Example-Based Learning for View-Based Human Face Detection

Kah-Kay Sung and Tomaso Poggio

Abstract—We present an example-based learning approach for locating vertical frontal views of human faces in complex scenes. The technique models the distribution of human face patterns by means of a few view-based "face" and "nonface" model clusters. At each image location, a difference feature vector is computed between the local image pattern and the distribution-based model. A trained classifier determines, based on the difference feature vector measurements, whether or not a human face exists at the current image location. We show empirically that the distance metric we adopt for computing difference feature vectors, and the "nonface" clusters we include in our distribution-based model, are both critical for the success of our system.

Index Terms—Face detection, object detection, example-based learning, example selection, pattern recognition, view-based recognition, density estimation, Gaussian mixture model.

1 Introduction

Feature and pattern detection is a classical computer vision problem with many potential applications, ranging from automatic target recognition to industrial inspection tasks [4], [9], [10], [6]. This paper presents an image-based feature and pattern-detection technique for finding human faces and other classes of slightly deformable objects under moderate amounts of lighting variation. To demonstrate our technique, we have developed a generic human face detection system that finds vertically oriented and unoccluded frontal views of human faces in gray-level images, over a range of scales. We stress that the underlying technique is fairly general, and can also be used for detecting image patterns in other problem domains, where target patterns may not be perfectly rigid or geometrically parameterizable.

1.1 The Face-Detection Problem

Face *detection* determines the location and size of each human face (if any) in an input image. A closely related problem is face *recognition*: Compare an input face image against models in a library of known faces and report if a match is found. In recent years, *face recognition* has attracted much attention because of its many possible applications in automatic access control systems and human-computer interfaces.

Face *detection* is interesting because it is usually the first important step of a fully automatic human face recognizer. It also has potential applications in surveillance and census systems. From an academic standpoint, face detection is interesting because faces make up a challenging class of naturally structured objects with fairly complex detailed pattern variations. A successful face detection system can provide valuable insight on how one might approach other similar feature and pattern detection problems.

Face detection is difficult because face patterns can have significantly variable image appearances. First, there are pattern variations due to differences in facial appearance, expression, and skin color. Second, certain common objects, such as glasses or a mustache, can either be present or absent from a face. Third, because faces are essentially 3D objects, lighting changes can cast or remove significant shadows from a particular face. As such, classical template-based matching techniques and geometric model-based object recognition approaches that have worked well for describing rigid and articulate objects, tend to perform poorly for detecting faces.

1.2 Existing Face-Detection Work

There have been three main approaches for modeling and detecting faces in images.

1.2.1 Correlation Templates

Correlation templates compute a difference measurement between a fixed target pattern and candidate image locations, and the output is thresholded for matches. While the class of all face patterns is probably too varied to be modeled by a single correlation template, there are some face-detection approaches that use several correlation templates to detect local facial subfeatures that are approximately rigid in appearance [2], [3].

A closely related approach to correlation templates is that of *view-based eigenspaces* [12]. The approach assumes that the set of all possible face patterns occupies a low-dimensional linear subspace, within a high-dimensional vector space of all possible image patterns. An image pattern is classified as a face if its distance from the subspace of faces is below a certain threshold. So far, this approach

• *K.-K. Sung is with the Department of Information Systems and Computer Science, National University of Singapore, Lower Kent Ridge Road, Singapore 119260. E-mail: sungkk@iscs.nus.edu.sg.*
• *T. Poggio is with the Center for Biological and Computational Learning, Massachusetts Institute of Technology, Cambridge, MA 02142, USA. E-mail: tp@ai.mit.edu.*

Manuscript received 16 Jan. 1995; revised 13 Oct. 1997. Recommended for acceptance by S. Dunn.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number 105880.

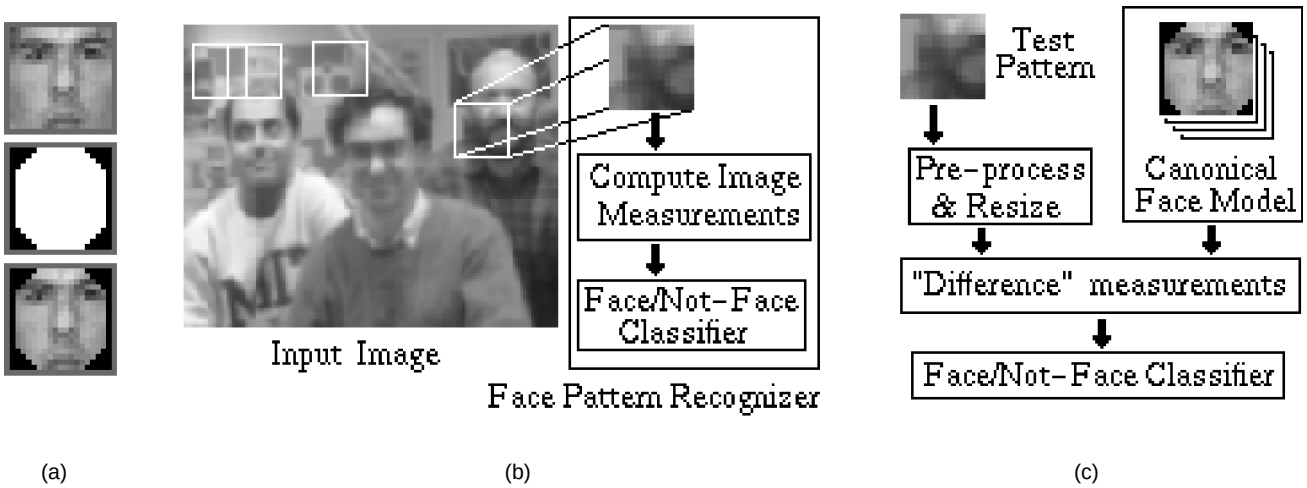


Fig. 1. Overview of our face-detection system. (a) A "canonical" face pattern, a 19 × 19 mask for eliminating near-boundary pixels of canonical face patterns, and the resulting "canonical" face pattern after applying the mask. (b) At each scale, the image is divided into many possibly overlapping windows. Each window pattern gets classified as either "a face" or "not a face," based on a set of local image measurements. (c) The key components of the *Face Pattern Recognizer* block in greater detail.

has only been demonstrated on images with little background clutter.

1.2.2 Deformable Templates

Deformable templates are similar to classical correlation templates with some built-in nonrigidity component. One deformable template approach uses parameterized curves and surfaces to model the nonrigid elements of faces and facial subfeatures, such as the eyes, nose, and lips [19]. The parameterized curves and surfaces are fixed elastically to a global template frame to allow for minor variations in position between facial features. The matching process aligns the parameterized curves and surfaces to their corresponding image features, while minimizing deformation "stress" in the template.

1.2.3 Image Invariants

Image-invariance schemes assume that there are certain spatial image relationships common and possibly unique to all face patterns, even under different imaging conditions. To detect faces, one has to compile such a set of image invariants and search the image for places where they occur. One image-invariance scheme is based on a set of observed brightness invariants between different parts of the human face [15].

1.3 Example-Based Learning and Face Detection

In this paper, we formulate the face-detection problem as one of learning to recognize face patterns from examples. We use an initial database of about 1,000 face images to construct a distribution-based generic face model with all its permissible pattern variations in a high-dimensional image vector space. We then train a decision procedure on a sequence of "face" and "nonface" examples, to empirically discover a set of operating parameters and thresholds that differentiates "face" patterns from "nonface" patterns. Our learning-based approach has the following key advantages over existing techniques:

1) Our modeling scheme does not depend much on domain specific knowledge or special handcrafting techniques to parameterize face patterns and their various sources of variation. This immediately eliminates potential errors due to inaccurate or incomplete knowledge.
2) Unlike most non-learning-based approaches that typically obtain their operating parameters and thresholds manually from a few trial cases, our detection scheme derives its parameters and thresholds automatically from many input-output examples. The resulting parameter and threshold values should therefore be statistically more reliable.
3) Our resulting system can also be made arbitrarily robust by increasing the size and variety of its training examples. Both *false positive* and *false negative* detection errors can be easily corrected by further training with the wrongly classified patterns. The system can also be similarly trained to detect human faces over a wider range of poses.

2 System Overview and Approach

In our view-based approach, faces are treated as a class of spatially local target patterns to be detected in an image. Fig. 1a shows a frontal human face image patch that just encloses the eyes and the mouth. To detect faces, we try to model the class of all such "face-like" image patches, and search the image exhaustively for these face-like patterns at all possible locations and scales. Each time a "matching" window pattern is found, the system reports a face of the current window size at the current location. We handle multiple scales by matching windows of different sizes against our face model.

Clearly, the most critical part of our system is the algorithm for classifying window patterns as "faces" or "nonfaces." Fig. 1c shows the key components of the algorithm which works as follows:

1) We use a normalized image window size of 19 × 19 pixels to model the distribution of canonical frontal face patterns.
2) To classify new window patterns at runtime, we first

resize each window pattern to 19×19 pixels. Next, we compute a set of "difference" measurements between the resized window pattern and the face distribution model. We then present the "difference" measurements to a trained classifier which determines whether or not the new window pattern contains a face.

Section 3 describes our distribution-based model for representing canonical face window patterns. Section 4 explains the distance measures we use for matching new window patterns against our distribution-based face model. Section 5 discusses the classifier training process, and in particular, our method of selecting a comprehensive but tractable set of training examples. In Section 6, we evaluate the system's overall performance, and identify the algorithm's critical components for generating correct results.

3 A Distribution-Based Face Model

Our window classification algorithm finds faces by matching each new window pattern against a canonical face model. One of the key difficulties in face detection is to account for the wide range of permissible pattern variations in face images. We address this problem by using a distribution-based modeling scheme to represent the set of all 19×19 pixel patterns that are canonical face views (see Fig. 1a). The scheme treats the class of all 19×19 pixel images as a vector space whose dimensionality equals the number of image pixels. The set of all 19×19 pixel canonical face patterns occupies some fixed volume in this multidimensional vector space. So, in theory, one can model faces by identifying this "face" volume, and representing it in some tractable fashion.

3.1 Identifying and Representing the Canonical Face Manifold

In practice, one does not have the set of all 19×19 pixel frontal face patterns to recover the volume of canonical face views exactly. Instead, we use a reasonably large example database of hand-cropped 19×19 pixel frontal face patterns, to obtain a coarse but fairly reliable representation of the actual face manifold. We also use a carefully chosen database of nonface patterns to help refine the boundaries of the face manifold, by carving out regions around the "face" sample distribution that do not correspond to canonical face views. Section 5.1 explains how we choose the special nonface patterns.

Our approach uses a few (six in our implementation) "face" clusters to piecewise approximate the multidimensional distribution of canonical face patterns, and a few (six in our implementation) "nonface" clusters to model the distribution of "nonface" patterns. Each cluster is a multidimensional Gaussian with a centroid location and a covariance matrix that describes the local data distribution (see Fig. 2).

Our piecewise-continuous modeling scheme serves two important functions. First, it performs generalization by smoothing the observed data sample distribution. This results in a stored data distribution function that is well defined even in regions of the image vector space where no data samples have been observed. Second, it serves as a tractable scheme for representing an arbitrary data distribution by means of a few Gaussian basis functions. The rest of this section describes our modeling process in greater detail.

3.2 Preprocessing

The first step of our modeling process is to normalize the individual sample patterns in the "face" and "nonface" training databases, before using them to synthesize "face" and "nonface" pattern clusters. We perform the following normalization steps on all patterns in both training databases:

1) **Image resizing:** This operation resizes all image patterns in both training databases to 19×19 pixels. We choose a 19×19 pixel window size to keep the dimensionality of the window vector space manageably small, but still large enough to preserve distinctive visual features of face patterns.
2) **Masking:** We use the 19×19 pixel binary mask in Fig. 1a to eliminate some near-boundary pixels of each window pattern. For our hand-cropped "face" patterns, these masked pixels are mostly background pixels. Removing them ensures that we do not wrongly introduce any unwanted background structure into our face representation. Masking also helps reduce the effective dimensionality of the 19×19 pixel window vector space from 361 dimensions to 283 dimensions (the number of unmasked pixels).
3) **Illumination gradient correction:** This operation subtracts a best-fit brightness plane from the unmasked window pixels. For face patterns, it helps reduce heavy shadows caused by extreme lighting angles.
4) **Histogram equalization:** This operation compensates for imaging effects due to changes in illumination brightness and differences in camera response curves.

Notice that we must also apply the same preprocessing steps to all new window patterns being classified at runtime.

3.3 Modeling the Distribution of "Face" Patterns

We use a database of 4,150 *normalized* frontal face patterns to synthesize six "face" pattern clusters in our 19×19 dimensional image vector space. The six face clusters serve as a model for our empirical face pattern distribution. Of the 4,150 database patterns, 1,067 are real face patterns, hand-cropped from several different image sources. Before normalization, we artificially enlarge the "face" database by adding to it slightly rotated versions of some real face patterns and their mirror images. This helps to ensure that the final database contains a reasonably dense and representative sample of canonical face patterns.

Each "face" pattern cluster is a multidimensional Gaussian function with a centroid location and a covariance matrix that describes the local data distribution around the centroid. We approximate the observed "face" distribution with only a small number of Gaussian clusters because the sample size is still very small compared to the image vector space dimensionality (we have 4,150 data samples in a 283-dimensional *masked* image vector space). Using a large

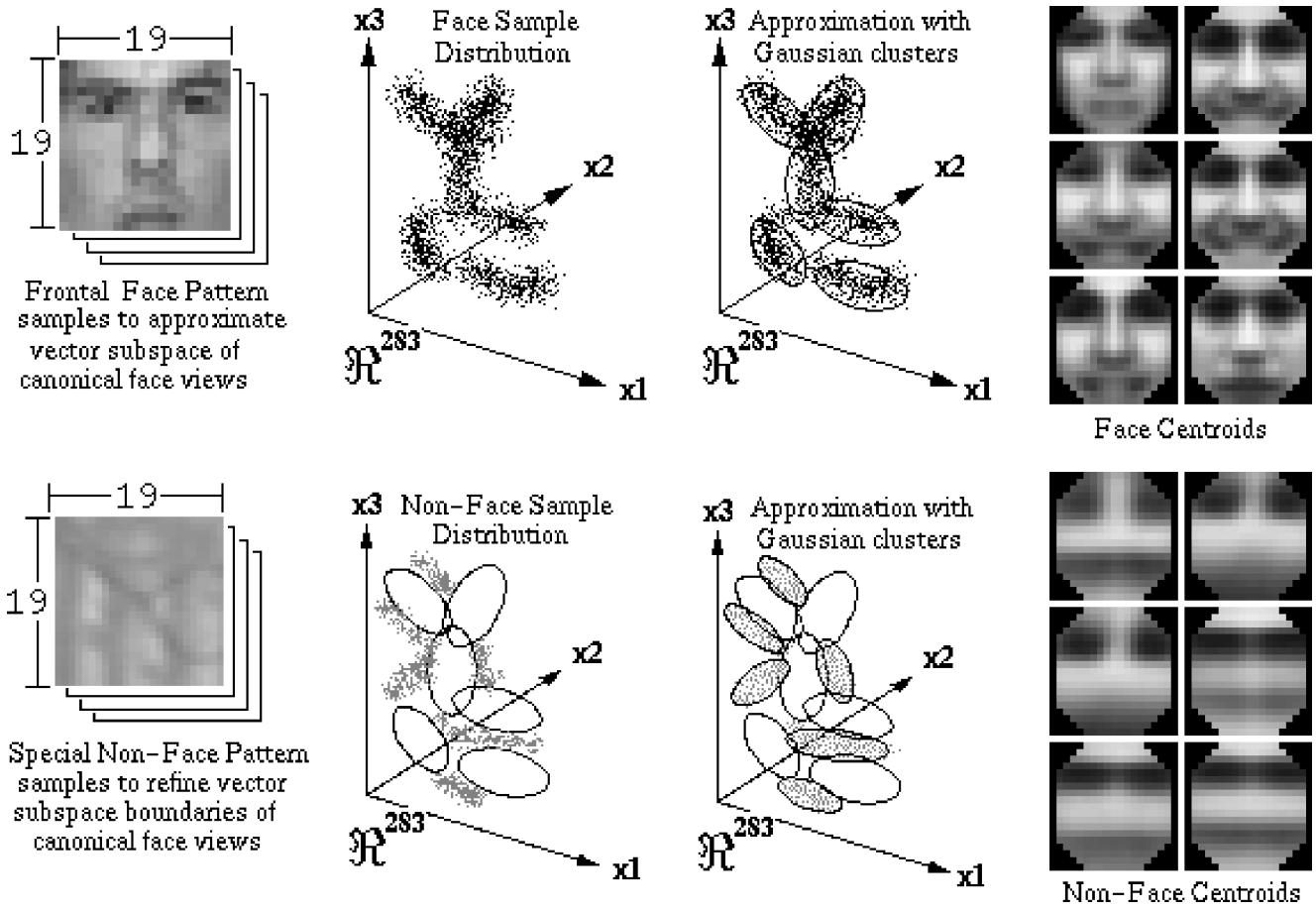


Fig. 2. Our distribution-based canonical face model. Top Row: We model an empirical distribution of face patterns using six multi-dimensional Gaussian clusters, whose centers are as shown on the right. Bottom Row: We also model a carefully chosen sample of nonface patterns using six multidimensional Gaussian clusters to help localize the boundaries of the face distribution. The final model consists of six Gaussian "face" clusters and six "nonface" clusters.

number of model clusters can lead to overfitting the observed data distribution with poor generalization results. We arrived at the chosen number of "face" clusters empirically. The system's performance, i.e., its face-detection rate versus the number of false alarms, does not change much with slightly fewer or more pattern clusters.

We use a modified version of the *k-means* clustering algorithm to compute the six face-pattern centroids and their cluster covariance matrices from the enlarged database of 4,150 normalized face patterns. Our clustering algorithm fits directionally elongated (i.e., elliptical) Gaussian distributions to the data sample. We adopt a nonisotropic Gaussian mixture model, because we believe that the actual face distribution can be locally more elongated along certain vector space directions than others.

Our *elliptical k-means* clustering algorithm uses an adaptively changing *normalized Mahalanobis* distance metric to partition the data sample into clusters. Let $\vec{x}$ be a new window pattern (in column vector form) and $\vec{\mu}$ be a Gaussian cluster centroid (also as a column vector). The normalized Mahalanobis distance between $\vec{x}$ and $\vec{\mu}$ is:

$$\mathcal{M}_n(\vec{x},\vec{\mu}) = \frac{1}{2}\left(d\ln 2\pi + \ln|\Sigma| + (\vec{x}-\vec{\mu})^T\Sigma^{-1}(\vec{x}-\vec{\mu})\right),$$

where d is the vector space dimensionality and Σ is the covariance matrix that encodes the cluster's shape and directions of elongation. The normalized Mahalanobis distance reduces the penalty of pattern differences along a cluster's major directions of data distribution, to form elliptical clusters instead of spherical clusters where there is a nonisotropic local data distribution. Section 4.2 explains the *normalized Mahalanobis* distance metric in greater detail.

The following is an outline of our *elliptical k-means* clustering procedure:

1) Obtain k (six in our case) initial pattern centers by performing vector quantization with *Euclidean* distances on the enlarged face database. Divide the data set into k partitions (clusters) by assigning each data sample to the nearest pattern center in Euclidean space.
2) Initialize the covariance matrices of all k clusters to be the identity matrix.
3) Recompute pattern centers to be the centroids of the current data partitions.
4) Using the current set of k pattern centers and their cluster covariance matrices, recompute data partitions by reassigning each data sample to the nearest pattern center in *normalized Mahalanobis* distance space. If the data partitions remain unchanged or if the maximum

Fig. 3. An example of a naturally occurring "nonface" pattern that resembles a face. Left: Viewed in isolation. Right: Viewed in the context of its environment.

number of *inner-loop* (i.e., Steps 3 and 4) iterations has been exceeded, proceed to Step 5. Otherwise, return to Step 3.

5) Recompute the covariance matrices of all k clusters from their data partitions.
6) Using the current set of k pattern centers and their cluster covariance matrices, recompute data partitions by reassigning each data sample to the nearest pattern center in *normalized Mahalanobis* distance space. If the data partitions remain unchanged or if the maximum number of *outer-loop* (i.e., Steps 3 to 6) iterations has been exceeded, proceed to Step 7. Otherwise, return to Step 3.
7) Return the current set of k pattern centers and their cluster covariance matrices.

The inner loop (i.e., Steps 3 and 4) is similar to the traditional *k-means* algorithm. Given a fixed distance metric, it finds a set of k pattern prototypes that partitions the sample data set. Our algorithm differs from the traditional *k-means* algorithm because of Steps 5 and 6 in the outer loop, where we try to iteratively refine and recover the cluster shapes, i.e., the cluster covariance matrices.

3.4 Modeling the Distribution of "Nonface" Patterns

There are many naturally occurring "nonface" patterns in the real world that look like faces when viewed in isolation (see Fig. 3). Because we are coarsely representing the face manifold with a few (six) Gaussian clusters, some of these face-like patterns may even be located nearer the "face" cluster centroids than some real "face" patterns. This may give rise to misclassifications, because, in general, we expect the opposite to be true, i.e., face patterns should lie nearer the "face" clusters than nonface patterns.

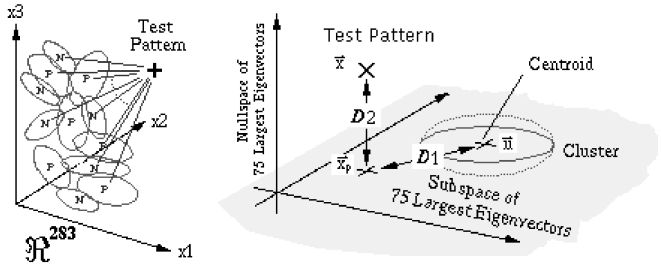
In order to reduce misclassification, we collect a comprehensive sample of these face-like patterns and explicitly model their distribution using a few (six in our implementation) "nonface" clusters. These "nonface" clusters carve out negative regions around the "face" clusters that do not correspond to face patterns. We also arrived at the chosen number "nonface" clusters empirically. The system's face-detection rate versus false-alarm ratio does not change much with slightly fewer or more "nonface" pattern clusters.

We use our *elliptical k-means* clustering algorithm to obtain six "nonface" centroids and their cluster covariance matrices from a database of 6,189 normalized face-like patterns. The database was incrementally generated in a "bootstrap" fashion by first building a reduced version of our face detection system with only "face" prototypes, and collecting all the *false positive* patterns it detects over a large set of real images. Section 5.1 elaborates further on our "boot-strap" data generation procedure.

4 Matching Patterns With the Model

To detect faces in an input image, our system matches window patterns at different image locations and scales against our distribution-based face model. Before each match, the system first applies the preprocessing operations of Section 3.2 to the current window pattern. Each match returns a set of "difference" measurements which is fed to a trained classifier that determines whether or not the current window pattern is a frontal face view.

This section describes the set of "difference" measurements we compute for each new window pattern. Each set of measurements is a vector of 12 distances between the normalized window pattern and the model's 12 cluster centroids in our multidimensional image vector space (see Fig. 4a). One can treat each distance measurement as the test pattern's actual distance from some local portion of the canonical face pattern manifold. The set of all 12 distances can thus be viewed as a crude "difference" notion between the test pattern and the entire "canonical face" pattern class.



(a) (b)

Fig. 4. Matching a test pattern against our distribution-based model. (a) Each set of measurements is a vector of 12 distances between the test pattern and the model's 12 cluster centroids. (b) Each distance measurement between the test pattern and a cluster centroid is a two-value distance metric. $\mathcal{D}_1$ is a Mahalanobis distance between the test pattern's projection and the cluster centroid in a subspace spanned by the cluster's 75 largest eigenvectors. $\mathcal{D}_2$ is the Euclidean distance between the test pattern and its projection in the subspace.

4.1 A Two-Value Distance Metric

We now define how we measure distance between a test pattern and each model cluster. The distance measure consists of two components (see Fig. 4b). The first value is a *normalized Mahalanobis* distance between the test pattern and the cluster centroid, in a lower-dimensional subspace spanned by the cluster's largest few eigenvectors. This distance component is directionally weighted to reflect the test pattern's location relative to the major elongation directions of the local data distribution. The second value is a normalized Euclidean distance between the test pattern and its projection in the lower-dimensional subspace. This is a uniformly weighted distance component that accounts for pattern differences not included in the first component due to possible modeling inaccuracies. We elaborate further on the two components below.

4.2 The Normalized Mahalanobis Distance

We begin with a brief review of the *normalized Mahalanobis* distance. Let $\vec{x}$ be a column vector test pattern, $\vec{\mu}$ be a column vector cluster centroid, and Σ be the covariance matrix describing the local data distribution near the centroid. The *normalized Mahalanobis distance* between the test pattern and the cluster centroid is given by:

$$\mathcal{M}_n(\vec{x},\vec{u}) = \frac{1}{2}\left(d\ln 2\pi + \ln|\Sigma| + (\vec{x}-\vec{\mu})^T\Sigma^{-1}(\vec{x}-\vec{\mu})\right),$$

where d is the vector space dimensionality and $|\Sigma|$ means the determinant of Σ.

One can reason about the *normalized Mahalanobis distance* as follows: If one models a local data distribution with a best-fit multidimensional Gaussian distribution centered at $\vec{\mu}$ with covariance matrix Σ, then all points at a given *normalized Mahalanobis distance* from $\vec{\mu}$ occupy a constant density surface in this multidimensional vector space. As a distance metric for classifying patterns, the *normalized Mahalanobis distance* is intuitively pleasing, because it measures pattern differences in a distribution-dependent manner that accounts for the location of all known patterns in the local target class.

4.3 The First Distance Component—Distance Within a Normalized Low-Dimensional Mahalanobis Subspace

The first distance component, $\mathcal{D}_1$, is a *normalized Mahalanobis* distance between the test pattern and the cluster centroid, measured within a lower-dimensional subspace spanned by the cluster's 75 largest eigenvectors. We arrived at our choice of 75 "significant" eigenvectors per cluster using the following criterion: For each cluster, eliminate as many trailing eigenvectors as possible, so that the sum of all the eliminated eigenvalues is still smaller than the cluster's largest eigenvalue. This procedure leaves us with approximately 75 eigenvectors for each cluster, which we standardize at 75 for simplicity.

Mathematically, we compute $\mathcal{D}_1$ by first projecting the test pattern onto the cluster's 75-dimensional vector subspace. Then we measure the normalized Mahalanobis distance between the test pattern's projection and the cluster centroid. Let $\vec{x}$ be the column vector test pattern, $\vec{\mu}$ be the cluster centroid, E_{75} be a matrix with 75 columns, where column i is a unit vector in the direction of the cluster's ith largest eigenvector, and W_{75} be a diagonal matrix of the corresponding 75 largest eigenvalues. The covariance matrix for the cluster's data distribution in the 75-dimensional subspace is given by $\Sigma_{75} = (E_{75}W_{75}E_{75}^T)$. The first distance value is:

$$\mathcal{D}_1(\vec{x},\vec{\mu}) = \frac{1}{2}\left(75\ln 2\pi + \ln|\Sigma_{75}| + (\vec{x}-\vec{\mu})^T\Sigma_{75}^{-1}(\vec{x}-\vec{\mu})\right).$$

This first value locates the test pattern relative to the cluster's major directions of data distribution. It ignores pattern differences in the cluster's smaller eigenvector directions, because the eigenvalues we recover in these directions may be significantly inaccurate due to insufficient data (see Section 4.5 for a detailed analysis). Using the smaller eigenvectors and eigenvalues to compute a distribution dependent distance can therefore lead to meaningless results.

4.4 The Second Distance Component—Distance From the Low-Dimensional Mahalanobis Subspace

The second distance component, $\mathcal{D}_2$, is a standard Euclidean distance between the test pattern and its projection in the 75-dimensional subspace. This distance component accounts for pattern differences not captured by the first component, namely, pattern differences in the cluster's smaller eigenvector directions. Because we may not have a reasonable estimate of the smaller eigenvalues, we simply assume an isotropic Gaussian data distribution in the smaller eigenvector directions, and hence a Euclidean distance measure.

Using the notation from the previous subsection, we can show that the second component is simply the L_2 norm of the displacement vector between $\vec{x}$ and its projection $\vec{x}_p$:

$$\mathcal{D}_2(\vec{x},\vec{\mu}) = \left|(\vec{x}-\vec{x}_p)\right|^2 = \left|\left(I - E_{75}E_{75}^T\right)(\vec{x}-\vec{\mu})\right|^2.$$

4.5 Relationship Between Our Two-Value Distance and the Mahalanobis Distance

There is an interesting relationship between our two-value distance metric, and the "complete" normalized Mahalanobis distance between an input pattern and a model cluster in our 19 × 19 pixel image vector space (see also [7] for a similar interpretation and presentation). Recall from Section 4.2 that the normalized Mahalanobis distance arises from fitting a multidimensional full-covariance Gaussian to an empirical data distribution. For high-dimensional vector spaces, modeling a distribution with a full-covariance Gaussian is often not feasible because one may not have enough data samples to estimate the covariance matrix accurately. Specifically, in a d-dimensional vector space, we need, in principle, at least $(d + 1)$ data samples to form a d-dimensional cluster of points to construct a d-dimensional full-covariance Gaussian distri-

bution. In practice, one would normally use a much larger number of data samples to accurately approximate the shape of the d-dimensional Gaussian cluster. In our application, we have, on the average, fewer than 700 data points (i.e., fewer than three times the minimum required number of data samples) to approximate each 283-dimensional *masked* "face" cluster. This, we feel, is still too small a data sample size for estimating the parameters of a full-covariance Gaussian model reliably.

One way of getting by with less data is to use a restricted Gaussian model with a diagonal covariance matrix, i.e., a multidimensional Gaussian distribution whose elongation axes are aligned to the vector space axes. In our domain of 19×19 pixel images, this corresponds to a model whose individual pixel values may have different variances, and whose pairwise pixel values are all uncorrelated. Clearly, this is a very poor model for face patterns which are highly structured with groups of pixels having very highly correlated intensities.

An alternative way of simplifying the Gaussian model is to preserve only a small number of "significant" eigenvectors in the covariance matrix. One can show that in order to construct a d-dimensional Gaussian model with h "significant" eigenvectors, we need only a minimum of $(h + 1)$ data samples. This can easily be a tractable number of data samples if h is small. Geometrically, the operation projects the original d-dimensional Gaussian cluster onto an h-dimensional subspace, spanned by the cluster's h most "significant" elongation directions. The projection remembers the most prominent pixel correlations among patterns in the target class. To exploit these pixelwise correlations for pattern classification, one computes a directionally weighted Mahalanobis distance between the test pattern's projection and the Gaussian centroid in this h-dimensional subspace—$\mathcal{D}_1$ of our two-value distance metric.

The orthogonal subspace is spanned by the $(d - h)$ remaining eigenvectors of the original Gaussian cluster. Because this subspace encodes pixel correlations that are less prominent and possibly less reliable due to limited training data, we simply assume an isotropic Gaussian distribution of data samples in this subspace, i.e., a diagonal covariance matrix Gaussian with equal variances along the diagonal. One can recover this diagonal covariance matrix with only one additional data sample beyond the $(h + 1)$ data samples we already require. In practice, one would use many more data samples. To measure distances in this subspace of isotropic data distribution, we use a directionally independent Euclidean distance—$\mathcal{D}_2$ of our two-value distance metric.

We can thus view our two-value distance metric as a robust approximate Mahalanobis distance that one uses when there is insufficient training data to accurately recover all the eigenvectors and eigenvalues of a full-covariance Gaussian model. The approximation uses its limited degrees of freedom to capture the most prominent pixel correlations in the data distribution. As the data sample size increases, one can preserve a larger number of principal components in the Gaussian model. This results in $\mathcal{D}_1$ becoming increasingly like the "complete" Mahalanobis distance with $\mathcal{D}_2$ vanishing in size and importance.

5 The Classifier

We use a multilayer perceptron (MLP) net classifier to identify "face" window patterns from "nonface" patterns based on their "difference" feature vectors of 12 distance measurements. The net has 12 pairs of input units, one output unit and 24 hidden units. Each hidden and output unit computes a weighted sum of its input links and performs sigmoidal thresholding on its output. During classification, the net is given a vector of the current test pattern's distance measurements to the 12 cluster centroids. Each input pair receives the two-value distance to a specific centroid. The output unit returns a 1 if the input distance vector arises from a "face" pattern, and a 0 otherwise. Our experiments in the next section show that the number of hidden units and network connectivity structure do not significantly affect the classifier's performance.

We train our MLP classifier on feature distance vectors from a database of 47,316 window patterns. There are 4,150 positive examples of "face" patterns in the database and the rest are "nonface" patterns. The net is trained with a standard back-propagation learning algorithm [14] until the output error stabilizes at a very small value.

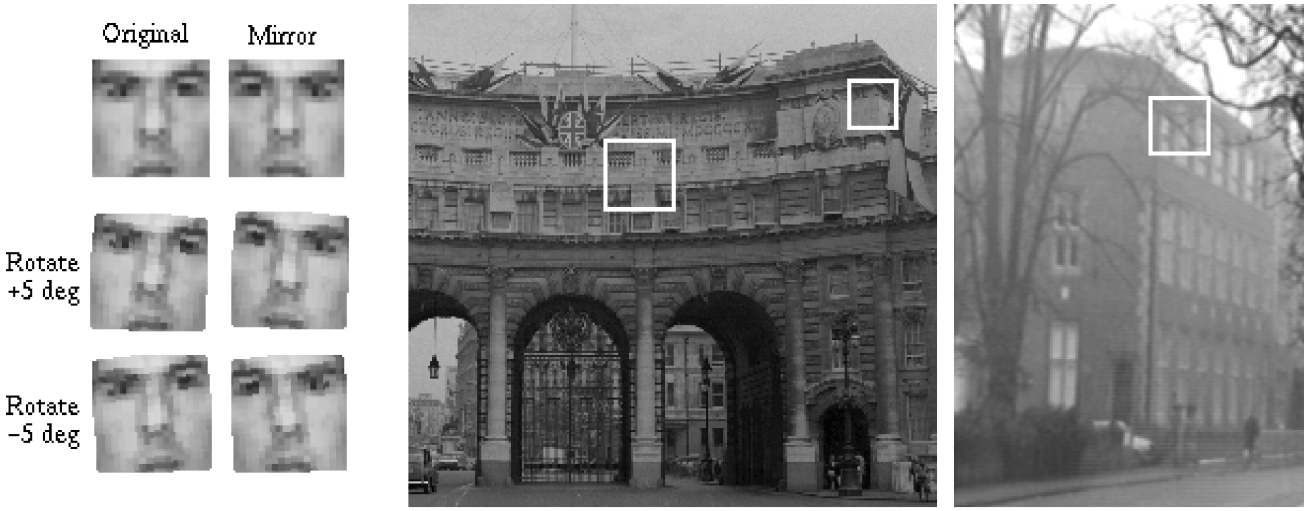
5.1 Generating and Selecting Training Examples

Because our face detection system learns its task from examples, it would be desirable to have as large a set of training examples as possible, in order to attain a comprehensive sampling of the input space. Unfortunately, there are real-world considerations that could seriously limit the size of training databases, such as shortage of free disk space and computation resource constraints.

How do we build a comprehensive but tractable database of "face" and "nonface" patterns? For "face" patterns, we simply collect all the frontal views of human faces we can find. Because we do not have many face patterns, we had to artificially enlarge our data set [13] by generating mirror images and slightly rotated versions of the original face patterns, as shown in Fig. 5a.

For "nonface" patterns, the task is more tricky. In essence, every square nonface window pattern of any size in any image is a valid training example. Clearly, our set of "nonface" patterns can grow intractably large if we are to include all possible "nonface" patterns in our training database. To constrain the number of "nonface" examples in our database, we use the following "bootstrap" strategy that incrementally selects only those "nonface" patterns with high utility value:

1) Start with a small set of "nonface" examples in the training database.
2) Train the MLP classifier with the current database of examples.
3) Run the face detector on a sequence of random images. Collect all the "nonface" patterns that the current system wrongly classifies as "faces" (see Fig. 5b). Add these "nonface" patterns to the training database as new negative examples.



(a) (b)

Fig. 5. (a) Artificially generating *virtual examples* of face patterns. For each original face pattern in our training database, we generate a few new face patterns using some simple image transformations. (b) Some false detects by an earlier version of our face detection system, marked by solid squares on the two images above. We use these false detects as new negative examples to retrain our system in a "boot-strap" fashion.

4) Return to Step 2.

At the end of each iteration, the "bootstrap" strategy enlarges the current set of "nonface" patterns with new "nonface" patterns that the current system classifies wrongly. We argue that this incremental example selection strategy makes sense, because we expect these new examples to help steer the classifier away from its current mistakes.

Notice that if necessary, we can apply the same "bootstrap" technique to enlarge the set of positive "face" patterns in our training database. Also, notice that at the end of each iteration, we may recluster our "face" and "nonface" databases to generate new model clusters that approximate the distribution of face patterns more accurately.

5.2 A Probabilistic Interpretation on Combining MLP Net Classifiers With the Distribution-Based Face Model

Section 3 describes how we model the distribution of "face" and "nonface" patterns using a few multidimensional Gaussian clusters in our 19×19 pixel image vector space. In Section 4, we introduce a two-value distance metric as a "difference" notion between an input window pattern and a model cluster. Each two-value distance measures the log-likelihood that the input pattern belongs to the local pattern distribution modeled by the cluster. The set of all 12 two-value distance measurements can thus be viewed as a model-centered coordinate system that locates the test pattern with respect to the entire "face" and "nonface" pattern distribution.

To identify "face" window patterns from "nonface" patterns, we train a MLP net classifier to combine the 12 distance measurements into a single "similarity" measure that can be thresholded for positive matches. From a probabilistic standpoint, one can treat the "similarity" measure as a conditional probability density function for face patterns: i.e., $P(\texttt{Class}(\vec{x}) = \texttt{Face} \mid \vec{x})$, where $\vec{x}$ is the input window pattern. Because we expect most "face" patterns to be located near the "face" model clusters and away from the "nonface" clusters, the resulting conditional probability density function should be high near the "face" clusters, low near the "nonface" clusters and low at places far away from the Gaussian mixture model.

6 Results and Performance Analysis

Fig. 6 shows some face-detection results by our system. The system detects frontal faces across scales, beginning with a window size of 19×19 pixels and ending with a window size of 100×100 pixels. At 19×19 pixels, the system scans the image for faces at one-pixel increments horizontally and vertically. To detect faces at a larger scale, we first shrink the image by an appropriate scaling factor, before scanning the scaled image for 19×19 pixel face patterns at one-pixel intervals. Between successive scales, the window width is enlarged (or equivalently, the image width is reduced) by a factor of 1.2.

Each time a "face" pattern is found, the system draws an appropriately sized dotted box at the corresponding window location in an output image. Notice that there can be multiple dotted boxes enclosing each face pattern in the output image, because the same face pattern can be detected at multiple scales and at slightly displaced window locations. The current system is not trained to detect faces with large off-plane rotation components.

6.1 Measuring the System's Performance

We ran our system on two test databases of new images, and counted the number of correct detections versus false alarms. The first test database consists of 301 frontal and near-frontal face mugshots of 71 different people. All the images are high quality digitized images with a fair amount of lighting variation. We use this database to obtain a "best case" detection rate for our system on high quality input patterns.

Fig. 6. Some face-detection results by our system. See text for details.

The second database contains 23 images with a total of 149 face patterns. There is a wide variation in image quality, ranging from high-quality CCD camera pictures to low-quality newspaper scans. Most of these images have complex background patterns with faces taking up only a very small percentage of the total image area. We use this database to obtain an "average case" performance measure for our system.

For the first database, our system correctly finds 96.3 percent of all the face patterns and makes only three *false detects*. All the face patterns that it misses have either strong illumination shadows or fairly large off-plane rotation components. For the second database, our system achieves a 79.9 percent detection rate with five *false positives*. The face patterns it misses are mostly either from low-quality newspaper scans or hand-drawn pictures. Also, many of the missed faces have either strong illumination shadows or fairly large rotation components. Most of the false detects are image patches with dark regions that look like eyes. Fig. 7 shows some failure modes of the system.

6.2 Analyzing the System's Components

We conducted some additional experiments to investigate how the following three aspects of our system architecture affect its face-detection and false-alarm rates:

1) the classifier architecture,
2) our two-value distance metric as a difference measure for computing distance feature vectors, and
3) the "nonface" clusters in our face model.

Fig. 7. Some face-detection results with mistakes. The top right image is very heavily quantized with only 46 gray levels. The system misses only one face. In the middle right image, the system misses some faces with significant illumination shadows. In the bottom left image, the system misses some faces with fairly large rotation components. Most of the false detects are images patches with eye-like dark regions. Note: The images here are not among those from our test databases.

6.2.1 Classifier Architecture

The first experiment looks at how varying the classifier architecture affects our system's performance. To do this, we create two new systems with different classifier architectures and compare their face-detection versus false-alarm statistics with those of our original system. Our two new classifier architectures are:

1) **A single perceptron unit.** We replace the original MLP net with a single perceptron unit connected directly to the 12 pairs of input terminals. The single perceptron unit is the simplest possible MLP net, and its purpose here is to provide an "extreme case" performance figure for network-based classifiers in this problem domain.
2) **A nearest neighbor classifier.** We perform nearest neighbor classification on the 24-value distance feature vectors computed by the matching stage. For each training pattern, we precompute and store its 24-value distance feature vector and output class at compile time. When classifying a new test pattern, we compute its 24-value distance feature vector and return the output class of the closest stored feature vector in Euclidean space. The nearest neighbor classifier provides us with a performance figure for a different classifier type in this problem domain.

6.2.2 The Distance Metric

The second experiment investigates how using a different distance metric for computing distance feature vectors in the matching stage affects the system's performance. We compare our two-value distance metric against three other distance measures:

1) The normalized Mahalanobis distance within a 75-dimensional vector subspace spanned by each model cluster's 75 largest eigenvectors—i.e., $\mathcal{D}_1$ only of our two-value distance.
2) The Euclidean distance between the test pattern and its projection in the 75-dimensional subspace—i.e., $\mathcal{D}_2$ only of our two-value distance.
3) The standard normalized Mahalanobis distance ($\mathcal{M}_n$) between the test pattern and the cluster centroid within the full image vector space.

To conduct this experiment, we repeat the previous set of classifier experiments three additional times, once for each new distance metric we are comparing. Because the three new distances are all single-value measurements, we also have to modify the classifier architectures accordingly to accept length-12 feature vectors instead of length-24 feature vectors.

6.2.3 "Nonface" Model Clusters

This experiment looks at how well the system performs with and without "nonface" clusters in the distribution-based model. We compare results from our original system against results from two new systems whose internal models contain only "face" clusters:

1) **A system with 12 "face" clusters and no "nonface" clusters.** In this system, we use the same total number of model clusters in the distribution-based face model. We obtain the 12 "face" clusters by performing elliptical *k-means* clustering with 12 centers on our enlarged face database of 4,150 patterns (see Section 3.3). The matching stage uses our two-value distance metric to compute a 24-value distance feature vector for each test pattern.
2) **A system with only six "face" clusters.** In this system, we preserve only the "face" clusters from the original system. The matching stage computes the same two-value distances between each test pattern and the six "face" centroids. The resulting distance feature vectors have only six pairs of values.

We generate two sets of performance statistics for each of the two new systems above. For the first set, we use a trained MLP net with 24 hidden units to classify new patterns from their distance feature vectors. For the second set, we replace the MLP net classifier with a trained single perceptron unit classifier.

6.3 Performance Figures and Interpretation

Tables 1 and 2 summarize the performance statistics for our three experiments.

6.3.1 Classifier Architecture

The vertical columns of Table 1 show how different classifier architectures affect the system's face-detection rate versus false-alarm instances. Qualitatively, the two network-based classifiers have very similar performance figures, especially on the first test database. Depending on the distance metric being used, the *nearest neighbor* classifier has either a somewhat higher face-detection rate with a lot more false alarms, or a much lower face-detection rate with somewhat fewer false alarms than the two network-based classifiers. The figures here suggest that the system's performance depends most critically on the type of classifier being used, and little on the detailed classifier configuration.

Why does the *nearest neighbor* classifier give seemingly poorer results than the two network-based classifiers? We believe this has to do with the much larger number of nonface patterns in our training database than face patterns (43,166 nonface patterns versus 4,150 face patterns). The good performance figures from the single perceptron classifier suggest that the two pattern classes are linearly highly separable in our 24 value distance feature space. Assuming that roughly the same fraction of face and nonface patterns lie along the linear class boundary, we get a boundary population with 10 times as many nonface samples as face samples. A new face pattern near the class boundary will therefore have a much higher chance of lying nearer a nonface sample than a face sample and hence be wrongly classified by a nearest-neighbor scheme.

TABLE 1
DETECTION RATES VERSUS NUMBER OF FALSE POSITIVES FOR DIFFERENT CLASSIFIER ARCHITECTURES AND DISTANCE METRICS

Classifier Architecture	Distance Metric							
	2-Value		Component $\mathcal{D}_1$		Component $\mathcal{D}_2$		Std. Mahalanobis	
Multi-Layer	96.3%	3	91.6%	21	91.4%	4	84.1%	9
	79.9%	5	85.1%	114	65.1%	5	42.6%	5
Single Unit	96.7%	3	93.3%	15	92.3%	3	93.0%	13
	84.6%	13	85.1%	94	68.2%	5	58.6%	11
Nearest Neighbor	65.1%	1	97.4%	208	53.9%	1	71.8%	5

Note: The four numbers for each entry are: Top left: detection rate for first database. Top right: number of false positives for first database. Bottom left: detection rate for second database. Bottom right: number of false positives for second database. We did not test the nearest neighbor *architecture on the second database because the test results from the first database already show that the* nearest neighbor *classifier is significantly inferior to the other two classifiers in this problem domain.*

6.3.2 The Distance Metric

The horizontal rows of Table 1 show how the system's performance changes with different distance metrics in the pattern matching stage. Both the network-based classifier systems produce significantly better performance statistics with our two-value distance metric than with the other three distance measures. The observation should not at all be surprising, since our two-value distance metric consists of both $\mathcal{D}_1$ and $\mathcal{D}_2$—two of the three other distance measures we are comparing against. A system that uses our two-value distance should therefore produce results that are at

least as good as a similar system that uses either $\mathcal{D}_1$ or $\mathcal{D}_2$ only.

Our two-value distance metric should also produce classification results that are at least as good as those obtained with a standard Mahalanobis distance metric, because both metrics are based on very similar Gaussian generative models of the local data distribution. With network-based classifiers, the two-value distance metric actually out-performs the standard Mahalanobis distance metric consistently. As discussed in Section 4.5, we believe this is because we have too few sample points in our local data distribution to accurately recover a full Gaussian covariance matrix for computing standard Mahalanobis distances. By naively trying to do so, we get a distance measure that poorly reflects the "difference" notion we want to capture.

TABLE 2
DETECTION RATES VERSUS NUMBER OF FALSE POSITIVES FOR DIFFERENT CLASSIFIER ARCHITECTURES AND COMPOSITION OF PROTOTYPES IN DISTRIBUTION-BASED MODEL

Classifier Architecture	Composition of Prototypes: 6 Face & 6 Non-Face		12 Face		6 Face	
Multi-layer Perceptron	96.3%	3	85.3%	21	59.7%	17
	79.9%	5	69.6%	74	60.9%	41
Single Perceptron	96.7%	3	52.1%	6	66.6%	25
	84.6%	13	49.7%	16	55.4%	56

Note. The four numbers for each entry are: Top left: detection rate for first database. Top right: number of false positives for first database. Bottom left: detection rate for second database. Bottom right: number of false positives for second database.

6.3.3 "Nonface" Model Clusters

Table 2 summarizes the performance statistics for comparing systems with and without "nonface" model clusters. As expected, the systems with "nonface" model clusters clearly outperform those without "nonface" clusters. Our results suggest that the "nonface" clusters give rise to a very discriminative set of additional distance features for identifying face patterns.

7 CONCLUSION

We have successfully developed a system for finding unoccluded vertical frontal views of human faces in images. The approach builds a distribution-based model of face patterns, and learns from examples a set of distance parameters for distinguishing between "face" and "nonface" window patterns. We stress again that the underlying technique is fairly general and has been recently used by ourselves and others for taking on feature-detection and pattern-recognition tasks in other problem domains [11], [17].

ACKNOWLEDGMENT

Support for the MIT Center of Biological and Computational Learning is provided in part by a grant from the U.S. National Science Foundation under contract ASC-9217041.

REFERENCES

[1] D. Beymer, A. Shashua, and T. Poggio, "Example Based Image Analysis and Synthesis," AIM-1431, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 1993.
[2] M. Bichsel, "Strategies of Robust Objects Recognition for Automatic Identification of Human Faces," PhD thesis, ETH, Zurich, 1991.
[3] R. Brunelli and T. Poggio, "Face Recognition: Features Versus Templates," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 15, no. 10, pp. 1,042–1,052, Oct. 1993.
[4] M.C. Burl, U. Fayyad, P. Perona, P. Smyth, and M.P. Burl, "A Trainable Tool for Finding Small Volcanoes in SAR Imagery of Venus," CNS TR-34, Calif. Inst. of Technology, 1993.
[5] R.O. Duda and P.E. Hart, *Pattern Classification and Scene Analysis*. New York: John Wiley and Sons Inc., 1973.
[6] W.E.L. Grimson and T. Lozano-Perez, "Model-Based Recognition and Localization From Sparse Range Data," A. Rosenfeld, ed., *Techniques for 3-D Machine Perception*. Amsterdam: North-Holland, 1985.
[7] G. Hinton, M. Revow, and P. Dayan, "Recognizing Handwritten Digits Using Mixture of Linear Models," *Proc. Advances in Neural Information Processings Systems*, vol. 7, 1995.
[8] M. Kirby and L. Sirovich, "Applications of the Karhunen-Loeve Procedure for the Characterization of Human Faces," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 12, no. 1, pp. 103–108, Jan. 1990.
[9] B. Kumar, D. Casasent, and H. Murakami, "Principal Component Imagery for Statistical Pattern Recognition Correlators," *Optical Eng.*, vol. 21, no. 1, Jan./Feb. 1982.
[10] A. Mahalanobis, A. Forman, N. Day, M. Bower, and R. Cherry, "Multi-Class SAR ATR Using Shift-Invariant Correlation Filters," *Pattern Recognition*, vol. 27, no. 4, pp. 619–626, 1994.
[11] M. Oren, C. Papageorgiou, P. Sinha, E. Osuna, and T. Poggio, "Pedestrian Detection Using Wavelet Templates," *Proc. CVPR*, 1997.
[12] A. Pentland, B. Moghaddam, and T. Starner, "View-Based and Modular Eigenspaces for Face Recognition," *Proc. CVPR*, pp. 84–91, 1994.
[13] T. Poggio and T. Vetter, "Recognition and Structure From One (2D) Model View: Observations on Prototypes, Object Classes, and Symmetries," AIM-1347, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 1992.
[14] D. Rumelhart and J. McClelland, *Parallel Distributed Processing*, vol. 1. Cambridge, Mass.: MIT Press, 1986.
[15] P. Sinha, "Object Recognition via Image Invariants: A Case Study," *Investigative Ophthalmology and Visual Science*, vol. 35, pp. 1,735–1,740, May 1994.
[16] K. Sung and T. Poggio, "Example-Based Learning for View-Based Human Face Detection," *Proc. Image Understanding Workshop*, vol. 2, pp. 843–850, 1994.
[17] K. Sung, "Learning and Example Selection for Object and Pattern Detection," PhD thesis, Massachusetts Institute of Technology, 1995.
[18] M. Turk and A. Pentland, "Eigenfaces for Recognition," *J. Cognitive Neuroscience*, vol. 3, no. 1, pp. 71–86, 1991.
[19] A. Yuille, P. Hallinan, and D. Cohen, "Feature Extraction From Faces Using Deformable Templates," *Int'l J. Computer Vision*, vol. 8, no. 2, pp. 99–111, 1992.

Kah-Kay Sung received his PhD degree in electrical engineering and computer science from the Massachusetts Institute of Technology in 1995. He is currently a lecturer at the Department of Information Systems and Computer Science, National University of Singapore. His research interests include computer vision and machine learning.

Tomaso Poggio received his PhD in theoretical physics from the University of Genoa in 1970 and was a member of the Max-Planck Institute fuer Biologische Kybernetik, Germany. He **w**orked on the visual system of the fly with W. Reichardt and on computational analysis of human and machine vision with D. Marr. Dr. Poggio holds the Uncas and Helen Whitaker Professorship of Vision Sciences and Biophysics at the Massachusetts Institute of Technology (MIT) Department of Brain and Cognitive Science. He is also affiliated with the MIT Artificial Intelligence Laboratory. In addition, he has been codirector of the MIT Center for Biological and Computational Learning. His current research focuses on the application of new learning techniques to time series analysis, object recognition, adaptive control, and computer graphics.