M.I.T Media Laboratory Perceptual Computing Section Technical Report No. 374
Appears in Proceedings of IMAGE'COM 96, Bordeaux, France, May 1996

Real-Time 3-D Tracking of the Human Body

Ali Azarbayejani, Christopher Wren, and Alex Pentland

MIT Media Laboratory, Cambridge, MA, USA
{*ali,cwren,sandy*}@*media.mit.edu*

Abstract

People are the central element in the whole enterprise of multimedia and communications and thus visual interpretation of humans and their movements is an important problem for computers. Here we describe a monocular and a stereo system for recovering 3-D descriptions of humans from images in real time. We discuss the technical details and present several applications using the systems for human interface.

1 Introduction

People are the central element in the whole enterprise of multimedia and communications. Applications such as video databases, wireless virtual reality interfaces, smart rooms, smart desks, very-low-bandwidth video compression, and security monitoring all have in common the need to track and interpret human action. The ability to find and follow people's head, hands, and body is therefore an important visual problem.

To address this need we have developed two real-time systems called `Pfinder` ("person finder") and `Spfinder` ("stereo person finder") which are monocular and binocular, respectively. These systems solve the problems of person tracking in arbitrarily complex scenes in which there is a single unoccluded person and fixed cameras. The systems collectively have been tested on thousands of people in several installations around the world, and have been found to perform quite reliably in a variety of contexts.

`Pfinder` has evolved over several years and has been used to recover a 3-D description of a person in a large room-size space. `Pfinder` has been used as a real-time interface device for information spaces[17], video games[15], and a distributed virtual reality populated by artificial life[5]. It has also been used as a pre-processor for gesture recognition systems, including one that can recognize a forty-word subset of American Sign Language with near perfect accuracy [18].

`Spfinder` is a recent extension to `Pfinder` in which a wide-baseline stereo pair of cameras is used to obtain 3-D models. `Spfinder` has been used in a smaller desk-area environment to capture accurate 3-D movements of head and hands. Applications have included self-calibration from watching a person [2] and visually-guided animation, in which a virtual character is driven by human movement.

`Pfinder` and `Spfinder` utilize a 2-D image analysis architecture with two complementary procedures for 2-D *tracking* and *initialization*. The initialization procedure obtains descriptions of the person from weak priors about the scene and is necessary for bootstrapping the systems at the start and when tracking breaks down. The tracking procedure recursively update the description based on strong priors from the previous frame and are necessary for efficiency. The tracking procedure can determine when it is in error and can then defer to the initialization procedure, which is slower but more reliable because it uses larger context. Initialization and tracking procedures for both systems are based largely on a Maximum *A Posteriori* probability (MAP) approach.

These 2-D procedures produce a set of "blob features" which `Pfinder` then uses along with camera-room calibration and `Spfinder` uses along with camera-camera calibration to recover 3-D models.

The organization of this paper is to describe the basic 2-D pattern recognition techniques used to bootstrap and track blob features in the `Pfinder` and `Spfinder` systems followed by a section on 3-D estimation of spatial geometry and a section describing some of the applications of these systems for real-time human interface. We begin with a brief review of other relevant research.

2 Background

`Pfinder` and `Spfinder` have descended from a variety of interesting experiments in human-computer interface and computer mediated communication. Initial exploration into this space of applications was by Krueger [9], who showed that even 2-D binary vision processing of the human form can be used as an interesting interface. More recently the Mandala group [1], has commercialized analog chromakey video processing to isolate colored gloves, etc., worn by users. In both cases, the focus is on the graphics interaction, whereas our systems focus on the visual analysis and go considerably beyond the primitive binary processing of these earlier systems.

Our systems are also related to body-tracking research such as Rehg and Kanade[13], Rohr [14], and Gavrila and Davis [7] that use kinematic models, and Pentland and Horowitz [11] and Metaxas and Terzopolous [10] who use dynamic models. However, in constrast to `Pfinder` and `Spfinder`, these other systems all require accurate initialization and local features, cannot deal with occlusion, and require massive computational resources.

Functionally, our systems are perhaps most closely related to the work of Bichsel[4] and Baumberg and Hogg [3]. These systems segment the person from the background in real time using only a standard workstation. Their limitation is that they did not analyze the person's shape or internal features, but only the silhouette of the person.

Consequently, they cannot track head and hands, recognize any but the simplest gestures, or determine body pose.

The features used by `Pfinder` and `Spfinder` are called "blobs". which is not an entirely new concept, but one to which little serious attention has been paid, in favor of stronger local features like points, lines, and contours. The blob representation that we use was developed originally by Kauth *et al* and Pentland [12, 8], for application to multispectral satillite (MSS) imagery. A broadly similar shape-color model has also been investigated by Schuster [16], and has been used to achieve fast tracking of human hands in a cluttered environment.

3 2-D processing

We first describe the color and spatial models used in `Pfinder` and `Spfinder` followed by a description of the tracking and initialization algorithms.

3.1 Modeling

The scene is modeled as a set of distinct *classes* including the room in the background and several classes covering the person in the foreground. In each image of the scene, every pixel must belong to one of the classes.

The person is modeled as a connected set of blobs, each of which serves as one class. Each blob has a spatial (x, y) and color (Y, U, V) Gaussian distribution, and a *support map* that indicates which pixels are members of the blob. We define $\mathbf{m}_k$ to be the mean (x, y, Y, U, V) of blob k, and $\mathbf{K}_k$ to be the covariance of that blob's distribution. Because of their different semantics, the spatial and color distributions are assumed to be independent. That is, $\mathbf{K}_k$ is block-diagonal, with uncoupled spatial and spectral components.

The background scene is modeled as a texture surface; each point on the texture surface is associated with a mean color value and a distribution about that mean. Color is expressed in the YUV space. The color distribution of each pixel is modeled with the Gaussian described by a full covariance matrix. Thus, for instance, a fluttering white curtain in front of a black wall will have a color covariance that is very elongated in the luminance direction, but narrow in the chrominance directions.

We define $\mathbf{m}_0$ to be the mean (Y, U, V) of a point on the texture surface, and $\mathbf{K}_0$ to be the covariance of that point's distribution. The spatial position of the point is treated implicitly because, given a particular image pixel at location (x, y), we need only consider the color mean and covariance of the corresponding texture location.

In each frame visible background pixels have their statistics recursively updated using a simple adaptive filter.

$$\mathbf{m}_t = \alpha\hat{\mathbf{y}} + (1-\alpha)\mathbf{m}_{t-1} \tag{1}$$

This allows us to compensate for changes in lighting and even for object movement. For instance, if a person moves a book it causes the texture map to change in both the locations where the book was, and where it now is. By tracking the person we can know that these areas, although changed, are still part of the texture model and thus update their statistics to the new value. The updating of the background class is done recursively, and even large changes in illumination can be substantially compensated within two or three seconds.

3.2 Tracking

Given a person model and a background model, we can now acquire a new image, interpret it, and update the scene and person models. To accomplish this there are several steps: (1) predicting the appearance of the user, (2) measuring the likelihoods of each pixel with respect to each class, (3) probabilistically classifying each pixel into a class, and (4) updating statistical models for the classes.

3.2.1 Predict Model Parameters

The first step is to update the spatial model associated with each foreground blob using the blob's dynamic model, to yield the blob's predicted spatial distribution for the current image:

$$\hat{\mathbf{X}}_{[n|n]} = \hat{\mathbf{X}}_{[n|n-1]} + \hat{\mathbf{G}}_{[n]}\left\{\hat{\mathbf{Y}}_{[n]} - \hat{\mathbf{X}}_{[n|n-1]}\right\} \tag{2}$$

where the estimated state vector $\hat{X}$ includes the blob's position and velocity, the observations $\hat{Y}$ are the mean spatial coordinates of the blob in the current image, and the filter $\hat{G}$ is constructed assuming simple Newtonian dynamics. Smaller blobs near the person's extremities (e.g., head, hands, and feet) are assumed to have less inertia than the larger blobs that describe the person's body.

3.2.2 Measure Likelihoods For Each Class

For each image pixel we must measure the likelihood that it is a member of each of the blob models and the background model.

For each pixel in the new image, we define $\hat{\mathbf{y}}$ to be the vector (x, y, Y, U, V). For each class k (i.e., for each blob and for the corresponding point on the scene texture model) we then measure the log likelihood

$$d_k = -(\hat{\mathbf{y}} - \mathbf{m}_k)^T \mathbf{K}_k^{-1} (\hat{\mathbf{y}} - \mathbf{m}_k) - \ln|\mathbf{K}_k| \tag{3}$$

Missing or implicit spatial components are assumed to contribute nothing to the membership likelihood.

Shadowing. Self-shadowing and cast shadows are a particular difficulty in measuring the membership likelihoods, however we have found the following approach sufficient to compensate for shadowing. First, we observe that if a pixel is significantly brighter (has a larger Y component) than the class statistics say it should, then we do not need to consider the possibility of shadowing. It is only in the case that the pixel is darker that there is a potential problem.

When the pixel is darker than the class statistics indicate, we therefore normalize the chrominance information by the brightness,

$$U^* = U/Y \tag{4}$$

$$V^* = V/Y \tag{5}$$

This normalization removes the effect of changes in the overall amount of illumination. For the common illuminants found in an office environment this step has been found to produce a stable chrominance measure despite shadowing.

The log likelihood computation then becomes

$$d_k = -(\hat{\mathbf{y}}^* - \mathbf{m}_k^*)^T {\mathbf{K}^*}_k^{-1} (\hat{\mathbf{y}}^* - \mathbf{m}_k^*) - \ln|\mathbf{K}_k^*| \tag{6}$$

where $\hat{\mathbf{y}}^*$ is (x, y, U^*, V^*) for the image pixel at location (x, y), $\mathbf{m}_k^*$ is the mean (x, y, U^*, V^*) of class k and ${\mathbf{K}^*}_k$ is the corresponding covariance.

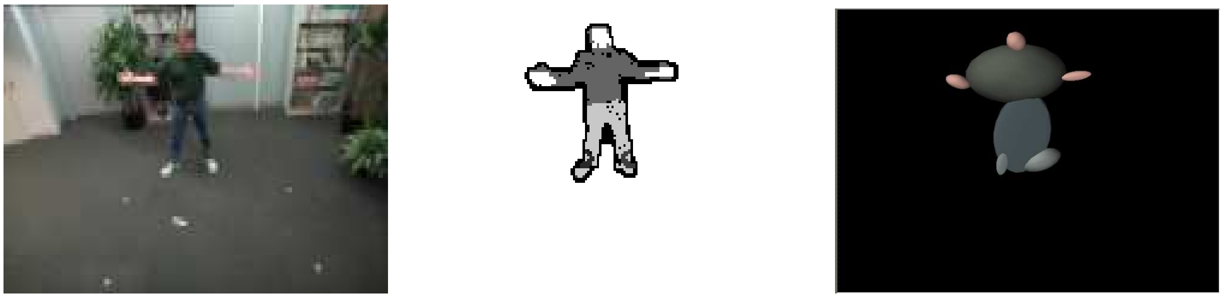

Figure 1: (left) the video input (n.b. color image shown here in black and white for printing purposes), (center) the segmentation of the user into blobs, (right) A 3-D model obtained from seven blob features with the `Pfinder` system.

3.2.3 Determine Support Map

The next step is to resolve the class membership likelihoods at each pixel into support maps, indicating for each pixel whether it is part of one of the blobs or of the scene. Spatial priors and connectivity constraints are used to accomplish this resolution.

Individual pixels are then assigned to particular classes: either to the scene texture class or a foreground blob. A classification decision is made for each pixel by comparing the computed class membership likelihoods and choosing the best one (in the MAP sense), e.g.,

$$s(x, y) = \text{argmax}_k(d_k(x, y)) \tag{7}$$

3.2.4 Update Models

Given the resolved support map $s(x, y)$, we can now update the statistical models for each blob and for the scene texture model. By comparing the new model parameters to the previous model parameters, we can also update the dynamic models of the blobs.

For each class k, the pixels marked as members of the class are averaged together to produce the model mean $\mathbf{m}_k$, and their second-order statistics measured to calculate the model's covariance matrix $\mathbf{K}_k$,

$$\mathbf{K}_k = E[(\hat{\mathbf{y}} - \mathbf{m}_k)(\hat{\mathbf{y}} - \mathbf{m}_k)^T] \tag{8}$$

This update can be written as $E[\hat{\mathbf{y}}\hat{\mathbf{y}}^T] - \mathbf{m}_k\mathbf{m}_k^T$ which allows it to be efficiently computed recursively.

3.3 Initialization

The initialization process builds the scene model by observing the scene without people in it, and then when a human enters the scene it begins to build up a model of that person.

The person model is built by first detecting a large change in the scene, and then building up a multi-blob model of the user over time. The model building process is driven by the distribution of color on the person's body, with blobs being added to account for each differently-colored region. Typically separate blobs are required for the person's hands, head, feet, shirt and pants.

3.3.1 Learning the scene

Before the system attempts to locate people in a scene, it must learn the background scene. To accomplish this the system begins by acquiring a sequence of video frames that do not contain a person. Typically this sequence is one second or more in order to obtain a good estimate of the color covariance associated with each image pixel. For computational efficiency, color models are built in both the standard (Y, U, V) and brightness-normalized (U^*, V^*) color spaces.

3.3.2 Detect Person

After the scene has been modeled, the system watches for large deviations from this model. New pixel values are compared to the known scene by measuring their Mahalanobis distance in color space from the class at the appropriate location in the scene model, as per Equation 3. If a changed region of the image is found that is of sufficient size to rule out unusual camera noise, then the system proceeds to analyze the region in more detail, and begins to build up a blob model of the person. The same initialization procedure is used to recover from tracking errors.

3.3.3 Building the Person Model

Modeling and subsequent analysis of the user utilizes the Gaussian blobs described above, incorporating both spatial and color information. The first model is a single blob covering the entire person; this model is used to obtain a better segmentation between the foreground (the person) and the background (the scene). In a manner similar to ISODATA, this original model is then successively split, the parameters recomputed, and the foreground re-segmented, until a minimum description criterion is achieved.

Contour In our full-body tracking systems we also utilize contour analysis of the foreground region to bootstrap blob features for building up a blob representation of the person. This is discussed in [19].

Occlusion When a blob can find no data to describe (as when a hand or foot is occluded), it is deleted from the person model. When the hand or foot later reappears, a new blob will be created by either the contour process (the normal case) or the color splitting process. This deletion/addition process makes the system very robust to occlusions and strong shadows. When a hand reappears after being occluded or shadowed, normally only a few frames of video will go by before the person model is again accurate and complete.

4 3-D processing

`Pfinder` and `Spfinder` use essentially the same 2-D processing techniques to produce blob features but differ in the

(a) Stereo sequence for self-calibration (b) 3-D view of sequence

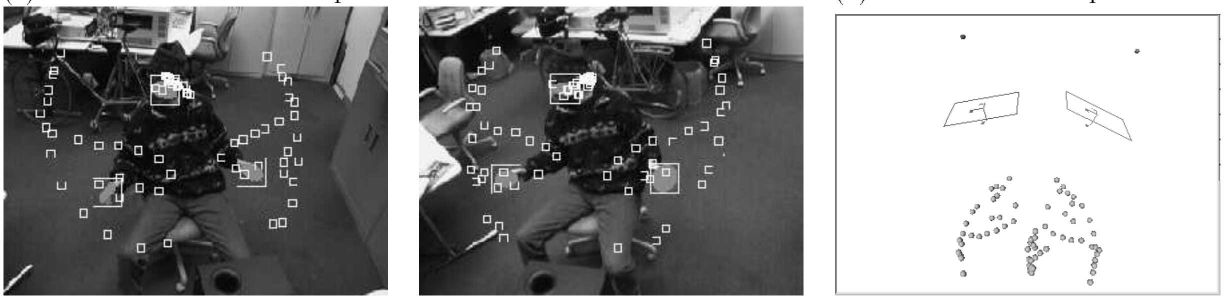

Figure 2: The blob representation can be used to facilitate stereo self-calibration: (a) the stereo pair, (b) 3-D calibration and reconstruction of hand and head trajectories

(a) Stereo sequence (b) 3-D estimate — frame 20

Figure 3: Real-time estimation of position, orientation, and shape of moving human head and hands. Experimentally, we find RMS errors of 1.5cm, 5 degrees, and 5% on translation, rotation, and shape, respectively along a linear 3-D trajectory.

way they obtain 3-D models. `Pfinder` uses only one camera and thus must use a simple 3-D model of the room and camera-room calibration to obtain a 3-D model. `Spfinder` uses two cameras, obtaining 3-D models from image correspondences and camera-camera calibration.

4.1 Monocular estimation (Pfinder)

The camera is calibrated to the floor plane and the person is assumed to be standing. Thus the user's body is in a vertical plane whose depth from the camera is determined by back-projecting the image location of the user's feet onto the floor plane. The 2-D blobs are then back-projected onto the vertical plane, resulting in a $2\frac{1}{2}$-D (2-D plus depth) model of the user. A more sophisticated kinematic model would allow a fully 3-D model to be driven with the same measurements.

4.2 Stereo estimation (Spfinder)

With two cameras, `Spfinder` can perform 3-D estimation from correspondences alone. In fact, since stereo cameras can be self-calibrated from correspondences on a moving person, no user model is necessary beyond that used in the 2-D processing. Here we briefly describe self-calibration and 3-D blob model estimation based on the calibrated stereo system.

4.2.1 Self-calibration

When a person first enters the space, the stereo-calibration is obtained recursively by using the means of blobs as point correspondences. It is well-known that calibration can be obtained in this way, but it is not usually done by tracking people [2].

The stereo pair shown in Figure 2(a) shows overlayed blobs and large white boxes marking the current feature locations, and small white boxes representing the subsequent feature tracks. The calibration parameters converge typically in the first 20 to 40 frames (roughly 2 to 4 seconds) if there is enough motion; longer if there is little motion. In this case, the subject waved his arms up and down to generate data and the system quickly converged to the state shown in Figure 2(b), which is a roughly overhead view showing the location of the cameras (COP and virtual image plane for each) and the 3-D trajectories of the hands and head.

4.2.2 3-D Modeling

We can represent shapes in both 2-D and 3-D by their low-order moments. Clusters of 2-D points have 2-D means and covariance matrices as described in Section 3 while 3-D shapes have 3-D means and covariance matrices

$$\mathbf{m} = (x_0, y_0, z_0)$$

$$\mathbf{K} = \begin{pmatrix} \sigma_x^2 & \sigma_{xy} & \sigma_{xz} \\ \sigma_{xy} & \sigma_y^2 & \sigma_{yz} \\ \sigma_{xz} & \sigma_{yz}^2 & \sigma_z^2 \end{pmatrix}$$

`Spfinder` obtains 3-D parameters from 2-D correspondences using estimation techniques described in [2]. A typical result is shown in Figure 3 for three blobs representing

Figure 4: Chris Wren playing with Bruce Blumberg's virtual dog in the ALIVE space with `Pfinder`.

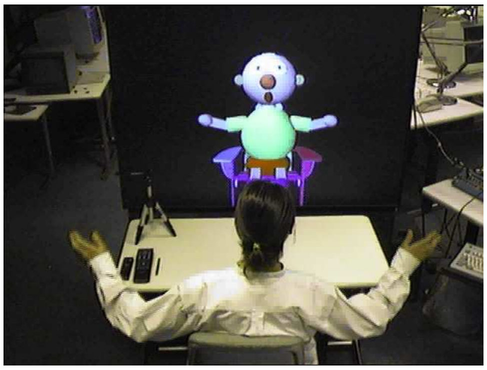

Figure 6: Ali Azarbayejani animating a 3-D character through the `Spfinder` visual interface

the head and two hands. The stereo pair indicates the three pairs of 2-D blobs used to obtain the 3-D blobs shown below. The 3-D view is roughly from overhead and the right side and shows the orientation of the wide-baseline stereo system and the shapes and locations of the 3-D blobs.

5 Applications

Although interesting by itself, the full implications of real-time human tracking only become concrete when the information is used to create an interactive application. `Pfinder` and `Spfinder` have been used to explore several different human interface applications that run at 10–30Hz on standard SGI Indy workstations with ordinary color CCD cameras (JVC-1280C).

5.1 Gesture Control for ALIVE, SURVIVE

In many applications it is desirable to have an interface that is controlled by gesture rather than by a keyboard or mouse. We have developed several of this kind of application which we call *Interactive Video Environments* (IVE), including the Artificial Life IVE (ALIVE) system [6]. ALIVE utilizes `Pfinder`'s support map polygon to define alpha values for video compositing (placing the user in a scene with some artificial life forms in real-time). `Pfinder`'s gesture tags and feature positions are used by the artificial life forms to make decisions about how to interact with the user, as illustrated in Fig. 4 [6].

`Pfinder`'s output can also be used in a much simpler and direct manner. The position of the user and the configuration of the user's appendages can be mapped into a control space, and sounds made by the user can be used to change the operating mode. This allows the user to control an application with their body directly. This interface has been used to navigate a 3-D virtual game environment called SURVIVE (Simulated Urban Recreational Violence IVE) [15] (illustrated in Fig. 5(left)), and an information landscape / virtual museum [17].

5.2 Recognition of American Sign Language

One interesting application attends only to the spatial statistics of the blobs associated with the users hands. Starner and Pentland [18] used this blob representation together with hidden Markov modeling to interpret a forty word subset of American Sign Language (ASL) with a 99% sign recognition accuracy. Thad Starner is shown using this system in Fig. 5(center).

5.3 Avatars and Telepresence

Using `Pfinder`'s estimates of the user's head, hands, and feet position it is possible to create convincing shared virtual spaces. The ALIVE system, for instance, places the user at a particular place in the virtual room populated by virtual occupants by compositing real-time 3-D computer graphics with live video. To make a convincing 3-D world, the video must be placed correctly in the 3-D environment, including graphics occluding the person and *vice versa*[6].

If you share `Pfinder`'s information about the user between geographically separate locations it is possible to create convincing telepresence without shipping video to the remote site, thus providing very low-bandwidth coding of human action, as in Darrell *et al* [5]. On the remote end information about the user's head, hand, and feet position is used to drive an video avatar that represents the user in the scene. One such avatar is illustrated in Fig. 5(right). It is important to note that the avatars need not be an accurate representation of the user, or be human at all.

5.4 Visually guided animation

With 3-D estimates of a person's body, head, and hands, 3-D animated characters can be driven anthropomorphically. This type of "motion capture" is an important technique in animation but is usually done by having the user wear a body suit with lots of wires, or by multiple camera systems that are carefully calibrated and track colored reflectors placed on various parts of the person's body.

The ability to capture human motion in an ordinary enviroment without encumbering the user with wires or reflectors opens up many new possibilities including computer games, 3-D avatars, and reduces the cost of motion capture for animation.

6 Conclusion

We have described two systems for tracking humans in 3-D. Both systems use 2-D pattern classification techniques for

Figure 5: (left) Chris Wren playing SURVIVE through the `Pfinder` visual interface, (center) Thad Starner signing American Sign Language through the `Pfinder` visual interface, Trevor Darrell controlling an avatar of himself through the `Pfinder` visual interface.

tracking blob features of people in images. `Pfinder` is monocular and uses a simple prior scene model calibrated to the camera to obtain a 3-D model. `Spfinder` is stereo and uses correspondences to obtain a 3-D model.

The systems run in real-time, from 10-30 Hz, on standard SGI Indy computers and make possible a variety of human interface applications, including gesture- and motion-based interfaces, ASL interpretation, 3-D "virtual set" compositing, visually-controlled avatars, and visual "motion capture" for 3-D animation.

References

[1] ACM SIGGRAPH'. *Mandala: Virtual Village*, SIGGRAPH-93 Visual Proceedings, Tomorrow's Realities, 1993.

[2] Ali Azarbayejani and Alex Pentland. Real-time self-calibrating stereo person tracking using 3-d shape estimation from blob features. Technical report, MIT Media Lab, Perceptual Computing Group, 1996.

[3] A. Baumberg and D. Hogg. An efficient method for contour tracking using active shape models. In *Proceeding of the Workshop on Motion of Nonrigid and Articulated Objects*. IEEE Computer Society, 1994.

[4] Martin Bichsel. Segmenting simply connected moving objects in a static scene. *Pattern Analysis and Machine Intelligence*, 16(11):1138–1142, Nov 1994.

[5] Trevor Darrell, Bruce Blumberg, Sharon Daniel, Brad Rhodes, Pattie Maes, and Alex Pentland. Alive: Dreams and illusions. In *Visual Proceedings, ACM Siggraph*, July 1995.

[6] Trevor Darrell, Pattie Maes, Bruce Blumberg, and Alex Pentland. A novel environment for situated vision and behavior. In *Proc. of CVPR–94 Workshop for Visual Behaviors*, pages 68–72, Seattle, Washington, June 1994.

[7] 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*. IEEE Computer Society, 1995. Zurich.

[8] R. J. Kauth, A. P. Pentland, and G. S. Thomas. Blob: An unsupervised clustering approach to spatial preprocessing of mss imagery. In *11th Int'l Symposium on Remote Sensing of the Environment*, Ann Arbor, MI, April 1977.

[9] M. W. Krueger. *Artificial Reality II*. Addison Wesley, 1990.

[10] D. Metaxas and D. Terzopoulos. Shape and non-rigid motion estimation through physics-based synthesis. *T-PAMI*, 15:580–591, 1993.

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

[12] Alex Pentland. Classification by clustering. In *Proceedings of the Symposium on Machine Processing of Remotely Sensed Data*. IEEE, IEEE Computer Society Press, June 1976.

[13] J.M. Rehg and T. Kanade. Visual tracking of high dof articulated structures: An application to human hand tracking. In *ECCV94*, pages B:35–46, 1994.

[14] K. Rohr. Towards model-based recognition of human movements in image sequences. *CVGIPiu*, 59(1):94–115, Jan 1994.

[15] Kenneth Russell, Thad Starner, and Alex Pentland. Unencumbered virtual environments. In *IJCAI-95 Workshop on Entertainment and AI/Alife*, 1995.

[16] Rolf Schuster. Color object tracking with adaptive modeling. In *Workshop on Visual Behaviors*, pages 91–96, Seattle, WA, June 1994. International Association for Pattern Recognition, IEEE Computer Society Press.

[17] Flavia Sparacino, Christopher Wren, Alex Pentland, and Glorianna Davenport. Hyperplex: a world of 3d interactive digital movies. In *IJCAI-95 Workshop on Entertainment and AI/Alife*, 1995.

[18] Thad Starner and Alex Pentland. Visual recognition of american sign language using hidden markov models. In *International Workshop on Automatic Face and Gesture Recognition*, Zurich, Switzerland, 1995.

[19] Chistopher Wren, Ali Azarbayejani, Trevor Darrell, and Alex Pentland. *Pfinder*: Real-time tracking of the human body. In *Photonics East, SPIE Proceedings Vol. 2615*, Bellingham, WA, 1995. SPIE.