of the fourth point can be determined from a single view. This illustrates another key feature of the kinematic model: it captures redundancy in the measurements, which leads to an overdetermined estimation problem.

Kinematic models play three main roles in tracking. First, they parameterize the DOFs on the object, and provide a mathematical representation for the output of the tracking algorithm— a trajectory in state space. Sec-

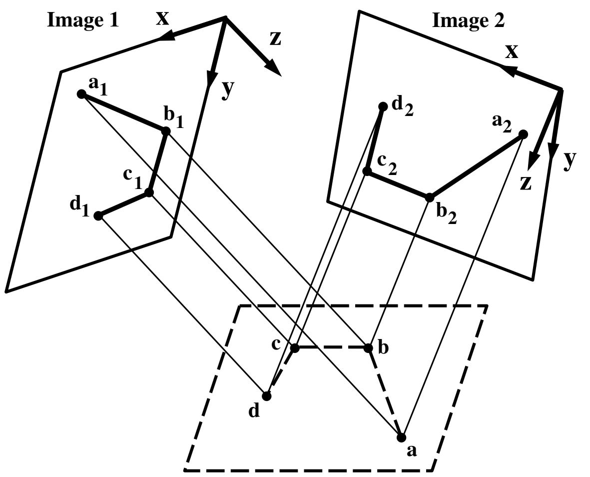


Figure 2.2: Use of an additional stereo image to reconstruct the 3D pose of the finger depicted in Fig. 2.1 (a).

ond, they express constraints on the motion of the rigid bodies making up the articulated object. These constraints lead to an over-determined estimation problem in the image measurements, which is desirable for good noise properties. Third, the kinematics also constrain the possible occlusions between the rigid bodies. Kinematic analysis plays an important role in the development of tracking algorithms for self-occluding motion in Chpt. 3.

This chapter begins with a brief description of the mathematical foundations of kinematic modeling. These representations originated in the robot manipulation literature, but have been adapted slightly to meet the requirements of visual tracking. This presentation is significantly more complete than any that has appeared in the visual tracking literature to date. The application of kinematic modeling techniques is illustrated for the hand. The resulting kinematic hand model is employed throughout this thesis. Calibration of kinematic and camera models are described, along with the effect of their errors.

The second half of the chapter describes the incorporation of kinematic models into tracking algorithms. The kinematics provide a forward model for the object, generating predicted images as a function of the estimated state. This chapter addresses the geometric component of the forward model, and ignores the effects of occlusion. In this chapter, every rigid body in the model is assumed to be completely visible to the camera. The forward model interacts with the input image through a residual error measure. Minimizing the residual through gradient-based algorithms brings the projection of the model into alignment with the input images.

The image intensities generated by the object determine the measurements that are available for tracking, and therefore the form of the residual error. Two residual errors are examined here. The first is a general template-based residual that can be applied to arbitrary articulated objects. The second residual is derived from geometric line and point features that approximate the template residual in the case of objects, like hands and bodies, made up of cylindrical links. The feature residual is a closed form expression that is amenable to analysis and real-time implementation on modest computing hardware.

2.2 Kinematic Modeling of Articulated Objects

I employ standard kinematic modeling techniques from robotics [59] to represent skeletal constraints for tracking. These models have been used for decades to solve robot control and path planning problems. They have good theoretical properties and support efficient on-line algorithms. Denavit-Hartenberg notation, for example, provides a standard description for kinematic chains like the finger. This notation has already been employed in hand models for computer graphics [52], but has not been used explicitly in hand or body tracking to date. One of the goals of this thesis is to explore the connections between articulated tracking and robot control more carefully than previous authors. For example, historical robot control issues like kinematic singularities have close parallels in hand tracking, as I will describe later in Sec. 2.5.3. Developing these parallels makes techniques from the robotics literature available for articulated tracking analysis.

All previous work on 3D human tracking employed some form of kinematic model. The two earliest systems, by O'Rourke and Badler [42] and Hogg [23], predated the widespread popularization of robot kinematic models by Paul [43]. They employed their own customized kinematic representations. The use of robot kinematic models for human body tracking was first proposed by Yamamoto and Koshikawa in [72]. This work did not present a detailed modeling framework, however, but relied on a separate software package for kinematic computations.

The kinematic models described in this section form the basis for all of the tracking algorithms in this thesis. Mathematical representations of object kinematics are presented here in detail. Following this description, a kinematic hand model is derived from an anatomical study. This illustrates both the usefulness of the modeling framework and the specific concerns of kinematic modeling for visual tracking. Models must be calibrated before they can be used, and the calibration process, along with the effects of calibration errors, is described at the end of the section.

2.2.1 Coordinate Frames and Transformations

An articulated object is made up of rigid bodies, called links, connected by joints. Each link has its own coordinate frame in the kinematic model, and pairs of link frames are connected by coordinate transformations. A coordinate transform from frame i to frame j, written $\mathbf{T}_i^j$, is specified by a rotation matrix $\mathbf{R}_i^j$ and translation vector $\mathbf{d}_i^j$, arranged in a 4x4 homogeneous

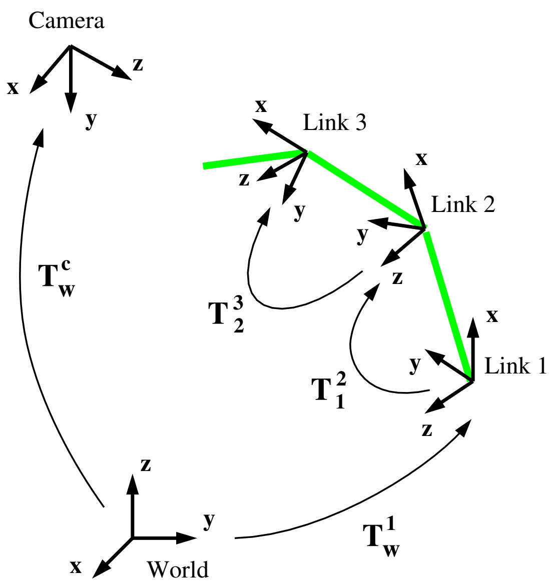


Figure 2.3: Illustration of the basic coordinate frames in the kinematic model.

transformation matrix:[2]

$$\mathbf{T}_i^j = \begin{bmatrix} \mathbf{R}_i^j & \mathbf{d}_i^j \\ \mathbf{0}^T & 1 \end{bmatrix} . \tag{2.1}$$

The transform satisfies the relation $\mathbf{p}_i = \mathbf{T}_i^j \mathbf{p}_j$, where $\mathbf{p}_i$ and $\mathbf{p}_j$ denote the coordinates with respect to frames i and j of the world point $\mathbf{p}$. Each $\mathbf{p}_i$ is a 4x1 vector with components $[x_i \, y_i \, z_i \, 1]$. The 3D *configuration* of an object like the hand is determined by the position of each of its links with respect to a world coordinate frame. One or more cameras, also positioned with respect to the world frame, provide images from which the hand configuration is estimated. The relationship between these frames is illustrated in Fig. 2.3.

In classical kinematic analysis, the shapes of the links are irrelevant. All that matters are the link frames and the transformations between them. For tracking, however, the visual appearance of the link must also be included

[2]All vectors are column vectors. Vectors and matrices are in bold face, scalars in italics. A superscript T denotes a matrix or vector transpose.

in the model. It is convenient to add an additional *shape coordinate frame* to each link, which positions the visible geometry relative to the link frame. Having an additional frame is useful, as the coordinate frame choice which is best for the kinematic description may not be the best for shape modeling. The choice of shape frame, like the choice of link coordinates, depends on the application. The specific choices made in hand modeling are described in the next section.

A series of links connected by joints forms a kinematic chain. The position of any link in the chain can be obtained by multiplying transformation matrices. For example, the position of the link 3 frame in Fig. 2.3 with respect to the camera is given by $\mathbf{T}_c^3 = (\mathbf{T}_w^c)^{-1}\mathbf{T}_w^1\mathbf{T}_1^2\mathbf{T}_2^3 = \mathbf{T}_c^w\mathbf{T}_w^1\mathbf{T}_1^2\mathbf{T}_2^3$. Joints are modeled by *parameterized* coordinate transformations, $\mathbf{T}_i^j(\mathbf{v})$, called joint transforms. A joint transform has the form of Eqn. 2.1, but is a matrix function of a vector $\mathbf{v}$ of kinematic parameters, such as joint angles and link lengths.

Link frames and joint transforms make up the topological part of the kinematic model— they specify the number of rigid bodies and their interconnections. The topological part of a human kinematic model comes directly from basic anatomy. A finger, for example, consists of three phalanges (rigid links) connected in series by the two knuckle joints. Kinematic parameters for the joint transforms make up the parametric part of the kinematic model. They consist of the object's DOFs and any fixed model parameters.

The two types of joint transforms used in this thesis are spatial transforms and Denavit-Hartenberg transforms. Spatial transforms model the six DOFs between two link frames that are not in physical contact. It is used in the hand model to position the palm relative to the world frame. I use quaternions to represent the rotational part of the spatial transform. Quaternions encode the axis-angle representation of rotation with four parameters.[3] Three

[3]See [38] for general information about quaternions and spatial transforms.

parameter representations, like Euler angles, have singularities at which their Jacobian loses rank, making tracking more difficult. These singularities are not a natural result of the kinematics, but an artifact of the parameterization. Since an object like the hand may achieve an arbitrary pose with respect to a given camera, it is difficult to ensure that singular configurations are avoided. Quaternions are the minimal singularity-free representation of the rotation group [60]. They have a long history of use in satellite control [69], and more recently in vision [21] and computer graphics [58]. The resulting spatial transform has seven parameters.

Since the four quaternion variables are not a minimal description of rotation, they are subject to a unit norm constraint that reduces their DOFs to three. Specifically, a quaternion vector $\mathbf{Q}$ must satisfy $\mathbf{Q}^T\mathbf{Q} = 1$ at all times. As a result, quaternion-based tracking is technically a constrained estimation problem. I follow the practice described in [22] of expressing the quaternion rotation matrix in a form that includes the normalization. The resulting quaternion estimate is re-normalized periodically to prevent the accumulation of numerical errors.

When two links are physically connected by a joint, the coordinate transformation between them must have fewer than six DOFs. The Denavit-Hartenberg (DH) notation [13] provides a consistent parameterization in this case. Each DH transform is composed of four basic transformations:

$$\mathbf{T}_i^{i+1}(\theta_i, d_i, a_i, \alpha_i) = \mathbf{Rot}_z(\theta_i)\mathbf{Trans}_z(d_i)\mathbf{Trans}_x(a_i)\mathbf{Rot}_x(\alpha_i) \ , \qquad (2.2)$$

where $\mathbf{Rot}(\cdot)$ represents a rotation about a given axis, and $\mathbf{Tran}(\cdot)$ a translation along it. See [59], Fig. 3-4, for an illustration of the general DH transform, which is widely used in robotics. The parameters $\{\theta_i, d_i, a_i, \alpha_i\}$, along with the choice of the link frame, can be used to model all lower pair joints of interest. The DH parameters can be divided into two groups: state variables, which represent the DOFs of the object at the joint, and fixed parameters, which describe the object's geometry and are unchanged by its

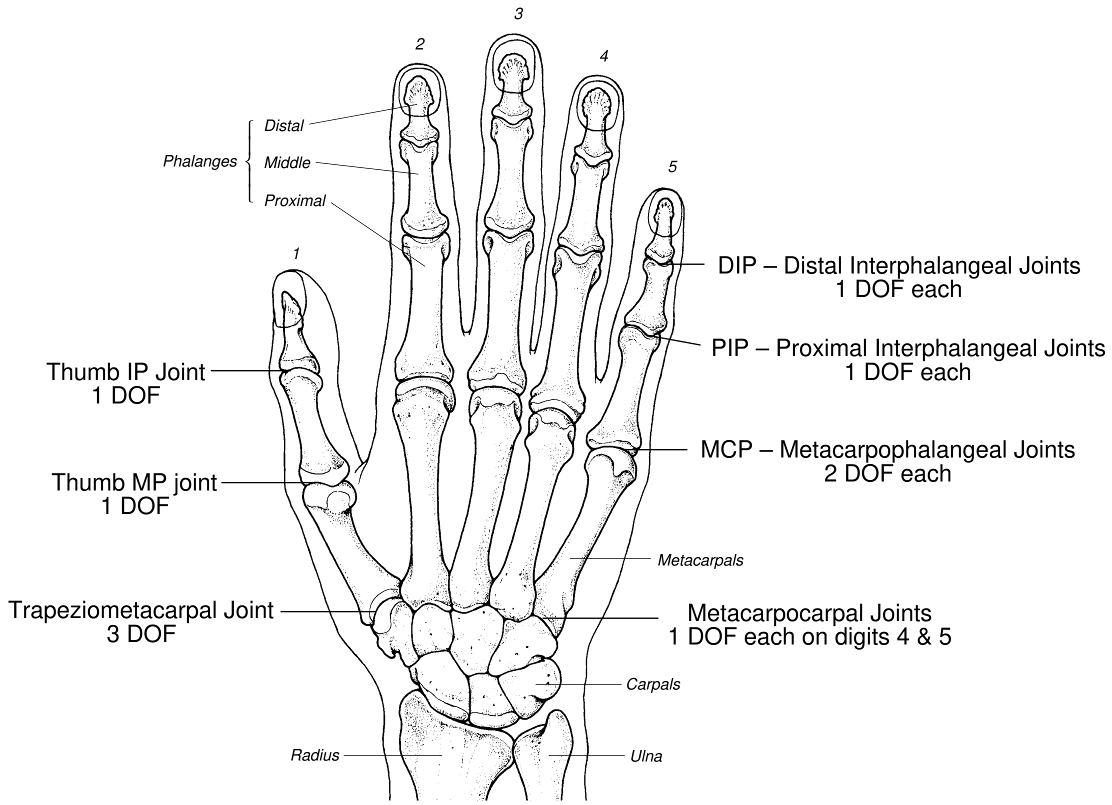


Figure 2.4: Hand skeleton and joints. This is Fig. 1 from [61], used with permission.

motion.

The kinematic representation described above can be applied to a wide variety of objects, from humans to industrial robots. In the next section, it is used to develop a hand kinematic model, which is employed in all of the tracking experiments in this thesis.

2.2.2 A Kinematic Hand Model

Kinematic models for visual tracking need only describe motion which a camera can measure. As a result, they can be considerably simpler than those

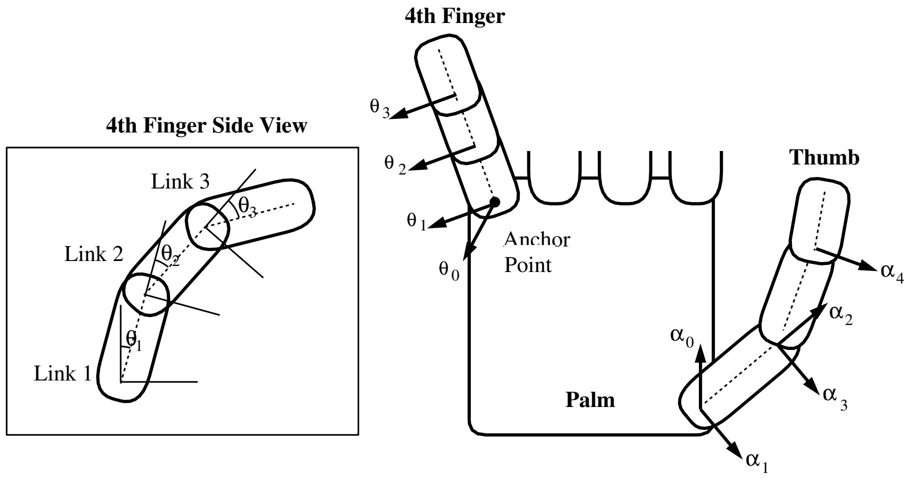


Figure 2.5: Kinematic models, illustrated for fourth finger and thumb. The arrows illustrate the joint axes for each link in the chain.

developed by the biomechanics community [3], which has a long history of skeletal kinematic modeling and analysis. The palm, illustrated in Fig. 2.4, provides a good example. Its four major metacarpal bones are completely invisible during tracking. Therefore, the kinematic hand model employed in this dissertation treats the palm as a single rigid body. The position of the palm in the world frame is given by a spatial transform of seven state variables: four quaternion states and three translation states. These spatial parameters are the first elements in the state vector, $\mathbf{q}$, for the hand.

Attached to the palm are five kinematic chains of three links each, comprising the four fingers and thumb. These chains and their kinematics are illustrated in Fig. 2.5. Each of the four fingers are modeled as planar mechanisms with four degrees of freedom (DOF). The abduction DOF moves the plane of the finger relative to the palm, while the remaining 3 DOFs determine the finger's configuration within the plane. Each finger has an *anchor point*, which is the position of the center of its metacarpophalangeal (MCP) joint in the frame of the palm.

Finger chains are built up from revolute joints, which constrain two links to a single rotational DOF around the joint axis. Figures 2.3 and 2.5 illustrate the link frame assignments for the revolute joint model. The frame for link i is chosen so that θ_i (in Eqn. 2.2) is the revolute joint angle, and the negative x axis passes through the joint center of link $i-1$. With this choice of coordinates, the DH kinematic parameters d_i and α_i are zero, and a_i equals the link length. Making these substitutions in Eqn. 2.2 gives the revolute joint transform

$$\mathbf{T}_i^{i+1} = \mathbf{Rot}_z(\theta_i)\mathbf{Trans}_x(L_i) \ , \tag{2.3}$$

where L_i is the length of the ith link. The link lengths are the fixed parameters in the kinematic model. They are determined before tracking begins through a calibration process described in Sec. 2.2.3. Once they have been specified, the state variables θ_i completely determine the configuration of the finger chains. Each finger contributes four joint variables to the state vector. The arrows in Fig. 2.5 illustrate the axes of the revolute joints of the fingers and thumb. The two DOFs at each finger MCP joint are modeled by a pair of revolute joint transforms, each with a single DOF. Arbitrary compound joints can be described in this manner. The shape frame for finger links is positioned at the joint center, immediately following the link rotation. Thus the transform between link and shape frames is given by

$$\mathbf{T}_i^s = \mathbf{Rot}_z(\theta_i) \tag{2.4}$$

Table 2.1 presents the kinematic model of the palm and first finger in its full detail. This is an excerpt from the table in Appendix A containing the complete hand kinematics. The table is a formatted version of a file the *DigitEyes* tracking system reads in when building its kinematic model. Each frame is numbered, and its entry in the column titled *Next* is a pointer to the frame that follows it in the chain. These pointers specify the topology of the kinematic model. Joint transforms are automatically created for links

Frame	Geometry	θ	d	a	α	Shape (in mm)	Next
0	Palm	0.0	0.0	0.0	0.0	x 56, y 86, z 15	1 8 ...
1		$\pi/2$	0.0	38.0	$-\pi/2$		2
2		0.0	-31.0	0.0	$\pi/2$		3
3		$\mathbf{q}_7$	0.0	0.0	$\pi/2$		4
4	Finger 1 Link 0	$\mathbf{q}_8$	0.0	45.0	0.0	Rad 10.0	5
5	Finger 1 Link 1	$\mathbf{q}_9$	0.0	26.0	0.0	Rad 10.0	6
6	Finger 1 Link 2	$\mathbf{q}_{10}$	0.0	24.0	0.0	Rad 9.0	7
7	Finger 1 Tip	0.0	0.0	0.0	0.0	Rad 9.0	Nil
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

Table 2.1: Kinematic and shape parameters for the palm and first finger. State variables are denoted $\mathbf{q}_i$, where $\mathbf{q}_0$-$\mathbf{q}_6$ are the palm pose and $\mathbf{q}_7$-$\mathbf{q}_{10}$ are the rotation angles for the first finger.

that are connected in the table. Links with an entry in the *Geometry* column have features that can be tracked. Frames with a state variable in the θ column have a revolute joint, while other frames have constant transforms. For example, two constant DH transforms, in frames 1 and 2, are used to position the base of the first finger (at frame 3) with respect to the palm. Note that the d and α parameters are nonzero only for constant transforms, in keeping with Eqn. 2.3. The *Shape* column describes the visible geometry of the link, which is used to render a model hand for visualization purposes. The other three fingers are similar to the first, and are contained in the Appendix.

Like the fingers, the thumb model is also constructed from the revolute joints of Eqn. 2.3. The thumb is the most difficult digit to model, due to its great dexterity and intricate actuation. It has five DOFs (see Fig. 2.4,) but one DOF at the trapeziometacarpal joint is dependent on the others. It acts to rotate the thumb longitudinally, bringing it into opposition with the fingers during grasping. Although the visual effect of this rotation is not pronounced, it is included in the current hand model for completeness.

This effect is modeled by placing an additional revolute DOF at the thumb MP joint, as shown in Fig. 2.5. Placing the oppositional DOF there, rather than at the base, helps limit its impact on the model. This choice was motivated by the experience of Rijpkema and Girard in their grasp modeling system [52]. They employed a similar thumb model and obtained realistic computer graphic animations of hand grasps. Aside from this extra joint, the thumb model is quite similar to that of the fingers, with two DOFs at the trapeziometacarpal joint and one each at the thumb MP and IP joints. The thumb occupies frames 29 through 36 in the kinematic table of Appendix A.

Real hands deviate from the above modeling assumptions in three main ways. First, most fingers are slightly nonplanar. This deviation could be modeled by allowing nonparallel joint axes, but the planar approximation has proved to be adequate in practice. Second, the last two joints of the finger (the distal and proximal interphalangeal joints) are driven by the same tendon and are not capable of independent actuation. It is simpler to include these DOFs separately, however, than to model the complicated angular relationship between them. The third deviation stems from the rigid palm assumption, which ignores the metacarpocarpal joints at the base of fingers 4 and 5 (see Fig. 2.4). When gripping an object, like a baseball, these joints permit the palm to conform to its surface, causing the anchor points to move by tens of millimeters. For free motions of the hand in space, however, this deviation is small enough to ignore.

The full hand model consists of 16 rigid bodies and a 28 dimensional state vector. The kinematic model described above is fairly standard, and closely related models have appeared in the user-interface, computer graphics, and biomechanics literature [61, 52, 66]. The most common difference between kinematic hand models is in their treatment of the metacarpophalangeal and trapeziometacarpal joints. This dissertation does not explore these subtleties of hand modeling in any significant detail. Kinematic modeling issues are secondary to the more basic concerns of real-time tracking and occlusion-handling which are the focus of this research. Once a solid foundation for visual articulated object tracking has been established, the development of accurate kinematic models for specific applications can be explored in earnest.

Articulated objects like the hand are subject to other motion constraints besides the kinematic joints which are the focus of this chapter. Regions of the state space may be inaccessible to the model, for example, due to joint limits and non-interpenetration. This leads to inequality constraints on the state estimates. Moreover, as a result of actuation and motor control patterns, groups of states will often be coupled during characteristic motions. For example, the fingers will follow similar state trajectories in making a fist. Since these constraints act on the state space at a level above the basic kinematics, they were not addressed in this work.

Kinematic models for the entire body could be developed using the methods described in this section. In fact, the body's kinematics are topologically quite similar to those of the hand, with the torso playing the role of the palm and the arms and legs taking on the role of the fingers. Like the fingers, the kinematic chains of the arms and legs are predominantly planar. One point of departure is the much greater flexibility of the torso compared to the hand as a result of the spinal column.

Adopting kinematic representations from robotics makes it possible to track any articulated object with the same mathematical framework. This generality is reflected in the software implementation of the *DigitEyes* tracking system. Any object that can be modeled using the techniques of this chapter can be tracked simply by changing the file illustrated in Table 2.1. This capability is exploited in Chpt 4, where different subsets of the whole-hand model are employed in separate experiments. To use a kinematic model for tracking, its fixed parameters must be determined from the actual, physical hand. This is accomplished in the kinematic calibration stage described next.

2.2.3 Kinematic Model Calibration and Errors

Calibrating the kinematic model by setting its fixed parameters is the most challenging aspect of hand modeling. The 21 joint axes, 15 link lengths, and 5 anchor points of the hand were determined in a three stage off-line calibration process. First, the joint axes were initialized following the finger and thumb anatomy of the previous section. Next, the link lengths were determined in two steps. In the first step, the distances between the three knuckles in each finger were measured with a ruler at the surface of the skin, to give a rough length for each link. Then, the resulting kinematic model was fit to each finger separately in two images taken with a calibrated camera: finger outstretched and finger curled. The link lengths were tuned manually until the projected hand model matched the images. Obtaining link lengths for the fingers and thumb took about four hours.

Finally, the anchor points were determined in the last stage. They are the most challenging parameters to calibrate, as they are difficult to measure on real hands, and difficult to identify in images. The anchor point calibration strategy exploited the known link lengths from the previous stage, and three images of the back of the hand with fingers extended: one looking straight down (called image 1) and two at oblique angles (images 2 and 3.) The first step was the arbitrary assignment of the palm origin to the MCP joint center of the first finger. Measurements with a ruler gave rough estimates of the anchor points with respect to this frame in the x and y axis directions (parallel to the plane of the palm, with the y axis pointing down the first finger.)

Given these preliminary anchor points, an interactive version of the tracking system was used to fit the complete hand model to image 1. After a few iterations, the anchor points were "released," freeing each finger and thumb to move independently of the palm. This allowed the base of each digit to shift until the error in its tip and edge positions was minimized. The original anchor point and the current base of each finger and thumb were overlaid on the hand image. Estimation was halted after a few iterations, and the original anchor points were manually adjusted to agree with the new base positions. This procedure was repeated with the two oblique images, to localize the anchor points along the z axis (out of the plane of the palm.) It took a few hours to calibrate the anchor points. The calibration procedure described above was performed once for my right hand, and the resulting kinematic model was used in all subsequent experiments. It is presented in Appendix A in its full detail.

The calibration goal of this dissertation was to obtain a useful kinematic model as quickly as possible. The experimental performance of this model on a wide variety of hand images indicates that this goal was achieved. However, calibration is likely to remain a nontrivial component of any future model-based articulated object tracking system. The adequacy of the hand model calibration is discussed further in Chpt. 4, and an approach to automatic, on-line calibration is discussed in Chpt. 6. The remainder of this section presents a taxonomy of kinematic model errors, and describes their effect on tracking performance.

Errors can occur in both the topological and parametric parts of the kinematic model. Topological errors, like incorrect joint axes, are the result of anatomical deviations from the model. For example, if a finger exhibits a large deviation from planarity, the joint axes of the planar finger model will be incorrect. As a result, it will be impossible to set the state variables so that the finger links are registered with the image. This type of error is easily detected by overlaying the model projection on the image.

Improper calibration can also produce errors in the link lengths and anchor points that make up the fixed model parameters. The effect of incorrect link lengths is particularly striking. If the links are too long, the finger develops obvious "kinks" in trying to fit its image. If the links are too short, the model finger tip never reaches its match in the image. Errors in the anchor points are the most difficult to detect and correct, as they may not be apparent unless the model is fit to the image under a wide range of viewing angles.

I encountered all three of these types of kinematic errors in the early stages of hand modeling. They proved to be fairly easy to detect using an interactive tracking system. The system I developed made it possible to fit hand models to images, see the result in 3D from an arbitrary viewpoint, and quickly modify the joint angles to observe their effect on registration. The importance of having an interactive system when developing these models cannot be over-emphasized. With this tool, the space of possible models could be searched efficiently and problems diagnosed quickly. The interactive system is described in more detail in Chpt. 4.

A calibrated kinematic model can be viewed as a mapping from the state space to the 3D positions of the shape frames, which contain the visible surfaces of the links. The next stage in this mapping is the projection of the 3D link geometry into the image plane. This is accomplished through a camera model, which maps points from the shape frames into image coordinates.

2.3 Camera Modeling and Calibration

As with the kinematics, cameras can also be modeled by transformations between coordinate frames. The imaging geometry of a pin-hole camera is modeled by a *projective* transform between the camera and image buffer coordinates [16]:

$$\mathbf{P}_b^c = \begin{bmatrix} \alpha_u & 0 & u_0 & 0 \\ 0 & \alpha_v & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} . \tag{2.5}$$

The intrinsic camera parameters, $\{\alpha_u, \alpha_v, u_0, v_0\}$, define the scale factors and origin for the camera's sensor array. The image coordinates of a 3D point $\mathbf{p}_c$ located in the camera frame are $\mathbf{w} = [x_b/z_b \ y_b/z_b]$, where $\mathbf{p}_b = [x_b \ y_b \ z_b] =$

$\mathbf{P}_b^c\mathbf{p}_c$. Let $\mathbf{S}[\cdot]$ denote the *scaling operator* that returns the first two elements of a vector divided by its third. Furthermore, let $\mathbf{T}_c^w$ specify the camera position with respect to the world frame (the extrinsic camera model.) The projection of a world point $\mathbf{p}_w$ into the camera image can then be written

$$\mathbf{w} = \mathbf{S}[\mathbf{P}_b^c\mathbf{T}_c^w\mathbf{p}_w] = \mathbf{S}[\mathbf{P}\mathbf{p}_w] \ , \tag{2.6}$$

where $\mathbf{P}$ is the 3x4 camera projection matrix.

When the distances between points on an object of interest are small compared to the distance to the camera, the perspective projection model can be approximated by orthographic projection

$$\mathbf{w} = \bar{\mathbf{P}}_b^c\mathbf{T}_c^w\mathbf{p}_w = \bar{\mathbf{P}}\mathbf{p}_w \ , \tag{2.7}$$

where

$$\bar{\mathbf{P}}_b^c = \begin{bmatrix} \alpha_u & 0 & 0 & u_0 \\ 0 & \alpha_v & 0 & v_0 \end{bmatrix} \tag{2.8}$$

is an orthographic transform, and $\bar{\mathbf{P}}$ is the 2x4 orthographic projection matrix. The fact that the camera and kinematic transformations have a similar algebraic form makes it easy to combine them in one representational framework.

Camera models are specified by the sets of intrinsic and extrinsic parameters. These parameters must be determined in a calibration stage before the model can be employed for tracking. I used Robert's calibration algorithm, described in [53], for all of the experiments in this thesis. The algorithm uses a single image of a cube of known size to determine both the intrinsic and extrinsic camera parameters. The procedure has two stages: First, the user manually identifies the position of six predetermined points in the cube image, and an approximate calibration matrix is generated. Second, the approximate model is refined in an iterative stage using additional, automatically detected image features and a standard numerical minimization package. The advantage of Robert's algorithm is that the image features are continuously updated in the iterative stage along with the camera model, reducing the effect of any initial errors in locating the image points.

Evaluating the accuracy of a calibrated camera model is a difficult task. In theory, image features from two faces of the cube image provide sufficient geometric constraints for calibration (see [16], Sec. 3.4.1.3). However, since numerical minimization is employed, there is no guarantee that the stopping point is the global minimum. A partial evaluation of the calibration accuracy was obtained when a pair of cameras were calibrated for stereo experiments. In this case, the epipolar lines for features in both images were examined and found to be accurate to within the image resolution. Additional experimental evaluation of Robert's algorithm is described in [53]. An advantage of calibrating with a cube target, as opposed to the series of grid positions that are traditionally employed, is that multiple cameras with convergent axes can be easily calibrated with respect to the same world frame (defined within the cube.) The calibration cube was manufactured out of PVC plastic to a tolerance of ± 0.003 in. on all dimensions, by K^2T, Inc.

2.4 Tracking Through Template Registration

Visual tracking is a sequential *image registration* problem. The state estimate in each frame minimizes the *residual error* between the projected object model and the image. Different tracking approaches are distinguished by the choice of residual function. In template registration, the residual error measures the intensity difference between an input image and the image predicted by the kinematic model. A set of templates describe the image appearance of each link. The position of each template in the image is given by the kinematic and camera models as a function of the state. State estimates are obtained by minimizing the residual numerically.

This section has four parts. First, deformation functions are developed that map templates to images as a function of the state. Next, deformations are incorporated into a *Sum of Squared Differences* (SSD) residual function. A gradient-based minimization algorithm is described in the third part. In the last part, the deformation function Jacobian is derived and its computation is discussed.

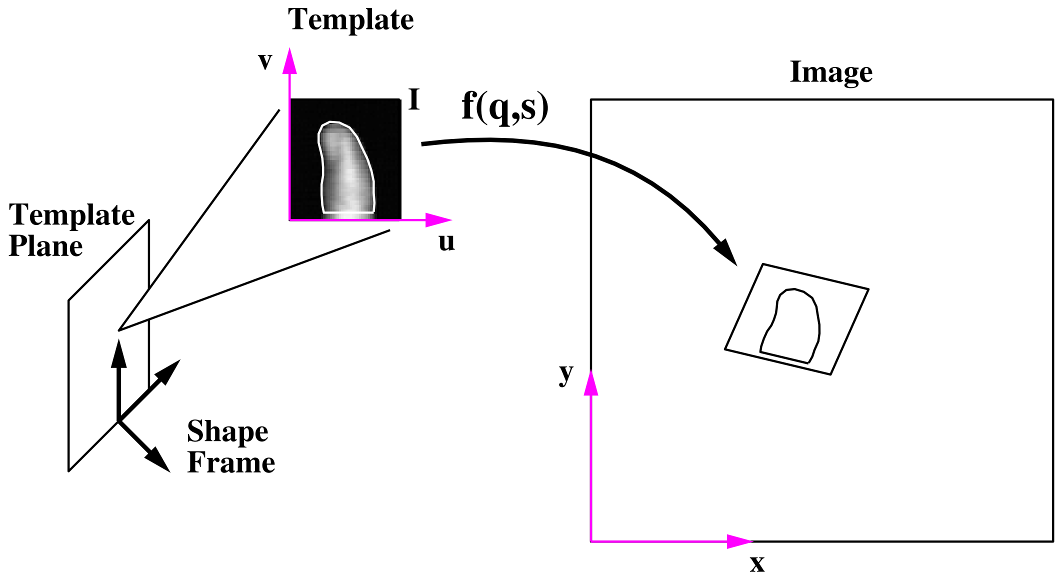


Figure 2.6: A finger tip template is mapped into the image by a deformation function. Its template plane positions it in shape coordinates. Only the template pixels, enclosed by the white boundary contour, are mapped to the image.

2.4.1 Kinematic Deformations of Templates

The appearance of each finger phalange is modeled by a collection of templates, each associated with a particular viewing direction. Each template has two parts, an image with an associated coordinate frame, and a *boundary contour* that inscribes the template pixels. Each template can be viewed as being "painted" on a plane which is located in the shape frame of the link, as illustrated in Fig. 2.6. The template plane determines the position in the shape frame of each template pixel.

Given the approximate pose of a link relative to the camera, the appropriate template plane can be chosen automatically. The number of required templates is a function of the shape and photometry of the link. For cylindrical links, like finger phalanges, a single view may be enough, while an object like the palm or body torso will require more. The number of required views for cylindrical objects can be reduced significantly by allowing the template plane to rotate around the axis of symmetry, maintaining a frontal camera orientation.

The template plane model merges geometric and photometric aspects of image appearance in a single framework. The orientation and position of the template plane relative to the camera capture the effects of foreshortening and rotation on the image of the link. The template pixels capture intensity variations at a finer scale resulting from the shape of the finger phalanges. A variety of features, from edges to textures, can be employed by changing the form of the template.

Given the state of the hand, the image appearance of each link can be synthesized by projecting its template plane through the camera model. The combination of kinematic and camera transforms is represented by a *deformation function* [51], $\mathbf{f}(\mathbf{q}, \mathbf{s})$, which maps template coordinates to image coordinates as a function of the state. If $\mathbf{s} = [u\ v]$ denotes a template pixel and $\mathbf{w} = [x\ y]$ denotes its corresponding image pixel, then $\mathbf{w} = \mathbf{f}(\mathbf{q}, \mathbf{s})$. This mapping is illustrated in Fig. 2.6 for a finger tip template. The deformation function is constructed from a series of coordinate transformations. Let the coordinate axes of template I_j, expressed in its shape frame, make up the column vectors of the 3x2 matrix $\mathbf{F}_j$. Combining this with Eqns. 2.2, 2.4, and 2.7 yields the orthographic deformation function

$$\mathbf{f}_j(\mathbf{q}, \mathbf{s}) = \bar{\mathbf{P}}\mathbf{T}_w^j(\mathbf{q})\mathbf{T}_j^s(\mathbf{q})\mathbf{F}_j\mathbf{s} \ . \tag{2.9}$$

Deformable template models have appeared in previous tracking and registration work. Their use for pattern recognition goes back at least to Widrow [70]. In 1981, Lucas and Kanade [36] proposed an image registration scheme using affine deformations that has become a standard solution to optical flow and point tracking problems. In joint work with Andy Witkin, I investigated an approach to 2D template tracking based on deformation models [51]. In our approach, the arbitrary (rigid or nonrigid) motion of the pixels was assumed to be the result of an unknown, but smooth, deformation function. This unknown deformation was approximated by its truncated Taylor Series, resulting in a family of polynomial deformation models. I developed a real-time system on an SGI GTX workstation that used these models to track a small window of pixels, selected by the user, through an image sequence. A related hierarchy of 2D motion models was published later by Bergen et. al. [6]. The kinematic deformation model of Eqn. 2.9 is a natural extension of this earlier work to a 3D tracking domain. A further extension of this paradigm occurs in Chpt. 3, in addressing self-occluding motion.

2.4.2 SSD Residual Error Function

The residual function for template registration measures the intensity difference between a deformed template and an input image. I employ the standard *Sum of Squared Differences* (SSD) error measure between filtered pixels. In the SSD approach, both the input image and the templates are convolved with a filter and subtracted, squared and summed to obtain the residual error. By changing the filter, different properties of the image can be emphasized. For example, using a Laplacian of Gaussian (LOG) filter produces a residual error which is sensitive to edge energy. Using Eqn. 2.9, the residual at a pixel $\mathbf{s}$ in template I_j can be written

$$R_j(\mathbf{q}, \mathbf{s}) = \hat{I}(\mathbf{f}_j(\mathbf{q}, \mathbf{s})) - \hat{I}_j(\mathbf{s}) \ , \tag{2.10}$$

where $\hat{I}$ and $\hat{I}_j$ are the filtered input image and template, respectively. The template error resulting from this residual choice is given by

$$E(\mathbf{q}) = \frac{1}{2} \int_{I_j} R_j(\mathbf{q}, \mathbf{s})^2 \, \mathbf{ds} = \frac{1}{2} \int_{I_j} [\hat{I}(\mathbf{f}_j(\mathbf{q}, \mathbf{s})) - \hat{I}_j(\mathbf{s})]^2 \, \mathbf{ds} \ . \tag{2.11}$$

Each template in the object model contributes an error term of the form of Eqn. 2.11.

The SSD residual is one possible choice from a large class of image similarity measures [55, 25]. It is a traditional choice for template matching applications, because it works well in practice. Any differentiable residual could be employed in Eqn. 2.11 to measure the error, and the rest of the framework would remain unchanged.

2.4.3 State Estimation by SSD Residual Minimization

The residual in Eqn. 2.10 is a nonlinear function of the state $\mathbf{q}$. There are two main sources of nonlinearity: trigonometric terms in the kinematic model from Eqn. 2.9, and intensity variations in the template and input images. Use of a perspective camera model introduces a secondary source of nonlinearity. The kinematic model is a smooth function of the state. SSD error functions are also observed empirically to be smooth and approximately quadratic around their minima [4]. As a result, Eqn. 2.11 can be treated as a smooth function of the state and minimized numerically through standard gradient-based methods [14]. The use of continuous variable optimization techniques is one of the key distinctions between the tracking approaches in this thesis and [72], and the earlier works of O'Rourke [42] and Hogg [23]. These optimization techniques make it possible to search much larger state spaces than classical interval analysis or constraint satisfaction approaches.

Given an error function like Eqn. 2.11, tracking can proceed by a simple gradient descent minimization algorithm. If $E_k(\cdot)$ denotes the state-dependent error for input image I_k, the state update is given by:

$$\mathbf{q}_k = \mathbf{q}_{k-1} - \rho \frac{\partial E_k}{\partial \mathbf{q}}(\mathbf{q}_{k-1}) \tag{2.12}$$

where ρ is the step size. The update step can be iterated when the inter-frame motion is large. The estimate from the previous frame, possibly modified by velocity-based prediction, serves as the starting point for minimization in the current frame. Sec. 2.5.3 discusses the use of more sophisticated minimization algorithms than gradient descent.

Differentiating Eqn. 2.11 yields

$$\frac{\partial E}{\partial \mathbf{q}} = \int_{I_j} R_j(\mathbf{q}, \mathbf{s}) \frac{\partial R_j}{\partial \mathbf{q}}(\mathbf{q}, \mathbf{s})\, \mathbf{ds} = \int_{I_j} R_j \frac{\partial \mathbf{f}_j}{\partial \mathbf{q}}^T \frac{\partial \hat{I}}{\partial \mathbf{w}}\, \mathbf{ds} \ , \tag{2.13}$$

where $\partial R_j / \partial \mathbf{q}$ denotes the *residual Jacobian*. The residual Jacobian is a product of two terms, the derivative of the deformation function, and the image gradient. Since the deformation function is a product of kinematic transforms (see Eqn. 2.9,) its derivative must take the form of a kinematic Jacobian. The derivation of this Jacobian and its on-line computation are discussed in the next section. The Jacobian maps state velocities to the image plane velocities of template pixels. It follows that the residual Jacobian at an image point is a weighted combination of the kinematic Jacobian of its associated link template point.

The key to the practical success of the gradient-based minimization approach is a high image sampling rate, which limits image motion between frames. Templates will generate useful error signals only when they "see" a significant portion of the link they are tracking, making it important to limit motion in the image plane. In the state space, a region of convergence (ROC) exists around the global minimum. Interframe motion must be small enough for the predicted state, which determines the starting point for minimization, to fall within the ROC at each image [64]. Analyzing the required sampling rate is difficult, as it depends on the object state, the form of the residual error measure, and the image properties. However, experimental results in Chpt. 4 indicate that image motions of five to ten pixels can be handled successfully, corresponding to a 15 Hz sampling rate under normal hand motion.

2.4.4 Deformation Function Jacobians

The deformation function of Eqn. 2.9 is a series of coordinate transformations. As a result, standard techniques from robotics (see [59], Sec. 5.1) can be employed to compute its Jacobian. Let $\mathbf{s}_j$ be a pixel in template I_j which projects to $\mathbf{w}_j$ in the image plane. Let $\mathbf{p}_j = \mathbf{F}_j \mathbf{s}_j$ denote the point's coordinates in the shape frame of link j. Suppose further that link frame i has a revolute joint with angle θ_i that effects the position of frame j. Then the basic Jacobian component, $\partial \mathbf{w}_j / \partial \theta_i$, can be derived as follows.

The first step is to reorganize Eqn. 2.9, letting $\mathbf{W}_j$ denote the point $\mathbf{p}_j$ in world coordinates prior to camera projection, obtaining

$$\mathbf{f}_j(\mathbf{q}, \mathbf{s}) = \bar{\mathbf{P}} \mathbf{W}_j = \bar{\mathbf{P}} \mathbf{T}_w^{s_j}(\mathbf{q}) \mathbf{p}_j = \bar{\mathbf{P}} [\mathbf{R}_w^{s_j}(\mathbf{q}) \mathbf{p}_j + \mathbf{d}_w^{s_j}(\mathbf{q})] \ , \tag{2.14}$$

where $\mathbf{R}_w^{s_j}$ and $\mathbf{d}_w^{s_j}$ are the rotation and translation components of $\mathbf{T}_w^{s_j}$, the position of link j's shape frame in world coordinates.

Separating the transform for $\mathbf{W}_j$ into components before and after frame i and differentiating with respect to time yields

$$\begin{aligned} \dot{\mathbf{W}}_j &= \frac{d}{dt}[\mathbf{R}_w^i (\mathbf{R}_i^{s_j} \mathbf{p}_j + \mathbf{d}_i^{s_j}) + \mathbf{d}_w^i] \\ &= \dot{\theta}_i \mathbf{n}_w^i \times \mathbf{R}_w^i (\mathbf{R}_i^{s_j} \mathbf{p}_j + \mathbf{d}_i^{s_j}) \ , \end{aligned} \tag{2.15}$$

where $\mathbf{n}_w^i$ is the rotation axis for joint i expressed in world coordinates. The Jacobian follows immediately as

$$\frac{\partial \mathbf{f}_j}{\partial \theta_i}(\mathbf{s}_j) = \bar{\mathbf{P}} \frac{\partial \dot{\mathbf{W}}_j}{\partial \dot{\theta}_i} = \bar{\mathbf{P}} [\mathbf{n}_w^i \times \{\mathbf{r}_w(\mathbf{s}_j) - \mathbf{d}_w^i\}] \ . \tag{2.16}$$

The term in braces is the moment arm for the rotation of point $\mathbf{s}_j$ about joint i, expressed in world coordinates. It is determined by $\mathbf{r}_w(\cdot)$, a function which gives the 3D position of a point in template coordinates with respect to the world frame. From the form of Eqn. 2.16, the Jacobian component for a revolute joint is obtained by projecting a spatial velocity vector into the image plane. In cases where perspective effects are significant, the orthographic mapping is replaced by an affine approximation to the perspective projection at each link.

Using Eqn. 2.16 in a tracking algorithm involves the following steps: First, the spatial positions of all frames are computed with respect to the world. Then the revolute joints are examined in sequence. For each joint, the template planes which it effects are processed in order. Each template pixel involved in Eqn. 2.13 makes a contribution to the Jacobian which is determined solely by its position with respect to the active joint axis. The total cost of the Jacobian computation depends on the number of templates, their size in pixels, the DOFs of the object, and its kinematic topology. Empirical evaluation of this cost and its ramifications for real-time implementation are presented in Chpt. 4. The compact derivation of Eqn. 2.16 and the simplicity of its computation are fortunate consequences of the highly regular structure of spatial kinematic models.

2.5 Tracking Through Feature Alignment

In the template registration approach to visual tracking, intensity errors are used to measure the *geometric* misalignment between the projected model and the input image. Templates provide a useful level of generality, and make it possible to exploit arbitrary texture cues. For a specific object like the hand, however, the constraints provided by template matching can be approximated by purely geometric error functions involving point and line features. The advantage of this is two-fold. First, geometric residual errors

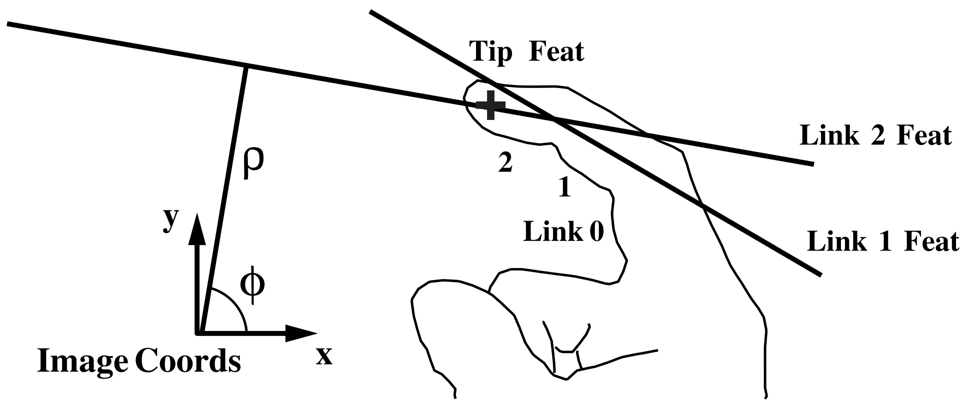


Figure 2.7: Features used in hand tracking are illustrated for finger links 1 and 2, and the tip. Each infinite line feature is the projection of the finger link central axis.

have a closed form expression that doesn't contain unmodeled quantities like image intensities. This is important in analyzing the performance of the estimator. Second, line and point features can be used to approximate template matching with a significantly lower computational cost. This feature-based approach is the basis for the real-time tracking results of Sec. 4.3.

2.5.1 Line Feature Residual and Jacobian

Finger phalanges can be viewed as cylinders and represented in the image by the projection of their central axis. Due to the lack of texture in images of the skin, the intensity gradient available for tracking comes from the contrast between the skin and its background, and is orthogonal to the projected cylinder axis. The image of each finger phalange can therefore be modeled as an infinite line feature passing through the projection of its cylinder axis. The line feature parameters $[a\, b\, \rho]$ satisfy the equation $ax + by - \rho = 0$. Figure 2.7 shows the line features for two links in an image of the first finger.

The residual error measures the perpendicular distance between the line feature and the projections of points along the cylinder axis. Point projections can be obtained from the deformation function of Sec. 2.4.1. In this case, only a single line in the template plane, corresponding to the central axis of the cylindrical link, is mapped through the deformation function. If $\mathbf{s}_j$ represents a point along the central axis, its contribution to the residual error is given by

$$l_j(\mathbf{q}, \mathbf{s}_j) = \mathbf{m}^T \mathbf{w}_j - \rho = \mathbf{m}^T \mathbf{f}_j(\mathbf{q}, \mathbf{s}_j) - \rho \ , \tag{2.17}$$

where $\mathbf{m} = [a\ b]$.

The Jacobian component generated by this residual is

$$\mathbf{J}^l_j(\mathbf{q}, \mathbf{s}_j) = \frac{\partial \mathbf{f}_j}{\partial \mathbf{q}}^T \mathbf{m} \ . \tag{2.18}$$

The role of the line feature in approximating the template residual can be seen by comparing Eqns. 2.18 and 2.13. In the line case, the normal vector $\mathbf{m}$ plays the same role as the image gradient. It corresponds to an image gradient field with a zero component along the central axis of the link.

2.5.2 Point Feature Residual and Jacobian

Links at the end of kinematic chains, like the fingertips of the hand, generate point features with parameters $[x\,y]$, as illustrated in Fig. 2.7. The tip residual measures the Euclidean distance in the image between the projected model point and the actual tip location, $\mathbf{c}_j$, in the image:

$$v_j(\mathbf{q}, \mathbf{s}_j) = \|\mathbf{v}_j(\mathbf{q}, \mathbf{s}_j)\| = \|\mathbf{f}_j(\mathbf{q}, \mathbf{s}_j) - \mathbf{c}_j\| \ . \tag{2.19}$$

Its Jacobian component is given by

$$\mathbf{J}^v_j = \frac{\partial \mathbf{f}_j}{\partial \mathbf{q}}^T \frac{\mathbf{v}_j}{\|\mathbf{v}_j\|} \ . \tag{2.20}$$

In this case, the unit vector in the $\mathbf{v}_j$ direction models an image gradient that is nonzero only along radial lines from the tip feature position.

The residual functions in Eqns. 2.17 and 2.19 measure distances in the image plane. The feature residuals for each link and tip in the model are concatenated into a single residual vector, $\mathbf{R}(\mathbf{q})$. The total error is then given by

$$\mathbf{E}(\mathbf{q}) = \frac{1}{2}\mathbf{R}(\mathbf{q})^T\mathbf{R}(\mathbf{q}) \ . \tag{2.21}$$

This error will be quadratic in the distances from the hand model projections to the image features. This agrees with the empirical observation that SSD residual errors are quadratic around their minimum.

Although these approximations were motivated by the hand, they are applicable to the body as well. The primary difference in between fingers and limbs is that clothing can provide image gradient constraints in arbitrary directions, unrelated to the central axis of the limb. However, clothing and background color will still often differ significantly, resulting in a strong edge constraint. If the interior texture is insignificant given the resolution of the camera, then the line and point models can be applied without modification.

2.5.3 State Estimation by Feature Residual Minimization

The state estimation problem can be achieved by minimizing the total error in Eqn. 2.21. This is a classical nonlinear least-squares problem, which can be solved numerically by Gauss-Newton minimization [14]. The GN state update equation is given by

$$\mathbf{q}_{k+1} = \mathbf{q}_k - [\mathbf{J}_k^T\mathbf{J}_k + \mathbf{S}]^{-1}\mathbf{J}_k^T\mathbf{R}_k \ , \tag{2.22}$$

where $\mathbf{J}_k$ is the Jacobian matrix for the residual $\mathbf{R}_k$, both of which are evaluated at $\mathbf{q}_k$. $\mathbf{S}$ is a constant diagonal *conditioning matrix* used to stabilize the least squares solution in the presence of kinematic singularities. Each entry in $\mathbf{S}$ weights one of the state variables, determining how strongly it is affected by the residual. Weights for quaternion, translation, and joint angle states were set at 1.0, 10.0, and 1000.0, respectively. These weights were chosen empirically and used in all of the experiments in this thesis. The above GN formulation is based on the rigid body tracking work of Lowe [35].

2.5.4 Visual Tracking and Kinematic Singularities

The use of a conditioning matrix in the state estimator of Eqn. 2.22 closely parallels methods for dealing with kinematic singularities in robot control (see [40], Eqn. 9.22.) In both the articulated tracking and control cases, the goal is to obtain a stabilized inverse solution in the case where the Jacobian has lost rank. One important difference between the estimation and control cases is the influence of the feature model on the singular configurations. The matrix $\mathbf{J}_k$ in the robot manipulator case maps from joint velocities to link frame velocities in 3D. While in the articulated tracking case, the Jacobian maps from joint velocities to residual velocities in the image plane. It follows that the necessary conditions for measurement singularities can be obtained by examining Eqns. 2.18 and 2.20.

As Eqn. 2.18 demonstrates, there are two possible conditions under which the state to line residual mapping is singular. The first case is when $\partial \mathbf{f}_j / \partial \mathbf{q} = \mathbf{0}$. As Eqn. 2.16 demonstrates, this can only occur when the link point velocity is directed along the camera axis (assuming orthographic projection). In this case, its projection into the image plane will be zero. This configuration occurs, for example, when the palm of the hand is flat and parallel to the image plane. All knuckle joint to link axis point mappings will be singular in this case.

The second singular case occurs when the line feature normal, $\mathbf{m}$, is orthogonal to the deformation Jacobian. This case is more serious, as it can produce a *singular subspace* in the state space. An example is when the plane of the finger contains the camera position. In this case, the link point

Jacobians will be nonzero for all finger articulations, but they will all lie in the line formed by intersecting the finger and image planes. This line will be parallel to the feature measurement lines produced by the finger, leading to loss of rank. The equations reflect the intuitively obvious fact that when the finger is curling towards the camera, the motion of its edges contain no information about the 3D motion.

Examination of the point feature Jacobian of Eqn. 2.20 indicates that it possesses the same two singular configurations that the line does. However, the orthogonal case is much less serious for a point feature, as it does not generate a singular subspace. This analysis demonstrates the value of the closed form approximations to the template residuals. They lead to an intuitive mathematical description of a basic property of articulated object tracking problems.

As in the robot manipulator case [40], state space neighborhoods of the singular points will exhibit marked sensitivity loss, in that large state space motions will have little effect on the image. This sensitivity loss makes accurate tracking in the neighborhood of singularities difficult. Experimental observations of the effects of near-singular tracking are discussed in Chpt. 4. The stabilization method of Eqn. 2.22, which has been used for rigid body tracking [35], also works for articulated state estimation problems.

2.5.5 Tracking with Multiple Cameras

Both the template registration and feature alignment approaches generalize easily to tracking with more than one camera. When multiple cameras are used, the residual vectors from each camera are concatenated to form a single global residual vector. This formulation exploits partial observations. If a finger link is visible in one view but not in the another due to occlusion, the single view measurement is still incorporated into the residual, and therefore the estimate. When this framework is augmented with occlusion-handling, the resulting algorithm can utilize any visible pixel from any camera position in estimating the state. Experimental articulated tracking results using two cameras were first reported in [46, 48]. Two camera results for human body tracking were presented more recently in [29, 32].

2.6 Discussion

Kinematic models made up of links and joints represent the most basic constraints on the motion of articulated kinematic chains, and make it possible to recover 3D motion from a single image sequence. A kinematic hand model was developed through anatomical analysis and calibrated using an interactive tracking system. Sections 2.4 and 2.5 described two approaches to estimating the model state from an image sequence.

The template registration approach of Sec. 2.4 belongs to the class of direct, energy-based vision algorithms which was popularized by deformable models [65] (including 2D Snakes [28],) and has been applied to a wide variety of problems [73, 53]. It is a direct method in which pixels are mapped to state estimates without an intervening feature detection stage. Its advantage is the direct enforcement of kinematic constraints on image interpretation. These constraints integrate information from different parts of the image, reducing the impact of localized interpretation errors on the final estimate. This will turn out to be particularly useful in tracking self-occluding objects in Chpt. 3.

Constraints in the classic energy-based approach take the form of a smoothness penalty term which is added to the residual error in forming the objective function. These soft constraints can be viewed as prior distributions over the state space [62, 63]. They are enforced explicitly, reflecting the fact that the size of the over-parameterized state space exceeds the actual DOFs in the scene. In contrast, kinematic constraints are enforced implicitly through the joint angle parameterization of articulated motion. Kinematic models are hard constraints— no amount of measurement error should cause the rigid bodies in a chain to separate from each other, or rotate in ways not permitted by their joints.

Section 2.5 demonstrates that the template residual functions for finger phalanges can be approximated by geometric expressions in line and point features. The result is a second tracking approach based on feature alignment. The residuals for point and line features have a closed form expression which makes the singularity analysis of Sec. 2.5.4 possible. In addition, these features can be detected through a simple algorithm which is suitable for real-time implementation, as Sec. 4.3.1 will demonstrate.

Chapter 3

Tracking Self-Occluding Objects with Layered Templates

Self-occlusion is an ubiquitous property of articulated object motion. It occurs when an *occluding* link blocks the camera's view of an *occluded* link. The images in Fig. 3.1 illustrate the effect of occlusion on the visual tracking problem. In this example, the hand rotates around the middle finger axis with the fingers held rigid. Suppose that the first and second fingers are each being tracked with a single template using the approach of Sec. 2.4. In figure (b), the two finger templates can be registered to the image independently, as both fingers are fully visible. In (a) and (c), however, the two templates overlap due to occlusion, and pixels in the overlapping region of the image must be assigned to the correct template for successful registration. Self-occlusion adds a combinatorial aspect to tracking— where templates overlap, their visibility to the camera must be determined in conjunction with their registration to the image.

This chapter extends the template registration approach of Chpt. 2 to handle self-occluding objects such as the two fingers of Fig. 3.1. The solution is based on a layered template representation of occlusion [1] with two

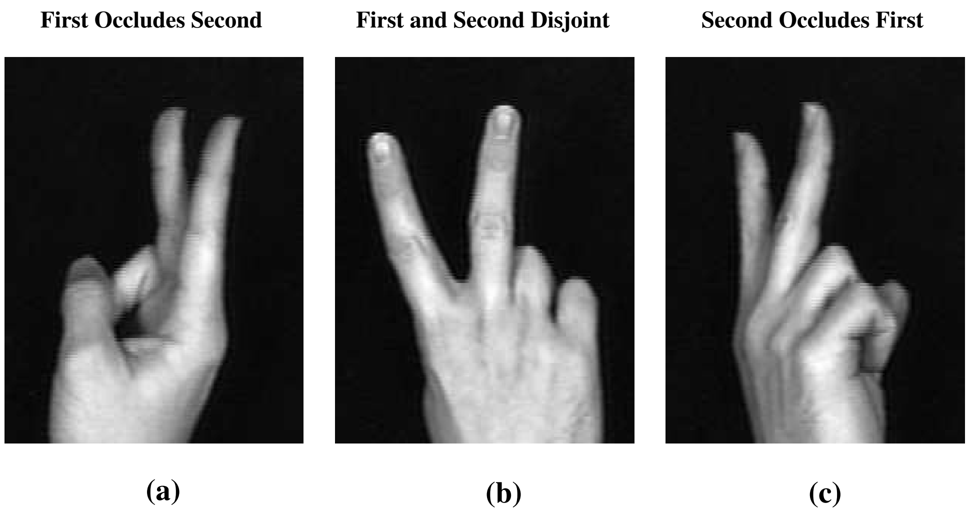


Figure 3.1: Three snapshots from a motion sequence, illustrating the different occlusion relations between the first and second fingers of the hand.

main components. The first component is a *visibility order* for overlapping templates, with the property that no template is occluded by a template that follows it in the list. The visibility order can be used to determine which template corresponds to a given region of pixels. The order between templates changes with the state, as in the transition from figure (a) to (c). In (a), the visibility order is {*Template 1, Template 2*}, while in (c) it is the reverse. The second component in the layered model is a set of *window functions* that block, or mask out, the contributions of occluded templates, as determined by the visibility order. Each template has an attached window function which moves with it as a function of the state.

Tracking using a layered representation requires the simultaneous solution of two problems: determining the visibility order for the templates that describe an object, and registering the overlapping templates to the input image. In bottom-up approaches to occlusion analysis, visibility order is estimated from image motion [11, 67] or contours [41]. This thesis explores an alternative, top-down approach which uses the kinematic model in conjunction with a high image sampling rate to partition the state space into regions with a *fixed* visibility order. In this approach, the visibility order for the current frame is predicted from the previous state estimate and used to constrain image interpretation.

The following sections develop the layered template representation in more detail, and describe its use in a model-based tracking algorithm for self-occluding objects. The first step is an analysis of the visibility orders for objects, like the hand, that are composed of planar kinematic chains. The next step is the incorporation of visibility-ordered templates into the registration algorithm of Sec. 2.4. Window functions provide a mathematical tool for arbitrating access to the image among overlapping templates. They are incorporated into a residual error function, which is minimized by gradient-based methods. The main computational step in gradient-based minimization is the Jacobian computation for layered templates, which is described in detail. This is followed by a discussion of image segmentation, and an outline of the complete tracking algorithm. The final contribution of this chapter is an analysis of the existence conditions for the invariant visibility orders employed in tracking. Occlusion ambiguities, in which the visibility order is not invariant, are introduced and their ramifications for tracking are discussed.

3.1 Model-based Occlusion Analysis

The tracking algorithm developed in this chapter is based on a simple observation: the occlusion relationships between the convex rigid bodies of an articulated object in motion rarely change instantaneously. As a result, the visibility order for the object templates is *invariant* under the small motions that occur between two frames of an image sequence, given a high sampling rate. This invariant order makes it possible to remove the discrete, combinatoric aspect of occlusion from the tracking problem, leaving only the

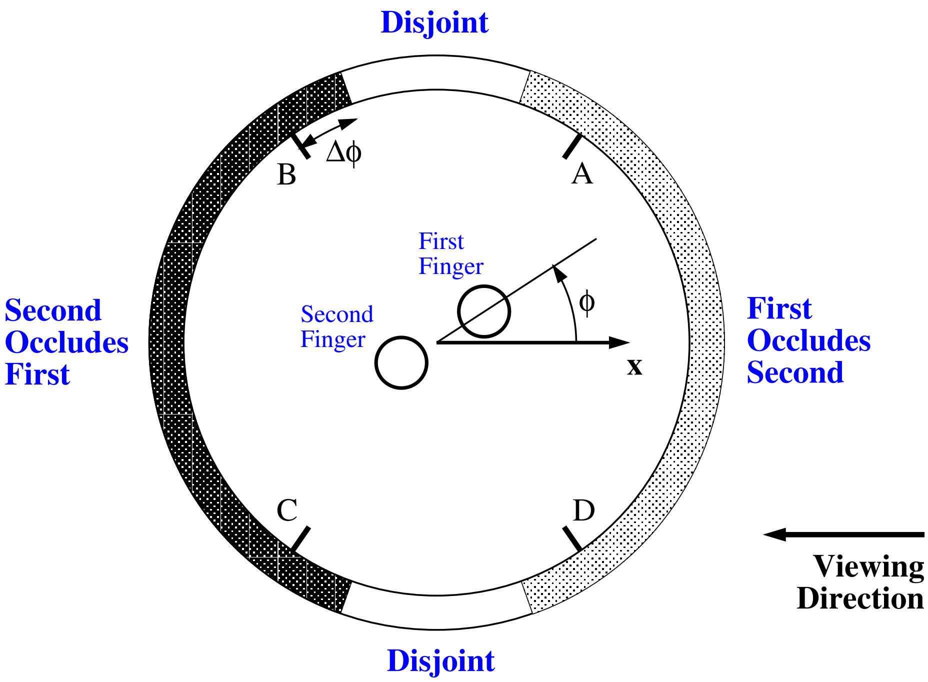


Figure 3.2: The partition of the rotation space (unit circle) into regions with an invariant visibility order. This is a top view of the scene in Fig. 3.1, with the camera located on the right. ϕ gives the rotation of the hand relative to the camera.

registration of overlapping templates. In this section, the use of an invariant visibility order in tracking is illustrated for the two finger motion sequence of Fig. 3.1.

Figure 3.2 shows the visibility order for the fingers in Fig. 3.1 as a function of the hand state. Since the hand has one DOF in this example, the space of rotations is a unit circle. The angles marked A, B, C, D denote *occlusion events*, points at which the occlusion relations change. Passing through $\phi = A$, for example, causes a transition from (a) to (b). The amount of hand rotation between frames is limited by the sampling rate to a small angle, $\Delta\phi$. Therefore, in local tracking the state estimate for the current frame is restricted to a *motion interval* of $\pm\Delta\phi$ around the previous estimate. If the ϕ_{k-1} is the state from the previous frame, then it follows that $\phi_k \in [\phi_{k-1} - \Delta\phi, \phi_{k-1} + \Delta\phi]$.

Since the occlusion events are sparsely distributed, the visibility order for the two templates will be constant from frame to frame across most of the image sequence. The template order in cases (a) and (c), for example, holds for nearly 90 degrees of hand rotation, which is much larger than $\Delta\phi$. When the motion interval contains an occlusion event, the visibility order will change. However, the transition always occurs between an occluded and a disjoint case. As a result, the onset of occlusion can be anticipated by assigning the occluded visibility order to the disjoint case near the event. This assignment is achieved by growing the occluded regions into the disjoint regions by the motion bound, $\Delta\phi$, resulting in the state space partition shown in Fig. 3.2 as dark and light grey bands.

The partition illustrated in Fig. 3.2 divides the state space into regions with a locally invariant visibility order. This partition has the following property: *Given the state of the object at time k, its membership in the state partition determines the visibility order at time $k+1$.* The occluded partitions (the light and dark grey sets in Fig. 3.2) contain all of the states that lie within $\pm\Delta\phi$ of an occluded configuration. The disjoint partitions (the white sets in Fig. 3.2) contain the states for which there are guaranteed to be no occlusions under bounded motion. These sets form a buffer zone in which the tracker can be configured for the next occlusion event. The partition is used in visual tracking problems to predict the visibility order for the current, unknown state from the previous state estimate. The predicted visibility order is used in turn to construct a layered template representation of the image, thereby reducing the tracking problem to the registration of overlapping templates between frames.

The construction of the partition in Fig. 3.2 depends on three properties of the motion and the estimator. First, the regions in state space in which the templates occlude each other must be separated by regions in which they are disjoint. Second, the change in state between frames must be small enough to take advantage of the disjoint regions. If $\Delta\phi$ is too large, growing the occluded regions will eliminate the disjoint regions entirely. Third, the state estimate must be accurate enough to make useful predictions about membership in the partition. These issues are addressed in more detail in the sections that follow.

3.2 Visibility Orders for Planar Kinematic Chains

A key step in the tracking algorithm for self-occluding motion is the construction of visibility orders for link templates. A *visibility order* for the bodies in an articulated object is an ordered list with the property that each body will not be occluded by any of the bodies that follow it. The next three sections present a set of rules for constructing invariant visibility orders for objects, like the hand, that are composed of planar kinematic chains. Section 3.4 discusses the existence of these invariant orders in the general case. The simplest type of visibility order is a binary occlusion relation between two bodies.

3.2.1 Binary Occlusion Relations

When the image plane projections of two objects overlap, and the visibility of one of them (object A) is completely unaffected by the other (object B), it is called a *binary occlusion* and A occludes B. If two solid objects have convex shapes, then any occlusion between them will be binary.[1]

Consider a pair of convex objects undergoing bounded motion, such as would occur between two frames in an image sequence. If the image plane

[1]Any two convex bodies can be separated by a plane which divides the viewing sphere in half, and for all view points in each half, the object it contains is completely visible.

projections of these bodies do not overlap under the allowed motion, no occlusion is possible. In this case, the bodies A and B are *disjoint*, which is written $A \equiv B$. The binary occlusion relation, *will-occlude(A,B)*, is true if A and B are not disjoint and A occludes B whenever their image plane projections overlap. This is written $A \succ B$. For example, if two occluding objects are located at distances Z_A and Z_B with respect to the camera, such that $Z_A < Z_B$ over some range of motion, then $A \succ B$.

The *will-occlude* relation describes a property of all possible occlusions of the two bodies under limited motion. For most articulated objects (see Sec. 3.4,) one of $A \equiv B$, $A \succ B$, and $B \succ A$ will be true for each pair of bodies in all configurations. Because these relations are fixed over a motion interval, they define a local occlusion invariant. The invariant is local because it only holds under bounded motions of the object around an operating point in state space. Since the occlusion relations are defined for objects with arbitrary degrees of freedom, they generalize the concept of depth sorting in constructing a layered representation. Section 3.4 discusses the construction of visibility orders for general articulated objects from the set of pairwise occlusion relations for its bodies.

3.2.2 Occlusion Relations for Revolute Joints

The on-line determination of visibility orders can be greatly simplified for an object like the hand through kinematic analysis. The first step is to define the will-occlude relation for two links connected by a revolute joint, a basic component of articulated kinematic chains. Using this definition and the kinematics of the hand, an algorithm can be derived for generating hand template visibility orders.

A revolute joint constrains two links to a single degree of freedom of relative motion. The first step in defining occlusion relations for the revolute case is to identify the fixed and moving bodies. There is a natural ordering

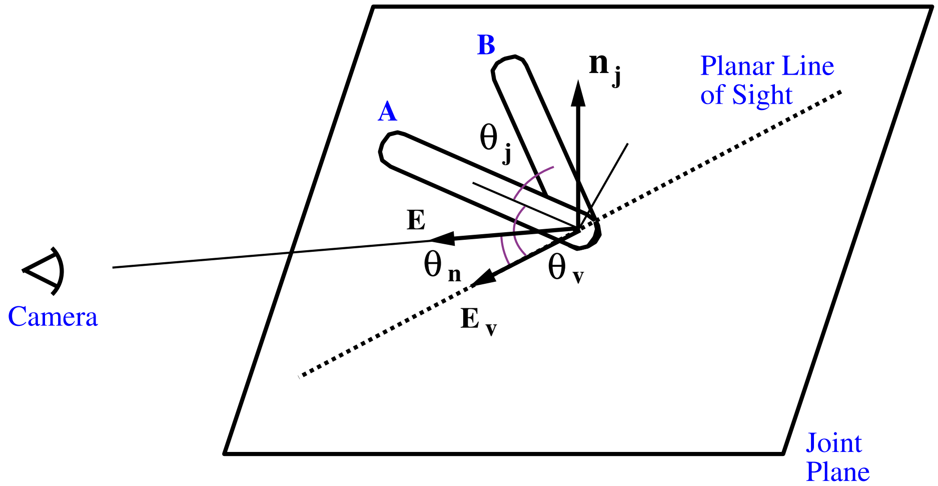


Figure 3.3: Occlusion properties of two links connected by a revolute joint.

between any two links in a kinematic chain. The link closest to the base of the chain is chosen as the fixed link, A. The joint angle, θ_j, positions link B relative to A's coordinate frame at the joint center, illustrated in Fig. 3.3. Both links lie in a *joint plane*, whose normal is defined by the joint axis, $\mathbf{n}_j$.

Since connected links overlap at their joints, there is always some degree of occlusion between adjacent links. However, for links like finger phalanges that are convex and roughly symmetric, only the visible surface along the axis of symmetry is important. Occlusion in this region is determined by the camera position relative to the joint center. In general, the camera viewpoint will lie outside the joint plane, as illustrated in Fig. 3.3. Consider, for the moment, a camera viewpoint located *in* the plane, along the planar viewing vector E_v from the joint center. Occlusion occurs whenever both links lie on the same side of the planar line of sight containing E_v. The following table summarizes the conditions for occlusion as a function of θ_j and the viewing angle θ_v, between E_v and A:

$$\begin{array}{lll} \theta_v > 0: & \theta_j \in [0, \theta_v] & \Rightarrow B \text{ occludes } A \\ & \theta_j \in [\theta_v - \pi, 0] & \Rightarrow A \text{ occludes } B \\ \theta_v < 0: & \theta_j \in [\theta_v, 0] & \Rightarrow B \text{ occludes } A \\ & \theta_j \in [0, \theta_v + \pi] & \Rightarrow A \text{ occludes } B \end{array} \tag{3.1}$$

In Fig. 3.3, $\theta_v > 0$ and $\theta_v - \pi < \theta_j < 0$, so that A is occluding B. Occlusion properties change at the boundaries of the intervals. Note that θ_j is bounded away from zero on both sides by noninterpenetration.

As the viewpoint moves out of the joint plane, the amount of occluded surface area decreases. When the general viewing vector, E, is parallel to $\mathbf{n}_j$ there is essentially no occlusion for all joint angles. E makes an angle θ_n with the joint plane, in which it has the projection E_v. It follows that any viewing direction can be represented in the joint coordinate frame by two angles: θ_v and θ_n. The occlusion conditions from Eqn. 3.1 apply only to viewpoints for which $|\theta_n| < \Delta_n$, for some fixed threshold Δ_n. For viewpoints above this threshold, the links are disjoint.

Given the state of an articulated object, Eqn. 3.1 can be applied to determine the occlusion at a revolute joint. To use this model for tracking, it must be extended to include bounded motions of the two links. Bounded change in the DOFs before link A in the kinematic chain will displace the joint coordinate frame, causing θ_n and θ_v to vary. The exact change in these angles will be a complex function of the state, but it can be approximated by restricting them to intervals, I_n and I_v, of a fixed size, centered around their current value. Bounded motion between B and A is modeled by an interval $I_j = [\theta_j^0 - \Delta_j, \theta_j^0 + \Delta_j]$, of width Δ_j containing θ_j. The intervals I_n and I_v are defined similarly. These intervals can be incorporated into Eqn. 3.1 by replacing inequalities with intersection tests. The normal, viewing, and joint angles at the current state are θ_n^0, θ_v^0, and θ_j^0, respectively. The revolute occlusion relations are

$$
\begin{array}{llll}
I_n \cap [-\Delta_n, \Delta_n] \neq \emptyset : & \theta_v^0 > 0 : & I_j \cap [0, \theta_v^0 + \Delta_v] \neq \emptyset & \Rightarrow B \succ A \\
& & I_j \cap [\theta_v^0 - \Delta_v - \pi, 0] \neq \emptyset & \Rightarrow A \succ B \\
& \theta_v^0 < 0 : & I_j \cap [\theta_v^0 - \Delta_v, 0] \neq \emptyset & \Rightarrow B \succ A \\
& & I_j \cap [0, \theta_v^0 + \Delta_v + \pi] \neq \emptyset & \Rightarrow A \succ B \\
\text{Otherwise} & & & \Rightarrow A \equiv B
\end{array} \qquad (3.2)
$$

3.2.3 Visibility Orders for Hand Templates

The kinematic properties of objects like the hand can be exploited in an algorithm for visibility ordering link templates. In this approach, templates are ordered within each finger chain using the revolute occlusion relation described above. Then the chains are compared as distinct objects, avoiding the complexity of testing each link against all the others. By exploiting the kinematic structure, the algorithm is efficient enough for on-line implementation. A more general approach to computing visibility orders from binary occlusion relations is described in Sec. 3.4.[2]

The hand consists of five planar kinematic finger chains and a rigid palm. As a result of planarity, the three joint axes in each finger are parallel and have the same joint plane. This greatly simplifies the application of revolute occlusion relations to finger ordering. A further simplification comes from the fact that all joint angles must be positive, reflecting physical limits on joint motion. As a result, each finger can be viewed as a convex planar shape. These two observations lead to a simple procedure for ordering templates within each link.

If the angle, θ_n, between the camera and the finger joint plane exceeds the threshold, Δ_n, described in Sec. 3.2.2, then the finger templates are disjoint and can be ordered arbitrarily. Otherwise, two applications of Eqn. 3.2 determine the ordering between links 1 and 2, and links 2 and 3 (see Fig. 2.5

[2]Note, however, that the revolute occlusion relation defined above applies only to pairs of links that share a joint. This definition would have to be extended to an arbitrary pair of links to meet the requirements of the general approach.

for the link numbers, which are the same for each finger.) Convexity imposes strong constraints on the global pose of the finger, making it possible to generate the entire visibility order directly from the two pairwise tests, according to the following table:

$$\begin{array}{rcl} 1 \succ 2 \ \text{ or } \ 2 \succ 3 & \Rightarrow & 1 \succ 3 \\ 2 \succ 1 \ \text{ or } \ 3 \succ 2 & \Rightarrow & 3 \succ 1 \\ 1 \equiv 2 \ \text{ and } \ 2 \equiv 3 & \Rightarrow & 1 \equiv 3 \end{array} \tag{3.3}$$

The thumb is also a planar mechanism with joint limits, and the finger template ordering rules can be applied to it without modification.

Occlusions between fingers are almost always binary. This observation simplifies visibility ordering by removing the need to consider individual templates. When the planes for two fingers are parallel, they can be ordered by distance from the camera. When the planes intersect, there are three possibilities, illustrated in Fig. 3.4 (a), (b), and (c). In (a), neither chain crosses the dividing line formed by the plane intersection. In this case, the two planes divide 3D space into four quadrants, with associated visibility orders given in the figure. In (b), one chain crosses the dividing line, but the other does not. In this case the quadrant labels are different. Note that the transition from (a) to (b) either leaves the visibility order unchanged, or changes a disjoint situation to an ordered one.

The only case where nontrivial interaction between the chains occurs is (c), where they both cross the dividing line. This case requires additional analysis within the plane of the finger. The first step is to choose the plane closest to the camera, in which the occlusion effect is most visible, and project the camera viewpoint into that plane. Due to convexity, the other chain will intersect this plane at one point. Figure 3.4 (d) shows a sample configuration of links in this case. If the line in the plane joining the projected viewpoint and the intersection point passes through the chain, then the chain comes first in the visibility order. Otherwise, the intersecting chain comes first.

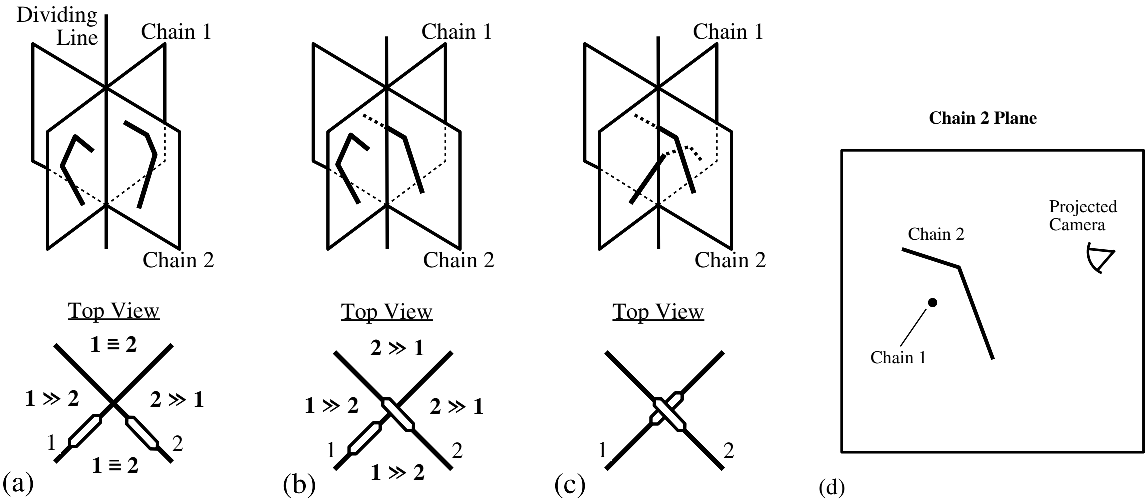


Figure 3.4: Types of intersections between two planar kinematic chains. In (a), chains are confined to separate sides of the dividing line at which their planes intersect. In (b) one chain crosses the line, and in (c) they both do. The viewpoint relative to the dividing line determines the visibility order. (d) shows the ordering test from (c) in the chain 2 plane.

In order for the plane intersection test to be valid under bounded motion, it is necessary to model the effect of chain motion within the plane and motion of the plane itself on the outcome. If the test is applied between fingers and thumb on the same hand, then palm motion will not effect the type of intersection, but may change the camera's quadrant. Since the decision hinges on whether each chain crosses the dividing line, this can be modeled by bounding the distance to the line for the closest part of each chain. The only nontrivial transition is from case (b) to (c). In this situation, a finger or thumb tip intersects the other chain's plane for the first time. The point of intersection can be predicted from the motion, or bounded by intersecting the bound on tip displacement with the plane.

Finger planes will intersect each other due to abduction. However, these planes are roughly parallel, and the intersections will almost always be of type

(a) in Fig. 3.4. As a result, there is a simple visibility ordering algorithm for the fingers: sort the anchor points for each finger based on distance to the camera along the optical axis. This determines the finger ordering. The thumb plane can intersect the finger planes in a variety of ways depending on the motion, and the intersection tests described above must be applied in this case. In most situations, the outcome of the test between the thumb and first finger can be applied to the rest of the fingers as well.

Finally, the plane of the palm sweeps out a volume in space in the direction of the camera axis. If the tip of a finger or the thumb intersects this volume, then the palm comes before that chain in the visibility order, otherwise after. A visibility order for hand templates can be constructed from the tests described above. These tests are simple to implement, making it possible to update the ordering on-line whenever a new state estimate is available. Note that the fundamental assumption in the above analysis is the planarity of the kinematic chains comprising the object. This modeling assumption is also valid for arms and legs, suggesting that the ordering tests described above could also be applied to human figures.

3.3 Estimation with Layered Templates

Using the techniques from the previous section, hand templates can be maintained in visibility order during tracking. This section describes an algorithm for registering an ordered set of overlapping templates to an input image. Tracking is achieved by applying this algorithm to each frame in a motion sequence, using the estimated state from the previous frame as the starting point for registration. Window functions are the key to registration. They model the appearance and disappearance of template pixels as a result of the image plane motion of overlapping templates. The resulting gradient-based minimization problem requires the derivation of Jacobians for layered templates, and algorithms for image segmentation. These components are

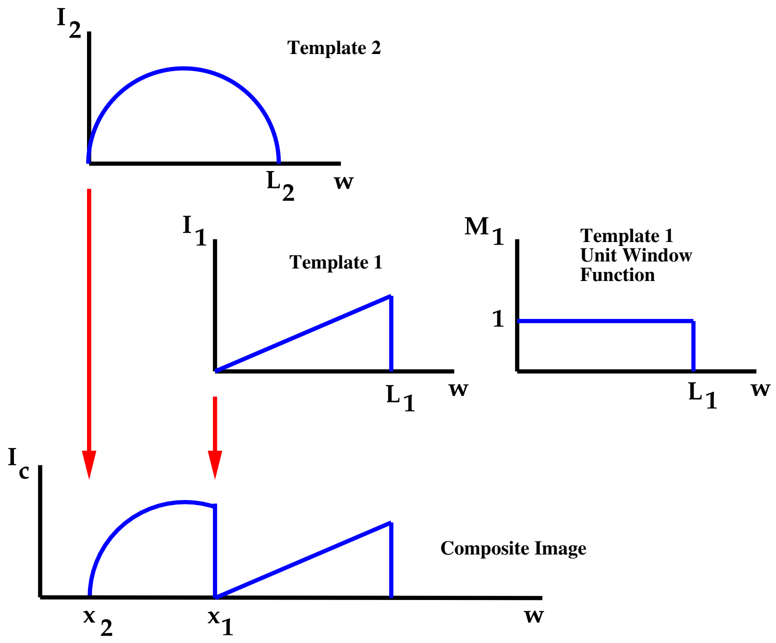


Figure 3.5: Image composition example for two 1D templates. Occlusion is modeled by the unit window function shown on the right.

discussed in detail in this section, and the complete algorithm is summarized in Sec. 4.4.1.

3.3.1 Window Functions

Given a visibility ordered set of templates, the effect of self-occlusion on the image can be modeled by *window functions*, whose position in the image is a continuous function of the state. Each template has an associated window, which masks out contributions to the image from templates below it in the visibility order. An example of a simple unit window function is given in Fig. 3.5 for the case of tracking two 1D occluding patterns with the visibility order, *Template 1* $\succ$ *Template 2*. In the forward model for this example, the two templates are combined to give a composite image:

$$I_c(x) = M_1(x - x_1)I_1(x - x_1) + [1 - M_1(x - x_1)]I_2(x - x_2) \tag{3.4}$$

where $I_{1,2}(\cdot)$ are the templates and M_1 is the window function for template 1. Given $m(\cdot, L)$, a unit window of length L for 1D images, it follows that $M_1(\cdot) = m(\cdot, L_1)$.

$I_c(\cdot)$ represents the *forward model* of the image as a function of the state. The 2D version of this function is formed by combining the deformable template model of Sec. 2.4.1 with the layered occlusion representation described above. A 2D version of Eqn. 3.4 can be written

$$I_c(\mathbf{q}, \mathbf{w}) = M_1(\mathbf{q}, \mathbf{w})I_1(\mathbf{f}_1^{-1}(\mathbf{q}, \mathbf{w})) + [1 - M_1(\mathbf{q}, \mathbf{w})]I_2(\mathbf{f}_2^{-1}(\mathbf{q}, \mathbf{w})) \ , \tag{3.5}$$

where $\mathbf{f}_{1,2}^{-1}$ are inverse deformation functions for the two templates that map from image coordinates to template coordinates as a function of the state. Since the functions $\mathbf{f}_{1,2}$ are affine in the image coordinates, their inverses are well-defined. $M_1(\mathbf{q}, \mathbf{w})$ denotes the window function for template 1, positioned in the image. It is defined for a general template, I_j, as

$$M_j(\mathbf{q}, \mathbf{w}) = m_j(\mathbf{f}_j^{-1}(\mathbf{q}, \mathbf{w})) \ , \tag{3.6}$$

where $m_j(\mathbf{s})$ is a 2D unit window in template coordinates, that is equal to one inside the template's boundary contour and zero everywhere else, as illustrated in Fig. 3.6.

The incorporation of the composite image in an SSD residual can be illustrated in the more complicated case of adding a background template, I_b, to Eqn. 3.5 obtaining

$$\begin{aligned} E(\mathbf{q}) &= \frac{1}{2}\int_I [\hat{I}(\mathbf{w}) - \hat{I}_c(\mathbf{q}, \mathbf{w})]^2 \, \mathbf{dw} \\ &= \frac{1}{2}\int_I [\hat{I}(\mathbf{w}) - M_1(\mathbf{q}, \mathbf{w})\hat{I}_1(\mathbf{f}_1^{-1}(\mathbf{q}, \mathbf{w})) - [1 - M_1(\mathbf{q}, \mathbf{w})] \times \\ &\quad \{M_2(\mathbf{q}, \mathbf{w})\hat{I}_2(\mathbf{f}_2^{-1}(\mathbf{q}, \mathbf{w})) + [1 - M_2(\mathbf{q}, \mathbf{w})]\hat{I}_b(\mathbf{w})\}]^2 \, \mathbf{dw} \ , \end{aligned} \tag{3.7}$$

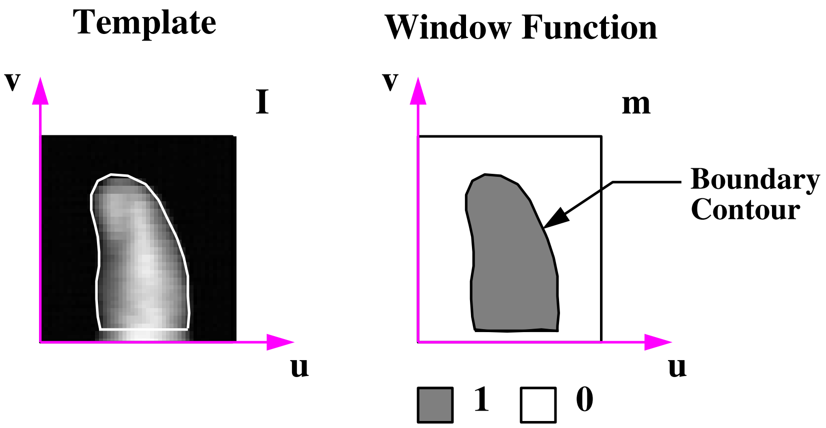


Figure 3.6: A template and its associated unit window function are illustrated for the finger tip.

where $M_{1,2}$ are window functions for the two filtered templates, $\hat{I}_{1,2}$, and $\hat{I}_b$ is the filtered background.

A set of templates in visibility order results in a recursive hierarchy of window functions, called the *window tree.* This tree is illustrated in Fig. 3.7 (also see [1], Fig. 2.) The path from the root of the tree to a leaf template captures the composition of masks that determine the template's visibility. Incorporation of the window tree into the tracking algorithm is discussed in Sec. 3.3.3. The window tree gives the structure of $\hat{I}_c(\cdot)$. In the case of Eqn. 3.7, $\hat{I}_c$ is formed by recursively descending a window tree of depth two. If all of its terms are multiplied out, aside from the $(1 - M_i)$ terms, the resulting sum consists of the paths from the root of the window tree to its three leaves.

The window functions in this section are based on the image compositing process described in [1, 67]. The window functions incorporate self-occlusion properties into the deformable template framework of Sec. 2.4.1.

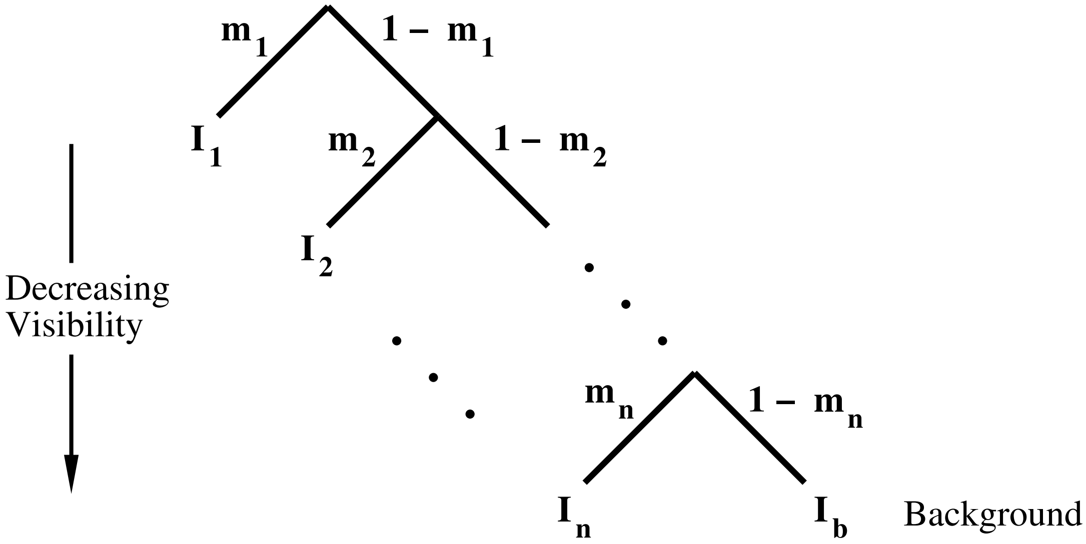


Figure 3.7: Tree of window functions generated by a set of templates, $I_1, I_2, \ldots, I_n$, in visibility order. I_b is the background template.

3.3.2 Minimization of Layered Template Error

The gradient descent tracking algorithm described in Sec. 2.4.3 can be applied to error functions such as Eqn. 3.7, registering the layered templates to an input image. Minimizing the error generates a segmentation of the image, which assigns each pixel to one of three templates. For each iteration of the tracking algorithm, the residual and its Jacobian will be evaluated at discrete pixel locations, summed over the input image. the following three steps must be performed at each pixel for template registration:

1. Assign the pixel to one of the hand templates or the background and determine its template coordinates, based on the current state estimate.

2. Compute the residual error between the input image pixel and its associated template pixel.

3. Compute the residual Jacobian from the composite image.

Two algorithms for obtaining the image segmentation in step 1 are described in Sec. 3.3.4. Once an image pixel has been assigned to a template, its corresponding template pixel is determined by the inverse deformation function, and the residual follows easily. The remaining step is the computation of the residual Jacobian.

3.3.3 Residual Jacobian Computation

Suppose an image pixel, $\mathbf{w}'$, originates from template I_j at template coordinate $\mathbf{s}_j$. Furthermore, let $\mathbf{p}'$ denote the 3D position of $\mathbf{s}_j$ in camera coordinates, as determined by the position of the template plane. The pixel at $\mathbf{w}'$ makes the following contribution to the residual

$$R' = \hat{I}(\mathbf{w}') - \hat{I}_c(\mathbf{q}, \mathbf{w}') \ . \tag{3.8}$$

There are two possible cases for the pixel $\mathbf{w}'$: either it is in the *interior* of I_j, or it is on the *boundary* of I_j and a second template I_k, where $j < k$. The Jacobian calculations in these cases rely on two assumptions: that the window functions are constant in the template interiors and fall to zero at their boundaries, and that the bodies are opaque, so that no more than two templates can effect a pixel value simultaneously.

For the interior pixel case, the ordered templates can be divided into a group, $\{I_1, \ldots, I_{j-1}\}$, that occludes I_j and a group, $\{I_{j+1}, \ldots, I_n\}$ that is occluded by it, as illustrated in Fig. 3.8 (a). Window functions and their gradients for the occluding templates are zero at $\mathbf{w}'$, leading to the simplification

$$\begin{aligned} R' &= \hat{I}(\mathbf{w}') - [1 - M_1(\mathbf{q}, \mathbf{w}')][1 - M_2(\mathbf{q}, \mathbf{w}')] \cdots [1 - M_{j-1}(\mathbf{q}, \mathbf{w}')] \times \\ & \quad M_j(\mathbf{q}, \mathbf{w}')\hat{I}_j(\mathbf{f}_j^{-1}(\mathbf{q}, \mathbf{s}')) - I_c^- \ . \end{aligned} \tag{3.9}$$

The second term in Eqn. 3.9 is produced by descending the window tree to node I_j. The gradients of its window functions are zero at $\mathbf{w}'$, so its only

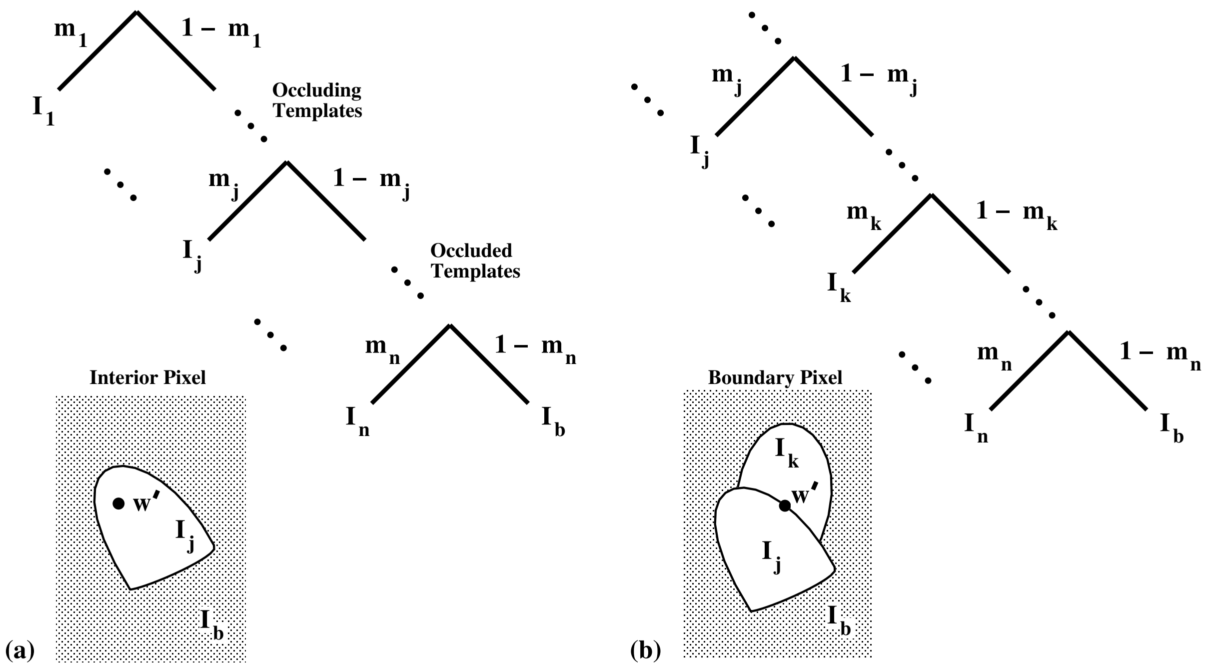


Figure 3.8: (a) A pixel $\mathbf{w}'$ in the interior of template I_j and the associated window tree, and (b) the same for a boundary pixel.

derivative contribution comes from $\hat{I}_j$. I_c^- contains the occluded templates' contributions to I_c. Its derivative is zero, as all of its terms contain $[1 - M_j(\mathbf{q}, \mathbf{w}')]$, which vanishes at $\mathbf{w}'$ along with its derivative.

Changing variables to template coordinates and differentiating Eqn. 3.9 gives the Jacobian contribution of an interior pixel:

$$\mathbf{J}_j^I(\mathrm{s}') = \frac{\partial}{\partial \mathbf{q}}[\hat{I}(\mathbf{f}_j(\mathbf{q}, \mathrm{s}')) - \hat{I}_j(\mathrm{s}')] = \frac{\partial \mathbf{f}_j}{\partial \mathbf{q}} \frac{\partial \hat{I}}{\partial \mathbf{w}} \quad . \tag{3.10}$$

This is the usual SSD Jacobian component, from Eqn. 2.13, that would be present if there was only a single template and no window functions. Its derivation and geometric interpretation are given in Sec. 2.4.4.

Now consider the boundary pixel case, illustrated in Fig. 3.8 (b). The point $\mathbf{w}'$ is on the boundary of m_j at coordinate $\mathbf{s}_j$, and lies in the interior of m_k at $\mathbf{s}_k$. This implies that the templates between j and k in the tree vanish at $\mathbf{w}'$ along with their derivatives. Furthermore, as in the interior case, the templates occluding I_j and occluded by I_k make no contribution to the Jacobian. This results in the simplified residual,

$$\begin{aligned} R' &= \hat{I}(\mathbf{w}') - [1 - M_1(\mathbf{q}, \mathbf{w}')] \cdots M_j(\mathbf{q}, \mathbf{w}') \hat{I}_j(\mathbf{f}_j(\mathbf{q}, \mathbf{s}_j)) - \\ & \quad [1 - M_1(\mathbf{q}, \mathbf{w}')] \cdots [1 - M_j(\mathbf{q}, \mathbf{w}')] \cdots M_k(\mathbf{q}, \mathbf{w}') \hat{I}_k(\mathbf{f}_k(\mathbf{q}, \mathbf{s}_k)) \end{aligned} \tag{3.11}$$

Since this is in the form of Eqn. 3.9, it has an interior Jacobian component as before. Window function gradients from the last two terms yield an additional component. In these terms, only $M_j(\mathbf{q}, \mathbf{w}')$ has a nonzero derivative at $\mathbf{w}'$. Substituting Eqn. 3.6 for M_j and differentiating yields the boundary Jacobian

$$\mathbf{J}^B_{j,k}(\mathbf{s}') = [I_k(\mathbf{s}_k) - I_j(\mathbf{s}_j)] \frac{\partial \mathbf{f}_j^{-1}}{\partial \mathbf{q}} \frac{\partial m_j}{\partial \mathbf{s}} \ , \tag{3.12}$$

This boundary component captures the effect of occlusion in covering and revealing pixels as the state changes.

The above discussion shows that the residual Jacobian for a template has two basic types of components: region contributions from Eqn. 3.10 and boundary contributions from Eqn. 3.12. This suggests a simple algorithm for Jacobian computation:

1. Scan the segmented image and compute the region contribution to the Jacobian at each visible pixel, using Eqn. 3.10.

2. Scan the discretized boundary of each template. If a boundary point is visible, identify the template it is occluding and compute the boundary Jacobian term from Eqn. 3.12.

The above algorithm, along with the segmentation algorithm described in the next section, forms the basis for gradient-based local tracking.

3.3.4 Algorithms for Image Segmentation

Each pixel in the input image must be assigned to a template in order to compute its contribution to the gradient. This segmentation problem is closely related to the visible surface determination problem in computer graphics: Given a set of polygons in camera coordinates, identify and scan-convert[3] the parts that are visible. Through this analogy, segmentation algorithms can be divided into two classes: list-priority and scan-line (see [18], Sec. 15.11.)

Templates are scanned sequentially in visibility order in the list-priority approach, and the most visible template is converted last. Each template is scan-converted independently, and its pixels in the input image are labeled. The visibility ordering ensures that each pixel is correctly labeled at the end of this first stage. The labeled pixels are then rescanned in a second stage to compute the Jacobian, as discussed in the previous section. Pixels contained by overlapping templates are processed multiple times, but template conversion and pixel labeling is simple and fast. This is the segmentation algorithm used in the experiments of Chpt. 4. Because of the visibility order, this approach is superior to the standard computer graphics depth sorting algorithm, which often splits polygons that can be correctly ordered ([18], Fig. 15.27.)

In contrast, scan-line algorithms sort the template edges on x and y, and scan the image one line at a time. When templates overlap, the visibility order determines the pixel assignment, and coherence is used to avoid unnecessary comparisons. The binary occlusion assumption plays the same role for coherence as polygon nonpenetration in the graphics case. Scan-line algorithms are more efficient than list-priority algorithms: Each pixel is processed once, avoiding redundant calculations. The Jacobian can be computed in one pass, avoiding a labeling stage. They are, however, more complicated

[3]In scan-conversion, polygons (specified by a set of vertices) are mapped into their component pixels in the frame buffer.

to implement.

The scan-line approach can also be used in situations where a visibility order is not available. In this case, the depth at each pixel in the scan-line determines the template order. In this approach, template order is computed in conjunction with segmentation. However, preservation of the ordering under bounded motion is not guaranteed, and this approach may require a prohibitively high sampling rate to work in practice. Moreover, in the case where the template ordering is fixed for a number of estimation steps, this version of the scan-line algorithm is inefficient, as it recomputes the visibility order each time.

3.4 The Existence of Visibility Orders

The existence of a visibility ordering algorithm for the hand raises the question of what other objects can be treated under the same framework. This section develops general existence conditions for visibility orders. These results apply to a multibody system with arbitrary degrees of freedom.

3.4.1 Existence Conditions for Occlusion Relations

A multi-body system has a local occlusion invariant if, for a given bounded motion, one of $A \equiv B$, $A \succ B$, $B \succ A$ is true for each pair of bodies, A and B. A visibility order can be constructed in this case, as shown in the next section. Bounded relative motion between the two bodies is modeled by $M(B)$, the union of all possible spatial positions of B with respect to A's coordinate frame.[4] In general, $M(B)$ will not be convex. But its convex hull, $CH[M(B)]$, can be partitioned from A by a separating plane if the occlusion is *unambiguous*. This is illustrated in Fig. 3.9 (a) for two 2D bodies viewed

[4]The spatial position of each body is defined with respect to the world coordinate frame. Above, the reference frame is shifted to A for convenience.

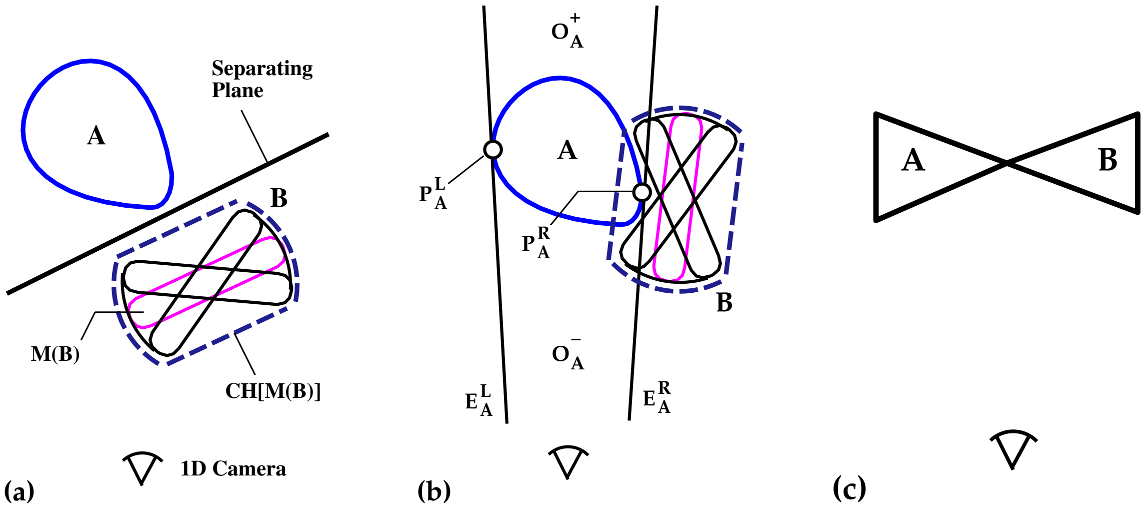


Figure 3.9: Occlusion relations for 2D objects viewed by a 1D camera. (a) Sufficient conditions for $A \succ B$, (b) geometric definition of occlusion ambiguity, and (c) degenerate configuration of two planar objects in point contact. No nonzero bound on relative translation can remove the occlusion ambiguity.

by a 1D camera. The relative motion in this case is rotation of B. The partition creates two half-spaces. If the image plane projections of A and the *motion image* of B, $CH[M(B)]$, don't overlap, $A \equiv B$. If they do overlap, the object in the half-space containing the camera will occlude the other object. In figure (a), $B \succ A$.

The case of occlusion ambiguity is illustrated in Fig. 3.9 (b), using the same two bodies. For this configuration, it is impossible to predict the occluder under the given motion bound. Ambiguity arises when $CH[M(B)]$ intersects the occluding limb of A. Referring to the figure, let $E_A^{L,R}$ denote the pair of line-of-sight tangents to A, with E_A^R closest to B. The points of contact, $P_A^{L,R}$, are the occluding limbs (in 3D this is a curve in the surface of A). The pair of tangents bound a region of space, O_A (a tangent cone in 3D,) which contains A and the camera viewpoint. O_A is divided into occluding and occluded regions, labeled O_A^+ and O_A^-. Occlusion ambiguity arises when $M(B)$ has a nonzero intersection with both regions. In this case, $CH[M(B)]$ intersects A and contains P_A^R, and both binary occlusion outcomes are possible. In general, the likelihood of an occlusion ambiguity decreases with the motion bound, but it can't be eliminated altogether, as figure (c) demonstrates.

When they exist, the set of ambiguous configurations will occupy a small subspace of the total configuration space, as they depend on a special combination of spatial proximity and viewing angle. An example of an ambiguous hand configuration is the "stop" gesture, with the hand held flat, fingers pressed together, and palm facing the camera. In this pose, rotation around the vertical axis changes the visibility order of the fingers. In a specific case like the hand, knowledge about ambiguous configurations can be used to aid tracking. Simple velocity-based prediction, for example, could be used to correctly interpret ambiguous cases. In general, high frame rates reduce the danger of an incorrect occlusion hypothesis, by making the mislabeled region of pixels as small as possible.

3.4.2 Visibility Ordering and Occlusion Graphs

The occlusion relations for a multi-body system can be represented by a directed *occlusion graph*. The graph is a pair (V, E), where the vertex set V contains all of the bodies. To construct the edge set, E, consider all pairs $x, y \in V$. Since there are no occlusion ambiguities, one of $x \equiv y$, $x \succ y$, or $y \succ x$ must be true. In the first case no edge is added, while the other two cases add directed edges (x, y) and (y, x) respectively. Consider the collection of 2D rigid bodies viewed by a 1D camera illustrated in Fig. 3.10. Figure 3.11 (a) shows the occlusion graph for the system under bounded translations in

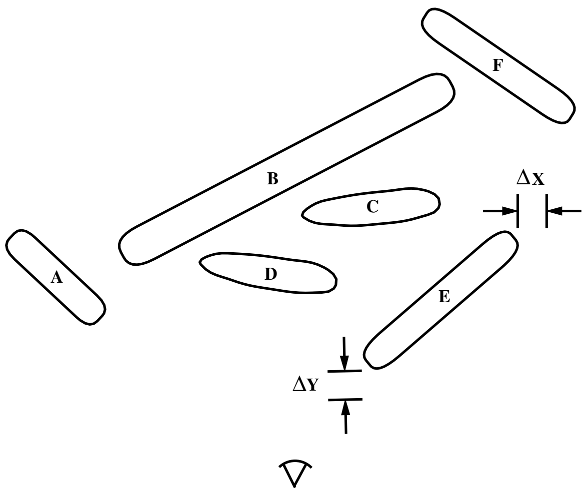


Figure 3.10: A collection of 2D rigid bodies under bounded translational motion relative to a 1D camera. Each body can translate by ΔX and ΔY, as shown for body E.

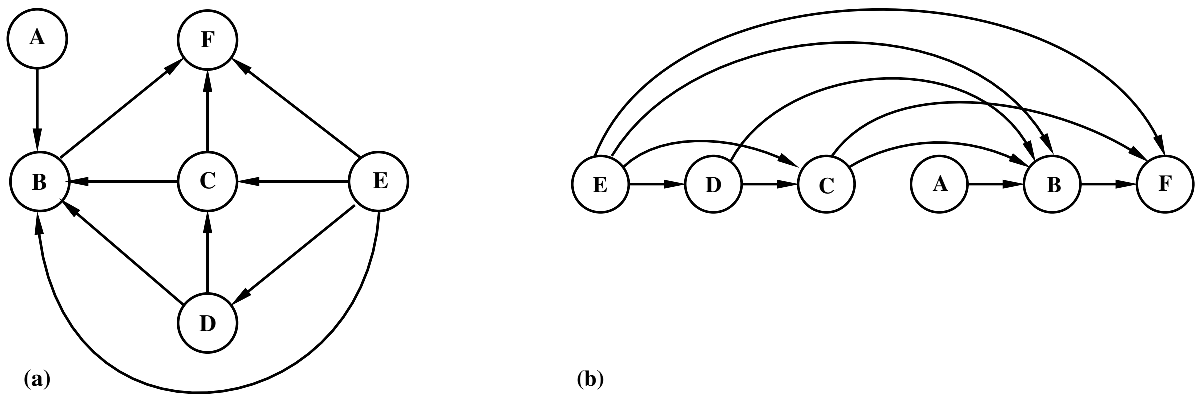


Figure 3.11: (a) Occlusion graph for the mechanism in Fig. 3.10, and (b) the visibility order produced by sorting the graph.

the plane.

When the object configuration admits a visibility ordering, it can be obtained by searching the occlusion graph. A configuration that can't be so ordered is illustrated in Fig. 3.12. In general, the occlusion graph must be *acyclic* to induce a natural order on the set of objects. The presence of occlusion cycles is fairly unusual, at least for convex bodies, as it involves a special arrangement of spacing and orientation. Cycles don't occur naturally in hand or body configurations, for example.

Figure 3.12: (a) A configuration of three objects and (b) its associated cyclic occlusion graph.

When the occlusion graph is acyclic, it can be topologically sorted by depth-first search [10] to produce a *visibility ordering.* Figure 3.11(b) shows the ordering produced by the sample occlusion graph. The sorted graph has the property that all edges are directed left to right. Taking the vertices in that order guarantees that no object will be occluded by an object that follows it in the list.

These results give sufficient conditions for the existence of a visibility ordering for an arbitrary object. Existence hinges primarily on the absence of occlusion ambiguities, which is determined by the relative motion and the temporal sampling rate. These results are useful in identifying the most likely configurations for occlusion ambiguities in a known object.

Looking beyond model-based tracking, there is increasing interest in layered representations for computer vision, because of their potential to simplify the 3D description of the world. Recently, several algorithms have been proposed for building layered descriptions of a scene from a single image or a motion sequence [41, 11, 67]. The results in this section provide general conditions under which a layered representation could be expected to exist, for a given type of moving object.

3.4.3 Occlusion Events and Global Models

The occlusion graph for an object is a function of its state. A discrete change in the topology of the graph can be viewed as an *occlusion event*, analogous to the visual events introduced by Koenderink and Van Doorn [31]. These events partition the configuration (state) space into hypervolumes over which the occlusion graph is constant. The state space partition is called the *occlusion meta-graph* for the object. The state partition in Fig. 3.2 can now be recognized as the meta-graph for the two finger model. Moreover, the visibility ordering rules for the hand described in Sec. 3.2.3 are, in fact, testing for occlusion events. These tests can be done efficiently in Cartesian space, in spite of the large number of DOFs, by exploiting the kinematic model.

The construction of an occlusion meta-graph for a two link planar mechanism is illustrated in Fig. 3.13. The 2 DOF state space is partitioned into three types of regions for which the occlusion graph is constant. The values of θ_1 can wrap around from $-\pi$ to π, but θ_2 is bounded away from both extremes, due to noninterpenetration. Each state is restricted to an interval of size $2\Delta\theta$ between frames. The construction technique is analogous to obstacle growing in the configuration space approach to manipulator path planning [8]. In general, the hypervolumes will be n dimensional regions bounded by curved surfaces.

The two link meta-graph shares with Fig. 3.2 the property that trajec-

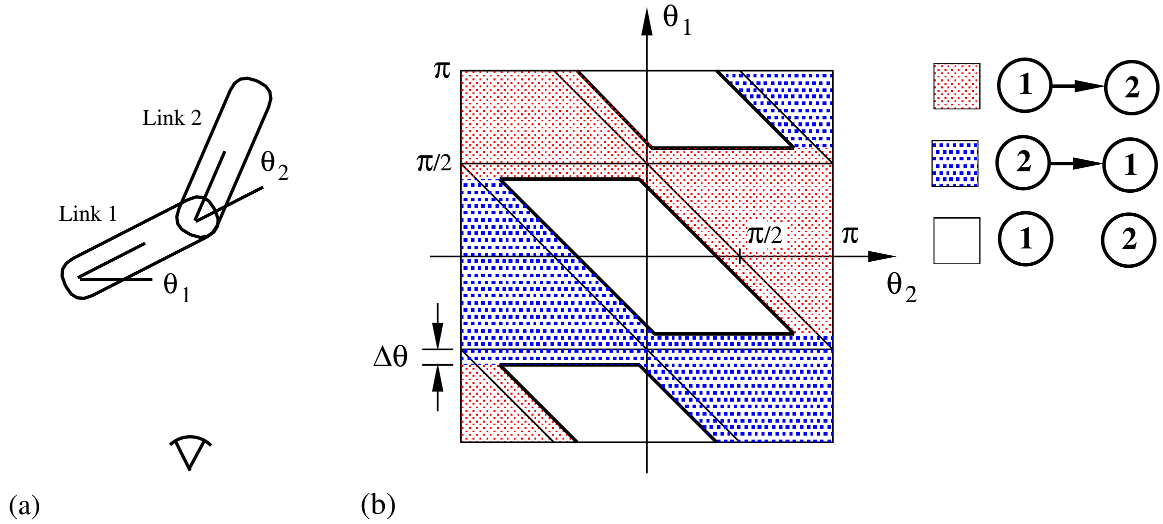


Figure 3.13: (a) Two link mechanism in 2D, and (b) associated occlusion meta-graph.

tories connecting the two occluding regions always pass through the disjoint region. Note that these occlusion events *do not* imply that an occlusion has taken place, but rather that the roles of the bodies involved in occlusion have changed. Occlusion ambiguities can also be identified in this representation as points where a square window of size $2\Delta\theta$ intersects two occluding regions. Since the left and right edges of the graph are excluded due to noninterpenetration, there are no occlusion ambiguities. This example illustrates the connection between the existence conditions for a visibility order and its use in a tracking algorithm. The explicit computation of meta-graphs for arbitrary moving bodies becomes challenging for even a few DOFs, making kinematic modeling essential in practice.

3.5 Discussion

The self-occlusion of articulated objects can be modeled by a layered template representation, which is updated over time by means of a kinematic model. Layered representations are constructed from visibility ordered templates, and practical ordering algorithms can be obtained from the object kinematics.

Window functions mask templates on the basis of the visibility order, leading to a direct minimization-based solution to self-occluding motion. By analyzing the structure of the window functions in the objective function, a simple algorithm for Jacobian computation is obtained.

The existence properties of the occlusion representation depend on the lack of occlusion ambiguities between pairs of rigid links. These existence results establish the applicability of the tracking framework to arbitrary articulated objects.

Chapter 4

Hand Tracking Experiments

The kinematic models and tracking algorithms presented in Chpt. 2 were used to construct a real-time articulated object tracking system, called *DigitEyes*. Two hand tracking experiments using the *DigitEyes* system are reported in this chapter, along with an off-line experiment in tracking self-occluding motion. These experiments validate the model-based tracking framework presented above, and demonstrate the potential of 3D human sensing, at frame rates of up to 10 Hz, using currently available computer hardware. All of these results are the first of their kind, demonstrating real-time high DOF tracking of hands using natural imagery, and with nontrivial amounts of self-occlusion.

The chapter begins with a description of the experimental objectives of the *DigitEyes* implementation, followed by a detailed discussion of its software architecture. This architecture made it possible to construct on-line and off-line versions of the system from the same basic set of modules. Next, the computational cost of hand tracking in *DigitEyes* is analyzed, and the special hardware used to achieve real-time performance is discussed. Following this, the first real-time experiment, tracking a 27 DOF hand model with two cameras, is presented. This result constitutes the first experimental demonstration of 3D high DOF tracking of unmarked, unadorned hands. In the second experiment, a simple 3D cursor user-interface was developed and tested using the *DigitEyes* system. Finally, experimental results are given for off-line tracking of two fingers in the presence of self-occlusions.

4.1 Experimental Objectives

The tracking experiments in this chapter were designed with two purposes in mind. The first was to validate the model-based tracking framework described in Chpts. 2 and 3 on real hand images. The *DigitEyes* real-time tracking system was indispensable in this task, as it made it possible to conduct experiments with millions of images in a reasonable amount of time.[1] Real-time hand tracking with one and two cameras provided experimental validation of the kinematic models and estimation framework, and a separate off-line tracking experiment tested the additional representations for self-occlusion. The second experimental goal was to evaluate the potential usefulness of vision-based hand tracking in applications. This was accomplished by applying the *DigitEyes* system to the 3D cursor user-interface problem, described in Sec. 4.3.4.

The two types of errors that are important in tracking are *residual errors* and *state errors*. Residual errors measure the difference between the input image and the image predicted by the state estimate, acting through the model. The residuals are defined mathematically by Eqns. 2.10, 2.17, and 2.19, and the state estimate minimizes them by definition. Backprojecting the estimated hand pose onto its associated image makes it possible to visually assess the degree of fit between the estimate and the measurement. A qualitative visual agreement between the back-projected model and the image is the most basic requirement for tracking performance, and is the basis

[1]The *DigitEyes* system was in daily operation for over a year. Assuming that the system ran for an hour each weekday at a sampling rate of 10 Hz, it follows that approximately 10 million images were processed!

for experimental validation in this thesis.

State error, on the other hand, is the difference between the tracker output and the ground truth for the physical system being tracked. It is synonymous with the *accuracy* of the tracker. Determining ground truth motion for a complicated object like the hand is extremely difficult, as the lack of a good noninvasive sensor is one of the motivations of this work. Although state variables such as joint angles provide a compact description of hand motion, obtaining ground truth for them is probably impractical. The most promising ground truth measure, discussed in more detail in Sec. 6, is to attach LEDs to the hand in a way that doesn't interfere with *DigitEyes*, and measure their absolute spatial position using stereo.

Track life [34] is a dynamic property of the estimator closely linked to the residual error. It refers to the length of time (number of frames) that the tracker remains on target, as measured by its ability to extract useful measurements from each image. All of the tracking algorithms in this thesis use the projected kinematic model to segment the input image into features or templates. Track loss occurs if the residual error grows so large that the model no longer projects to the correct parts of the image. When track loss occurs, the estimator loses correspondence with the image and the state error can grow arbitrarily large. However, the residual error in each frame may be small enough to prevent track loss, and yet the state error may remain large due to model error or singularities. Thus track life, like the residual error, is a weaker criteria than tracking accuracy.

4.2 Software Architecture

The *DigitEyes* system was designed with a modular software architecture, that makes it possible to quickly assemble individualized tracking systems for both videotaped and real-time imagery, using both templates and point and line features. The system runs on Sun and SGI workstations, as well

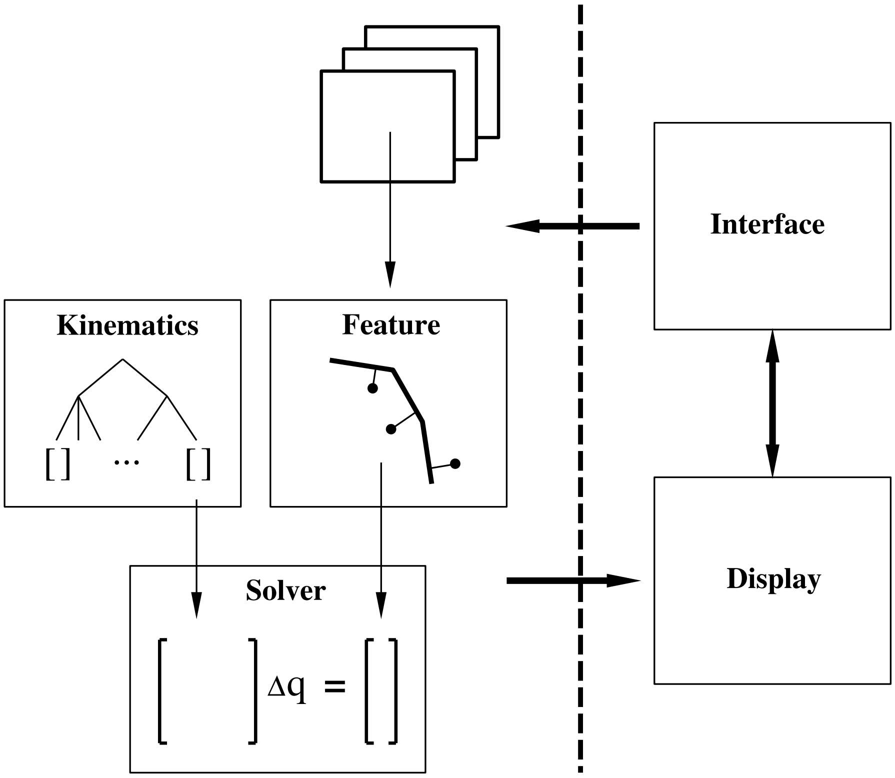


Figure 4.1: Software architecture for tracking system.

as on a special board for real-time image processing, called IC40. All of the software is written in C. The major components of the software architecture are shown in Fig. 4.1. The interface and display modules were written for an SGI Indigo 2 workstation, using GL and the FORMS user-interface toolkit. The solver and image processing components will run on all three hardware platforms.

The kinematics module is the heart of the system. Its primary data structure is a tree of link frames connected by kinematic transforms. The tree structure captures the topology of the kinematic model, and represents the transformations between the links, along with their kinematic parameters, features, and shape models. The tree is constructed automatically from an initialization file, such as the one in Appendix A, and is made up of base and chain nodes. Base nodes, such as the palm frame in the hand model, have a spatial transform and multiple children. Chain nodes have Denavit-Hartenberg transforms and a single child. Arbitrary branched, open, kinematic chains can be constructed from these two elements. Each node also contains a set of kinematic parameters, divided into state variables and fixed parameters. In a static node, all of the parameters are fixed. Active base nodes have seven state variables and active chain nodes have one. Each node may contain geometry, in which case it has both feature points and a polygonal solid model defined with respect to a shape frame.

Functions in the kinematics module descend the link tree recursively, updating the spatial position of the link and shape frames, and computing Jacobians with respect to the active variables. The output of this positioning operation is used in two ways. First, the Jacobian matrix used for estimation is built from columns distributed through the link tree. It combines with the feature residuals to form a linear system. Second, for display purposes, the positioned shape models can be rendered on an SGI workstation from a user-controlled viewpoint.

There are two types of feature modules that interface with the same kinematics module. For tracking experiments with point and line features, a single 3D feature point in each frame combines with the point and line residuals. The Jacobian matrix is constructed from the contributions of each of these points. For experiments with templates, points sampled from the template plane form the Jacobian that is used during gradient descent. The same basic software for computing point Jacobians is used in both cases.

The interface gives the user control over the display of the estimated model. Since each shape frame is positioned using the estimated state, rendering these shapes on a graphics workstation gives visual feedback of the estimator's performance. Models can be rendered from the same viewpoint as the calibrated camera, or from a viewpoint specified interactively by the user. The 3D cues provided by the shaded model, rendered from the calibrated camera viewpoint, make it possible to visually gauge registration and state errors simultaneously. Additional control over the solver and image processing is available in the interactive version of the system. The interface to the interactive system forms the basis of the 3D cursor application described in Sec. 4.3.4.

4.3 Real-Time Hand Tracking

The *DigitEyes* system is the first real-time 3D hand tracking system based on video images of unmarked, unadorned hands. Its successful performance can be attributed to two factors: the use of kinematic models to constrain image interpretation and ameliorate the effects of noise, and the use of a high image sampling rate to minimize the size of the search space, and make linearized LS methods feasible. The achievement of high image sampling rates is one of the most challenging system-level issues in constructing a real-time vision-based tracking system [2]. Its feasibility depends on two factors: the computational requirements of the estimation problem, and the delay involved in getting the images into processor memory. These issues are taken up in the next two sections. They are followed by experimental real-time tracking results for a full hand model using two cameras, and a 3D cursor user-interface using a single camera.

4.3.1 The *DigitEyes* System

The *DigitEyes* real-time tracking system is based on the feature alignment approach of Sec. 2.5. Point and line features have two computational advantages that make real-time tracking possible on a conventional microprocessor:

- They can be detected by searching along lines in the image, removing the quadratic cost of area-based image processing. This eliminates the need for special hardware to do correlations, for example.

- Each feature contributes one number to the residual vector, and a column to the Jacobian. This leads to small matrices which can be processed quickly.

This section describes the hardware implementation of the *DigitEyes* system and discusses its computational requirements.

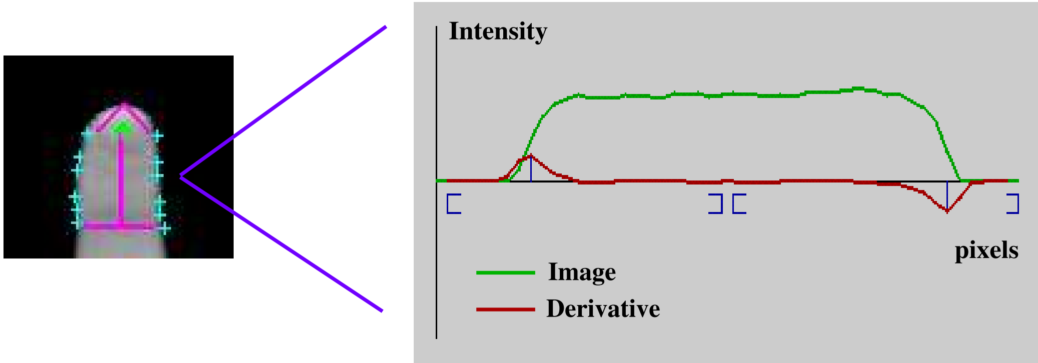


Figure 4.2: A single link tracker is shown along with its detected boundary points. One slice through the finger image of a finger is also depicted. Peaks in the derivative give the edge locations.

Fast Feature Detection

A fast feature detection algorithm was developed for images without significant amounts of self-occlusion. It is based on searching images along *slices*, lines that are perpendicular to the projected model cylinder axis. As a result of the high sampling rate, the actual finger phalange position in the image will be close to the model projection, and will be intersected by several slices. For each slice, the derivative of the 1D image profile is computed. Peaks in the derivative with the correct sign correspond to the intersection of the slice with the finger silhouette. The extracted intensity profile and peak locations for a single slice are illustrated in Fig. 4.2. Line fitting to each set of two or more detected intersections produces the feature for the link. The residual follows as the perpendicular distance from the detected feature line to a point on the base of the projected axis. If only one silhouette line is detected for a given link, the cylinder radius can be used to extrapolate the axis line location. Currently, the length of the slices (search window) is fixed by hand. Finger tip positions are measured through a similar procedure.

Computational Requirements

The cost of computing the forward kinematics, residual Jacobian, and state estimate determine the processing requirements for hand tracking. The forward kinematics computation is a sequence of matrix multiplications whose cost is determined by the kinematic topology. The computational costs of the residual Jacobian and state estimate are a function of the size of the feature and state spaces. They consist of the Jacobian matrix computation, using the technique of Sec. 2.4.4, and a linear system solution.

Component	*Time* (ms/iter.)	*Details*
Overlay Display	10.0	
Forward Kinematics	18.0	
Feature Detection	46.2	2.46 ms/link feature 1.85 ms/tip feature
State Estimation	72.0	Jacobian: 35 ms Linear Solve: 37 ms
Total Time	146.0	

Table 4.1: Computational cost (measured in milliseconds) associated with the main components of hand tracking for a full hand model. Overlay display refers to drawing model backprojections as overlays on live video.

Table 4.1 shows the average computation time for the components of the hand tracking system described above. These costs were measured for a full hand model of 36 link frames, 28 states, and 35 residuals, running on a 68040 CPU (the IC40 board described below.) The measurements were obtained by timing with a stopwatch. The total computation time requirements for a single iteration of the estimator, 146 ms, lead to a sampling rate of 6.66 Hz for the full hand. For contrast, the required computation time was also measured for a 6 DOF hand model, in which the palm pose was estimated using measurements from three fingers. In this case, the total cost was 67 ms/iter., for an sampling rate of nearly 15 Hz.

Hardware Architecture

Most modern workstations have the computational power required for real-time hand tracking, as Table 4.1 illustrates. What workstations lack, however, is the ability to transfer images into working memory at high speeds, due to the limitations of system bus bandwidths. While this will eventually improve, some specialized hardware is currently required to reduce image transfer time.

The *DigitEyes* system is built around a special board for real-time image processing, called IC40, manufactured by Eltec, Inc. Each IC40 board contains a 68040 CPU, 5 MB of dual-ported RAM, a digitizer, and a video generator. The key feature of this system is the on-board digitizer, which can write directly to CPU memory, thereby removing the bus bottleneck present in most workstation-based systems.[2] The IC40 can deliver digitized images to the processor memory at video rate with no computational overhead. Another important attribute of the IC40 is its video generator, which is used to overlay backprojections of the estimated hand configuration on the input video signal. The overlay makes possible on-line visual assessment of the

[2]I am grateful to Omead Amidi and Yuji Mesaki for their help in obtaining the IC40 and making it operational.

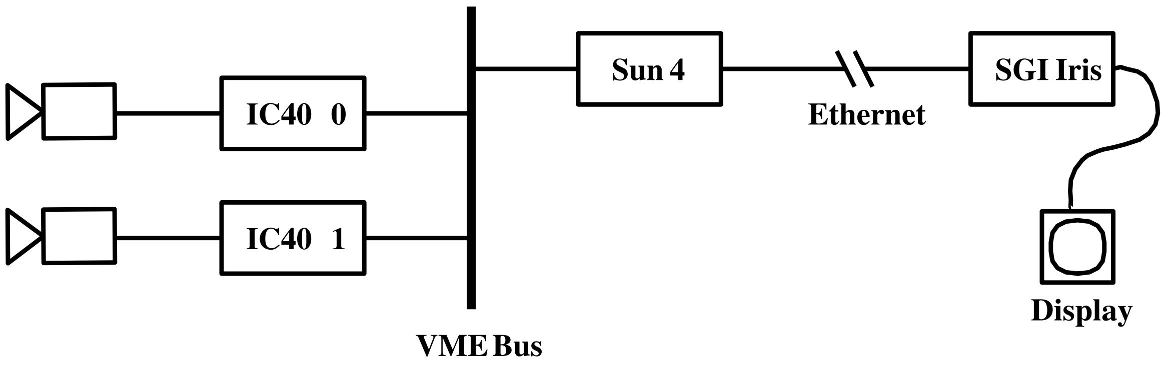


Figure 4.3: The hardware architecture for the stereo version of the *DigitEyes* hand tracking system.

quality of the model registration. Ordinary C code is cross-compiled using gcc on a Sun, and down-loaded to the board for execution. The IC40 does not run any operating system in the *DigitEyes* implementation.

In the single camera version of the system, all image processing and state estimation is done on the IC40 board, and state estimates are communicated to a Sun workstation over the VME bus. The Sun passes the estimated states to a Silicon Graphics Indigo 2 workstation through a TCP/IP connection. The Indigo 2 asynchronously renders and displays the model using the estimated state. The overall system organization is shown in Fig. 4.3.

In the stereo implementation, there is an IC40 board for each camera. The total computation is divided into two parts: feature extraction and state estimation. Feature extraction is done in parallel by each board, then the extracted features are passed over the VME bus to the Sun workstation. Both IC40 boards are memory mapped on the Sun, and a simple semaphore is used to synchronize feature acquisition between them. A solver module running on the Sun combines the two feature vectors, as described in Sec. 2.5.5, and solves the resulting linear system to obtain the state estimate. Each board has its own camera model, and uses it to compute its own forward kinematics. The estimated state is passed back to each board at the end

Figure 4.4: Experimental test bed for the *DigitEyes* system.

of the estimation cycle, and is used to reposition the feature trackers. The experimental testbed for hand tracking is depicted in Fig. 4.4.

4.3.2 Algorithm Summary

The feature alignment-based tracking algorithm described in Sec. 2.5 is the basis for the *DigitEyes* real-time tracking system. This section summarizes the main steps in the algorithm and its fixed parameters, along with error sources that impact tracking performance.

Table 4.2 summarizes the fixed parameters in the feature alignment tracking algorithm. These model and camera parameters are determined through the calibration process of Secs. 2.2.3 and 2.3. The initial state is set for each application and the user is required to place their hand in the known configuration prior to tracking. The sampling rate is a function of the complexity of the model, and will be described in more detail below. The weights for the Gauss-Newton algorithm (see Sec. 2.5.3) were set empirically and used in all of the experiments in Sec. 4.3.

Parameters	Description
Camera Model	11 extrinsic (pose) and intrinsic (image scale and origin) parameters
Kinematic Model	Joint axes, link lengths, and anchor points
Initial State	Starting point for tracking, $\mathbf{q}_0$
Sampling Rate	Frequency at which images are processed
Feature Window Size	Size of slice in search for finger edges, set at 20 pixels
Gauss-Newton Weights	Stabilizes quaternion (1.0), translation (10.0), and joint angle (1000.0) state estimates

Table 4.2: Table of fixed parameters for feature alignment tracking algorithm.

Tracking begins with the user's hand in the initial configuration. This is aided by overlaying the projected hand model with the video image during the positioning stage. Once the system is initialized, tracking proceeds through the following steps:

1. *Update link frame positions with respect to the camera, using the current state estimate.*

2. *Project link frames into image through camera model and initialize search windows.*

3. *Process image slices and find edge points.*

4. *Compute line and tip feature measurements from edge points.*

5. *Compute residual Jacobian for each measured feature.*

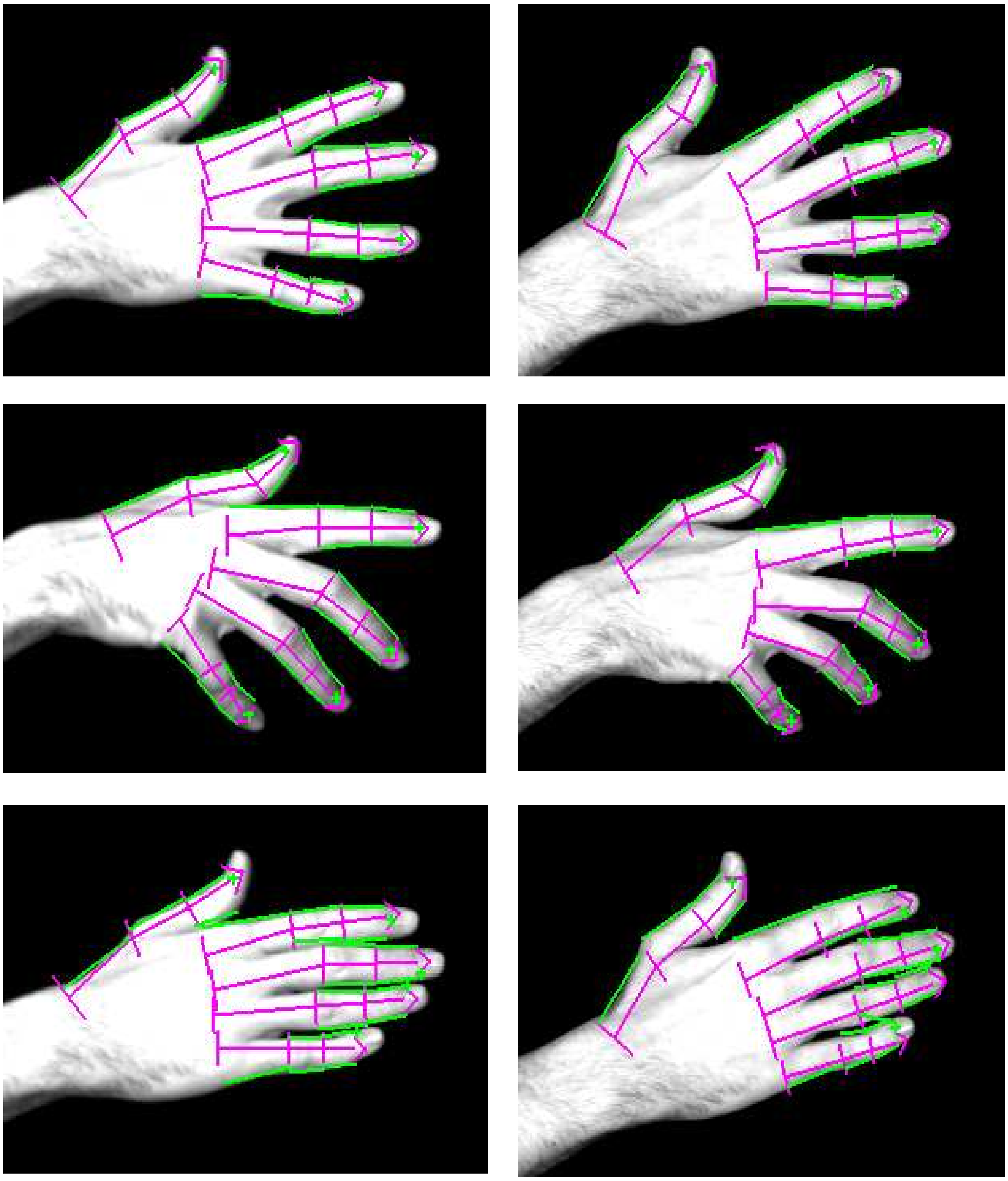
6. *Compute state correction through pseudo-inverse (Eq. 2.22) by solving the linear system.*

7. *Update the state estimate.*

The algorithm outlined above was used in all of the real-time experiments in this thesis. Accurate tracking requires accurate kinematic and camera models and a sufficient number of iterations of the estimation algorithm between frames. Model accuracy ensures that the residual minima will correspond to the minimum error state, while adequate iterations ensure that the minima will be reached for each frame. Track life for the feature alignment algorithm is determined by the alignment between the image and model projections in each frame. Track loss occurs when the search window constructed around a projected link of the model fails to contain the correct feature. Tracking accuracy impacts track life through the size of the residual. The residual grows with the distance between the projected model and the detected features. If it becomes too large, features may lie outside their associated search windows. Excessive hand velocity can also lead to track loss, as the feature displacement in the image between frames may exceed the search window size. This maximum displacement is determined by the hand velocity in conjunction with the sampling rate.

4.3.3 Whole Hand Tracking

The most ambitious tracking experiment attempted with the *DigitEyes* system was full 27 DOF hand tracking using two cameras. Two Sony XC-75 cameras were positioned 1.5 feet apart with optical centers verging near the



Camera 0 View **Camera 1 View**

Figure 4.5: Three pairs of hand images from the continuous motion estimate plotted in Figs. 4.7 and 4.8. Each stereo pair was obtained automatically during tracking by storing every fiftieth image set to disk. The samples correspond to frames 49, 99, and 149.

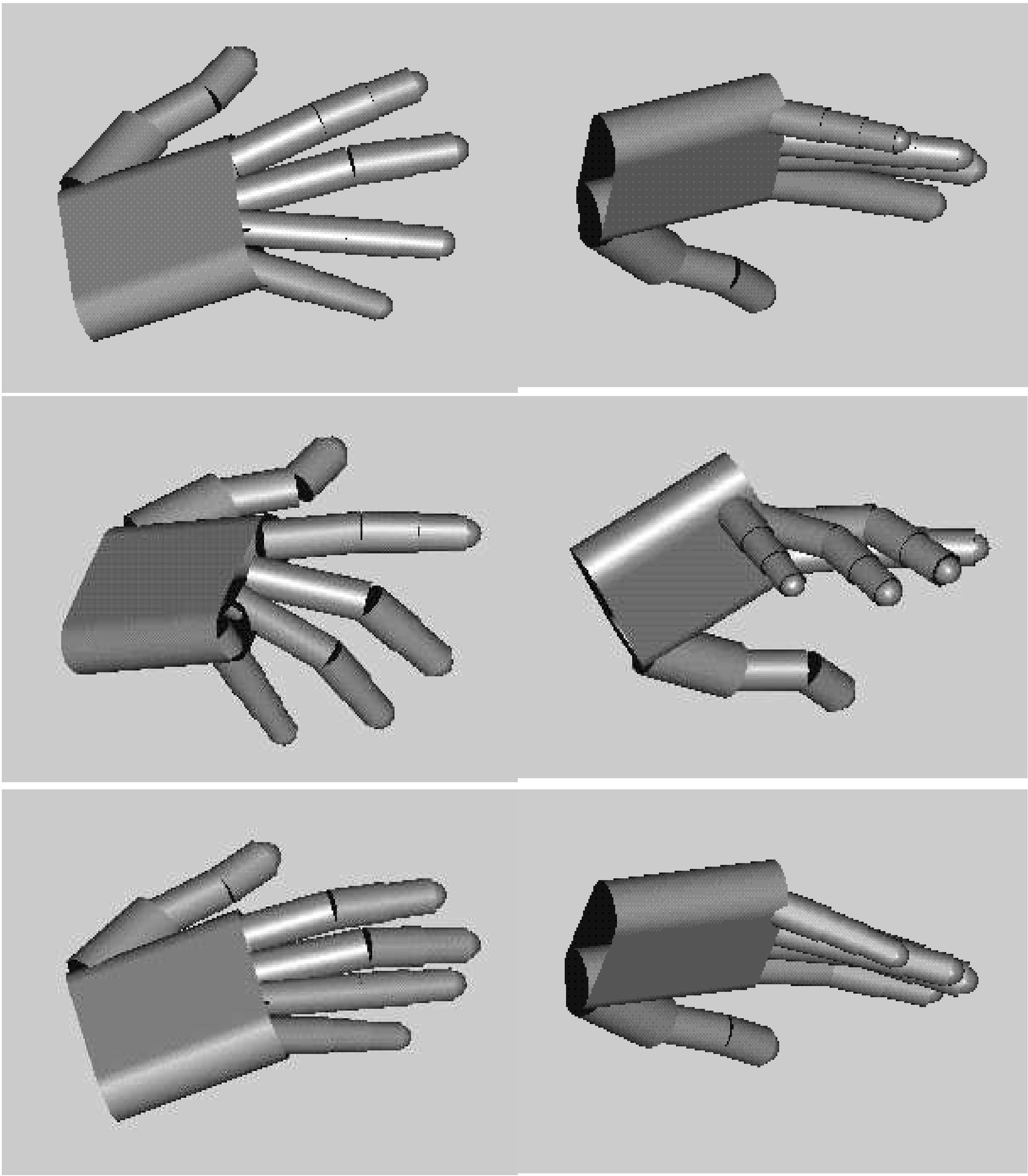


Figure 4.6: Estimated hand state for the image samples in Fig. 4.5, rendered from the Camera 0 viewpoint (left) and a viewpoint underneath the hand (right).

middle of the tracking area, and intersecting the table surface at approximately 45 degrees. They were both calibrated to the same coordinate frame, located in the tabletop. The distance from the cameras to the tabletop was approximately five feet. The tracker incorporated the full hand model from Appendix A. Line and point features from 13 of the 15 finger phalanges were employed in tracking. No features were extracted from the proximal phalanges of the middle two fingers, due to the impossibility of avoiding occlusions of these features during motion. No features were extracted from the palm, due to a desire to keep the feature extraction code simple and uniform.

Tracking began with the hand in a pre-arranged position on the tabletop. Because the hand motion had to avoid occlusions for successful tracking, the available range of travel was not large. It was sufficient, however, to demonstrate recovery of articulated DOFs in conjunction with palm motion. Figure 4.5 shows sample images, trackers, and features from both cameras at three points along a 200 frame sequence. The sample images were obtained automatically during tracking by writing every 50th image to disk.[3] Figure 4.6 shows the estimated model configurations corresponding to these sample points. In the left column, the estimated model is rendered from the calibrated viewpoint of the first camera. In the right column, it is shown from an arbitrary viewpoint, demonstrating the 3D nature of the tracking result. State estimates were logged by a program running on the Sun. The graphical model figures were rendered off-line, using the logged states.

Close examination of the sample images and backprojected models shows some of the residual error properties of the tracker. The first thing to note is that the fit is quite good overall, indicating the basic adequacy of both the measurements and the kinematic model. The most obvious indications of small errors are misalignments between anchor points and knuckle positions, and projected and actual joint centers and finger tips. A more interesting

[3]Samples obtained at 50 frame intervals were found to capture the most significant hand poses during tracking.

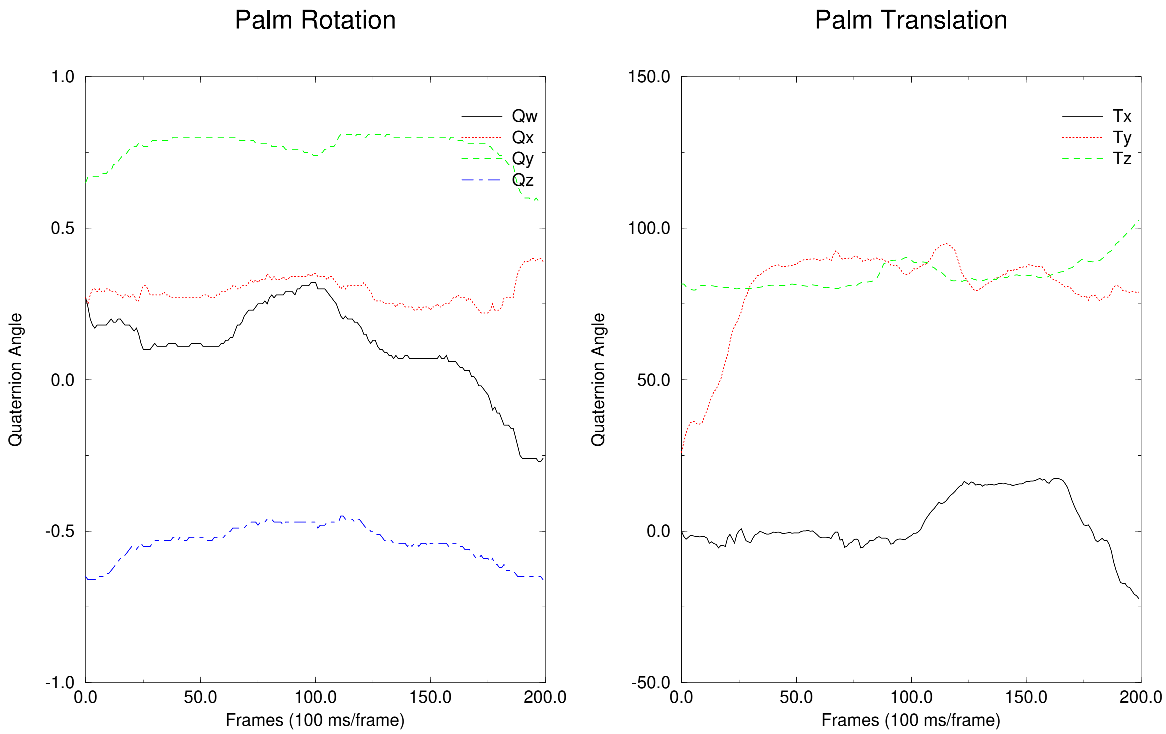


Figure 4.7: Estimated palm rotation and translation for motion sequence of entire hand. $\mathbf{Q}_w$-$\mathbf{Q}_z$ are the quaternion components of rotation, while $\mathbf{T}_x$-$\mathbf{T}_z$ are the translation. The sequence lasted 20 seconds.

error is visible in the images from frame 99. From the shading cues in the images, it is clear that the PIP joint[4] on the fourth (little) finger is strongly bent. Yet examination of the estimated model pose, particularly in the synthesized view from under the palm, shows that the estimated PIP joint angle is zero, and the estimator placed all of the bending at the MCP joint.

This error is the result of the fourth finger being in a singular configuration, in which none of the line features give information about its pose. In this case, only the tip position contains information about the degree of bending, and the system is free to assign angles among all three joints to achieve it.

[4]See Fig. 2.4 for the joint labels.

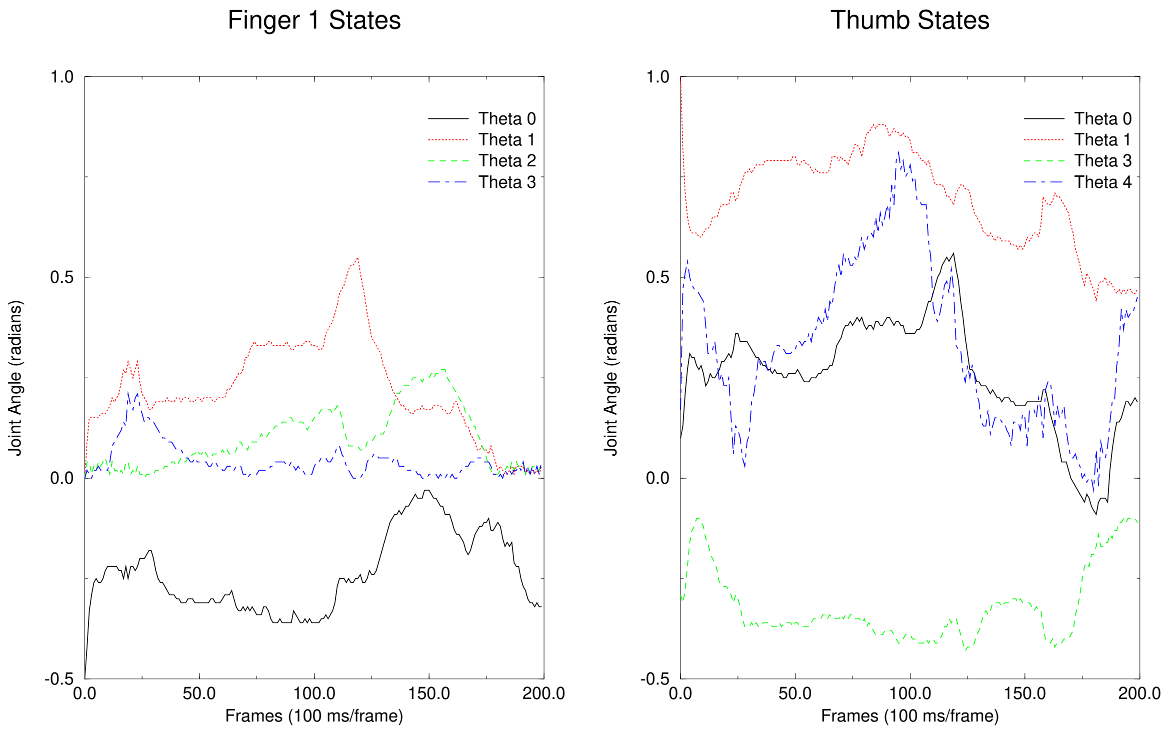


Figure 4.8: Estimated joint angles for the first finger and thumb. The other three fingers are similar to the first. Refer to Fig. 2.5 for variable definitions.

This error is due fundamentally to a lack of constraint on the state in that particular image. In this case, stereo did not remove the problem, since the configuration was singular with respect to both cameras.

As with traditional stereo, the tracker will benefit from cameras whose centers are widely separated. Unlike traditional stereo, this does not make it more difficult to process the images, since no direct correspondences are being computed (see Sec. 2.5.5.) Since the real-time system could not handle occlusions, a wide camera baseline was not possible. Incorporation of the occlusion handling results from Chpt. 3 would remove this limitation, leading to much more accurate 3D tracking performance. A subset of the estimated state trajectories for the motion sequence are given in Figs. 4.7 and 4.8.

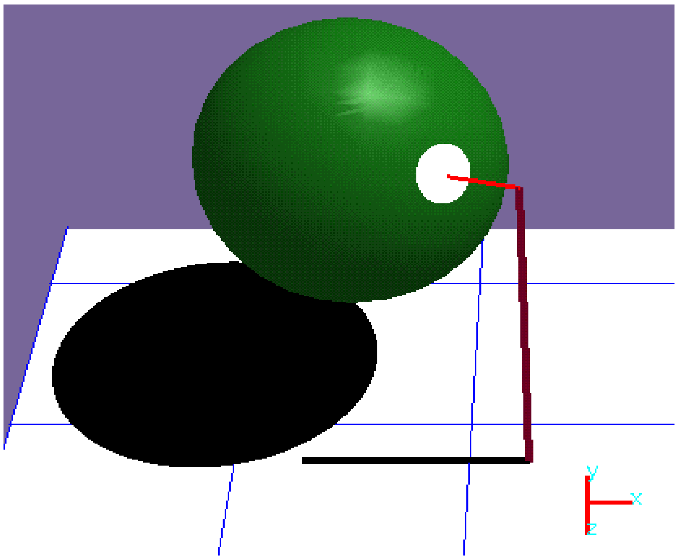


Figure 4.9: A sample graphical environment for a 3D mouse. The 3D cursor is at the tip of the "mouse pole", which sits atop the ground plane (in the foreground, at the right). The sphere is an example of an object to be manipulated, and the line drawn from the mouse to the sphere indicates its selection for manipulation.

4.3.4 3D Mouse User-Interface

Hand motion estimated in real-time by the *DigitEyes* system using a simplified hand model was employed to drive a 3D mouse interface [46, 47]. Figure 4.9 shows an example of a simple 3D graphical environment, consisting of a ground plane, a 3D cursor (drawn as a pole, with the cursor at the top), and a spherical object (for manipulation.) Shadows generate additional depth cues. The interface problem is to provide the user with control of the cursor's three DOFs, and thereby the means to manipulate objects in the environment.

In the standard "mouse pole" solution [71], the 3D cursor position is controlled by clever use of a standard 2D physical mouse. Normal mouse motion controls the base position of the pole on the ground plane. Depressing one of the mouse buttons switches reference planes, causing mouse motion in one direction to control the pole (cursor) height. By switching between planes, the user can place the cursor arbitrarily. Commanding continuous motion with this interface is awkward, however, and tracing an arbitrary, smooth space curve is nearly impossible. *DigitEyes* was used to develop a 3D virtual mouse, that permitted simultaneous hand-based control of the cursor's DOFs.

This application of the *DigitEyes* system served two purposes. First, it provided a qualitative test of the system's ability to recover 3D information using a single image sequence. Second, it demonstrated the capability of the tracking framework to provide adequate sensing for a practical application. Experience with the interface suggests areas for future improvement of the system.

In the *DigitEyes* solution to the 3D mouse problem, the 3 input DOFs are derived from a partial hand model, which consists of the first and fourth fingers of the hand, along with the thumb. The palm is constrained to lie in the plane of the table used in the interface, and thus has 3 DOF. The first finger has 3 articulated DOFs, while the fourth finger and thumb each have a single DOF allowing them to rotate in the plan of the table (abduct). The hand model is illustrated in Fig. 4.10. A single camera oriented at approximately 45 degrees to the table top acquires the images used in tracking. The palm position in the plane controls the base position of the pole, while the height of the index finger above the table controls the height of the cursor. This particular mapping has the important advantage of decoupling the controlled DOFs, while making it possible to operate them simultaneously. For example, the user can change the pole height while leaving the base position constant. The fourth finger and thumb have abduction DOFs in the plane, and are used as "buttons". The cost of estimating the reduced hand model was measured at 96.4 ms/iter. by timing with a stopwatch (see Sec. 4.3.1.) This gives an estimation rate of 10 Hz.

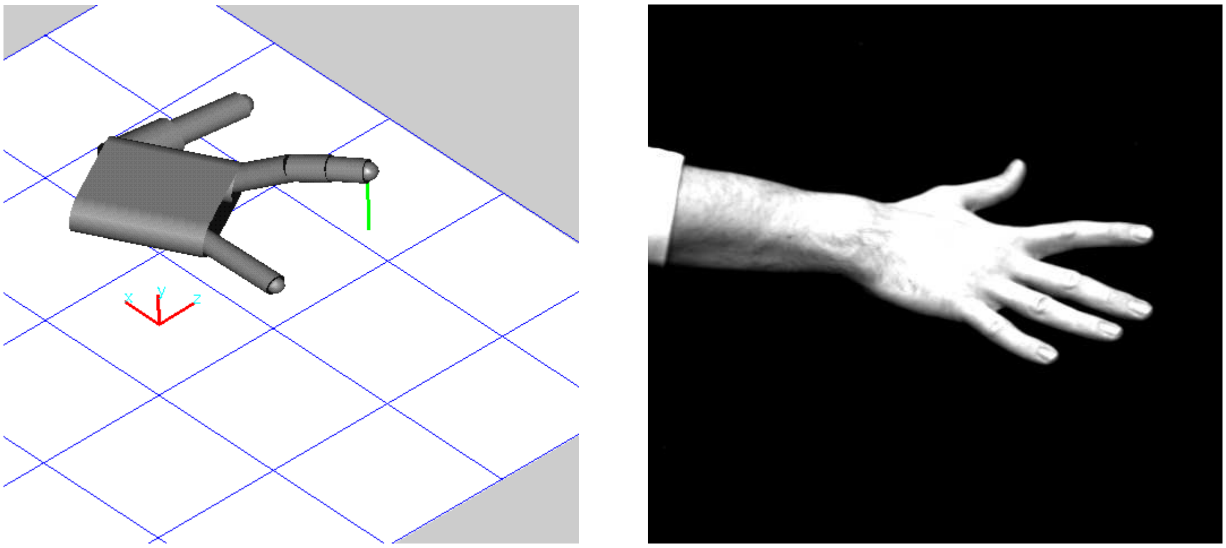


Figure 4.10: The hand model used in the 3D mouse application is illustrated for frame 200 in the motion sequence from Fig. 4.12. The vertical line shows the height of the tip above the ground plane. The input hand image (frame 200) demonstrates the finger motion used in extending the cursor height.

Figures 4.11 – 4.13 give experimental results from a 500 frame motion sequence in which the estimated hand state was used to drive the 3D mouse interface. Figures 4.11 and 4.12 show the estimated hand state for each frame in the image sequence. Frames were acquired at 100 ms sampling intervals. The pole height and base position derived from the hand state by the 3D mouse interface are also depicted in Fig. 4.12. The motion sequence has four phases. In the first phase (frame 0 to 150), the user's finger is raised and lowered twice, producing two peaks in the pole height, with a small variation in the estimated pole position. Second, around frame 150 the finger is raised again and kept elevated, while the thumb is actuated, as for a "button event". The actuation period is from frame 150 to frame 200, and results in some change in the pole height, but negligible change in pole position. Third, from 200 to 350, the pole height is held constant while the pole position is

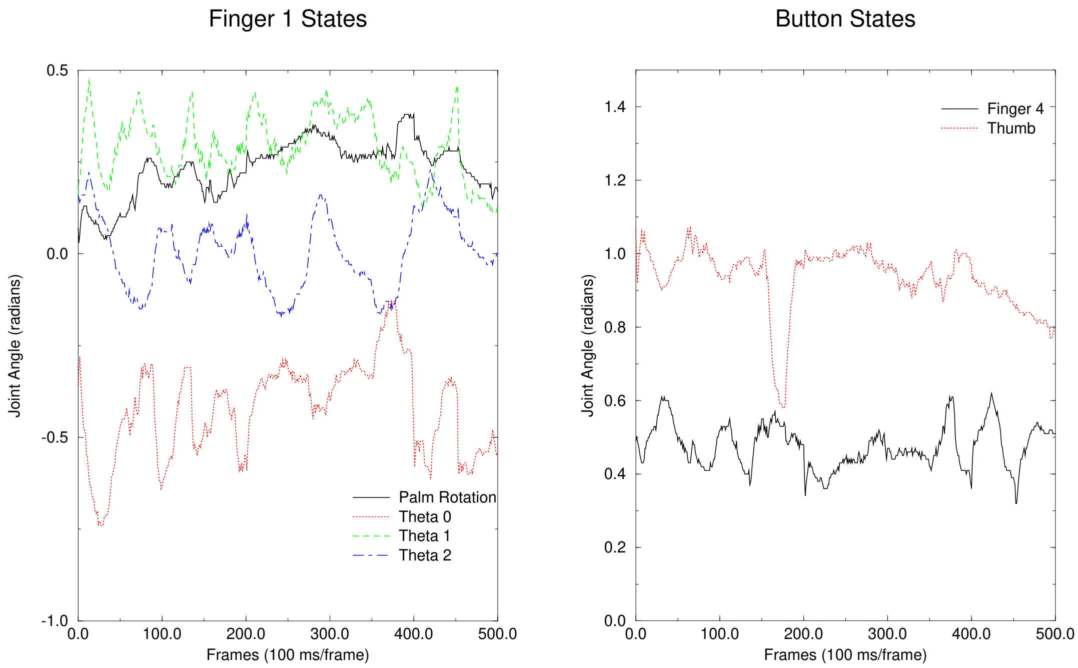


Figure 4.11: Palm rotation and finger joint angles for mouse pole hand model depicted in Fig. 4.10. Joint angles for thumb and fourth finger, shown on right, are used as buttons. Note the "button event" signaled by the thumb motion around frame 175.

varied. Finally, from 350 to the end of the sequence all states are varied simultaneously. Sample mouse pole positions throughout the sequence are illustrated in Fig. 4.13. This is the same scene as in Fig. 4.9, except that the mouse pole height and position change as a function of the estimated hand state.

4.3.5 Evaluation of 3D Mouse Performance

Following extensive experimentation with the 3D mouse interface, three important attributes of the hand tracker's performance were identified. These factors seemed to have the largest impact on the successful use of the inter-

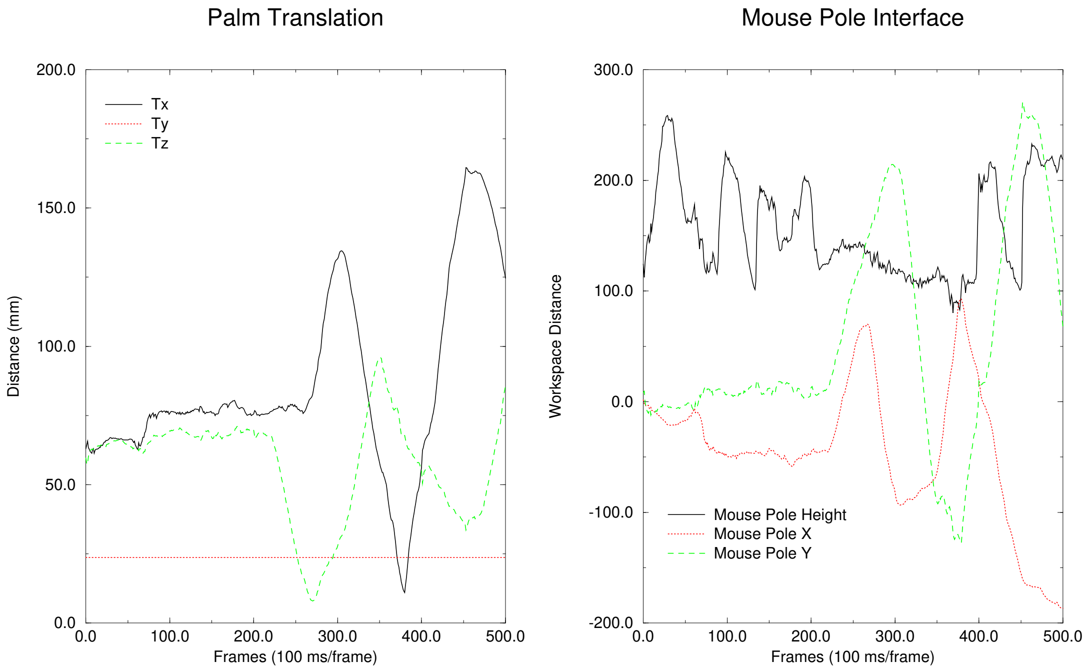


Figure 4.12: Translation states for mouse pole hand model are given on the left. The Y axis motion is constrained to zero due to tabletop. On the right are the mouse pole states, derived from the hand states through scaling and a coordinate change. The sequence events goes: 0-150 finger raise/lower, 150-200 thumb actuation only, 200-350 base translation only, 350-500 combined 3 DOF motion.

face:

- Sampling rate
- Sensitivity
- Latency

The quality of the interface as a whole seemed to depend on another set of three properties, which are closely linked to the tracker attributes above.

- Maximum hand speed

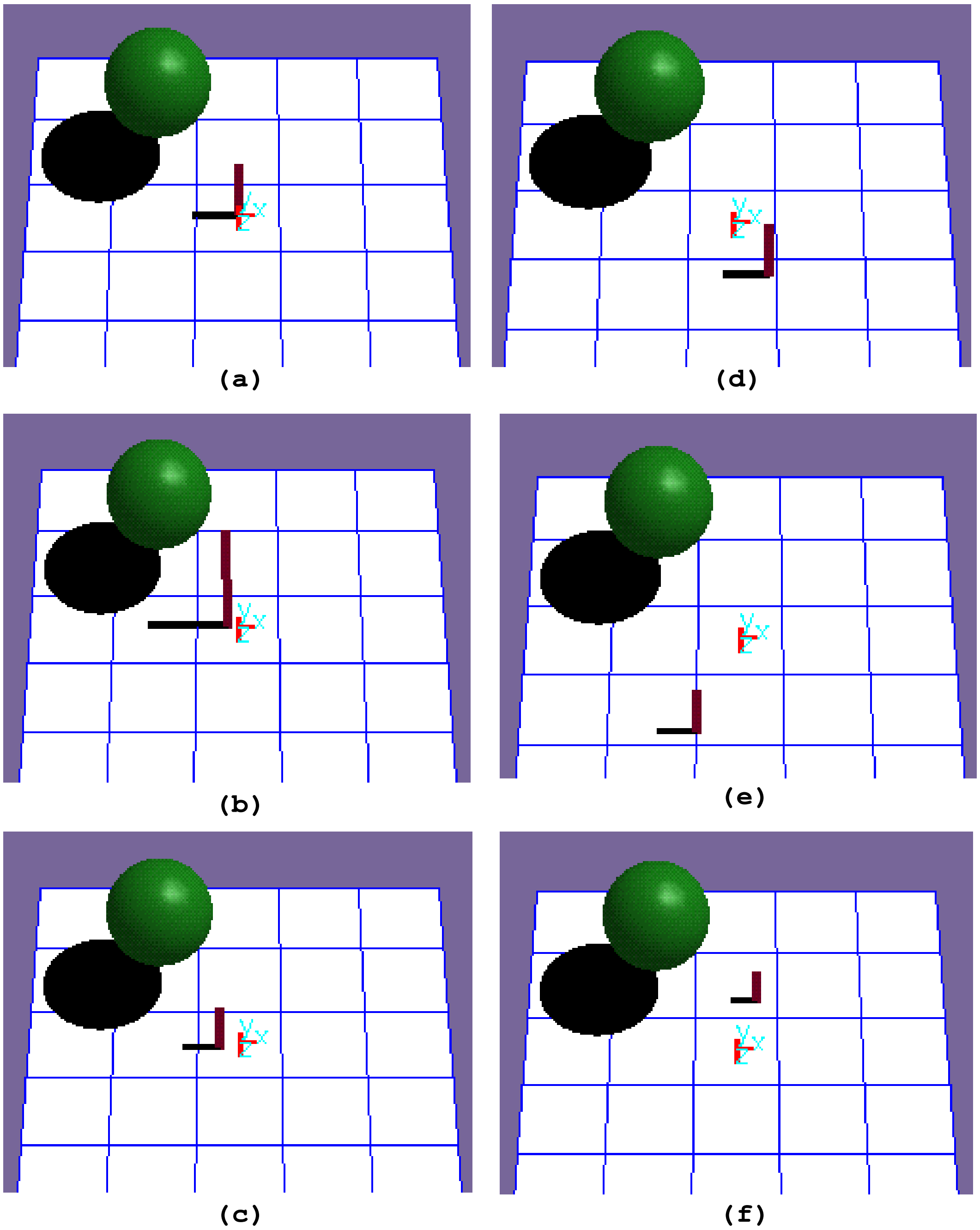


Figure 4.13: The mouse pole cursor at six positions during the motion sequence of Fig. 4.11. The pole is the vertical line with a horizontal shadow, and is the only thing moving in the sequence. Samples were taken at frames 0, 30, 75, 260, 300, and 370 (chosen to illustrate the range of motion).

- Transient DOF coupling
- Resolution

To illustrate the impact of the tracker's performance on an application, each of the above issues is examined in turn. The maximum possible speed of the user's hand across the tabletop is a function of the sampling rate of the estimation algorithm, in relation to the error surface properties of the residual. In the specific case of the virtual mouse interface, the tracker could tolerate hand motions of about 2.5 in/sec before track loss began. This was measured experimentally by timing repeated hand translations in the plane, keeping the tracker on the edge of convergence by observing the real-time overlay of the backprojected model and images.

Transient coupling between DOFs is a second factor that is affected by the sampling rate. State coupling is a natural consequence of the kinematic constraints which make tracking possible. These constraints lead to transient effects in the estimator, however, that can negatively impact performance. An example of transient coupling occurs around frame 150 in the Button State and Mouse Pole Interface plots from Figs. 4.11 and 4.12. When the thumb is actuated for a button event, the pole height drops initially, and then rises back to its previous level over the course of about 20 frames. This behavior is the result of an initial tendency of the estimator to spread residual error over all of the states that can reduce it. Only after the thumb has had time to rotate, and absorb most of its residual error, are the other residuals able to reassert their control over their own DOFs. The duration of these transient effects is primarily a result of the sampling rate. More iterations/sec. make the estimator "stiffer," and reduce the effect of these disturbances. Interestingly, very similar experimental observations have be made in the domain of robot control [30].

The last property of the interface, the resolution with which the cursor position can be controlled, is largely a function of the estimator sensitivity.

As described in Sec. 2.5.3, the sensitivity of a state varies with position in the state space. The large scale effect of this is that the ease of use of the interface depends strongly on the palm orientation relative to the camera. Consider rotating the palm on the table as the pole height is varied. The orientation at which the plane of the finger contains the camera is a singular configuration, and pole height becomes extremely difficult to measure. The sensitivity of the estimator to the finger motion decreases as this singularity is approached. The effective resolution in the cursor position is determined by the state sensitivity. The more sensitive the state, the larger the range of image displacements that are produced by a given range of state space motion. This in turn leads to a larger resolution in state space, and greater ability to control the cursor at a fine level of detail.

The effects of latency were not studied in detail for the virtual 3D mouse problem, as they were not extremely significant. Latency refers to the time delay between hand motion and the response of the interface. Long latency times make control of the interface impossible. As a result of the virtual 3D mouse interface design, the total latency was determined by the estimator cycle time, the communication delay to the Indigo 2, and the model rendering time. These last two additional effects added around 30 ms to the 100 ms cycle time. The effect of the total latency was noticeable, but did not make the cursor uncontrollable.

4.4 Tracking Self-Occluding Hand Motion

The representations and algorithms for self-occluding motion described in Chpt. 3 were implemented in an off-line version of the *DigitEyes* system. This section gives a complete summary of the resulting tracking algorithm, and presents experimental results for a two finger motion sequence with significant self-occlusion.

4.4.1 Algorithm Summary

The template-based tracking algorithm for self-occluding motion described in Chpt. 3 is summarized along with a discussion of the primary error sources. Table 4.3 lists the fixed parameters in the template-based tracking algorithm. In addition to the camera and kinematic models, a template model must be specified for each link. These templates are obtained manually from a set of reference images before tracking begins.

Parameters	Description
Camera Model	11 extrinsic (pose) and intrinsic (image scale and origin) parameters
Kinematic Model	Joint axes, link lengths, and anchor points
Template Model	Sufficient views for each link in object
Initial State	Starting point for tracking, $\mathbf{q}_0$
Sampling Rate	Frequency at which images are processed
Step size	Scales state correction in gradient-descent algorithm.

Table 4.3: Table of fixed parameters for feature alignment tracking algorithm.

Tracking begins with the user's hand in the initial configuration. This is aided by overlaying the projected hand model with the video image during the positioning stage. Once the system is initialized, tracking proceeds through the following steps:

1. *Update link frame positions with respect to the camera, using the current state estimate.*

2. *Project link templates into image through camera model.*

3. *Segment image pixels, assigning them to templates.*

4. *Compute residual and Jacobian for each segmented pixel.*

5. *Compute state correction through gradient-descent minimization (Eq. 2.12.)*

6. *Update the state estimate.*

The accuracy of the tracking algorithm depends, as in Sec. 4.3.2, on the accuracy of the kinematic and camera models. In addition, errors in the template models, such as unexpected shading variations, can cause the minimum residual state to differ from the correct state, degrading the tracking accuracy. A sufficient number of iterations of the estimation algorithm between frames is also required for accurate tracking. Track life in the template-based algorithm is determined by the shape of the state space error surface, which is minimized during estimation. For each image, there is a region of convergence (ROC) centered around the minimum residual state. Track loss occurs if the starting point for minimization lies outside this ROC in any frame. This could happen as a result of errors in the camera, kinematic, or template models. It could also occur if the hand velocity between frames is too large, resulting in a state displacement outside of the ROC. The maximum state displacement is determined by the hand velocity in conjunction with the sampling rate.

In addition to the basic requirements for template-based tracking described above, there are four necessary conditions for tracking self-occluding objects:

1. There are no points in the state space where the occlusion properties change instantaneously. This ensures that all regions of occlusion are separated by disjoint regions.

2. The sampling rate is high enough to prevent occlusion ambiguities. The product of the sampling rate and maximum state velocity must be less than the minimum distance through a disjoint region of the state space. When this condition is met, the disjoint regions can be grown by the motion interval without bringing them into contact.

3. The composite Jacobian formed from all camera viewpoints must be full rank. This ensures that measurements are available for each state in the linearized system. If this condition is not met, there will be no estimates for some states, and any occlusion properties that depend on these states cannot be determined. This could cause the visibility order prediction to fail.

4. The occlusion graph for the tracked object must be acyclic at all points in the state space.

If these conservative requirements are met, the prediction of the visibility order will succeed for all possible motions of the object. In practice, there may be points in the state space where one or more of these conditions are violated. For example, occlusion ambiguities arise in hand tracking during a gesture like "stop", as discussed in Sec. 3.4.1. In practice it may be necessary to use other information, such as the velocity of the object, to disambiguate these cases.

4.4.2 Two Finger Tracking Results

The main representations and algorithms for self-occluding motion described in Chpt. 3 have been implemented in an off-line version of the *DigitEyes* system. This section presents the first experimental results using this tracker, for a two finger motion sequence with significant self-occlusion, depicted in Fig. 4.14.

In the sequence, my index finger curls into my palm while my hand and remaining fingers are held still. An 80 frame sequence was digitized from videotape and sampled for an effective frame rate of approximately 15 Hz. This resulted in an average finger tip displacement between frames of about three pixels. The camera was positioned at approximately 45 degrees to the table top, facing the palm. As a result of this camera position, the first finger

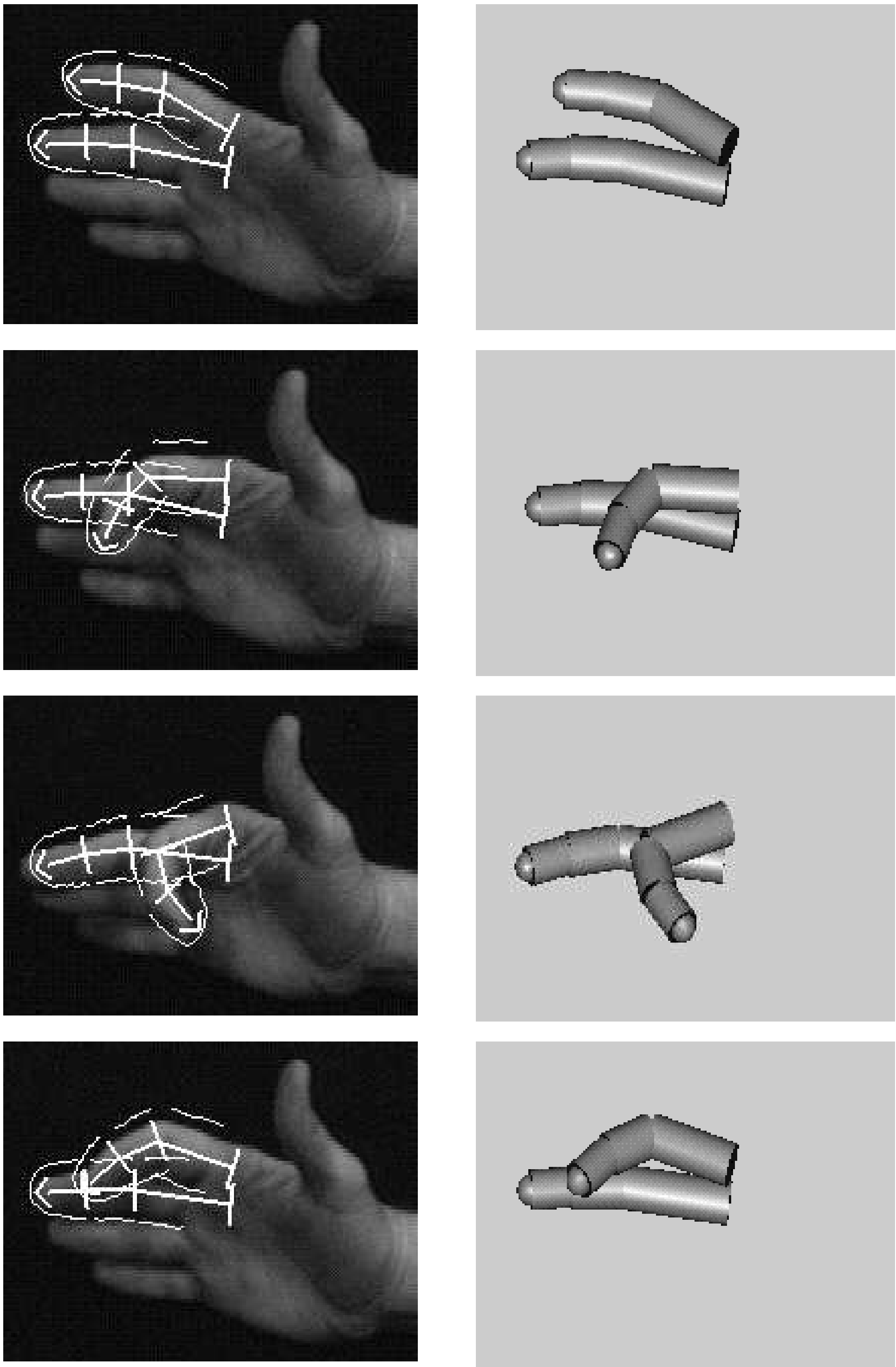


Figure 4.14: Sample input images and associated state estimates for frames 0, 13, 30, and 75 in the motion sequence. The two finger hand model is rendered with respect to the calibrated camera model using the estimated state. The overlays show the template boundaries and projection of cylinder center axes. These frames were selected for their representative self-occlusions.

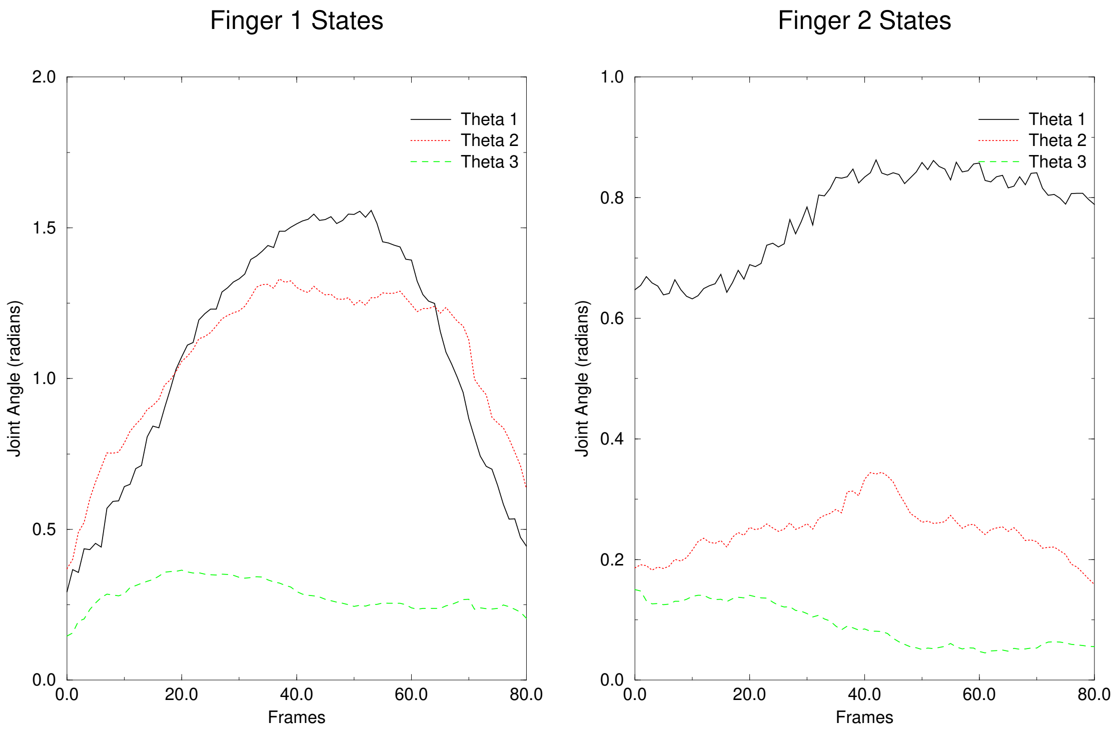


Figure 4.15: Estimated planar finger motions for two finger model.

occludes all of the remaining fingers during its motion. In this example, the visibility order does not change throughout the sequence. As a result, the order was computed once before tracking began.

The sequence was tracked with a 9 DOF kinematic model containing the index and middle fingers (3 planar joints per finger) of the full hand model, and allowing full translation. Each finger link was modeled with a single template. The template models for each link were manually segmented from a reference image that was not part of the sequence, but similar in pose and taken under identical lighting conditions. Both the input images and templates were convolved with a 13x13 LOG filter. LOG filtering was chosen to emphasize the edge properties of the templates. The filter size was selected empirically. Unit window functions were used for all links and the state was

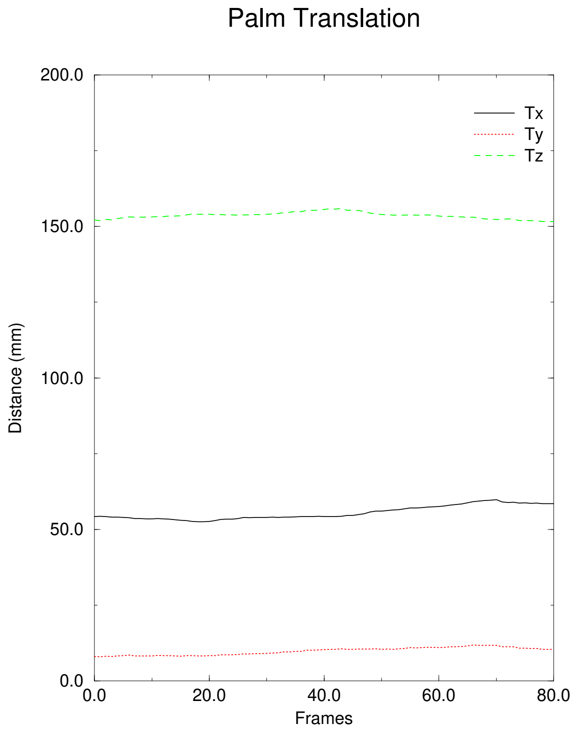


Figure 4.16: Estimated translation state of two finger model.

initialized by hand. The gradient descent algorithm was iterated twice for each frame in the sequence. By limiting the motion to the occluding bodies, any perturbation on the occluded links would be visible as a nonzero motion. However, there was a small amount of middle finger motion towards the middle of the sequence which is visible in the images and was recovered by the estimator.

Figure 4.14 shows the estimated model configurations corresponding to the sample points, rendered from the calibrated camera viewpoint. The estimated state trajectories for the motion sequence are shown in Figs. 4.15 and 4.16. Deviations in the states of the unmoved finger are on the order of 0.2 radians. Deviances in translation are equally small. There are four potential sources of error in the template-based tracker: image noise, kinematic model error, photometric template model error, and template shape error. Of these, the last two were noticeable in the residual, and warrant further study.

These results demonstrate the potential of the direct template registration approach to tracking self-occluding objects. From a classical feature detection perspective, the images in the sequence are quite difficult. All of the phalanges of the middle finger are partially occluded during some portion of the motion sequence, and the index finger is silhouetted against the fingers and palm for most of its motion. A significant advantage of the window-based approach is that it can tolerate any amount of occlusion and continue to extract useful information from the pixels that are visible. The successful tracking of this complicated motion testifies again to the power of the kinematic model in constraining the interpretation of the image.

All of the experiments in this chapter employed a black cloth backdrop to ensure high contrast between the hand and its background. Invariance to background was not addressed, as it is believed to be less important than the kinematic and self-occlusion issues which were the focus of this thesis. In practice, applications can be designed with a constrained background, as the 3D virtual mouse interface demonstrates. However, a background template can be added to the framework tested in this section, making it possible to exploit a fixed background image in tracking.

4.5 Summary

This chapter describes experimental results in hand tracking, both in real-time and using off-line image sequences. Two algorithms were tested: the line and point feature-based algorithm of Sec. 2.5, and the layered template algorithm for self-occluding motion described in Chpt. 3. The presented results include the first experimental demonstration of 27 DOF visual tracking, and the first tracking results for articulated motion with significant amounts of occlusion. All experiments were conducted with natural images of unmarked hands.

Chapter 5

Previous Work

Research in human motion analysis spans a wide variety of disciplines from biomechanics to human-computer interaction and virtual reality, from computer vision to computer graphics. Previous work in this area can be classified along three overlapping lines. The first body of work, which includes this thesis, is concerned with 3D analysis of human motion. It is distinct from a second body of work in 2D gesture recognition. This work is concerned solely with the mapping from image sequences to a set of discrete classes. In principle, the 3D tracking approach can be applied to this problem as well [15], but work in this area often takes a learning approach and tries to avoid 3D model specification. A third body of work develops special purpose algorithms for human sensing applications. It is not concerned with developing general frameworks, as is the case in the previous two areas.

5.1 3D Motion Analysis

In 3D tracking approaches, a model of the articulated object is employed to constrain image interpretation [50, 48, 47, 29, 32, 39, 33, 15, 72, 44, 23, 42]. A second class of 3D analysis problems attempt to recover both 3D structure and motion (or pose) simultaneously [24, 57, 68, 45]. This latter class is a significant departure from the approach of this thesis, and won't be considered in detail. Since none of the articulated tracking work prior to this thesis dealt explicitly with self-occlusions, the comparison in these sections will be concerned only with the use of kinematics to provide geometric constraints in tracking. Connections between Chpt. 3 and other work on occlusion are described later.

The two earliest systems for visual human motion analysis, by O'Rourke and Badler [42] and David Hogg [23], approached model-based recovery of human motion using the respective AI search techniques of constraint propagation and heuristic search of a discretized state space. Both works stand out in the complexity of the model constraints they applied to the tracking problem. O'Rourke's system was capable of incorporating occlusion and rigid body noninterpenetration constraints in pose determination. Hogg's system also included postural models.

The treatment of occlusion constraints in O'Rourke's system provides an interesting complement to their role in this thesis. In the example in his paper, the onset of occlusion is detected at the image level, and then the model is positioned so as to achieve the occlusion constraint. In that particular case, the prediction of occlusion from the model would be very difficult, as the arm is in a singular configuration. One could imagine using the occlusion prediction mechanism in Chpt. 3 to rule out unlikely occlusion events, reducing the number of possibilities that had to be searched.

Hogg's work is probably the most relevant to this thesis, as it dealt with motion explicitly and presented results for an image sequence of a walking figure that are still impressive by today's standards. From a conceptual viewpoint, the two biggest distinctions between this work and Hogg's lie in the kinematic representation and search method. Robotic kinematic models provide powerful tools for converting articulated tracking into a continuous estimation problem. These techniques make it possible to handle a much larger state space and integrate kinematic constraints and image interpretation directly.

The idea of applying robotic kinematic models to human motion tracking was first proposed by Yamamoto and Koshikawa [72]. They presented 2D tracking results for a three DOF system of a human arm and torso. This work extends the tracking framework in their paper significantly in several directions: explicit presentation of robot kinematic models based on DH notation, analysis of singular configurations, 3D real-time tracking results, application to a user-interface domain, and high DOF tracking. Their more recent publications [29, 32] present off-line 3D tracking results using two cameras, following the approach to integrating multiple views described in Sec. 2.5.5. These results are very interesting, as they provide evidence that the techniques in this thesis are applicable to body tracking as well.

Works in the area of physics-based modeling have also addressed articulated body motion [65, 39, 44]. One of the applications of deformable models presented in [65] is 3D tracking of a single finger from a stereo image sequence. Pentland and Horowitz [44] give an example of tracking the motion of a human figure using optical flow and an articulated deformable model. In a related approach, Metaxis and Terzopoulos [39] track articulated motion using deformable superquadric models.

Although most researchers working in the gesture recognition area have pursued 2D approaches, there are a few works that investigate 3D analysis [15, 33]. Dorner describes a system for interpreting American Sign Language from image sequences of a single hand in [15]. In her system, the user wears a glove with different colors to aid in finger segmentation. Another 3D approach based on wearing gloves with fiducial points is described in [33].

5.2 2D Gesture Analysis

There has been a large amount of work on applying static and dynamic gesture recognition approaches to hand imagery. Three representative works based on learning approaches are [19, 12, 56]. In [12], Darrell and Pentland describe a system for learning and recognizing dynamic hand gestures. Their approach tries to avoid explicit models by building a library of template models on-line. Work by Segen [56] takes a neural network approach to 2D hand gesture recognition. Some sample interfaces based on gestural control of computer graphics models are described. Freeman describes a gesture recognition system based on orientation histograms in [19]. All of these systems obtain real-time performance.

Although many frameworks for human motion analysis are possible, an approach based on full-state 3D tracking has four main advantages. First, by tracking all of the hand's DOFs, the end-user is provided with the maximum possible flexibility for interface applications. (See [61, 27] for examples of interfaces requiring a whole-hand sensor.) In addition, a general modeling approach based on 3D kinematics makes it possible to track any subset of hand or body states with the same basic algorithm. Another benefit of full state tracking is invariance to unused hand motions. The motion of a particular finger, for example, can be recognized from its joint angles regardless of the pose of the palm relative to the camera. Finally, modeling the hand kinematics in 3D eliminates the need for application- or viewpoint-dependent user modeling.

5.3 Application-Specific Human Sensing

Many authors have used hand and body images to test 2D tracking and registration algorithms. Many of these approaches are applicable to user interface or surveillance domains. A glove-based approach that uses motion parallax to control a graphical environment is described in [9]. In the domain of human figures, two approaches to 2D tracking are [5, 25]. Huttenlocher et. al. have applied the Hausdorff distance measure to register images of moving people [25]. In a related effort, Baumberg and Hogg [5] describe a real-time pedestrian tracking system based on active shape models. Approaches to human motion analysis based on more invasive approaches, such as mechanical sensors [74, 7] or active targets [37], have a long history.

5.4 Layered Representations

The layered representation for self-occlusion presented in Chpt. 3 is related to other work in tracking and motion coding. Layered representations based on clustering optical flow are presented in [1, 11, 67]. This work is largely concerned with automatically generating layered, velocity-based representations of a motion sequence that could serve as a model for coding or recognition. A coding approach based on global image models is presented in [26]. A layered representation based on the occluding contours of a single image is described in [41]. These works are complementary to the approach in this thesis, which is concerned with making the best use of available models. In addition, the kinematic representation of self-occlusions is a generalization of layered representations based on depth ordering in the scene, since it is designed to exploit orderings within configuration space.

As a result of modeling self-occlusion in the image plane, tracking can be formulated as a direct optimization problem over an image-based residual error. The approach of coupling the image interpretation (feature detection) problem directly to the model was popularized by deformable models [65] (including 2D Snakes [28]) and has since been applied to a variety of other domains [51, 73].

Chapter 6

Conclusion and Future Work

A vision-based sensor can provide a passive, noninvasive solution to human motion tracking problems, since it can be located in the user's environment rather than on their person. To achieve these goals, computer vision algorithms have been developed that can estimate 3D articulated motion from ordinary intensity images of unmarked hands or bodies at video rates.

This dissertation has presented new results in applying kinematic models to articulated object tracking. By adopting the representations and tools of robotics, powerful computational and analytic tools are brought to bear on the visual tracking problem. Using these techniques, kinematic models are developed for the hand and incorporated into a real-time tracking system called *DigitEyes*. The kinematic model plays an additional role in predicting visibility orders for tracking self-occluding motion. The resulting tracking algorithms were tested on natural hand image sequences and applied to a 3D mouse user-interface problem. These experiments demonstrate the potential of 3D visual human sensing.

Future Work

- Hand model calibration could be accomplished on-line by adapting fixed model parameters in an estimation loop with a longer time constant than state estimation. Model parameters would be initialized to standard values, and after a period of adaptation would conform to any user.

- A real-time implementation of the self-occlusion handling tracker would be extremely interesting. Such a system would allow unconstrained hand motion during tracking for the first time. Such an approach should be employed with multiple cameras to ensure accurate estimation.

- Ground truth 3D hand data should be obtained to measure the absolute accuracy of the tracker as a function of the number of cameras. An initial strategy is to attach LEDs to the palm and track the six DOF palm motion using the fingers. This would avoid interference between the two sensors and provide a base accuracy assessment.

- Alternative window functions should be investigated and their effect on tracker performance should be analyzed.

- It would be interesting to combine the top-down occlusion prediction from the kinematic model with a bottom-up occlusion analysis stage in a synergistic approach.

- Applications of the *DigitEyes* sensor to graphics, puppetry, and user-interface applications should be developed to improve our understanding of the necessary performance level for real applications.

Appendix A

Whole Hand DH Model

The next page gives the full DH model for my right hand, which was used in all of the experiments in this thesis.

Frame	Geometry	θ	d	a	α	shape (in mm)	Next
0	Palm	0.0	0.0	0.0	0.0	x 56.0, y 86.0, z 15.0	1 8 15 22 29
1		$\pi/2$	0.0	38.0	$-\pi/2$		2
2		0.0	-31.0	0.0	$\pi/2$		3
3		$\mathbf{q}_7$	0.0	0.0	$\pi/2$		4
4	Finger 1 Link 0	$\mathbf{q}_8$	0.0	45.0	0.0	Rad 10.0	5
5	Finger 1 Link 1	$\mathbf{q}_9$	0.0	26.0	0.0	Rad 10.0	6
6	Finger 1 Link 2	$\mathbf{q}_{10}$	0.0	24.0	0.0	Rad 9.0	7
7	Finger 1 Tip	0.0	0.0	0.0	0.0	Rad 9.0	–
8		$\pi/2$	0.0	37.0	$-\pi/2$		9
9		0.0	-9.0	0.0	$\pi/2$		10
10		$\mathbf{q}_{11}$	0.0	0.0	$\pi/2$		11
11	Finger 2 Link 0	$\mathbf{q}_{12}$	0.0	56.0	0.0	Rad 10.0	12
12	Finger 2 Link 1	$\mathbf{q}_{13}$	0.0	27.0	0.0	Rad 10.0	13
13	Finger 2 Link 2	$\mathbf{q}_{14}$	0.0	22.0	0.0	Rad 9.0	14
14	Finger 2 Tip	0.0	0.0	0.0	0.0	Rad 7.0	–
15		$\pi/2$	0.0	33.0	$-\pi/2$		16
16		0.0	6.0	0.0	$\pi/2$		17
17		$\mathbf{q}_{15}$	0.0	0.0	$\pi/2$		18
18	Finger 3 Link 0	$\mathbf{q}_{16}$	0.0	53.0	0.0	Rad 9.0	19
19	Finger 3 Link 1	$\mathbf{q}_{17}$	0.0	25.0	0.0	Rad 9.0	20
20	Finger 3 Link 2	$\mathbf{q}_{18}$	0.0	20.0	0.0	Rad 8.0	21
21	Finger 3 Tip	0.0	0.0	0.0	0.0	Rad 7.0	–
22		$\pi/2$	0.0	30.0	$-\pi/2$		23
23		0.0	26.0	0.0	$\pi/2$		24
24		$\mathbf{q}_{19}$	0.0	0.0	$\pi/2$		25
25	Finger 4 Link 0	$\mathbf{q}_{20}$	0.0	38.0	0.0	Rad 9.0	26
26	Finger 4 Link 1	$\mathbf{q}_{21}$	0.0	19.0	0.0	Rad 8.0	27
27	Finger 4 Link 2	$\mathbf{q}_{22}$	0.0	17.0	0.0	Rad 7.0	28
28	Finger 4 Tip	0.0	0.0	0.0	0.0	Rad 6.0	–
29		$-\pi/2$	15.0	43.0	$-\pi/2$		30
30		$-\pi$	38.0	0.0	0.0		31
31		$\mathbf{q}_{23}$	0.0	0.0	$\pi/2$		32
32	Thumb Link 0	$\mathbf{q}_{24}$	0.0	46.0	$-\pi/2$	Rad 14.0	33
33		$\mathbf{q}_{25}$	0.0	0.0	$\pi/2$		34
34	Thumb Link 1	$\mathbf{q}_{25}$	0.0	34.0	0.0	Rad 10.0	35
35	Thumb Link 2	$\mathbf{q}_{26}$	0.0	25.0	0.0	Rad 10.0	36
36	Thumb Tip	0.0	0.0	0.0	0.0	Rad 8.0	–

Table A.1: The Denavit-Hartenberg kinematic model for my right hand. It was calibrated using the procedure of Sec. 2.2.3.

Bibliography

[1] E. H. Adelson. Layered representation for image coding. Technical Report 181, MIT Media Lab, 1991.

[2] O. Amidi, Y. Mesaki, T. Kanade, and M. Uenohara. Research on an autonomous vision guided helicopter. In *Fifth World Conf. on Robotics Res.*, pages 14–11 to 14–22, Cambridge, MA, 1994. SME.

[3] K. N. An, E. Y. Chao, W. P. Cooney, and R. L. Linscheid. Normative analysis of human hand for biomechanical analysis. *Journal of Biomechanics*, 12:775–788, 1989.

[4] P. Anandan. A computational framework and an algorithm for the measurement of visual motion. *International Journal of Computer Vision*, 2:283–310, 1989.

[5] A. Baumberg and D. Hogg. An efficient method for contour tracking using active shape models. In J. Aggarwal and T. Huang, editors, *Proc. of Workshop on Motion of Non-Rigid and Articulated Objects*, pages 194–199, Austin, Texas, 1994. IEEE Computer Society Press.

[6] J. Bergen, P. Anandan, and et. al. Hierarchical model-based motion estimation. In *Second European Conf. on Computer Vision*, pages 237–252, Santa Margherita Liguere, Italy, 1992. Springer-Verlag.

[7] T. Calvert, J. Chapman, and A. Patla. Aspects of the kinematic simulation of human movement. *IEEE Computer Graphics and Applications*, pages 41–50, November 1982.

[8] J. Canny. *The Complexity of Robot Motion Planning.* MIT Press, Cambridge, MA, 1988.

[9] R. Cipolla, Y. Okamoto, and Y. Kuno. Qualitative visual interpretation of 3d hand gestures using motion parallax. In *IAPR Workshop on Machine Vision Applications*, Tokyo, Japan, December 1992.

[10] T. Cormen, C. Leiserson, and R. Rivest. *Introduction to Algorithms.* MIT Press, Boston, MA, 1990.

[11] T. Darrell and A. Pentland. Robust estimation of a multi-layered motion representation. In *Proc. of IEEE Workshop on Visual Motion*, pages 173–178, Princeton, NJ, 1991.

[12] T. Darrell and A. Pentland. Space-time gestures. In *Looking at People Workshop*, Chambery, France, 1993.

[13] J. Denavit and R. Hartenberg. A kinematic notation for lower pair mechanisms. *J. Applied Mechanics*, 22:215–221, 1955.

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

[15] B. Dorner. Hand shape identification and tracking for sign language interpretation. In *Looking at People Workshop, IJCAI*, Chambery, France, 1993.

[16] O. Faugeras. *Three-Dimensional Computer Vision: A Geometric Viewpoint.* MIT Press, Cambridge, MA, 1993.

[17] S. Fels and G. Hinton. Building adaptive interfaces with neural networks: The glove-talk pilot study. In *Proc. of Third Intl. Conf. on Human-Computer Interaction*, pages 683–688. North-Holland.

[18] J. Foley, J. van Dam, S. Feiner, and J. Hughes. *Computer Graphics: Principles and Practice.* Addison-Wesley, 1990.

[19] B. Freeman and M. Roth. Orientation histograms for hand recognition. Technical Report TR-94-03, Mitsubishi Electric Research Lab, May 1994.

[20] D. Gennery. Tracking known three-dimensional objects. In *Proceedings of AAAI-82*, pages 13–17, 1982.

[21] D. Gennery. Visual tracking of known three-dimensional objects. *Int. J. Computer Vision*, 7(3):243–270, 1992.

[22] M. Gleicher and A. Witkin. Through-The-Lens camera control. *Computer Graphics*, 26(2):331–340, 1992.

[23] D. Hogg. Model-based vision: a program to see a walking person. *Image and Vision Computing*, 1(1):5–20, 1983.

[24] R. Holt, A. Netravali, T. Huang, and R. Qian. Determining articulated motion from perspective views: A decomposition approach. In J. Aggarwal and T. Huang, editors, *Proc. of Workshop on Motion of Non-Rigid and Articulated Objects*, pages 126–137, Austin, Texas, 1994. IEEE Computer Society Press.

[25] D. Huttenlocher, G. Klanderman, and W. Rucklidge. Comparing images using the hausdorff distance. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 15(9):850–863, September 1993.

[26] R. Jasinschi, J. Moura, and et. al. Video compression via constructs. In *Proc. Int. Conf. Acoust., Speech, and Sig. Proc.*, 1995. Accepted for publication.

[27] S. B. Kang and K. Ikeuchi. Grasp recognition using the contact web. In *Proc. IEEE/RSJ Int. Conf. on Int. Robots and Sys.*, Raleigh, NC, 1992.

[28] Micheal Kass, Andy Witkin, and Demitri Terzopoulos. Snakes: Active contour models. *Int. J. Computer Vision*, 1(4):321–331, 1987.

[29] S. Kawada, A. Sato, and et. al. Multi-image tracking of human bodies in motion. In *Proc. of National Conf., Inst. of Electronics and Communication Engineers of Japan*, pages 7D–6, Tokyo, Japan, March 1994. In Japanese.

[30] P. Khosla. *Real-time Control and Identification of Direct-Drive Manipulators*. PhD thesis, Carnegie Mellon Univ., Dept. of ECE, 1986.

[31] J. Koenderink and A. van Doorn. The singularities of the visual mapping. *Biological Cybernetics*, 24, 1976.

[32] T. Kondo, T. Yamagiwa, and et. al. Measurement of Kansei in human action based on a robot model. In *Proc. of National Conf., Inst. of Electronics and Communication Engineers of Japan*, pages 6D–6, Tokyo, Japan, March 1994. In Japanese.

[33] J. Lee and T. Kunii. Constraint-based hand animation. In Thalmann and Thalmann, editors, *Models and Techniques in Computer Animation*, pages 110–127. Springer-Verlag, 1993.

[34] X. Li and Y. Bar-Shalom. Stability evaluation and track life of the PDAF for tracking in clutter. *IEEE Trans. Automatic Control*, 36(5):588–602, May 1991.

[35] D. Lowe. Robust model-based motion tracking through the integration of search and estimation. *Int. J. Computer Vision*, 8(2):113–122, 1992.

[36] B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo. In *Seventh Int. Joint Conf. on Artificial Intelligence (IJCAI-81)*, pages 674–679, Vancouver, B.C., 1981.

[37] R. Mann and E. Antonsson. Gait analysis– precise, rapid, automatic, 3-d position and orientation kinematics and dynamics. *BULLETIN of the Hospital for Joint Diseases Orthopaedic Institute*, XLIII(2):137–146, 1983.

[38] J. McCarthy. *An Introduction to Theoretical Kinematics.* MIT Press, 1990.

[39] D. Metaxis and D. Terzopoulos. Shape and nonrigid motion estimation through physics-based synthesis. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 15(6):580–591, 1993.

[40] Y. Nakamura. *Advanced Robotics: Redundancy and Optimization.* Addison-Wesley, 1991.

[41] M. Nitzberg and D. Mumford. The 2.1-d sketch. In *Proc. Third Int. Conf. on Comp. Vision*, Osaka, Japan, 1990.

[42] J. O'Rourke and N. Badler. Model-based image analysis of human motion using constraint propagation. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 2(6):522–536, 1980.

[43] R. P. Paul. *Robot Manipulators.* MIT Press, 1981.

[44] A. Pentland and B. Horowitz. Recovery of nonrigid motion and structure. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 13(7):730–742, 1991.

[45] R. Rashid. Towards a system for the interpretation of moving light displays. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 2(6):574–581, 1980.

[46] J. Rehg and T. Kanade. Digiteyes: Vision-based human hand tracking. Technical Report CMU-CS-TR-93-220, Carnegie Mellon Univ. School of Comp. Sci., 1993.

[47] J. Rehg and T. Kanade. Digiteyes: Vision-based hand tracking for human-computer interaction. In J. Aggarwal and T. Huang, editors, *Proc. of Workshop on Motion of Non-Rigid and Articulated Objects*, pages 16–22, Austin, Texas, 1994. IEEE Computer Society Press.

[48] J. Rehg and T. Kanade. Visual tracking of high dof articulated structures: an application to human hand tracking. In J. Eklundh, editor, *Proc. of Third European Conf. on Computer Vision*, volume 2, pages 35–46, Stockholm, Sweden, 1994. Springer-Verlag.

[49] J. Rehg and T. Kanade. Visual tracking of self-occluding articulated objects. Technical Report CMU-CS-TR-94-224, Carnegie Mellon Univ. School of Comp. Sci., 1994.

[50] J. Rehg and T. Kanade. Visual tracking of self-occluding articulated objects. In *Proc. Intl. Conf. Computer Vision*, Boston, MA, 1995. Accepted for publication.

[51] J. Rehg and A. Witkin. Visual tracking with deformation models. In *Proc. of the IEEE Intl. Conf. on Robotics and Automation*, pages 844–850, Sacramento, CA, April 1991.

[52] H. Rijpkema and M. Girard. Computer animation of knowledge-based human grasping. *Computer Graphics*, 25(4):339–348, 1991.

[53] L. Robert. Camera calibration without feature extraction. *Computer Vision Graphics and Image Processing*, 1995. Accepted for Publication.

[54] B. Robertson. Caught in the act. *Computer Graphics World*, pages 23–28, September 1994.

[55] A. Rosenfeld and A. Kak. *Digital Picture Processing*. Academic Press, New York, 1982. 2nd ed.

[56] J. Segen. Gest: A learning computer vision system that recognizes hand gestures. In R. Michalski and G. Tecuci, editors, *Machine Learning IV*. Morgan Kaufmann, 1993.

[57] T. Shakunaga. Pose estimation of jointed structures. In *IEEE Conf. on Comput. Vis. and Patt. Rec.*, pages 566–572, Maui, Hawaii, 1991.

[58] K. Shoemake. Animating rotations with quaternion curves. *Computer Graphics*, 19(3):245–254, 1985.

[59] M. Spong. *Robot Dynamics and Control*. John Wiley and Sons, 1989.

[60] J. Stuelpnagel. On the parameterization of the three-dimensional rotation group. *SIAM Review*, 6(4):422–430, 1964.

[61] D. Sturman. *Whole-Hand Input*. PhD thesis, Massachusetts Institute of Technology, 1992.

[62] R. Szeliski. *Bayesian Modeling of Uncertainty in Low-Level Vision.* Kluwer Academic Pub., Boston, MA, 1989.

[63] R. Szeliski and D. Terzopoulos. Physically-based and probabilistic modeling for computer vision. In B. Vemuri, editor, *Proc. SPIE 1570, Geometric Methods in Computer Vision*, pages 140–152, San Diego, CA, 1991.

[64] D. Terzopoulos and R. Szeliski. Tracking with kalman snakes. In A. Blake and A. Yuille, editors, *Active Vision*, pages 3–20. MIT Press, 1992.

[65] D. Terzopoulos, A. Witkin, and M. Kass. Constraints on deformable models: Recovering 3D shape and non-rigid motion. *Artificial Intelligence*, 36(1):91–123, 1988.

[66] D. Thompson, W. Buford, and et. al. A hand biomechanics workstation. *Computer Graphics*, 22(4):335–343, 1988.

[67] J. Wang and E. Adelson. Layered representation for motion analysis. In *Proc. IEEE Conf. Comput. Vis. and Pattern Rec.*, pages 361–366, 1993.

[68] J. Webb and J. Aggarwal. Structure from motion of rigid and jointed objects. *Artificial Intelligence*, 19:107–130, 1982.

[69] J. Wertz. *Spacecraft Attitude Determination and Control.* Reidel, Boston, 1978.

[70] B. Widrow. The "Rubber-Mask" technique— I. Pattern measurement and analysis. *Pattern Recognition*, 5:175–197, 1973.

[71] A. Witkin. 1992. Personal communication.

[72] M. Yamamoto and K. Koshikawa. Human motion analysis based on a robot arm model. In *IEEE Conf. Comput. Vis. and Pattern Rec.*, pages 664–665, 1991. Also see Electrotechnical Laboratory Report 90-46.

[73] A. Yuille. Deformable templates for face recognition. *J. of Cognitive Neuroscience*, 3(1):59–70, 1991.

[74] T. Zimmerman, J. Lanier, C. Blanchard, S. Bryson, and Y. Harvill. A hand gesture interface device. In *Proc. Human Factors in Comp. Sys. and Graphics Interface (CHI+GI'87)*, pages 189–192, Toronto, Canada, 1987.