M.I.T Media Laboratory Perceptual Computing Section Technical Report No. 353
published in IEEE Transactions on Pattern Analysis and Machine Intelligence July 1997, vol 19, no 7, pp. 780-785

Pfinder: Real-Time Tracking of the Human Body

Christopher Wren, Ali Azarbayejani, Trevor Darrell, Alex Pentland

MIT Media Laboratory Preceptual Computing Section
20 Ames Street; Cambridge MA 02139 USA
{cwren, ajazar, trevor, sandy}@media.mit.edu
http://pfinder.www.media.mit.edu/projects/pfinder/

Abstract

Pfinder is a real-time system for tracking people and interpreting thier behavior. It runs at 10Hz on a standard SGI Indy computer, and has performed reliably on thousands of people in many different physical locations. The system uses a multi-class statistical model of color and shape to obtain a 2-D representation of head and hands in a wide range of viewing conditions. Pfinder has been successfully used in a wide range of applications including wireless interfaces, video databases, and low-bandwidth coding.

1 Introduction

Applications such as video databases, wireless virtual reality interfaces, smart rooms, very-low-bandwidth video compression, and security monitoring all have in common the need to track and interpret human behavior. 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 a real-time system called Pfinder ("person finder") that substantially solves the problem for arbitrarily complex but single-person, fixed-camera situations. Use of image-to-image registration techniques [1, 10] as a preprocessing step allow Pfinder to function in the presence of camera rotation and zoom, but real-time performance cannot be achieved without special-purpose hardware. The system provides interactive performance on general-purpose hardware, has been tested on thousands of people in several installations around the world, and has performed quite reliably.

Pfinder has been used as a real-time interface device for information, and performance spaces[18], video games[18], 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 [17].

Pfinder adopts a Maximum *A Posteriori* Probability (MAP) approach to detection and tracking of the human body using simple 2-D models. It incorporates *a priori* knowledge about people primarily to bootstrap itself and to recover from errors. The central tracking and description algorithms, however, can equally well be applied to tracking vehicles or animals, and in fact we have done informal experiments in these areas. Pfinder is a descendant of the vision routines originally developed for the ALIVE system [9], which performed person tracking but had no explicit model of the person and required a controlled background. Pfinder is a more general, and more accurate, method for person segmentation, tracking, and interpretation.

2 Background

The notion of grouping atomic parts of a scene together to form blob-like entities based on proximity and visual appearance is a natural one, and has been of interest to visual scientists since the Gestalt psychologists studied grouping criteria early in this century [6].

In modern computer vision processing we seek to group image pixels together and to segment images based on visual coherence, but the features obtained from such efforts are usually taken to be the boundaries, or contours, of these regions rather than the regions themselves. In very complex scenes, such as those containing people or natural objects, contour features have proven unreliable and difficult to find and use.

The blob representation that we use was developed by Pentland and Kauth *et al* [13, 8] as a way of extracting an extremely compact, structurally meaningful description of multi-spectral satellite (MSS) imagery. In this method feature vectors at each pixel are formed by adding (x, y) spatial coordinates to the spectral (or textural) components of the imagery. These are then clustered so that image properties such as color and spatial similarity combine to form coherent connected regions, or "blobs," in which all the pixels have similar image properties. This blob description method is, in fact, a special case of recent Minimum Description Length (MDL) algorithms [4, 16].

The Pfinder system is related to body-tracking research such as Rehg and Kanade[14], Rohr [15], and Gavrila and Davis [7] that use kinematic models, or Pentland and Horowitz [12] and Metaxas and Terzopolous [11]

Figure 1: (left) video input (n.b. color image shown here in greyscale), (center) segmentation, (right) a 2-D representation of the blob statistics

who use dynamic models. However, in contrast to Pfinder these other systems all require accurate initialization and use local image features. Consequently, they have difficulty with occlusion and require massive computational resources.

Functionally, our systems are perhaps most closely related to the work of Bichsel[3] and Baumberg and Hogg [2]. These systems segment the person from the background in real time using only a standard workstation. Their limitation is that they do not analyze the person's shape or internal features, but only the silhouette of the person. Consequently, they cannot track head and hands, determine body pose, or recognize any but the simplest gestures.

3 Steady State Tracking

We will first describe Pfinder's representations and operation in the "steady-state" case, where it has already found and built representations of the person and scene. In the following sections we will then describe the model-building, and error recovery processes.

3.1 Modeling The Person

We can represent these 2-D regions by their low-order statistics. Clusters of 2-D points have 2-D spatial means and covariance matrices, which we shall denote $\boldsymbol{\mu}$ and $\mathbf{K}$. The blob spatial statistics are described in terms of their second-order properties; for computational convenience we will interpret this as a Gaussian model:

$$\Pr(\mathbf{O}) = \frac{\exp[-\frac{1}{2}(\mathbf{O}-\boldsymbol{\mu})^T \mathbf{K}^{-1}(\mathbf{O}-\boldsymbol{\mu})]}{(2\pi)^{\frac{m}{2}}|\mathbf{K}|^{\frac{1}{2}}} \quad (1)$$

The Gaussian interpretation is not terribly significant, because we also keep a pixel-by-pixel *support map* showing the actual occupancy. We define $s_k(x, y)$, the support map for blob k, to be

$$s_k(x, y) = \begin{cases} 1 & (x, y) \in k \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

The aggregate support map $s(x, y)$ over all the blob models represents the segmentation of the image into spatio-color classes.

Like other representations used in computer vision and signal analysis, including superquadrics, modal analysis, and eigenvector representations, blobs represent the global aspects of the shape and can be augmented with higher-order statistics to attain more detail if the data supports it. The reduction of degrees of freedom from individual pixels to blob parameters is a form of regularization which allows the ill-conditioned problem to be solved in a principled and stable way.

Each blob has a spatial (x, y) and color (Y, U, V) component. Color is expressed in the YUV color space. We could additionally use motion and texture measurements as part of the blob descriptions, but current hardware has restricted us to use position and color only. 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. Each blob can also have a detailed representation of its shape and appearance, modeled as differences from the underlying blob statistics. The ability to efficiently compute compact representations of people's appearance is useful for low-bandwidth applications[5].

The statistics of each blob are recursively updated to combine information contained in the most recent image with knowledge contained in the current class statistics and the priors.

3.2 Modeling The Scene

We assume that the majority of the time Pfinder will be processing a scene that consists of a relatively static situation such as an office, and a single moving person. Consequently, it is appropriate to use different types of model for the scene and for the person.

We model the scene surrounding the human as a texture surface; each point on the texture surface is associated with a mean color value and a distribution about that mean. 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 $\boldsymbol{\mu}_0$ 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. The scene texture map is considered to be class zero.

One of the key outputs of Pfinder is an indication of which scene pixels are occluded by the human, and which are visible. This information is critical in low-bandwidth coding, and in the video/graphics compositing required for "augmented reality" applications.

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

$$\boldsymbol{\mu}_t = \alpha \mathbf{y} + (1-\alpha)\boldsymbol{\mu}_{t-1} \tag{3}$$

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 process is done recursively, and even large changes in illumination can be substantially compensated within two or three seconds.

3.3 The Analysis Loop

Given a person model and a scene model, we can now acquire a new image, interpret it, and update the scene and person models. To accomplish this there are several steps: predict the appearance of the user in the new image using the current state of our model; for each image pixel and for each blob model, calculate the likelihood that the pixel is a member of the blob; resolve these pixel-by-pixel likelihoods into a support map; and update the statistical models all blob models. Each of these steps will now be described in more detail.

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

$$\hat{\mathbf{X}}_{[n|n-1]} = \boldsymbol{\Phi}_{[n-1]} \hat{\mathbf{X}}_{[n-1|n-1]} \tag{4}$$

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 the Kalman gain matrix assuming simple Newtonian dynamics.

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

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

$$\begin{aligned} d_k = & -\tfrac{1}{2}(\mathbf{y}-\boldsymbol{\mu}_k)^T \mathbf{K}_k^{-1}(\mathbf{y}-\boldsymbol{\mu}_k) - \\ & \tfrac{1}{2}\ln|\mathbf{K}_k| - \tfrac{m}{2}\ln(2\pi) \end{aligned} \tag{5}$$

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 predicted by the class statistics, 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 shadow.

When the pixel is darker than the class statistics indicate, we therefore normalize the chrominance information by the brightness, $U^* = U/Y$, and $V^* = V/Y$. 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

$$\begin{aligned} d_k = & -\tfrac{1}{2}(\mathbf{y}^*-\boldsymbol{\mu}_k^*)^T \mathbf{K}_k^{*-1}(\mathbf{y}^*-\boldsymbol{\mu}_k^*) - \\ & \tfrac{1}{2}\ln|\mathbf{K}_k^*| - \tfrac{m}{2}\ln(2\pi) \end{aligned} \tag{6}$$

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

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.,

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

Connectivity constraints are enforced by iterative morphological "growing" from a single central point, to produce a single region that is guaranteed to be connected (see Figure 2). The first step is to morphologically grow out a "foreground" region using a mixture density comprised of all of the blob classes. This defines a single connected region corresponding to all the parts of the user. Each of the individual blobs are then morphologically grown, with the constraint that they remain confined to the foreground region.

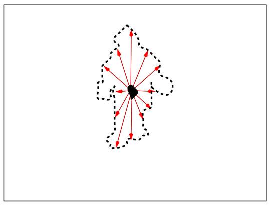

Figure 2: The morphological grow operation

This results in a set of connected blobs that fill out the foreground region. However the boundaries between blobs can still be quite ragged due to misclassification of individual pixels in the interior of the figure. We therefore use simple 2-D Markov priors to "smooth" the scene class likelihoods.

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 used to estimate the new model mean $\boldsymbol{\mu}_k$:

$$\hat{\boldsymbol{\mu}}_k = E[\boldsymbol{y}] \tag{8}$$

and the second-order statistics become the estimate of the model's covariance matrix $\mathbf{K}_k$,

$$\hat{\mathbf{K}}_k = E[(\boldsymbol{y} - \boldsymbol{\mu}_k)(\boldsymbol{y} - \boldsymbol{\mu}_k)^T] \tag{9}$$

This process can be simplified by re-writing it in another form more conducive to iterative calculation. The first term can be built up as examples are found, and the mean can be subtracted when it is finally known:

$$E[(\mathbf{y} - \boldsymbol{\mu}_k)(\mathbf{y} - \boldsymbol{\mu}_k)^T] = E[\mathbf{y}\mathbf{y}^T] - \boldsymbol{\mu}_k \boldsymbol{\mu}_k^T \tag{10}$$

For computational efficiency, color models are built in two different color spaces: the standard (Y, U, V) space, and the brightness-normalized (U^*, V^*) color space.

Errors in classification and feature tracking can lead to instability in the model. One way to ensure that the model remains valid is to reconcile the individual blob models with domain-specific prior knowledge. For instance, some parameters (e.g., color of a person's hand) are expected to be stable and to stay fairly close to the prior distribution, some are expected to be stable but have weak priors (e.g., shirt color) and others are both expected to change quickly and have weak priors (e.g., hand position).

Intelligently chosen prior knowledge can turn a class into a very solid feature tracker. For instance, classes intended to follow flesh are good candidates for assertive prior knowledge, because people's normalized skin color is surprisingly constant across different skin pigmentation levels and and radiation damage (tanning).

4 Initialization

Pfinder's initialization process consists primarily of building representations of the person and the surrounding scene. It first 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 added to account for each differently-colored region. Typically separate blobs are required for the person's hands, head, feet, shirt and pants.

The process of building a blob-model is guided by a 2-D contour shape analysis that recognizes silhouettes in which the body parts can be reliably labeled. For instance, when the user faces he camera and extends both arms (what we refer to as the "star fish" configuration) then we can reliably determine the image location of the head, hands, and feet. When the user points at something, then we can reliably determine the location of the head, one hand, and the feet.

These locations are then integrated into blob-model building process by using them as prior probabilities for blob creation and tracking. For instance, when the face and hand image positions are identified we can set up a strong prior probability for skin-colored blobs.

The following subsections describe the blob-model building process in greater detail.

4.1 Learning The Scene

Before the system attempts to locate people in a scene, it must learn the scene. To accomplish this Pfinder begins by acquiring a sequence of video frames that do not contain a person. Typically this sequence is relatively long, a 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.

4.2 Detect Person

After the scene has been modeled, Pfinder 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 5.

If a changed region of the image is found that is of sufficient size to rule out unusual camera noise, then Pfinder proceeds to analyze the region in more detail, and begins to build up a blob model of the person.

4.3 Building the Person Model

To initialize blob models, Pfinder uses a 2D contour shape analysis that attempts to identify the head, hands, and feet locations. When this contour analysis does identify one of these locations, then a new blob is created and placed at that location. For hand and face locations, the blobs have strong flesh-colored color priors. Other blobs are initialized to cover clothing regions. The blobs introduced by the contour analysis compete with all the other blobs to describe the data.

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 Pfinder very robust to occlusions and dark 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.

The blob models and the contour analyzer produce many of the same features (head, hands, feet), but with very different failure modes. The contour analysis can find the features in a single frame if they exist, but the results tend to be noisy. The class analysis produces accurate results, and can track the features where the contour can not, but it depends on the stability of the underlying models and the continuity of the underlying features (i.e., no occlusion).

The last stage of model building involves the reconciliation of these two modes. For each feature, Pfinder heuristically rates the validity of the signal from each mode. The signals are then blended with prior probabilities derived from these ratings. This allows the color trackers to track the hands in front of the body—when the hands produce no evidence in the contour. If the class models become lost due to occlusion or rapid motion, the contour tracker will dominate and will set the feature positions once they are re-acquired in the contour.

5 Limitations

Pfinder explicitly employs several domain-specific assumptions to make the vision task tractable. When these assumptions break, the system degrades in specific ways. Due to the nature of Pfinder's structure and since the model of the user is fairly weak, the system degrades gracefully and recovers in two or three frames once the assumption again holds.

Pfinder expects the scene to be significantly less dynamic than the user. Although Pfinder has the ability to compensate for small, or gradual changes in the scene or the lighting, it cannot compensate for large, sudden changes in the scene. If such changes occur, they are likely to be mistakenly considered part of the foreground region, and an attempt will be made to explain them in the user model.

test	hand	arm
translation (X,Y)	0.7 pixels (0.2% rel)	2.1 pixels (0.8% rel)
rotation (Θ)	4.8 degrees (5.2% rel)	3.0 degrees (3.1% rel)

Table 1: Pfinder Estimation Performance

Another limitation, related to the dynamic scene problem, is that system expects only one user to be in the space. Multiple users don't cause problems in the low level segmentation or blob tracking algorithms, but do cause significant difficulties with the gesture recognition system that attempts to explain the blob model as a single human figure.

6 Performance

We find RMS errors in pfinder's tracking on the order of a few pixels, as shown in Table 1. Here, the term "hand" refers to the region from approximately the wrist to the fingers. An "arm" extends from the elbow to the fingers. For the translation tests, the user moves through the environment while holding onto a straight guide. Relative error is the ratio of the RMS error to the total path length.

For the rotation error test, the user moves an appendage through several cycles of approximately 90 degree rotation. There is no guide in this test, so neither the path of the rotation, nor even its absolute extent, can be used to directly measure error. We settle for measuring the noise in the data. The RMS distance to a low-pass filtered version of the data provides this measure.

7 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 has been used to explore several different human interface applications.

7.1 A Modular Interface

Pfinder provides a modular interface that allows client applications to request subsets of the information that Pfinder provides. A wide range of data is exported through the Pfinder interface: blob model statistics; polygon representation of the support map; video texture bounding box with alpha map semantically labeled features (e.g. head, right hand, etc.); and static gestures (e.g. standing, sitting, pointing, etc.).

7.2 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. One such application is the Artificial Life IVE (ALIVE) system[9]. 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. 3(a).

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 are 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 as in SURVIVE (Simulated Urban Recreational Violence IVE) [18] (illustrated in Fig. 3(b)).

7.3 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 [17] used this blob representation together with hidden Markov modeling to interpret a forty word subset of American Sign Language (ASL). Using this approach they were able to produce a real-time ASL interpreter with a 99% sign recognition accuracy. Thad Starner is shown using this system in Fig. 3(c).

7.4 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, that is, video of the person must be able to occlude, or be occluded by, the graphics[5].

The high level description of the user is also suitable for very-low bandwidth telepresence applications. 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[5]. One such avatar is illustrated in Fig. 3(d). It is important to note that the avatars need not be an accurate representation of the user, or be human at all.

8 Hardware

Pfinder is implemented on the SGI architecture using the VL (Video Library) interface. A typical frame rate on sixteenth resolution (160x120 pixel) frames is 10Hz using a 200MHz R4400 processor Indy with Vino Video. For input we use a JVC-1280C, single CCD, color camera. It provides an S-video signal to the SGI digitizers.

9 Conclusion

Pfinder demonstrates the utility of stochastic, region-based features for real-time image understanding. This approach allows meaningful, interactive-rate interpretation of the the human form without custom hardware. The technique is stable enough to support real applications as well as higher-order vision techniques.

10 References

[1] A. Azarbayejani and A.P. Pentland. Recursive estimation of motion, structure, and focal length. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 17(6):562–575, June 1995.

[2] 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.

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

[4] T. Darrell and A.P. Pentland. Cooperative robust estimation using layers of support. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 17(5):474–487, May 1995.

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

[6] Willis Davis Ellis. *A source book of gestalt psychology*. Harcourt, Brace, New York, 1938.

[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] Pattie Maes, Bruce Blumberg, Trevor Darrell, and Alex Pentland. The alive system: Full-body interaction with animated autonomous agents. *ACM Multimedia Systems*, 5:105–112, 1997.

(a) (b) (c) (d)

Figure 3: (a) Chris Wren playing with Bruce Blumberg's virtual dog in the ALIVE space, and (b) playing SURVIVE. (c) Real-time reading of American Sign Language (with Thad Starner doing the signing) (d) Trevor Darrell demonstrating vision-driven avatars.

[10] S. Mann and R. W. Picard. Video orbits: characterizing the coordinate transformation between two images using the projective group. *IEEE T. Image Proc.*, 1997. To appear.

[11] D. Metaxas and D. Terzopoulos. Shape and non-rigid motion estimation through physics-based synthesis. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 15:580–591, 1993.

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

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

[14] J.M. Rehg and T. Kanade. Visual tracking of high dof articulated structures: An application to human hand tracking. In *European Conference on Computer Vision*, pages B:35–46, 1994.

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

[16] H. S. Sawhney and S. Ayer. Compact representations of videos through dominant and multiple motion estimation. *IEEE transactions on pattern analysis and machine intelligence*, 18(8):814–31, 1996.

[17] Thad Starner and Alex Pentland. Real-time american sign language recognition from video using hidden markov models. In *Proceedings of International Symposium on Computer Vision*, Coral Gables, FL, USA, 1995. IEEE Computer Society Press.

[18] C. Wren, F. Sparacino, A. Azarbayejani, T. Darrell, T. Starner, Kotani A, C. Chao, M. Hlavac, K. Russell, and Pentland A. Perceptive spaces for peformance and entertainment: Untethered interaction using computer vision and audition. *Applied Artificial Intelligence*, 11(4):267–284, June 1997.