Monocular Tracking of the Human Arm in 3D: Real-Time Implementation and Experiments

Enrico Di Bernardo[†‡], Luis Goncalves[†], and Pietro Perona[†‡]

† California Institute of Technology, 116-81, Pasadena, CA 91125, USA
‡ Università di Padova, Italy
{dibe,luis,perona}@vision.caltech.edu

Abstract

We have developed a system capable of tracking a human arm in 3D and in real time. The system is based on a previously developed algorithm for 3D tracking which requires only a monocular view and no special markers on the body [9]. In this paper we describe our real-time system and the insights gained from real-time experimentation.

1. Introduction and Motivation

Observing the human body in motion is key to a large number of activities and applications such as *security*, *character animation*, *virtual reality*, *human-machine interfaces*, and *biomechanics studies*.

All of the current techniques for tracking the human body require either employing dedicated human operators or using ad-hoc sensors. This results in a number of limitations:*practicality* (the user needs to wear markers or other ad-hoc equipment which may be impractical, uncomfortable, constrain the user to a limited work space, be difficult to transport),*cost* (computational and sensory hardware and human operator time),*timeliness* (the data may not be available in real-time, but only after a lag required to process a batch of images, allow communication between human operators).

Our goal is to make tracking of the human body cheaper, more practical and faster by making it automatic and non-invasive.

1.1. Automatic human motion estimation

Previous work on human motion estimation using vision can be coarsely grouped into three types :

- gesture classification [6, 7]
- systems which track or classify periodic motions with 1 degree of freedom [18, 15, 16]
- estimation of 3D unconstrained motion [8]; of the hand from a monocular view [17, 14]; of the body, with the use of multiple cameras and special markers [3]

We are interested in estimating 3D unconstrained motion. In particular we want to study how accurately can one track in 3d the human body with the simplest, cheapest, and most convenient setup: a single gray-scale camera and no special markers.

We describe here a real-time system based on previous work of estimating the motion of a human arm. In a batch experiment [9], we determined that our method was able to track the arm with a depth accuracy of 8% w.r.t the workspace dimension. Since the arm itself can be used as a 3D mouse, we felt that there was much insight to be gained by implementing our algorithm in real-time. In this paper we describe the real-time implementation of our arm tracker, and the results of a number of experiments designed to asses its performance.

2. The estimation system

We model the arm in 3D and use the current estimate of arm position to predict the arm projection in the image. The difference between the predicted image and the actual arm image is used as an error measurement to update the estimated arm position with a recursive estimator. Thus, rather than extracting features explicitly from the image, we make direct comparisons between the actual image and the expected image. This method is inspired by Dickmann's work on lane following [5].

2.1. The arm model

In order to generate the predicted image, we need to "render" a 3D model of the arm from the camera's point of view. We choose a simple 3D model (Fig.1) in which the upper and lower arm are modeled as truncated right-circular cones, and the shoulder and elbow joints are modeled as spherical joints.

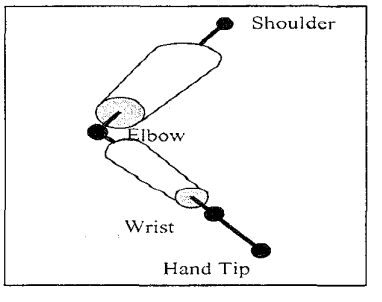

Figure 1. ***The arm model:*** *Limbs are modeled as truncated right-circular cones. The elbow and shoulder joints are modeled as spherical joints, and the hand tip is assumed to be along the forearm axis.*

1015-4651/96 $5.00 © 1996 IEEE
Proceedings of ICPR '96

In order to keep the model as simple as possible and to have as few degrees of freedom as possible (only four positional DOF with two spherical joints), we chose not to model (or render) the shape of the hand, but simply assume it to extend along the axis of the forearm. This strong simplification provided us with a reasonable starting point for experimentation.

Our model thus requires 7 fixed parameters to describe it's shape: the longitudinal lengths of the hand, forearm and upper arm (3), and the diameters of the two limb segments at each end (4). Each of these parameters was measured with approximately 5% accuracy (we have not yet systematically studied the effect of model inaccuracies). Furthermore, we assume that the 3D position of the shoulder is known. There is a natural hierarchy to the segmentation of the human body, and shoulder position is determined by tracking the torso. Since we attempt to track only the arm, we assume the shoulder position is known.

2.2. The Recursive Estimator

The state of the system consists of the four spherical joint angles and their velocities. In order to recursively estimate it, we use an implicit version of the Extended Kalman Filter [11, 4, 10]. The dynamics of the system is described by a random walk in the spherical joint velocities:

$$\begin{cases} \vec{\theta}(t+\Delta T) &= \vec{\theta}(t) + \vec{\theta_v}\Delta T \\ \vec{\theta_v}(t+1) &= \vec{\theta_v}(t) + \vec{w} \end{cases} \quad (1)$$

where $\vec{\theta}$ is the vector of spherical coordinates, $\vec{\theta_v}$ is the vector of the angular velocities and $\vec{w}$ is a vector of gaussian noise.

The measurement equation, instead of being in the standard form

$$\vec{y} = \vec{h}(\vec{\theta}, \vec{\theta_v}) \quad (2)$$

is an implicit and non-linear relation between the state and the image values

$$\vec{h}(\vec{y}, \vec{\theta}, \vec{\theta_v}) = \vec{0} \quad (3)$$

This kind of problem can be transformed to the classical formulation of the extended KF. The key is to obtain a linearization of the measurement equation. This involves calculating the jacobian of the measurements with respect to the state (and the image intensities), which in turn involves knowing a precise camera calibration. Although a straightforward calculation, it is rather tedious, and we refer the details to [2].

2.3. The Error Measurements

The measurement process is explained in detail in [9]. We pre-process the image in order to obtain the blurred version of a binary image which assumes value 1 where the arm and the body are and value 0 elsewhere. Then the difference between this pre-processed image and the predicted image is calculated at 20 points on both sides of each (predicted) limbs' contours and the predicted hand tip position (Fig. 3). If the predicted and the real image fall exactly on each other (and in the absence of measurement noise and modeling error), all these differences are zero, otherwise, the deviations from this ideal value constitutes the innovation process used by the Kalman filter for the update of the state estimate (Fig.4).

3. The Real-Time System

Figure 5 diagrams the implementation hardware, which consists of a video camera, a video processing board, a Pentium 90 PC, and an SGI workstation.

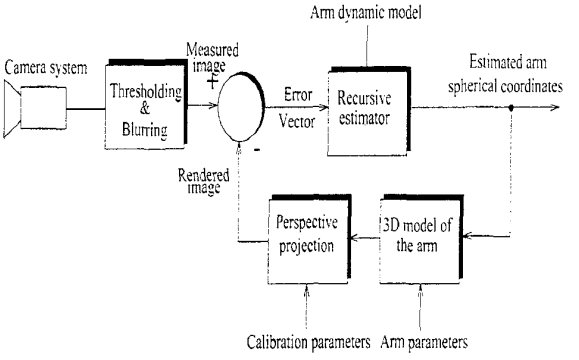

Figure 2. ***The estimation system:*** *Real and rendered arm views are compared to provide an error signal to a recursive estimator.*

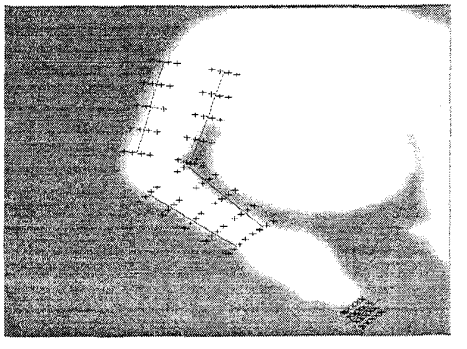

Figure 3. ***The measurements for the recursive estimator:*** *The (+) indicate locations on the thresholded and blurred image where intensity values are compared with those from the predicted arm position (outlined by lines).*

3.1. The Camera

The camera used is a commercial Canon L1 cam-corder. It has 480×640 pixels resolution interlaced. The viewing angle used was 65 degrees in the horizontal direction. The camera was calibrated in order to compensate for radial distortion and origin displacement.

3.2. The C80 Board

A TI TMS3020C80 based signal and image processing board was used to perform the image preprocessing. The C80 chip includes 5 cpu-s (one floating point main processor and four integer DSP parallel processors) with a combined maximum theoretical throughput of over 1.3 billion instructions per second at 33 MHz.

The incoming camera image is de-interlaced and digitized by the board, and then the background subtraction algorithm is applied. The background subtraction algorithm we used is similar to the one proposed by Russel, Starner and Pentland in [19]. We acquire 40 frames of the static background in order to build some statistics of it (mean and variance of luminance, I, and chrominance, Cr and Cb, for each

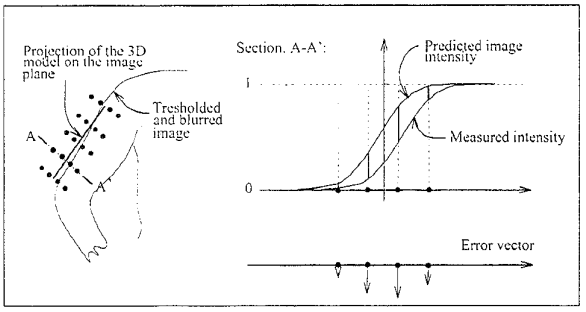

Figure 4. ***Detail of the measurement process:*** *Transverse to the predicted arm contour, the acquired and pre-processed image is sampled at 4 points. The difference between the expected and measured intensity values generates an error vector.*

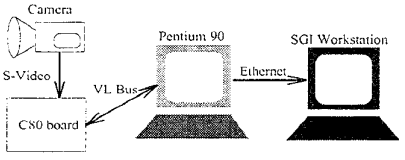

Figure 5. ***System configuration:*** *The hardware architecture comprises a commercial camera, a video board, a Pentium 90 and an SGI workstation.*

pixel location). After this initialization period, when a new image frame is acquired, we classify each pixel to be part of the foreground if it satisfies one of the following conditions:

$$\begin{array}{ll} \frac{|l(x,y)-\mu_l(x,y)|}{\sigma_l} & > T_l \\ \frac{|Cr(x,y)-\mu_{Cr}(x,y)|}{\sigma_{Cr}} & > T_c \\ \frac{|Cb(x,y)-\mu_{Cb}(x,y)|}{\sigma_{Cb}} & > T_c \end{array}$$

where $\mu_s(x,y)$ and $\sigma_s(x,y)$ are the mean and the standard deviation of the pixel (x,y) and s can be the luminance l or the two chrominances Cr and Cb while $l(x,y)$, $Cr(x,y)$ and $Cb(x,y)$ are the pixel (x,y) values for the current image and T_l and T_c are the two thresholds. Typical threshold values range from 2 to 5, depending on the imaging conditions.

As a result of the background subtraction algorithm, we obtain a binary image with value 1 for the foreground 0 for the background. The last step before taking the measurements is convolving with a gaussian kernel ($\sigma = 1.66 pixels$) in order to obtain a profile that fits the one coming from the model (as shown in Fig.4). The convolution is computed only in the 4 pixels surrounding each of the measure points. All the operations that have been described so far are executed in parallel in the 4 fast DSP, dividing the image in blocks that can fit in the internal memory of each of the processors.

3.3. The Pentium 90

The update and prediction steps of the Kalman Filter are implemented on a Pentium 90 (the host computer). Given the estimate of position and velocity of the four spherical coordinates at time t, the position of the arm at time $t+1$ is predicted and the projection of the 3D model of the arm on the image plane is computed. The host computer then communicates to the C80 board (via dual ported RAM, since the C80 board resides on the host computer local VESA bus) the coordinates of the required measurements. The C80 board reads the coordinates from the parallel RAM, computes the image values at those locations along with the two jacobians (with respect to the state and to the pixel intensities), and writes them back onto the DPRAM. The host uses these values to compute the error vector and then update the state of the filter at time $t+1$.

3.4. The SGI workstation

The current estimate of the arm's position and velocity is sent via ethernet from the Pentium 90 to an SGI graphics workstation for the 3D rendering of the arm. This provides the operator with visual feedback of his arm movements as they are reconstructed by the recursive estimator. The SGI display shows both a rendered perspective view of the reconstructed 3D arm, as well as three scale bars showing respectively the X, Y and Z coordinates of the hand tip w.r.t. the shoulder reference frame (see Fig.7).

3.5. Overall System Characteristics

The real-time system, as described above, is capable of performing one complete estimation cycle in 90 msec (11 Hz). The bottleneck component is the C80 board. Image acquisition and background subtraction operations take about 60 ms per cycle. The board spends the remaining 30 ms performing convolutions to calculate the image measurements and their jacobians. The cycle time for the recursive estimator running on the Pentium is about 20 ms and then for the remaining 70 ms the host is waiting for the C80 to provide new measurements.

4. System Evaluation

There are several performance criteria that can be considered in assessing the usability of a real-time system. In this section we introduce some such criteria and evaluate our system according to them.

4.1. Robustness

Of foremost importance is the system's robustness. An ideal system would be able to track the arm at all times, whereas our system so far still looses track of the arm every now and then if the user is not experienced. The main reason for the loss of tracking is that typical arm movements are too quick. If between one frame and the next the subject's arm moves so much that there isn't much overlap between the predicted arm position in the new frame and the actual arm position, the measurements from the new frame will not carry any information, and thus the estimation process will fail.

Since our system incorporates a model of the dynamics of arm motion, losing track of the arm does not translate into a maximum allowable arm velocity. Rather, with the current dynamic model of a random walk in joint velocities, a more correct measure would be the maximum allowable joint accelerations. However, the amount of mismatch in joint space necessary to cause enough mismatch in the image plane for the hand(?arm?) to be lost depends on the relative configuration of the arm as well as the orientation of the arm with respect to the camera. Because of this, it is difficult to quantify the maximum allowable joint accelerations (or more in line with the mechanisms of tracking loss, the 3D

acceleration of the hand tip). Based on our experimentation, movements need to be made approximately 3 times slower than normal. With some practice, a user is able to adapt his movement so that the system does not loose track of his arm.

One may improve the system's tracking ability in several ways. Most obviously, increasing the cycle rate from 11 Hz to 30 Hz should bring us within the natural movement limit. Another aid would be to perform multi-resolution measurements. Measurements on a pyramid of down-sampled images could increase the range over which a mismatch in arm positions can still give useful measurements. Finally, an improved dynamical model of arm motion may increase the prediction accuracy of the recursive filter.

Another factor limiting the robustness of our system is due to the pre-processing block currently in use. Not only the arm but also the rest of the body is detected as foreground, and thus, if some part of the arm crosses over into the image region occupied by, say, the torso, the measurements coming from that part of the arm will be incorrect. Fortunately, the jacobian of the estimated state of the system with respect to those image intensities will be zero, so that those measurements do not affect the update of the estimated state. Therefore, the system is insensitive to occlusions overall. However, if enough of the arm is occluded, tracking may be lost. To correct this problem, a more sophisticated pre-processing block must be used. Loss of tracking also occurs because of the problem of shadows. Although the preprocessing block was designed to compensate for the presence of shadows, sometimes it fails. When this occurs, some of the background is considered as foreground, and if the arm is nearby, some of the measurements may be affected.

4.2. Quality of data

Robustness issues aside, there are several performance measures that can be used to assess the quality of the output produced by the system. Their relative weight depends on the intended use of the system.

4.3. Absolute Positioning Accuracy

We compared the computed 3D position of the hand-tip with the actual 3D position, with the arm held still in a certain pose. Using a pre-recorded sequence for which a ground-truth trajectory was known, we found that our system had a maximum absolute positioning accuracy of 8% along the line-of-sight. Note that for many applications, this kind of measure is irrelevant. For instance, for the purposes of a human-machine interface, it is often sufficient that different poses be identified as unique. The virtual space can be distorted and it is not essential that there be an exact correspondence with points in real space, since the user can correct the position using visual feedback.

4.4. Repeatability

A more useful performance measure for a human-machine interface is one which measures the repeatability with which a given position can be reached. We measured the variance of the virtual hand-tip position when the user places his hand-tip at a specified location in 3D. Using this measure, we found our system to produce results which are repeatable with a standard deviation of approximately 1 cm (less than 1% of the distance to the camera). The data for this calculation was obtained by having a user repeatedly move his hand-tip between four marked locations distributed in the arm's workspace. Figure 6 shows scatter plots of the repeatability measurements for two of the locations.

4.5. Resolution

The final performance measure that we consider is one which measures the resolution of positional control. In our specific case, we chose to measure the resolution with which the virtual hand-tip position can be controlled. The data was generated by having a user repeatedly try to line up his virtual hand-tip coordinates with those of a randomly chosen virtual point. This virtual point was shown on the SGI display by hi-lighting it's XYZ coordinates on the scales (recall Fig. 7). We found the system to have a resolution standard deviation of approximately 1cm.

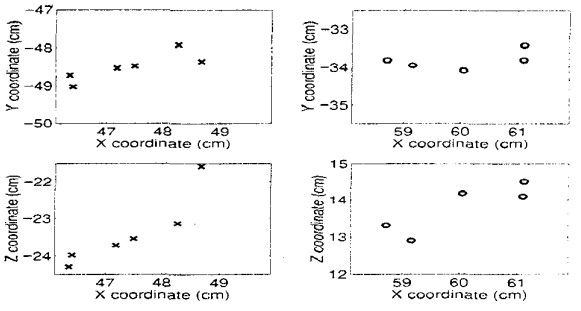

Figure 6. ***Repeatability of hand-tip position estimation:*** *Scatter plots of estimated hand-tip position when the user repeatedly moves to the same locations in the workspace. Left: a location on the tabletop, Right: a location along a tightly strung string between the tabletop and the ceiling. The coordinates are relative to the shoulder position; X is forwards (towards camera), Z is upwards, and Y to the left. The camera was 130 cm from the shoulder, looking downwards with an inclination of 30 degrees to the horizon, and viewing angles of 65 degrees horizontally, 50 degrees vertically.*

4.6. Static versus dynamic performance measures

The three measures described above are all 'static' performance measures, that is, measures applied when the arm is stationary, rather than moving. 'Dynamic' measures would take into account, for example, the joint angles, velocities and accelerations as a function of time (over complete trajectories). Since we did not have available a method for determining the ground truth of those quantities as a function of time, we were unable to compute any dynamic performance measures. Note that it is necessary to use a dynamic performance measure to asses the tracking qualities of the recursive estimator (such as convergence, stability). We can, however, remark qualitatively on the estimator's tracking: because of the random walk velocity arm dynamics model, there was an obvious "lagging behind" of the estimator whenever a sudden change in direction occured.

4.7. Practicality

In the introduction we claimed that a system like this would be more practical to use than the usual tools. We set up an experiment: we render in the virtual 3D space (represented on the SGI screen using perspective projection) a small cube in a random location inside the arm's reachable space. The task consists of reaching this location with the hand, grabbing the cube and moving it to a new random location (Fig. 7). It takes an experienced user between 10 to 20 seconds to complete the task. The difficulty in executing the task derives from the slowness with which the movement

must be made, and from the visual interface between the operator and the virtual enviroment: although rendered in a perspective view, it is still difficult to accurately perceive the position of the target. The displaying of XYZ scales aids this, but the user then moves his arm sequentially, along one axis at a time. Using a 3D trackball instead of our system, the same task is typically done in less than 5 seconds. When the frame rate of our system increases to 30Hz, we expect the task time to be comparable to that of the trackball.

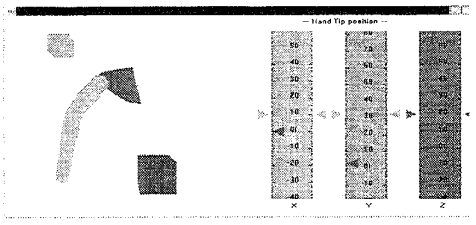

Figure 7. ***An experimental task:*** *The user has to grab hold of a virtual box and relocate it to a target position. The SGI display shows a perspective view of arm, box, and target location. To further aid the user, target and initial box locations are indicated on the XYZ scales as well.*

5. Conclusions and Future Work

We have implemented in realtime a system previously described in [9]. We achieved a speedup factor of 110 over the offline system. From our experimentation with the system as is, we conclude that the fundamental design is sound, and that it is worthwhile to continue work in this direction. Although our initial result with the offline version of the system was quite encouraging (8% maximum error in estimated depth, relative to distance from camera), experimentation with the real-time system has revealed that, with respect to performance measure's which are more relevant to a human-machine-interface, the performance is even better (1% positioning resolution).

Below we describe some of the next steps we will take to improve our system's robustness and performance level.

To avoid the problem of the body-hidden arm as well as the problem of the foreground-confused shadows, we must experiment with other pre-processing blocks. Possibilities include using the gradient magnitude of the image, segmenting the image based on color, and/or optical flow.

As mentioned previously, the higher the frame processing rate, the more robust and useful the system becomes. One way to increase the rate is to use faster hardeware (50MhZ C80 boards are now available). Also, we can experiment with more efficient algorithms. For example, our current C80 board does not support hardware-driven double buffering. If it did, cpu usage would be reduced by 33% per cylce (a frame rate increase of 150%). Finally, the use of multiresolution images may allows us to track without making as many measurements per image (currently 100).

With a better model of arm dynamics, it may be possible to increase robustness without increasing the frame rate. There is much knowledge from studies of human motion which we may be able to incorporate into our system. First of all, we could incorporate knowledge of joint limits. There is also the knowledge that some postures are more common than others, and that, say, when you are reaching in free space, there is a 'standard' position for the elbow. Another example is the fact that for ballistic arm movements there is a standard hand speed profile which is common to all such movements, modulo a time and intensity scaling.

Finally, we hope to add more links of the body to the model. First on the list is a more accurate model of the shoulder complex (rather than modelling it as a simple spherical joint). Then we can add the torso, head and neck, and the second arm.

Acknowledgments

This work is supported in part by the California Institute of Technology; a fellowship from the "Ing.A.Gini" foundation; the Office of Naval Research grant ONR NOOO14-93-1-0990; an NSF National Young Investigator Award; the Center for Neuromorphic Systems Engineering as a part of the National Science Foundation Engineering Research Center Program; and by the California Trade and Commerce Agency, Office of Strategic Technology.

References

[1] C. G. Atkeson and J. M. Hollerbach. Kinematic features of unconstrained vertical arm movements. *The Journal of Neuroscience*, 5(9):2318–2330, September 1985.

[2] E. Di Bernardo, L. Goncalves, and P. Perona. State-guided image feedback for arm motion estimation. Technical report, California Institute of Technology, in preparation.

[3] N.A. Borghese, M. Di Rienzo, G. Ferrigno, and A. Pedotti. Elite: A goal oriented vision system for moving objects detection. *Robotica*, 9:275–282, 1991.

[4] R.S. Bucy. Non-linear filtering theory. *IEEE Trans. A.C. AC-10, 198*, 1965.

[5] E. D. Dickmanns and V. Graefe. Applications of dynamic monocular machine vision. *Machine Vision and Applications*, 1:241–261, 1988.

[6] W. T. Freeman and M. Roth. Orientation histograms for hand gesture recognition. Technical Report 94-03a, Mitsubishi Electric Research Labs., 201 Broadway, Cambridge, MA 02139, 1994.

[7] W. T. Freeman and C. Weissman. Television control by hand gestures. Technical Report 94-24, Mitsubishi Electric Research Labs., 201 Broadway, Cambridge, MA 02139, 1994.

[8] D.M. Gavrila and L.S. Davis. Towards 3-D model-based tracking and recognition of human movement: a multi-view approach. In International Workshop on Automatic Face and Gesture Recognition, p272-277,Zurich, 1995.

[9] L. Goncalves, E. Di Bernardo, E. Ursella, and P. Perona. Monocular tracking of the human arm in 3d. In *International Conference Computer Vision*, pages 764–770, June 1995.

[10] A.H. Jazwinski. *Stochastic Processes and Filtering Theory*. Academic Press, 1970.

[11] R.E. Kalman. A new approach to linear filtering and prediction problems. *Trans. of the ASME-Journal of basic engineering.*, 35-45, 1960.

[12] F. Lacquaniti, J.F. Soechting, and S.A. Terzuolo. Path constraints on point-to-point arm movements in three-dimensional space. *Neuroscience*, 17(2):313–324, 1986.

[13] Jadran Lenarcic and Andreja Umek. Simple model of human reachable workspace. *IEEE Trans. Systems, Man, and Cybernetics*, 24(8):1239–1246, August 1994.

[14] P. Nesi and A. Del Bimbo. Hand Pose Tracking for 3-D Mouse Emulation. In International Workshop on Automatic Face and Gesture Recognition,p302-307, Zurich, 1995.

[15] S.A. Niyogi and E.H. Adelson. Analyzing gait with spatiotemporal surfaces. *Proceedings of the Workshop on Motion of non-rigid and articulated objects*, pages 64–69, 1994.

[16] R. Polana and R.C. Nelson. Recognizing activies. In *Proceedings of ICPR*, 1994.

[17] J.M. Rehg and T. Kanade. Digiteyes: Vision-based hand tracking for human-computer interaction. In *Proceedings of the workshop on Motion of Non-Rigid and Articulated Bodies*, pages 16–24, November 1994.

[18] K. Rohr. Incremental recognition of pedestrians from image sequences. In *Proc. IEEE Comput. Soc. Conf. Comput. Vision and Pattern Recogn.*, pages 8–13, New York City, June, 1993.

[19] K. Russell, T. Starner, and A. Pentland. Unencumbered virtual environments. Technical report, Massachusetts Institute of Technology, obtained from the WWW, 1995.