DigitEyes: Vision-Based Human Hand Tracking

James M. Rehg Takeo Kanade

31 December 1993
CMU-CS-93-220

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

A portion of this report will appear in *Proceedings of European Conf. on Computer Vision*, May 1994, Stockholm, Sweden.

This research was 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 authors and should not be interpreted as representing the official policies, either expressed or implied, of NASA or the U.S. government.

Keywords: Human Motion Analysis, Human-Computer Interaction, Nonrigid Motion, Gesture Recognition, Visual Tracking, Model-Based Vision

Abstract

Passive sensing of human hand and limb motion is important for a wide range of applications from human-computer interaction to athletic performance measurement. High degree of freedom articulated mechanisms like the human hand are difficult to track because of their large state space and complex image appearance. This article describes a model-based hand tracking system, called *DigitEyes*, that can recover the state of a 27 DOF hand model from gray scale images at speeds of up to 10 Hz. We employ kinematic and geometric hand models, along with a high temporal sampling rate, to decompose global image patterns into incremental, local motions of simple shapes. Hand pose and joint angles are estimated from line and point features extracted from images of unmarked, unadorned hands, taken from one or more viewpoints. We present some preliminary results on a 3D mouse interface based on the *DigitEyes* sensor.

Contents

1 Introduction

Sensing of human hand and limb motion is important in applications from Human-Computer Interaction (HCI) to athletic performance measurement. Current commercially available solutions are invasive, and require the user to don gloves [19] or wear targets [10]. This paper describes a noninvasive visual hand tracking system, called *DigitEyes*. We have demonstrated hand tracking at speeds of up to 10 Hz using line and point features extracted from gray scale images of unadorned, unmarked hands.

Most previous real-time visual 3D tracking work has addressed objects with 6 or 7 spatial degrees of freedom (DOF)[7, 9]. We present tracking results for branched kinematic chains with as many as 27 DOF (in the case of a human hand model). We show that simple, useful features can be extracted from natural images of the human hand. While difficult problems still remain in tracking through occlusions and across complicated backgrounds, these results demonstrate the potential of vision-based human motion sensing.

This paper has two parts. First, we describe the 3D visual tracking problem for objects with kinematic chains. Second, we show experimental results of tracking a 27 DOF hand model using two cameras, and describe a simple 3D mouse interface using a single camera.

2 The Articulated Mechanism Tracking Problem

Visual tracking is a sequential estimation problem: given an image sequence, recover the time-varying state of the world [7, 9, 18]. The solution has three basic components: state model, feature measurement, and state estimation. The state model specifies a mapping from a state space, which characterizes all possible spatial configurations of the mechanism, to a feature space. For the hand, the state space encodes the pose of the palm (seven states for quaternion rotation and translation) and the joint angles of the fingers (four states per finger, five for the thumb), and is mapped to a set of image lines and points by the state model. A state estimate is calculated for each image by inverting the model to obtain the state vector that best fits the measured features. Features for the unmarked hand consist of finger link and tip occluding edges, which are extracted by local image operators.

Articulated mechanisms are more difficult to track than a single rigid object for two reasons: their state space is larger and their appearance is more complicated. First, the state space must represent additional kinematic DOFs not present in the single-object case, and the resulting estimation problem is more expensive computationally. In addition, kinematic singularities are introduced that are not present in the six DOF case. Singularities arise when a small change in a given state has no effect on the image features. They are currently dealt with by stabilizing the estimation algorithm. Second, high DOF mechanisms produce complex image patterns as their DOFs are exercised. This is illustrated in Fig. 1, where changes in the pose of a model hand are shown to yield dramatic changes in its silhouette. People exploit this observation in making shapes from shadows cast by their hands.

To reduce the complexity of the hand motion, we employ a high image acquisition rate (10-15 Hz depending on the model) which limits the change in the hand state, and therefore image feature location, between frames. As a result, state estimation and feature measurement are *local*, rather than global, search problems. In the state space, we exploit this

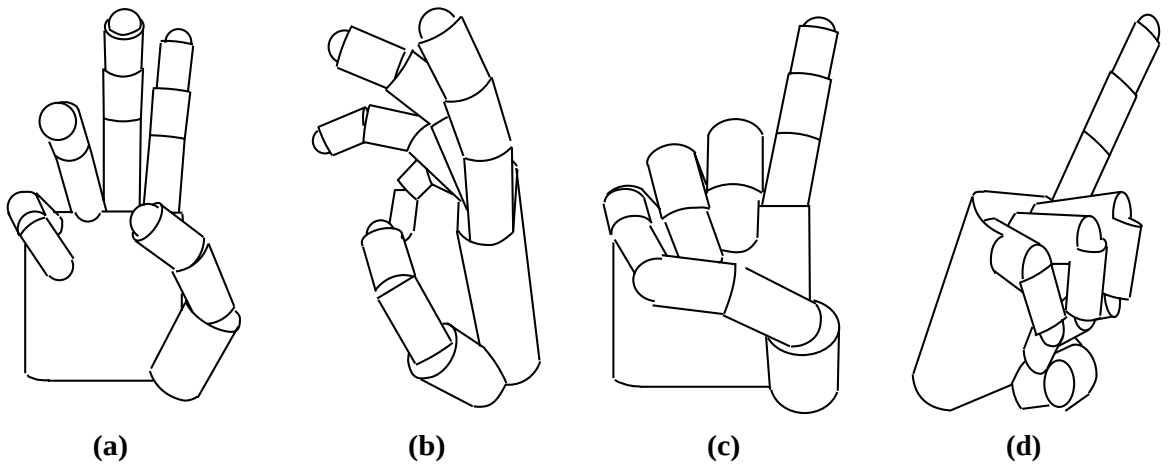


Figure 1: Changes in the hand state yield significant changes in appearance, as these four configurations of the model hand illustrate. Views (a) and (b) differ only in the pose of the hand, as do (c) and (d); while views (a) and (c) differ only in the values of the finger joint angles. Finger links are modeled with cylinders, and finger tips with hemispheres.

locality by linearizing the nonlinear state model around the previous estimate. The resulting linear estimation problem produces state corrections which are integrated over time to yield an estimated state trajectory. In the image, the projection of the previous estimate through the state model yields coordinate frames for feature extraction. We currently assume that the closest available feature is the correct match, which limits our system to scenes without occlusions or complicated backgrounds.

Previous work on tracking general articulated objects includes [18, 12, 11]. In [18], Yamamoto and Koshikawa describe a system for human body tracking using kinematic and geometric models. They give an example of tracking a single human arm and torso using optical flow features. Pentland and Horowitz [12] give an example of tracking the motion of a human figure using optical flow and an articulated deformable model. In [6], Dorner describes a system for interpreting American Sign Language from image sequences of a single hand. Dorner's system uses the full set of the hand's DOFs, and employs a glove with colored markers to simplify feature extraction. A much earlier system by O'Rourke and Badler [11] analyzed human body motion using constraint propagation. In other hand-specific work, Kang and Ikeuchi describe a range sensor-based approach to hand pose estimation [8], used in their Assembly Plan from Observation system.

Two recent works [14, 4] have addressed pose estimation of articulated objects from a single view. Dhome et. al. recover the pose of an industrial robot arm from a single image and a CAD model [4]. They use a kinematic representation that decouples rotation and translation to allow for more efficient global search of the state space. In [14], Shakunaga derives constraints on joint angles from point and line measurements and gives an algorithm for pose recovery.

In addition to this work on articulated object tracking, several authors have applied general motion techniques to human motion analysis. In contrast to *DigitEyes*, these approachs analyze a subset of the total hand motion, such as a set of gestures [2] or the rigid motion of

the palm [1]. Darrell and Pentland describe a system for learning and recognizing dynamic hand gestures in [2]. Their approach avoids the problems of hand modeling, but doesn't address 3D tracking. In [1], Blake et. al. describe a real-time contour tracking system that can follow the silhouette of a rigid hand under an affine motion model.

None of these earlier approachs have demonstrated tracking results for the full state of a complicated mechanism like the human hand, using natural image features. Although there has been a significant amount of gesture recognition work on unmarked hand images, these approachs don't produce 3D motion estimates, and it would be difficult to apply them to problems like the 3D mouse interface described in Subsect. 6.1. See [16] for several other examples of novel user interfaces based on a whole-hand sensor.

In order to apply the *DigitEyes* system to specific applications, such as HCI, two practical requirements must be met. First, the kinematics and geometry of the target hand must be known in advance, so that a state model can be constructed. Second, before local hand tracking can begin, the initial configuration of the hand must be known. We achieve this in practice by requiring the subject to place their hand in a certain pose and location to initiate tracking. A 3D mouse interface based on visual hand tracking is presented in Subsect. 6.1.

In the sections that follow, we describe the *DigitEyes* articulated object tracking system in more detail, along with the specific modeling choices required for hand tracking.

3 State Model for Articulated Mechanisms

The state model encodes all possible mechanism configurations and their corresponding image feature patterns as a two-part mapping between state and feature spaces. The first part is a kinematic model which captures all possible spatial link positions, while the second part is a feature model which describes the image appearance of each link shape.

3.1 Kinematic Model: Application to the Human Hand

We model kinematic chains, like the finger, with the Denavit-Hartenburg (DH) representation, which is widely used in robotics [15]. In this representation, each finger link has an attached link coordinate frame, and the transformations between these frames model the kinematics. Since feature models require geometric information not captured in the kinematics, the DH description of each link is augmented with an additional transform from the link frame to a *shape frame*. A solid model in the shape frame generates features through projection into the image.

We model the hand as a collection of 16 rigid bodies: 3 individual finger links (called phalanges) for each of the five digits, and a palm. From a kinematic viewpoint, the hand consists of multi-branched kinematic chains attached to a six DOF base. We make several simplifying assumptions in modeling the hand kinematics. First, we assume that each of the four fingers of the hand are 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 DOF determine the finger's configuration within the plane. Fig. 2 illustrates the planar finger model. Each finger has an *anchor point*, which is the position of its base joint center in the frame of the palm, which is assumed to be rigid. The base joint is the one farthest

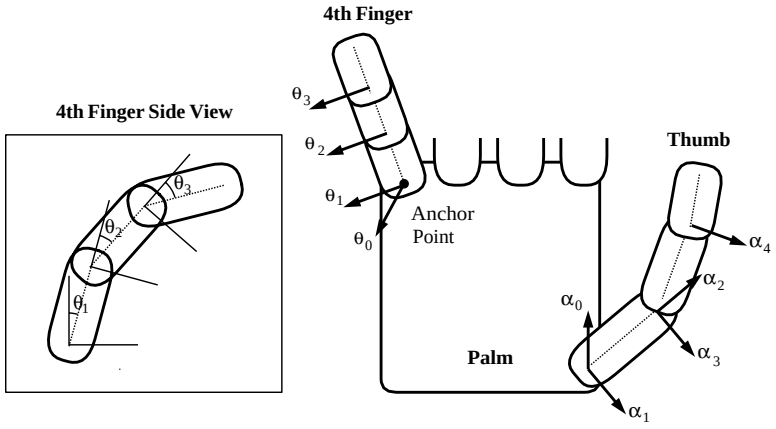


Figure 2: Kinematic models, illustrated for fourth finger and thumb. The arrows illustrate the joint axes for each link in the chain.

(kinematically) from the finger tip. We use a four parameter quaternion representation of the palm pose, which eliminates rotational singularities at the cost of a redundant parameter. The total hand pose is described by a 28 dimensional state vector.

The 3D shape of the hand is determined by the shape of its links and palm. These shapes can be given by solid models, or a class of deformable models as in [12]. Shape models are described with respect to the shape frame, which is positioned relative to the link coordinate frame. In general, the DH transform between two links is series of four transforms

$$T_i^{i+1} = \mathrm{Rot}_{z,\theta_i}\mathrm{Trans}_{z,d_i}\mathrm{Trans}_{x,a_i}\mathrm{Rot}_{x,\alpha_i} \quad . \tag{1}$$

In our framework, the shape frame is located after the first transform, and so the kinematic to shape frame transform is just $\mathrm{Rot}_{z,\theta_i}$.

The thumb is the most difficult digit to model, due to its great dexterity and intricate kinematics. We currently employ the thumb model used in Rijpkema and Girard's grasp modeling system [13] (see Fig. 2). They were able to obtain realistic animations of human grasps using a five DOF model. The DH parameters for the first author's right hand, used in the experiments, can be found in Table 1.

Real fingers deviate from our modeling assumptions in three ways. First, most fingers deviate slightly from planarity. This deviation could be modeled with additional kinematic transforms, but we have found the planar approximation to be adequate in practice. Second, the last two joints of the finger, counting from the palm outwards, are driven by the same tendon and are not capable of independent actuation. It is simpler to model the DOF explicitly, however, than to model the complicated angular relationship between the two joints. The third and most significant modeling error is change in the anchor points during motion. We have modeled the palm as a rigid body, but in reality it can flex. In gripping a baseball, for example, the palm will conform to its surface, causing the anchor points to deviate from their rest position by tens of millimeters. Fortunately, for free motions of the hand in space, the deviation seems to be small enough to be tolerated by our system.

The modeling framework we employ is general. To track an arbitrary articulated structure, one simply needs its DH parameters and a set of shape models that describe its visual

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	–
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
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 1: Kinematic and shape parameters for the first finger and thumb of the first author's right hand, which are used in the experiments. State variables are denoted $\mathbf{q}_i$, where $\mathbf{q}_0$-$\mathbf{q}_3$ contain the quaternion for palm rotation and $\mathbf{q}_4$-$\mathbf{q}_6$ contain palm translation. The "Next" field gives the number of the next frame in the kinematic chain. The other three fingers are similar to the first.

appearance. Within the subproblem of hand tracking, this allows us to develop a suite of hand models whose DOFs are tailored to specific applications.

3.2 Feature Model: Description of Hand Images

The output of the hand state model is a set of features consisting of lines and points generated by the projection of the hand model into the image plane. Each finger link, modeled by a cylinder, generates a pair of lines in the image corresponding to its occlusion boundaries. The bisector of these lines, which contains the projection of the cylinder central axis, is used as the link feature. The link feature vector $[a\, b\, \rho]$ gives the parameters of the line equation $ax + by - \rho = 0$. Using the central axis line as the link feature eliminates the need to model the cylinder radius or the slope of the pair of lines relative to the central axis, which is often significant near the finger tips. We use the entire line because the endpoints are difficult to measure in practice. Fig. 3 shows two link feature lines extracted from the first two links of a finger.

Each finger tip, modeled by a hemisphere, generates a point feature by projection of the center into the image. The finger tip feature vector $[x\, y]$ gives the tip position in image coordinates, as illustrated in Fig. 3. The total hand appearance is described by a $(3m + 2n)$-dimensional vector, made up of link and tip features, where m and n are the number of

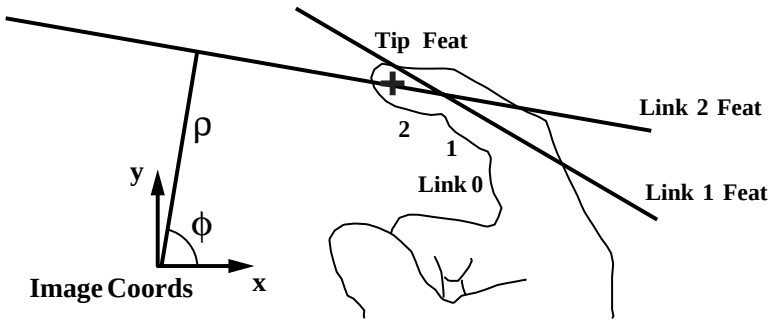


Figure 3: Features used in hand tracking are illustrated for finger links 1 and 2, and the tip. Each infinite line feature is the projection of the finger link central axis.

finger links and tips, respectively, in the model.

Other feature choices for hand tracking are possible, but the occlusion contours are the most powerful cue. Hand albedo tends to be uniform, making it difficult to use correlation features. Shading is potentially valuable, but the complicated illuminance and self-shadowing of the hand make it difficult to use.

4 Feature Measurement: Detection of Finger Links and Tips

Local image-based trackers are used to measure hand features. These trackers are the projections of the spatial hand geometry into the image plane, and they serve to localize and simplify feature extraction. A finger link tracker, drawn as a "T"-shape, is depicted along with its measured line feature in Fig. 4. The stem of the "T" is the projection of the cylinder center axis into the image. The image sampling rate ensures that the true feature location is near the projected tracker.

Once the link tracker has been positioned, line features are extracted by sampling the image in slices perpendicular to the central axis. 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. 5. Line fitting to each set of two or more detected intersections gives a measurement of the projected link 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.

Using local trackers and sampling along lines in the image reduces the pixel processing requirements of feature measurement, permitting fast tracking.

5 State Estimation for Articulated Mechanisms

State estimation proceeds by making incremental state corrections between frames. One cycle of the estimation algorithm goes as follows: The current state estimate is used to predict feature locations in the next frame and position feature trackers. After image acquisition and feature extraction, measured and predicted feature values are compared to produce a state correction, which is added to the current estimate to obtain a new state estimate. The difference between measured and predicted states is modeled by a residual vector, and the state correction is obtained by minimizing its magnitude squared. A high image sampling rate allows us to linearize the nonlinear mapping from state to features around an operating point, which is recomputed at each frame, to obtain a linear least squares problem in the model Jacobian. The following subsections describe the residual model and estimation algorithm in detail.

5.1 Residual Model: Link and Tip Image Alignment

The tip residual measures the Euclidean distance in the image between predicted ($\mathbf{c}_i$) and measured ($\mathbf{t}_i$) tip positions. The residual for the ith tip feature is a vector in the image plane defined by

$$\mathbf{v}_i(\mathbf{q}) = \mathbf{c}_i(\mathbf{q}) - \mathbf{t}_i \ , \tag{2}$$

where $\mathbf{c}_i$ is the projection of the tip center into the image as a function of the hand state.

The link residual is a scalar that measures the deviation of the projected cylinder axis from the measured feature line. It is illustrated for a single finger link in Fig. 4. The residual at a point along the axis equals the perpendicular distance to the feature line. We incorporate the orthographic camera model into the residual equation by setting $\mathbf{m} = [a\, b\, 0]^t$ and writing

$$l_i(\mathbf{q}) \ = \ \mathbf{m}^t \mathbf{p}_i(\mathbf{q}) - \rho \ , \tag{3}$$

where $\mathbf{p}_i(\mathbf{q})$ is the 3D position of a point on the cylinder link in camera coordinates, and $[a\ b\ \rho]$ are the line feature parameters. The total link residual consists of one or more point residuals along the cylinder axis (at the base and tip), each given by (3). Note that both residuals are linear in the model point positions.

The feature residuals for each link and tip in the model are concatenated into a single residual vector, $\mathbf{R}(\mathbf{q})$. If the magnitude of the residual vector is zero, the hand model is perfectly aligned with the image data.

5.2 Estimation Algorithm: Nonlinear Least Squares

The state correction is obtained from the residual vector by minimizing

$$\mathbf{H}(\mathbf{q}) = \frac{1}{2} \| \mathbf{R}(\mathbf{q}) \|^2 \ . \tag{4}$$

We employ a modified Gauss-Newton (GN) algorithm to solve this nonlinear least squares problem [3]. The source of nonlinearity in the state model for articulated mechanisms is

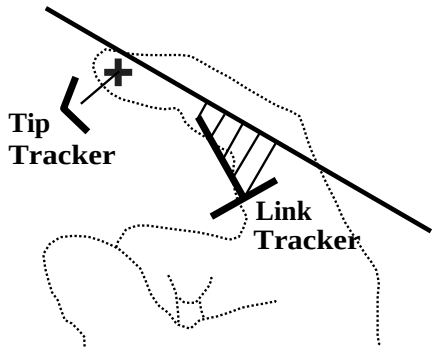


Figure 4: Image trackers, detected features, and residuals for a link and a tip are shown using the image from Fig. 3. Slashed lines denote the link residual error between the T-shaped tracker and its extracted line measurement. Similarly, the tip tracker (carat shape) is connected to its point feature (cross) by a residual vector.

trigonometric terms in the forward kinematic model. The other source of nonlinearity, inverse depth coefficients in the perspective camera model, is absent in our orthographic formulation.

Let $\mathbf{R}(\mathbf{q}_j)$ be the residual vector for image j. The GN state update equation is given by

$$\mathbf{q}_{j+1} = \mathbf{q}_j - [\mathbf{J}_j^t \mathbf{J}_j + \mathbf{S}]^{-1} \mathbf{J}_j^t \mathbf{R}_j \quad , \tag{5}$$

where $\mathbf{J}_j$ is the Jacobian matrix for the residual $\mathbf{R}_j$, both of which are evaluated at $\mathbf{q}_j$. $\mathbf{S}$ is a constant diagonal conditioning matrix used to stabilize the least squares solution. $\mathbf{J}_j$ is formed from the link and tip residual Jacobians. The same basic approach was used by Lowe in his rigid body tracking system [9].

Other tracking work has employed Kalman Filtering to incorporate dynamic constraints into state estimation [1, 7, 17, 5]. The update rule in (5) can be viewed as the limiting case of this filter, in which the estimate is a function of the measurements alone. The complicated dynamics of the hand and its ability to accelerate rapidly weaken the effectiveness of dynamic constraints (compared, for example, to satellite tracking problems). Time smoothing may be useful in some applications, but the kinematic hand model provides a much stronger constraint on feature locations and potential matchs.

In the remainder of this section, we derive the link and tip Jacobians and discuss their computation. To calculate the link Jacobian we differentiate (3) with respect to the state vector, obtaining

$$\frac{\partial l_i(\mathbf{q})}{\partial \mathbf{q}} \;=\; \mathbf{m}^t \frac{\partial \mathbf{p}_i(\mathbf{q})}{\partial \mathbf{q}} \quad . \tag{6}$$

The above gradient vector for link i is one row of the total Jacobian matrix. Geometrically, it is formed by projecting the *kinematic Jacobian* for points on the link, $\partial \mathbf{p}_i(\mathbf{q})/\partial \mathbf{q}$, in the direction of the feature edge normal. Similarly, the tip Jacobian is obtained as

$$\frac{\partial \mathbf{v}_i(\mathbf{q})}{\partial \mathbf{q}} \;=\; \frac{\partial \mathbf{p}_i(\mathbf{q})}{\partial \mathbf{q}} \quad . \tag{7}$$

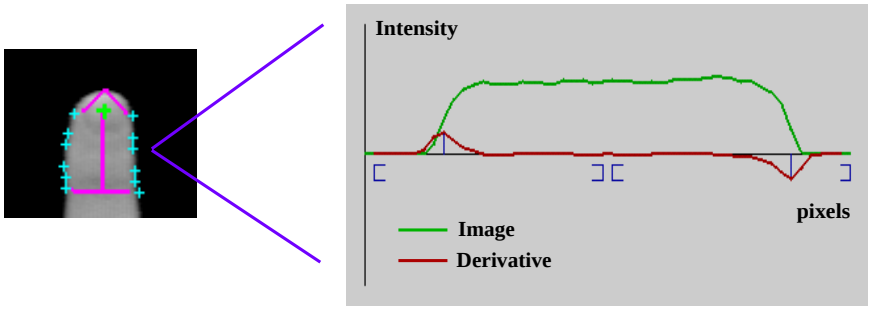


Figure 5: A single link tracker is shown along with its detected boundary points. One slice through the finger image of a finger is also depicted. Peaks in the edge locations.

The kinematic Jacobians in (6) and (7) are composed of terms of the form $\partial \mathbf{p}_i / \partial \mathbf{q}_j$, which arise frequently in robot control. As a result, these Jacobian entries can be obtained directly from the model kinematics by means of some standard formulas (see [15], Chapter 5). There are three types of Jacobians, corresponding to joint rotation, spatial translation, and spatial rotation DOFs. All points must be expressed in the frame of the camera producing the measurements. For a revolute (rotational) DOF joint $\mathbf{q}_j$ we have

$$\frac{\partial \mathbf{p}_i}{\partial \mathbf{q}_j} = \mathbf{w}_j \times (\mathbf{p}_i - \mathbf{d}_c^j) \ , \tag{8}$$

where $\mathbf{w}_j$ is the rotation axis for joint j expressed in the camera frame, and $\mathbf{d}_c^j$ is the position of the joint j frame in camera coords. There will be a similar calculation for *each* camera being used to produce measurements.

The Jacobian calculation for the palm DOFs must reflect the fact that palm motion takes place with respect to the world coordinate frame, but must be expressed in the camera frame. We obtain the translation component as

$$\frac{\partial \mathbf{p}_i}{\partial \mathbf{v}} = \mathbf{R}_c^w \ , \tag{9}$$

where $\mathbf{v}$ is the palm velocity with respect to the world frame and $\mathbf{R}_c^w$ is the camera to world rotation. Similarly, if $\mathbf{q}_j$ is a component of the quaternion specifying palm rotation, we obtain

$$\frac{\partial \mathbf{p}_i}{\partial \mathbf{q}_j} = [\mathbf{R}_c^w \mathbf{J}_w]_j \times \mathbf{p}_i \ , \tag{10}$$

where $\mathbf{J}_w$ is a Jacobian mapping quaternion velocity to angular velocity, and $[\cdot]_j$ denotes the jth column of a matrix.

The details of the derivation are contained in Appendix A.

5.3 Tracking with Multiple Cameras

The tracking framework presented above generalizes easily to 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 can exploit 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.

6 Experimental Results

To test the articulated tracking framework described above, we developed two hand tracking systems based on reduced and full-state hand models, using one and two cameras. The reduced hand model was used with a single camera to provide input to a 3D cursor interface. The full hand model was tracked using two image sequences. In both cases we provide recorded state trajectory estimates along with graphical output.

6.1 3D Graphical Mouse Using a Single Camera

For the first tracking experiment, we applied the *DigitEyes* system to a 3D mouse interface problem. Figure 6 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, the 3D cursor position is controlled by clever use of a standard 2D physical mouse. Normal mouse motion controls the pole base position in the plane, while depressing one of the mouse buttons switchs 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.

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. 7. 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 controled 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".

Figures 8 – 10 give experimental results from a 500 frame motion sequence in which the estimated hand state was used to drive the 3D mouse interface (Implementation details are given in Sec. 7.) Figures 8 and 9 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. 9. The motion sequence has four phases. In the first phase (frame 0 to 150), the user's finger

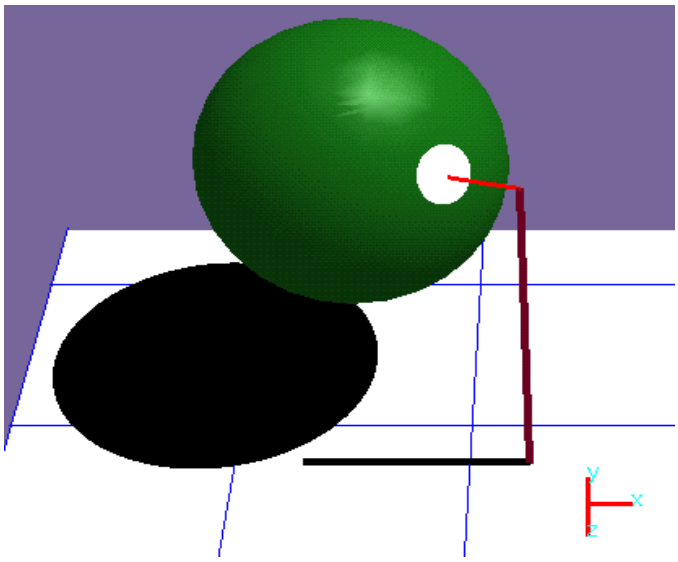


Figure 6: A sample graphical environment for a 3D mouse. The 3D cursor is at the tip of the "mouse pole", which sits atop the ground plane (in the foreground, at the right). The sphere is an example of an object to be manipulated, and the line drawn from the mouse to the sphere indicates its selection for manipulation.

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 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. 10 (at the end of the report.) This is the same scene as in Fig. 6, except that the mouse pole height and position change as a function of the estimated hand state. A hand image from the middle of the sequence (frame 200) is shown in Fig. 7 along with the estimated hand model state.

These results demonstrate fairly good decoupling between the desired states and a useful dynamic range of motion. The largest coupling error occurs around frame 150 when the pole height drops as the thumb is actuated. This coupling could be compensated for by storing a list of estimated pole heights and restoring the height to its previous value when the onset of thumb actuation is detected. In this experiment, the mouse state is generated from the hand state by a simple scaling and coordinate change. An unfortunate side-effect of scaling is to amplify the noise in the estimator. More sophisticated schemes based on smoothing the state prior to its use would likely improve the output quality.

This example illustrates an important advantage of hand tracking with kinematic models: absolute 3D distances (such as finger height above a table) can be measured from a single camera image. The ability to recover 3D spatial quantities from hand motion is one of the advantages our system has over approachs based on gesture recognition.

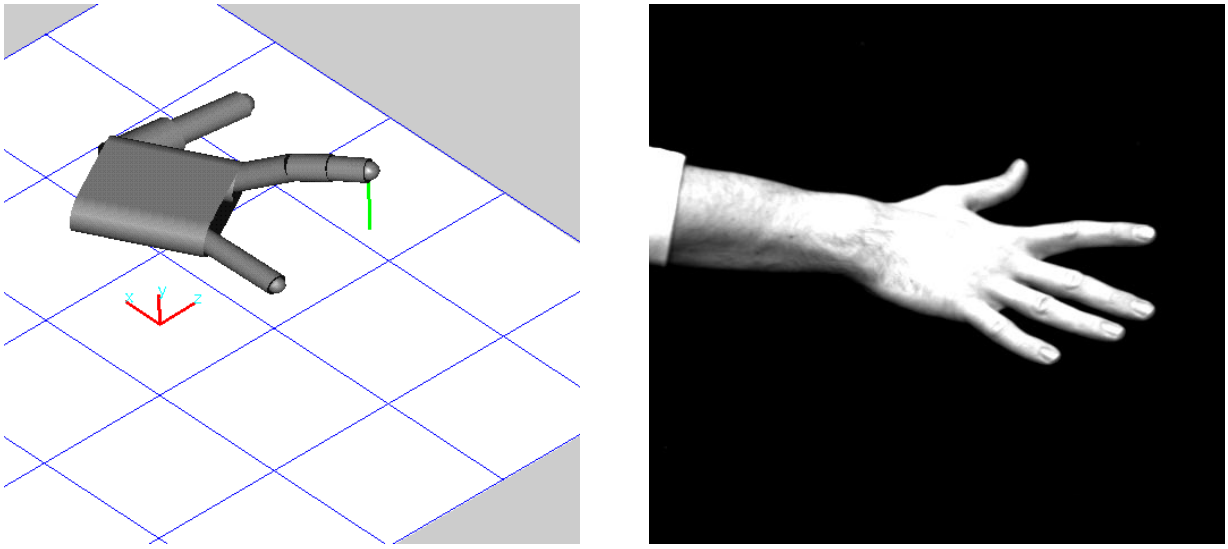


Figure 7: The hand model used in the 3D mouse application is illustrated for frame 200 in the motion sequence from Fig. 9. The vertical line shows the height of the tip above the ground plane. The input hand image (frame 200) demonstrates the finger motion used in extending the cursor height.

6.2 Whole Hand Tracking With Two Cameras

In the second tracking experiment, the *DigitEyes* system was used to track a full 27 DOF hand model, using two camera image sequences. Because the hand motion must avoid occlusions for successful tracking, the available range of travel is not large. It is sufficient, however, to demonstrate recovery of articulated DOFs in conjunction with palm motion. Figure 11 shows sample images, trackers, and features from both cameras at three points along a 200 frame sequence. The two cameras were set up about a foot and a half apart with optical centers verging near the middle of the tracking area, intersecting the table surface at approximately 45 degrees. Fig. 12 shows the estimated model configurations corresponding to the sample points. In the left column, the estimated model is rendered from the viewpoint of the first camera. In the right column, it is shown from an arbitrary viewpoint, demonstrating the 3D nature of our tracking result. A subset of the estimated state trajectories for the motion sequence are given in Figs. 13 and 14.

Direct measurement of tracker accuracy is difficult due to the lack of ground truth data. We plan to use a Polhemus sensor to measure the accuracy of the 6 DOF palm state estimate. Obtaining ground truth measurements for joint angles is much more difficult. One possible solution is to wear an invasive sensor, like the DataGlove, to obtain a baseline measurement. By fitting the DataGlove inside a larger unmarked glove, the effect of the external finger sensors on the feature extraction can be minimized.

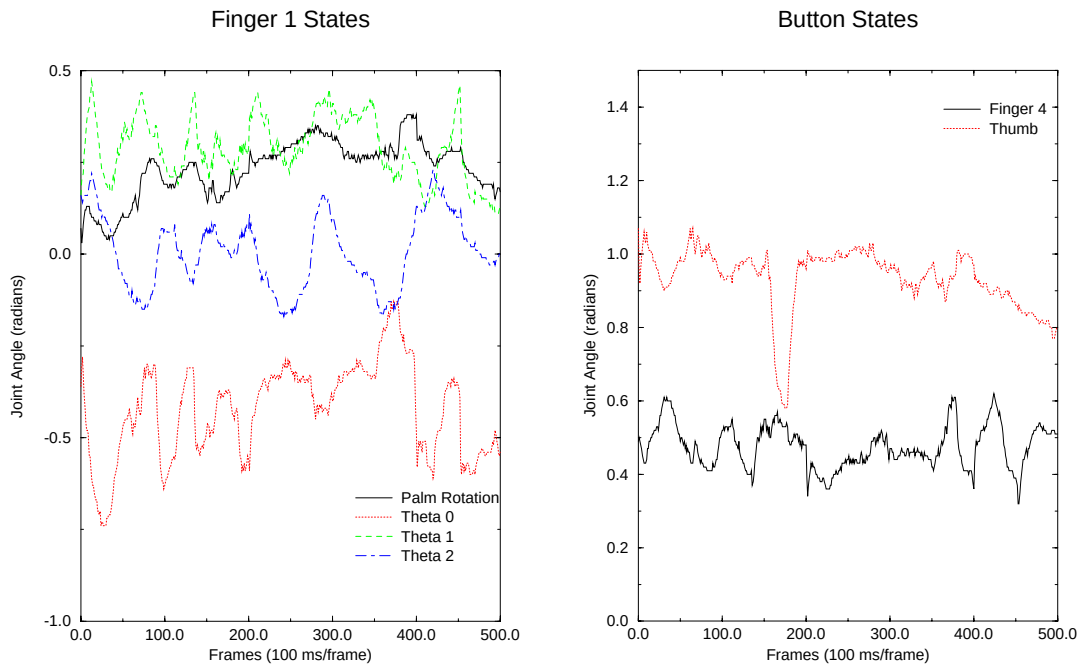


Figure 8: Palm rotation and finger joint angles for mouse pole hand model depicted in Fig. 7. Joint angles for thumb and fourth finger, shown on right, are used as buttons. Note the "button event" signaled by the thumb motion around frame 175.

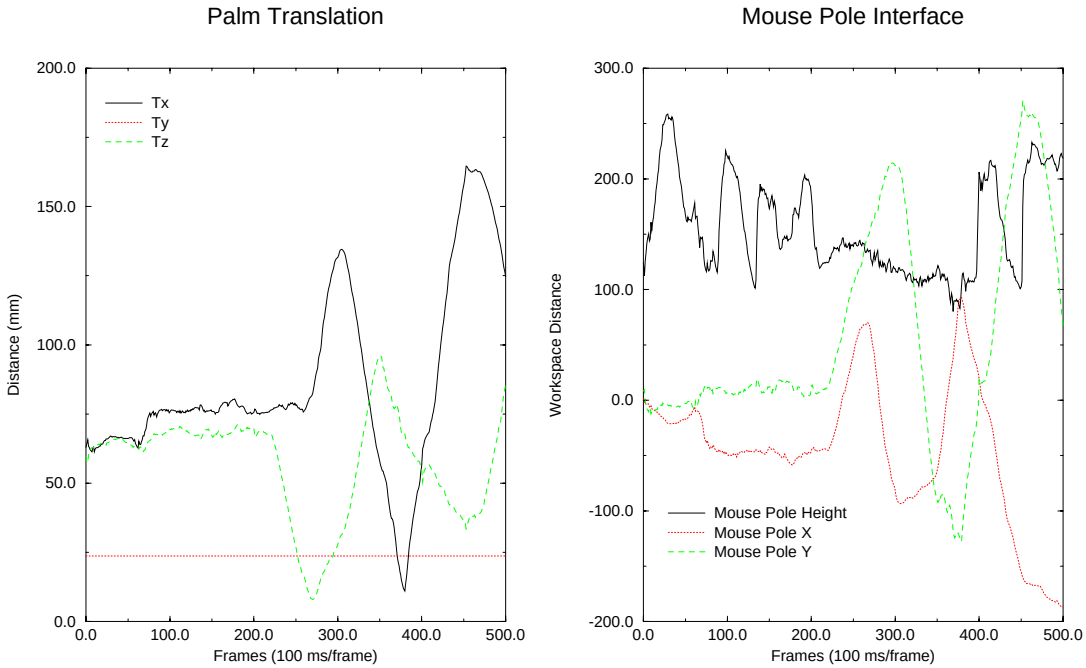


Figure 9: Translation states for mouse pole hand model are given on the left. The Y axis motion is constrained to zero due to tabletop. On the right are the mouse pole states, derived from the hand states through scaling and a coordinate change. The sequence events goes: 0-150 finger raise/lower, 150-200 thumb actuation only, 200-350 base translation only, 350-500 combined 3 DOF motion.

7 Implementation Details

The *DigitEyes* system is built around a special board for real-time image processing, called IC40. Each IC40 board contains a 68040 CPU, 5 MB of dual-ported RAM, a digitizer, and a video generator. The key feature of this board is its ability to deliver digitized images to processor memory at video rate with no computational overhead. This removes an important bottleneck in most workstation-based tracking systems. Ordinary C code can be compiled and down-loaded to the board for execution.

In the multicamera 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 a Sun workstation, which combines them and solves the resulting least squares problem to obtain a state estimate. Estimated states are passed over the Ethernet to a Silicon Graphics Indigo 2 workstation for model rendering and display. The overall system organization is shown in Fig. 15. Our experimental testbed for hand tracking is depicted in Fig. 16.

The generality of our tracking framework is reflected in the software organization of the *DigitEyes* system. Different trackers can be generated simply by changing the kinematic description of the mechanism. Feature tracking code for the IC40 boards is generated automatically from the kinematic description. This makes it possible to experiment with a variety of kinematic models, tailored to specific hand tracking applications.

8 Conclusion

We have presented a visual tracking framework for high DOF articulated mechanisms, and its implementation in a tracking system called *DigitEyes*. We have demonstrated real-time hand tracking of a 27 DOF hand model using two cameras. We will extend this basic work in two ways. First, we will modify our feature extraction process to handle occlusions and complicated backgrounds. Second, we will analyze the observability requirements of articulated object tracking and address the question of camera placement.

References

[1] A. Blake, R. Curwen, and A. Zisserman. A framework for spatiotemporal control in the tracking of visual contours. *Int. J. Computer Vision*, 11(2):127–145, 1993.

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

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

[4] M. Dhome, A. Yassine, and J. Lavest. Pose estimation of a robot arm from one greylevel image. In *Workshop on Comput. Vis. for Space Appl.*, pages 298–310, Antibes, France, 1993.

[5] E. Dickmanns and V. Graefe. Applications of dynamic monocular machine vision. *Mach. Vis. Appl.*, 1:241–261, 1988.

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

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

[8] 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.

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

[10] 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.

[11] 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.

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

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

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

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

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

[17] J. J. Wu, R. E. Wink, T. M. Caelli, and V. G. Gourishankar. Recovery of the 3-d location and motion of a rigid object through camera image (an extended kalman filter approach). *Int. J. Computer Vision*, 2(4):373–394, 1989.

[18] 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.

[19] 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.

9 Appendix A: Spatial Transform Jacobian

Given the camera and hand position in world coordinates, we outline the derivation of the Jacobian for a point expressed in the camera frame under rotation and translation of the palm. We start with the basic result

$$\dot{\mathbf{p}}_i = \mathbf{R}_c^w \mathbf{v} + \mathbf{R}_c^w \omega \times \mathbf{p}_i, \tag{11}$$

where $\mathbf{v}, \omega$ give the velocity of the base frame in world coordinates. Eqn 9 follows immediately. Substituting the additional relation

$$\omega = \mathbf{J}_\omega \dot{\mathbf{q}}, \tag{12}$$

where $\mathbf{q}$ is the quaternion parameterization of rotation and $\mathbf{J}_\omega$ is a four by three Jacobian matrix, and differentiating with respect to $\mathbf{q}_i$ yields Eqn 10.

To obtain Eqn 12, we start with the relation

$$\dot{\mathbf{R}}(\mathbf{q}) = \mathbf{S}(\omega)\mathbf{R}(\mathbf{q}),$$

and solve it for $\mathbf{S}(\omega)$, a skew symmetric matrix in the angular velocity. The other side is then a matrix of linear equations in the $\dot{\mathbf{q}}_i$. Eqn 12 results from equating the individual components of ω with their linear representations in $\dot{\mathbf{q}}$.

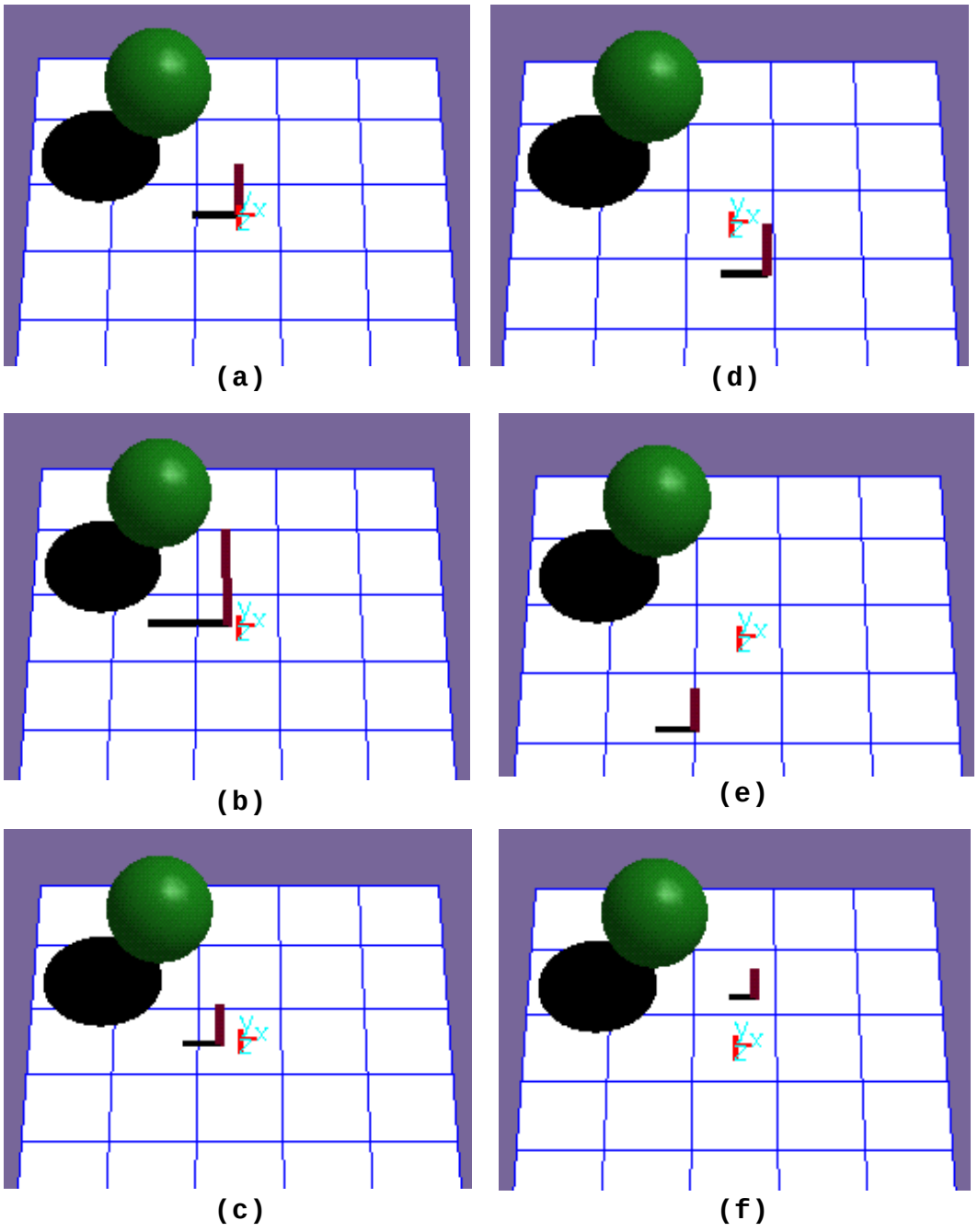
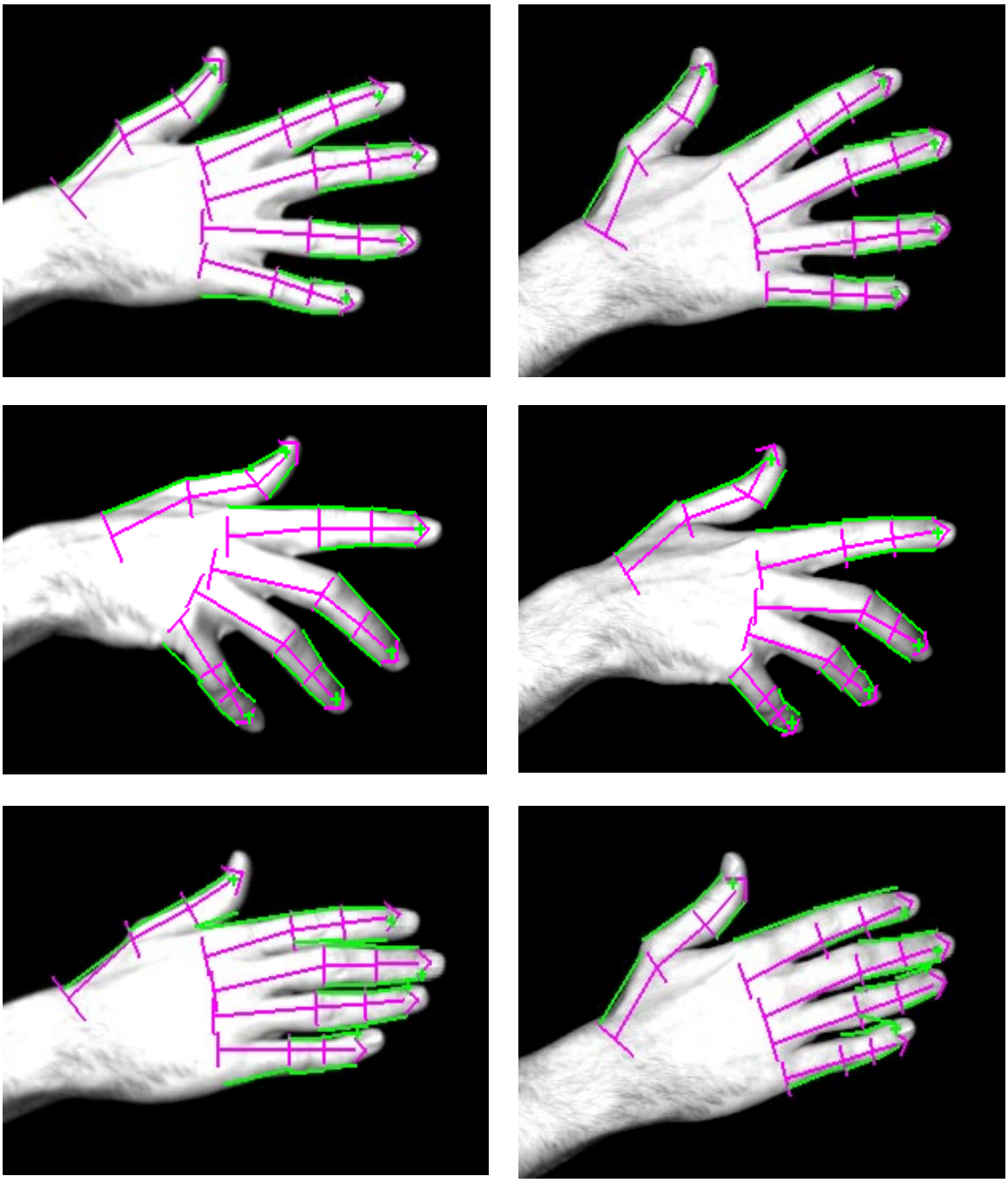


Figure 10: The mouse pole cursor at six positions during the motion sequence of Fig. 8. The pole is the vertical line with a horizontal shadow, and is the only thing moving in the sequence. Samples were taken at frames 0, 30, 75, 260, 300, and 370 (chosen to illustrate the range of motion).



Camera 0 View **Camera 1 View**

Figure 11: Three pairs of hand images from the continuous motion estimate plotted in Figs. 13 and 14. 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.

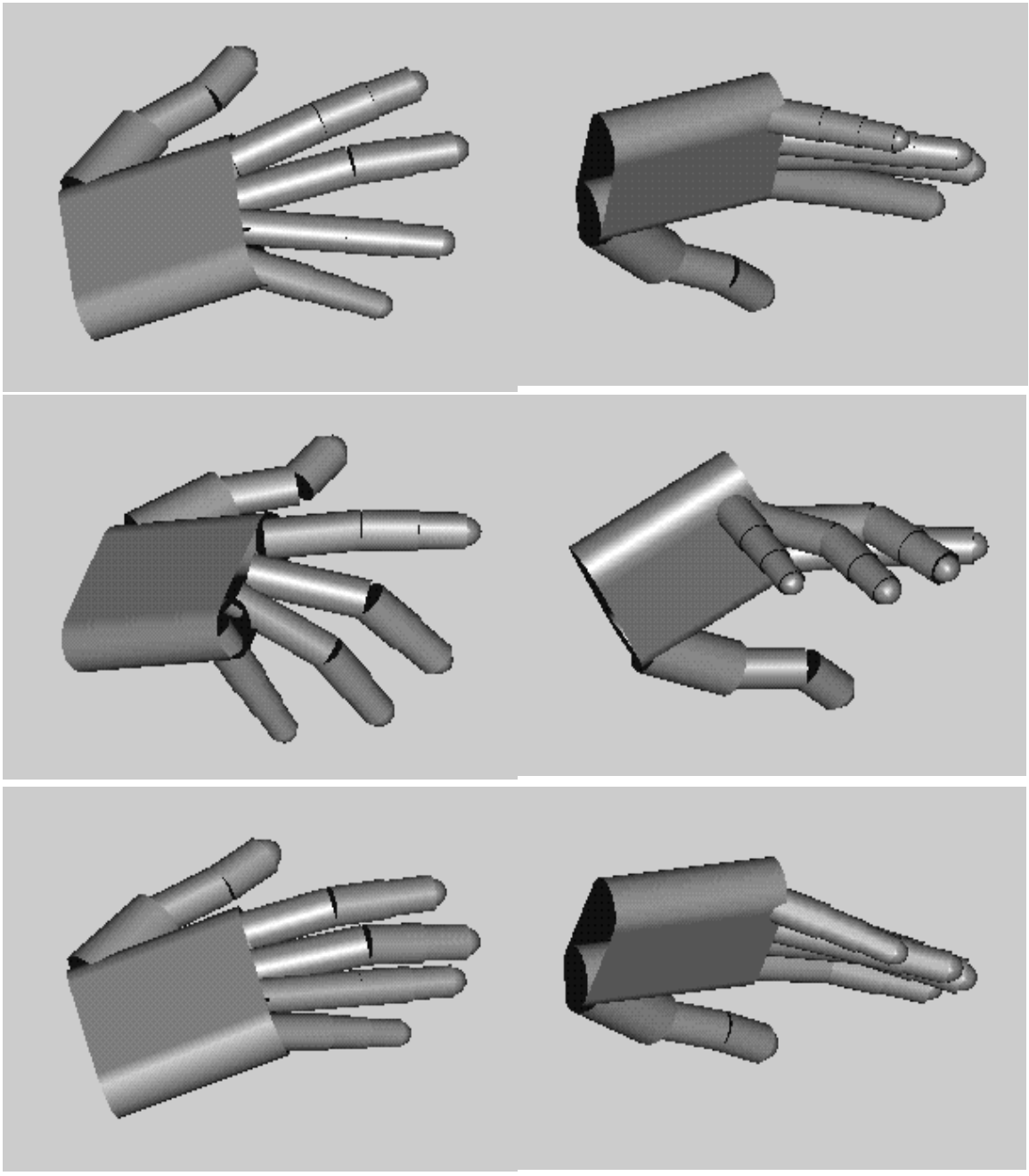


Figure 12: Estimated hand state for the image samples in Fig. 11, rendered from the Camera 0 viewpoint (left) and a viewpoint underneath the hand (right).

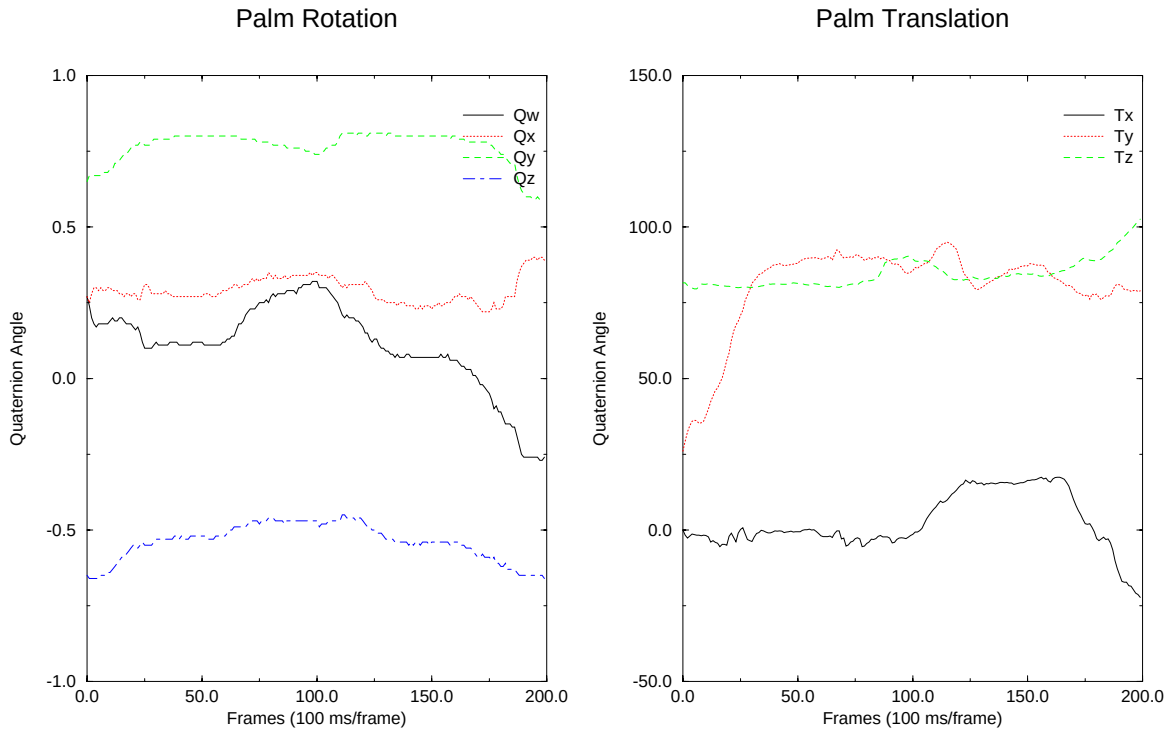


Figure 13: Estimated palm rotation and translation for motion sequence of entire hand. $\mathbf{Q}_w$-$\mathbf{Q}_z$ are the quaternion components of rotation, while $\mathbf{T}_x$-$\mathbf{T}_z$ are the translation. The sequence lasted 20 seconds.

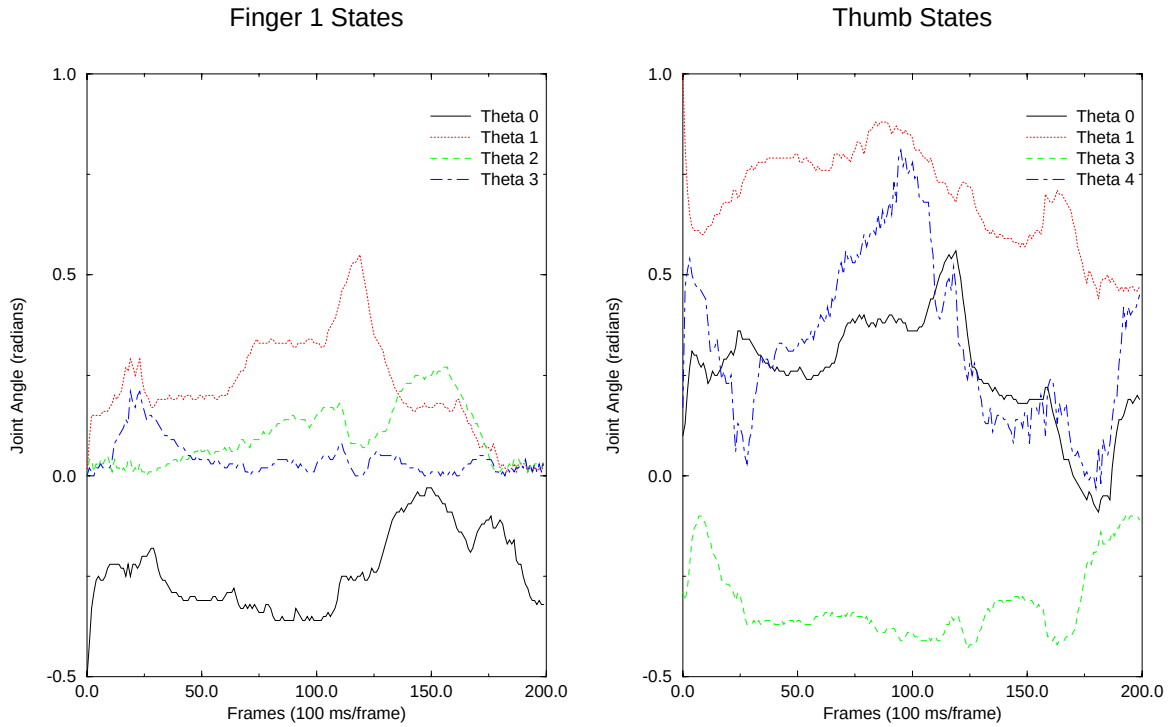


Figure 14: Estimated joint angles for the first finger and thumb. The other three fingers are similar to the first. Refer to Fig. 2 for variable definitions.

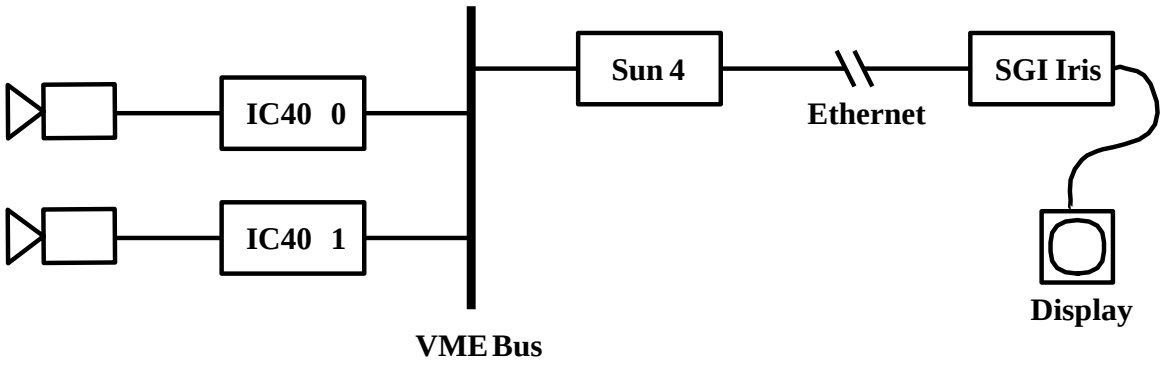


Figure 15: The hardware architecture for our current hand tracking system.

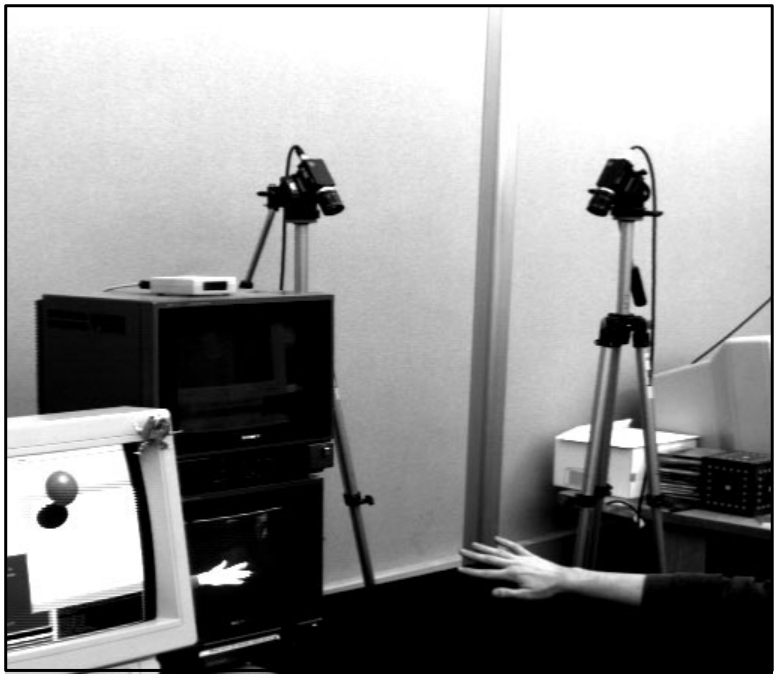


Figure 16: Experimental test bed for the *DigitEyes* system.