MASSACHUSETTS INSTITUTE OF TECHNOLOGY
ARTIFICIAL INTELLIGENCE LABORATORY
and
CENTER FOR BIOLOGICAL AND COMPUTATIONAL LEARNING

A.I. Memo No. 1521 December, 1994
C.B.C.L. Paper No. 112

Example-based Learning for View-based Human Face Detection

Kah-Kay Sung and Tomaso Poggio
This publication can be retrieved by anonymous ftp to publications.ai.mit.edu.

Abstract

Finding human faces automatically in an image is a difficult yet important first step to a fully automatic face recognition system. It is also an interesting academic problem because a successful face detection system can provide valuable insight on how one might approach other similar object and pattern detection problems. This paper presents 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 "non-face" prototype 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, whether or not a human face exists at the current image location. We show empirically that the prototypes we choose for our distribution-based model, and the distance metric we adopt for computing difference feature vectors, are both critical for the success of our system.

This report describes research done at the Artificial Intelligence Laboratory and within the Center for Biological and Computational Learning in the Department of Brain and Cognitive Sciences at the Massachusetts Institute of Technology. This research is sponsored by grants from ARPA-ONR under contract N00014-92-J-1879; and by a grant from the National Science Foundation under contract ASC-9217041 (this award includes funds from ARPA provided under the HPCC program). Additional support is provided by the North Atlantic Treaty Organization, ATR Audio and Visual Perception Research Laboratories, Mitsubishi Electric Corporation, Sumitomo Metal Industries, and Siemens AG. Support for the A.I. Laboratory's artificial intelligence research is provided by ARPA-ONR contract N00014-91-J-4038. Tomaso Poggio is supported by the Uncas and Helen Whitaker Chair at MIT's Whitaker College.

1 Introduction

Feature and pattern detection in images is a classical computer vision problem with many potential applications, ranging from automatic target recognition to industrial inspection tasks in assembly lines. While there has been some successful pattern detection systems in the past, especially those using pictorial templates [4] [10] [11] or geometric models [7] for describing objects, most such systems are limited by some serious constraints. For example, in some systems, the target objects must be rigid or at least made up of rigid parts. In others, lighting must be controlled or the imaging geometry must be known.

This paper presents a new feature and pattern detection technique for finding slightly deformable objects under reasonable amounts of lighting variation. To demonstrate our technique, we focus on the specific problem of finding human faces in images. We have developed a generic human face detection system that operates successfully under a reasonably wide range of lighting conditions and imaging environments. Our system detects vertically oriented and unoccluded frontal views of human faces in grey-level images. It handles faces within a range of scales as specified by the user, and works under different lighting conditions, even with moderately strong shadows. We stress that our ultimate goal is to propose a general methodology for taking on feature detection tasks in multiple domains, including hand-printed character recognition, industrial inspection, medical image analysis and terrain classification, where target patterns may not be rigid or geometrically parameterizable, and where imaging conditions may not be within the user's control.

1.1 The Face Detection Problem

We begin by describing the problem of human face detection. Given as input an arbitrary image, which could be a digitized video signal or a scanned photograph, determine whether or not there are any human faces in the image, and if there are, return an encoding of the location and spatial extent of each human face in the image. An example encoding might be to fit each face in the image with a bounding box, and return the image coordinates of the box corners. A related problem to face detection is *face recognition*: given an input image of a face, compare the input face against models in a library of known faces and report if a match is found. In recent years, the face recognition problem has attracted much attention because of its many possible applications in automatic access control systems and human-computer interfaces.

Why is automatic face *detection* an interesting problem? Applications wise, face detection has direct relevance to the face recognition problem, because the first important step of a fully automatic human face recognizer is usually one of identifying and locating faces in an unknown image. So far, the focus of face recognition research has been mainly on distinguishing individual faces from others in a database. The task of finding faces in an arbitrary background is usually avoided by either hand segmenting the input image, or by capturing faces against a known uniform background.

Face detection also has potential applications in human-computer interfaces and surveillance systems. For example, a face finder can make workstations with cameras more user friendly by turning monitors on and keeping them active whenever there is someone in front of the terminal. In some security and census systems, one could determine the number of people in a scene by counting the number of visible human faces.

From an academic standpoint, face detection is interesting because faces make up a challenging class of naturally structured but slightly deformable objects. There are many other object classes and phenomena in the real world that share similar characteristics, for example different hand or machine printed instances of the character 'A', tumor anomalies in MRI scans and material defects in industrial products. A successful face detection system can provide valuable insight on how one might approach other similar pattern and feature detection problems.

Face detection is difficult for three main reasons. First, although most faces are similarly structured with the same facial features arranged in roughly the same spatial configuration, there can be a large component of non-rigidity and textural differences among faces. For the most part, these elements of variability are due to the basic differences in "facial appearance" between individuals — person 'A' has a larger nose than person 'B', person 'C' has eyes that are farther apart than person 'D', while person 'E' has a darker skin complexion than person 'F'. Even between images of the same person's face, there can still be significant geometrical or textural differences due to changes in expression and the presence or absence of facial makeup. As such, traditional fixed template matching techniques and geometric model-based object recognition approaches that work well for rigid and articulate objects tend to perform inadequately for detecting faces.

Second, face detection is also made difficult because certain common but significant features, such as glasses or a moustache, can either be present or totally absent from a face. Furthermore, these features, when present, can cloud out other basic facial features (eg. the glare in one's glasses may de-emphasize the darkness of one's eyes) and have a variable appearance themselves (eg. glasses come in many different designs). All this adds more variability to the range of permissible face patterns that a comprehensive face detection system must handle.

Third, face detection can be further complicated by unpredictable imaging conditions in an unconstrained environment. Because faces are essentially 3-dimensional structures, a change in light source distribution can cast or remove significant shadows from a particular face, hence bringing about even more variability to 2D facial patterns.

1.2 Existing work

As discussed above, the key issue and difficulty in face detection is to account for the wide range of allowable facial pattern variations in images. There have been three main approaches for dealing with allowable pattern variations, namely: (1) the use of correlation templates, (2) deformable templates, and (3) spatial image invariants.

Correlation Templates Fixed correlation templates work like matched filters. They 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 fixed correlation templates, there are some face detection approaches that use a bank of several correlation templates to detect major facial subfeatures in the image [2] [3]. The assumption here is that the degree of non-rigidity in these facial subfeatures is small enough to be adequately described by a few fixed templates. At a later stage, the technique infers the presence of faces by analyzing the spatial relationship between the detected subfeatures.

A closely related approach to correlation templates is that of *view-based eigen-spaces* [12]. The approach assumes that the set of all possible face patterns occupies a small and easily parameterizable sub-space in the original high dimensional input image vector space. To detect faces, the approach first recovers a representation of the sub-space of face patterns as a reference for matching. An image pattern is classified as "a face" if its distance from the sub-space of faces is below a certain threshold, according to an appropriate distance metric.

Typically, the approach approximates the sub-space of face patterns using clusters and their principal components from one or more sets of face images. Mathematically, each set of face images is modeled as a cluster of points in the original high-dimensional input image vector space, and each cluster is characterized by the principal components of the distribution of its elements. The principal components, or eigenvectors, capture the different types of variation that can occur between face images in the set, with the largest eigenvectors usually corresponding to the key variations types — those due to illumination changes and predominant differences in facial appearance between individuals. The face sub-space is formed using only a subset of the largest eigenvectors from each of the clusters, so as to preserve only the semantically meaningful variations. One commonly used distance metric is the Euclidean distance of a pattern from the nearest face sub-space location, which may be interpreted as a difference measure between the observed pattern and some "typical" face pattern [12]. So far, this approach has only been demonstrated on images with not-so-cluttered backgrounds.

The view-based eigen-space approach has also been used for detecting and locating individual facial subfeatures. The purpose of these sub-feature detectors has mainly been one of registering mugshot faces with model faces for recognition, rather than as a preprocessor for face detection.

Deformable Templates Deformable templates are similar in principle to classical correlation templates, except that the former has some built-in non-rigidity component. To find faces in an image with a deformable template, one essentially fits the template to different parts of the image and thresholds the output for matches.

One deformable template approach uses hand constructed parameterized curves and surfaces to model the non-rigid elements of faces and facial subfeatures, such as the eyes, nose and lips [20]. The parameterized curves and surfaces are fixed elastically to a global template frame to allow for minor positional variations between facial features. The matching process attempts to align the template with one or more pre-processed versions of the image, such as the peak, valley and edge maps. An energy functional constrains alignment by attracting the parameterized curves and surfaces to corresponding image features, while penalizing deformation "stress" in the template. The best fit configuration is found by minimizing the the energy functional, and the minimum value also serves as a closeness measure for the match.

Image Invariants Image-invariance schemes assume that even though faces may vary greatly in appearance for a variety of reasons, there are some spatial image relationships common and possibly unique to all face patterns. To detect faces, one has to compile such a set of image invariants and check for positive occurrences of these invariants at all candidate image locations.

One image-invariance scheme is based on a set of observed brightness invariants between different parts of a human face [15]. The underlying observation is that that while illumination and other changes can significantly alter brightness levels at different parts of a face, the local ordinal structure of brightness distribution remains largely unchanged. For example, the eye regions of a face are almost always darker than the cheeks and the forehead, except possibly under some very unlikely lighting conditions. Similarly, the bridge of the nose is always brighter than the two flanking eye regions. The scheme encodes these observed brightness regularities as a *ratio template*, which it uses to pattern match for faces. A ratio template is a coarse spatial template of a face with a few appropriately chosen sub-regions, roughly corresponding to key facial features. The brightness constraints between facial parts are captured by an appropriate set of pairwise brighter-darker relationships between corresponding sub-regions. An image pattern matches the template if it satisfies all the pairwise brighter-darker relationships.

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 1000 face mugshots to construct a distribution-based generic face model with all its permissible pattern variations in a high dimensional image window vector space. We then train a decision procedure on a sequence of "face" and "non-face" examples, to empirically discover a set of operating parameters and thresholds that separates "face" patterns from "non-face" patterns. Our learning-based approach has the following distinct advantages over existing techniques:

First, our modeling scheme does not depend much on domain specific knowledge or special hand-crafting techniques to parameterize face patterns and their various sources of variation. This immediately eliminates one potential source of modeling error — that due to inaccurate or incomplete knowledge. Unlike *deformable template* approaches or *image invariance* techniques that build models based on prior knowledge of facial structure, our scheme essentially builds models that describe the distribution of face patterns it receives. Therefore, as long as we provide our scheme with a sufficiently comprehensive set of example face patterns, we can expect our face models to be more accurate and more descriptive than the manually synthesized ones.

Second, 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 a large number of input-output examples. This makes our scheme potentially superior in two ways: (1) For a given set of free thresholds and parameters, we can expect our scheme to arrive at a statistically more reliable set of operating values because it is able to process a wider sample of training data automatically. (2) Because our scheme automatically learns thresholds and parameters, it can be easily made to learn appropriate high-dimensional and non-linear relationships on image measurements for distinguishing between "face" and "non-face" patterns. These relationships, even if they do exist, may be too complex for human observers to discover manually.

Our resulting system also has the following desirable characteristics. Performance wise, it can 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. Functionality wise, the system is also highly extensible. For instance, to detect human faces over a wider range of poses, we can apply the same example-based learning technique to create a few separate systems that each model and identify faces at a different pose, and have the systems run in parallel.

2 System Overview and Approach

In our view-based approach, faces are treated as a class of local target patterns to be detected in an image. Because faces are essentially structured objects with the same key features geometrically arranged in roughly the same fashion, it is possible to define a semantically stable "canonical" face structure in the image domain for the purpose of pattern matching. Figure 1(a) shows the canonical face structure adopted by our system. It corresponds to a square portion of the human face whose upper boundary lies just above the eyes and whose lower edge falls just below the mouth. The face detection task thus becomes one of appropriately representing the class of all such "face-like" window patterns, and finding instances of these patterns in an image.

Our system detects faces by exhaustively scanning an image for these face-like window patterns at all possible scales. Figure 2(a) describes the system's task at one fixed scale. The image is divided into multiple, possibly overlapping sub-images of the current window size. At each window, the system attempts to classify the enclosed image pattern as being either "a face" or "not a face". This involves taking a set of local image measurements to extract pertinent information from the enclosed pattern, and using a pre-defined decision procedure to determine, based on the measurements obtained, whether or not the enclosed pattern "resembles" our canonical face structure. Each time a "matching" window pattern is found, the system reports a face at the window location, and the scale as given by the current window size. Multiple scales are handled by examining and classifying windows of different sizes. Our system performs an equivalent operation by examining and classifying fixed sized window patterns on scaled versions of the image. The idea is to resize the image appropriately, so that the desired window size scales to the fixed window dimensions assumed by the image measurement routines.

Clearly, the most critical part of our system is the algorithm for classifying window patterns as "faces" or "non-faces". The rest of this paper focuses on the algorithm we developed. Figure 2(b) shows the key components of the algorithm. Basically, the approach is one of appropriately modeling the distribution of canonical face patterns in some high dimensional image window vector space, and learning a functional mapping of input pattern measurements to output classes from a representative set of "face" and "non-face" window patterns. More specifically, our approach works as follows:

1. We require that the window pattern to be classified be 19×19 pixels in size and appropriately masked (see Figure 1). All window patterns of different dimensions must first be re-scaled to this size and masked before further processing. Matching with a fixed sized window simplifies our algorithm because it allows us to use the same classification procedure for all scales. The masking operation applies some prior domain knowledge to the matching problem by ignoring certain near-boundary pixels that may fall outside the spatial boundaries of a face.
2. Using a database of canonical "face" window patterns and a similar database of "non-face" window patterns, we construct a distribution-based model of canonical face patterns in the masked 19×19 dimensional image window vector space. Our modeling scheme uses a few "face" pattern prototypes to piece-wise approximate the distribution manifold of canonical face patterns in the masked 19×19 dimensional image window vector space. It also uses a few "non-face" pattern prototypes to help capture the concavities in the distribution manifold more accurately. Together, these "face" and "non-face" pattern prototypes serve as a distribution-based model for the class of canonical face views. The "face" pattern prototypes are synthesized offline by performing clustering on the example database of "face" window patterns, and the "non-face" prototypes are similarly synthesized from the database of "non-face" window patterns. The prototypes are

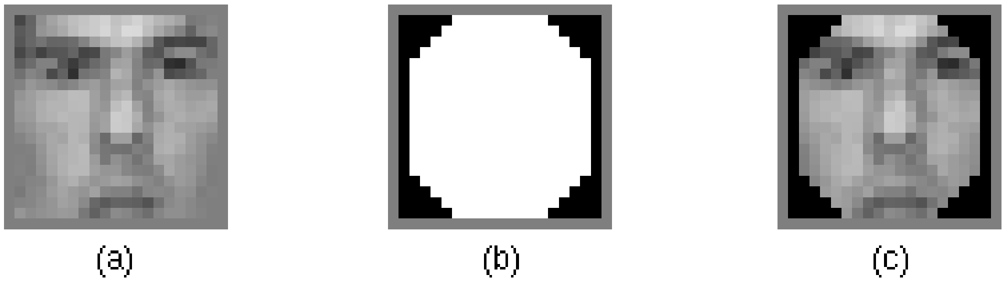

Figure 1: **(a):** A "canonical" face pattern. **(b):** A 19×19 mask for eliminating near-boundary pixels of canonical face patterns. **(c):** The resulting "canonical" face pattern after applying the mask.

hard-wired into the face detection system at compile time. Each prototype is a multi-dimensional Gaussian cluster with a centroid location and a covariance matrix that describes the local data distribution around the centroid.

3. For each new window pattern to be classified, we compute a set of image measurements as input to the "face" or "non-face" decision procedure. Each set of image measurements is a vector of distances from the new window pattern to the window pattern prototypes in the image window vector space. To compute our vector of distances, we define and use a new "Mahalanobis-like" distance metric for measuring the distance between the input window pattern and each prototype center. The distance metric takes into account the shape of each prototype cluster, in order to penalize distances orthogonal to the local data distribution. The resulting vector of distances coarsely encodes the new window pattern's location relative to the overall distribution of canonical face patterns in the image window vector space. It serves as a notion of "difference" between the new window pattern and the class of all canonical face patterns.

4. We train a *multi-layer perceptron* (MLP) net to identify new window patterns as "faces" or "non-faces" from their vector of distance measurements. When trained, the multi-layer perceptron net takes as input a vector of distance measurements and outputs a '1' if the vector arises from a face pattern, and a '0' otherwise. We use a combined database of 47316 "face" and "non-face" window patterns to train the multi-layer perceptron classifier.

The following sections describe our window pattern classification algorithm in greater detail. Section 3 describes our distribution-based model for canonical face window patterns. It explains the pre-processing and clustering operations we perform for synthesizing "face" and "non-face" window pattern prototypes. Section 4 describes our distance metric for computing classifier input vectors. 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 identify the algorithm's critical components in terms of generating correct results, and evaluate the system's overall performance.

3 A Distribution-based Face Model

Our window classification algorithm identifies faces by comparing each new masked window pattern with a canonical face model, and declaring a "face" when the match is good. This section describes our canonical face model. We use a distribution-based modeling scheme that tries to represent canonical faces as the set of all masked 19×19 pixel patterns that are canonical face views. More specifically, our distribution-based modeling scheme works as follows: Suppose we treat each unmasked pixel of a 19×19 image as a vector dimension, then the class of all 19×19 pixel images forms a vector space whose dimensionality equals the number of unmasked image pixels. Each 19×19 image pattern maps to a specific vector space location, and the set of all 19×19 pixel canonical face patterns maps to a fixed region in this multi-dimensional vector space. So, in theory, one can model the class of all canonical face views by identifying the portion of this multi-dimensional image vector space that corresponds to canonical face patterns, and representing the region 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 canonical face patterns to recover the sub-space of canonical face views exactly. Figure 3 explains how we localize and represent the region of canonical face patterns with limited data. Basically, we use a reasonably large example database of canonical face patterns and a carefully chosen database of non-face patterns to infer the spatial extent of canonical face views in the multi-dimensional image vector space. We assume that our "face" database contains a sufficiently representative sample of canonical face patterns that evenly populates the actual vector sub-space of canonical face views. So by building a model of the "face" sample distribution, one can still obtain a coarse but fairly reliable representation of the actual canonical face manifold. Our "non-face" data samples are specially selected patterns that lie near the boundaries of the canonical face manifold. We use the "non-face" patterns to help localize and refine the boundaries of the canonical face manifold by explicitly

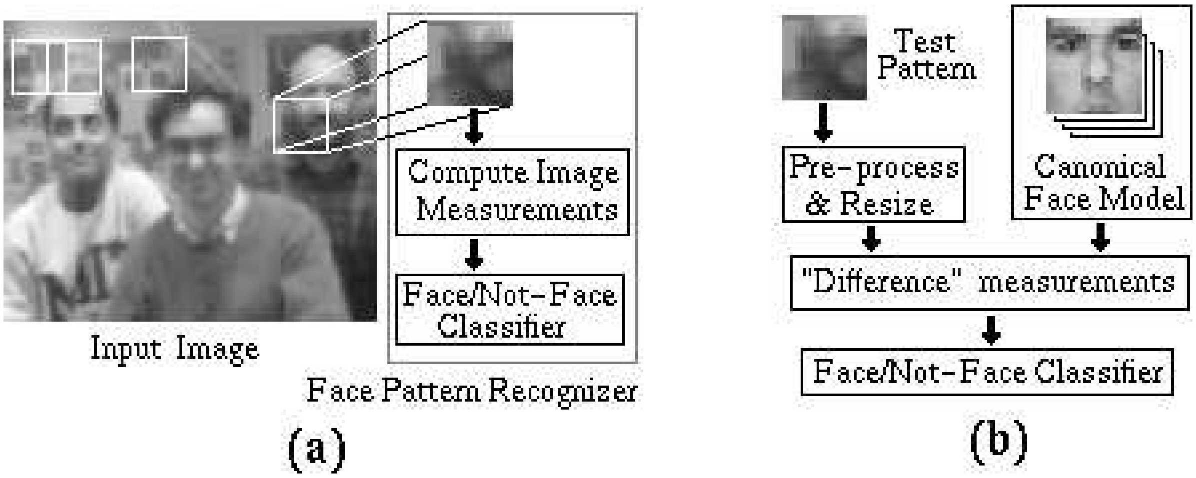

Figure 2: Overview of our face detection system. **(a):** The system's task at each scale. The image is divided into many possibly overlapping windows. Each window pattern gets classified as either "a face" or "not a face", based on a set of local image measurements. **(b):** The key components of the *Face Pattern Recognizer* block in greater detail. The Face Pattern Recognizer block classifies new patterns as "faces" or "non-faces". The algorithm uses a distribution-based model to represent the space of all possible canonical face patterns. For each new pattern to be classified, it computes a set of "difference" measurements between the new pattern and the canonical face model. A trained classifier identifies the new pattern as being either "a face" or "not a face", based on the set of "difference" measurements.

carving out regions around the "face" sample distribution that do not correspond to canonical face views. We shall explain how we synthesize our special database of "non-face" patterns in Section 5.2.

Our approach uses six prototype "face" clusters to piece-wise approximate the multi-dimensional distribution of canonical face patterns, and six "non-face" clusters to model the distribution of "non-face" patterns. Qualitatively, one can view the six "face" clusters as a coarse distribution-based representation of the canonical face manifold. The six "non-face" clusters help define boundaries in and around the manifold by carving out nearby regions in the vector space that do not correspond to face patterns. Each prototype cluster is a multi-dimensional Gaussian with a centroid location and a covariance matrix that describes the local data distribution. We adopt a piece-wise continuous modeling scheme because we believe that the actual face pattern manifold is continuous and smoothly varying in our multi-dimensional image vector space – i.e., more often than not, a face pattern with minor spatial and/or grey-level perturbations still looks like another valid face pattern. Similarly, a non-face pattern with minor variations would most likely still appear as a non-face pattern. The piecewise smooth modeling scheme serves two important functions. First, it performs generalization by applying a prior smoothness assumption to 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 bottom right image of Figure 3 shows the 12 prototype centroids in our canonical face model. The six "face" prototypes are synthesized by clustering the database of canonical face patterns, while the six "non-face" prototypes are similarly derived from the database of non-face patterns. The rest of this section describes our process for synthesizing prototypes in greater detail.

3.2 Preprocessing

The first step of synthesizing prototypes is to normalize the sample window patterns in the databases. Normalization compensates for certain sources of image variation. It reduces the range of possible window patterns the subsequent stages have to consider, thus making the modeling and classification tasks easier. Our preprocessing stage consists of the following sequence of image window operations:

1. **Window resizing:** Recall that our scheme performs modeling and classification on 19×19 grey-level window patterns. We choose a 19×19 window size to keep the dimensionality of the window vector space manageably small, but also large enough to preserve details that distinguish between "face" and "non-face" window patterns. This operation re-scales square window patterns of different sizes to 19×19 pixels.

2. **Masking:** We use the 19×19 binary pixel mask in Figure 1(b) to zero-out some near-boundary pixels of each window pattern. For "face" patterns, these masked pixels usually correspond to background pixels irrelevant to the description of a face. Zeroing out these pixels ensures that our modeling scheme does not wrongly encode any unwanted background structure in our canonical face representation. It also reduces the dimensionality of our image window vector space, which helps to make

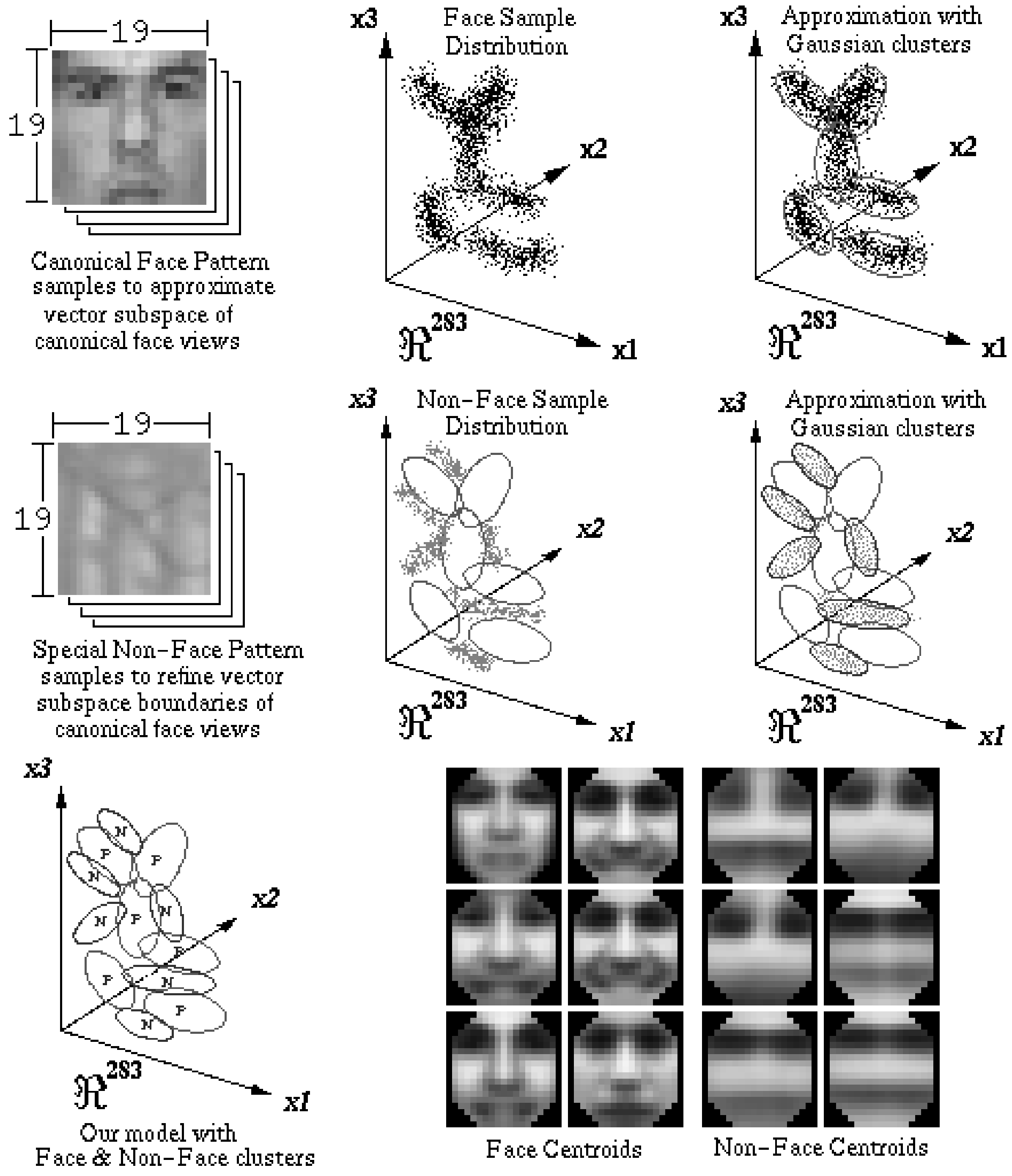

Figure 3: Our distribution-based canonical face model. **Top Row:** We use a representative sample of canonical face patterns to approximate the volume of canonical face views in a *masked* 19×19 pixel image vector space. We model the "face" sample distribution with 6 multi-dimensional Gaussian clusters. **Center Row:** We use a selection of non-face patterns to help refine the boundaries of our Gaussian mixture approximation. We model the "non-face" sample distribution with 6 Gaussian clusters. **Bottom Row:** Our final model consists of 6 "face" clusters and 6 "non-face" clusters. Each cluster is defined by a centroid and a covariance matrix. The 12 centroids are shown on the right. **Note:** All the distribution plots are fictitious and are shown only to help with our explanation. The 12 centroids are real.

the modeling and classification tasks a little more tractable.

3. **Illumination gradient correction:** This operation subtracts a best-fit brightness plane from the unmasked window pixels. For face patterns, it does a fair job at reducing the strength of heavy shadows caused by extreme lighting angles.

4. **Histogram equalization:** This operation adjusts for several geometry independent sources of window pattern variation, including changes in illumination brightness and differences in camera response curves.

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

3.3 Modeling the Distribution of "Face" Patterns — Clustering for Positive Prototypes

We use a database of 4150 normalized canonical "face" patterns to synthesize 6 "face" pattern prototypes in our multi-dimensional image vector space. The database consists of 1067 real face patterns, obtained from several different image sources. We artificially enlarge the original database by adding to it slightly rotated versions of the original face patterns and their mirror images. This helps to ensure that our final database contains a reasonably dense and representative sample of canonical face patterns.

Each pattern prototype is a multi-dimensional Gaussian cluster with a centroid location and a covariance matrix that describes the local data distribution around the centroid. Our modeling scheme approximates the observed "face" sample distribution with only a small number of prototype clusters because the sample size is still very small compared to the image vector space dimensionality (4150 data samples in a 283 dimensional *masked* image vector space). Using a large number of prototype clusters could lead to overfitting the observed data distribution with poor generalization results. Our choice of 6 pattern prototypes is somewhat arbitrary. 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 prototypes.

We use a modified version of the *k-means* clustering algorithm to compute 6 face pattern centroids and their cluster covariance matrices from the enlarged database of 4150 face patterns. Our clustering algorithm differs from the traditional *k-means* algorithm in that it fits directionally elongated (i.e. elliptical) Gaussian distributions to the data sample instead of isotropic Gaussian distributions. We adopt a non-isotropic mixture model because we believe the actual face data distribution may in fact be locally more elongated along certain vector space directions than others.

To implement our *elliptical k-means* clustering algorithm, we use an adaptively changing *normalized Mahalanobis* distance metric instead of a standard *Euclidean* distance metric to partition the data sample into clusters. The normalized Mahalanobis distance metric returns a directionally dependent distance value between a new pattern $\vec{x}$ (in column vector form) and a Gaussian cluster centroid $\vec{\mu}$ (also as a column vector). It has the form:

Figure 4: An example of a naturally occuring "non-face" pattern that resembles a face. **Left:** The pattern viewed in isolation. **Right:** The pattern viewed in the context of its environment.

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

where Σ is the covariance matrix that encodes the cluster's shape and directions of elongation. The normalized Mahalanobis distance differs from the Euclidean distance in that it reduces the penalty of pattern differences along a cluster's major directions of data distribution. This penalty adjustment feature allows the clustering algorithm to form elliptical clusters instead of spherical clusters where there is a non-isotropic local data distribution. Section 4.2 explains the *normalized Mahalanobis* distance metric in greater detail.

The following is a crude outline of our *elliptical k-means* clustering algorithm:

1. Obtain k (6 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. Re-compute 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, re-compute data partitions by re-assigning 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 *inner-loop* (i.e. Steps 3 and 4) iterations has been exceeded, proceed to **Step 5**. Otherwise, return to **Step 3**.

5. Re-compute the covariance matrices of all k clusters from their respective data partitions.

6. Using the current set of k pattern centers and their cluster covariance matrices, re-compute data partitions by re-assigning 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*

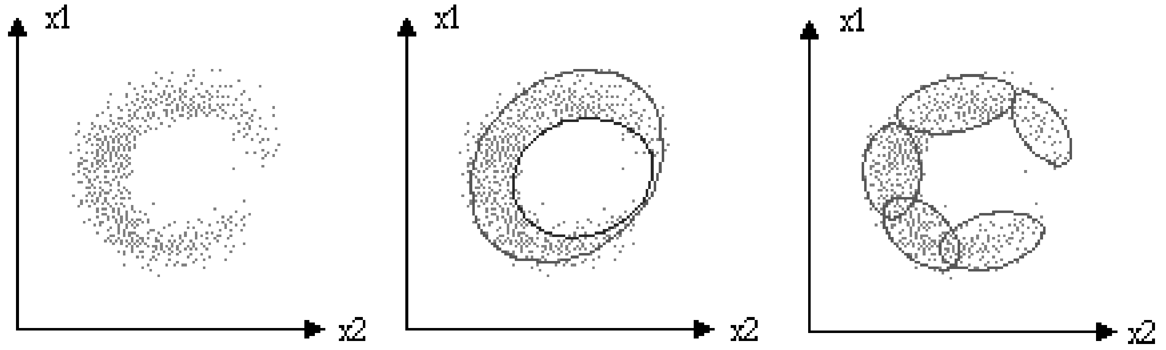

Figure 5: An illustration on how "negative" prototypes can help model concavities in a distribution with fewer Gaussian basis functions. **Left:** A hypothetical data distribution in 2 dimensions with a concave surface. **Center:** The distribution can be reasonably approximated with one *positive* Gaussian prototype encircling the distribution and one *negative* prototype carving out the concave region. **Right:** To model the same distribution with only positive prototypes, one has to use more Gaussian basis functions. The difference in number of basis functions can be a lot larger in a higher dimensional space.

(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 analogous to the traditional *k-means* algorithm. Given a fixed distance metric, it finds a set of k pattern prototypes that stably 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 as well.

3.4 Modeling the Distribution of "Non-Face" Patterns — Clustering for Negative Prototypes

There are many naturally occuring "non-face" patterns in the real world that look like faces when viewed in isolation. Figure 4 shows one such example. In our multi-dimensional image window vector space, these "face-like" patterns lie along the boundaries of the canonical face manifold. Because we are coarsely representing the canonical face manifold with 6 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 misclassification problems, because in general, we expect the opposite to be true, i.e. face patterns should lie nearer the "face" cluster centroids than non-face patterns.

In order to avoid possible misclassification, we try to obtain a comprehensive sample of these face-like patterns and explicitly model their distribution using 6 "non-face" prototype clusters. These "non-face" clusters carve out negative regions around the "face" clusters that do not correspond to face patterns. Each time a new window pattern lies too close to a "non-face" prototype, we favor a "non-face" hypothesis even if the pattern also lies near a "face" prototype. Our choice of using exactly 6 pattern prototypes to model the "non-face" distribution is also somewhat arbitrary, as in the case of modeling the "face" pattern distribution. The system's face detection rate versus false alarm ratio does not change much with slightly fewer or more "non-face" pattern prototypes.

We use our *elliptical k-means* clustering algorithm to obtain 6 "non-face" prototypes and their cluster covariance matrices from a database of 6189 face-like patterns. The database was incrementally generated in a "boot-strap" 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 random images. Section 5.2 elaborates further on our "boot-strap" data generation technique.

3.5 Summary and Remarks

One of the key difficulties in face detection is to account for the wide range of permissible face pattern variations that cannot be adequately captured by geometric models, correlation templates and other classical parametric modeling techniques. Our approach addresses this problem by modeling the distribution of frontal face patterns directly in a fixed 19×19 pixel image vector space. We infer the actual distribution of face patterns in the image vector space from a sufficiently representative set of face views, and a carefully chosen set of non-face patterns. The distribution-based modeling scheme statistically encodes observable face pattern variations that cannot otherwise be easily parameterized by classical modeling techniques. To generalize from the slow varying nature of the face pattern distribution in the image vector space, and to construct a tractable model for the face pattern distribution, we interpolate and represent the observed distribution in a piecewise-smooth fashion using a few multi-dimensional Gaussian clusters.

Our final distribution-based model consists of 6 "face" clusters for coarsely approximating the canonical face pattern manifold in the image vector space, and 6 "non-face" clusters for representing the distribution of a specially selected "non-face" data sample. The non-face patterns come from non-face regions in the image vector space near the canonical face manifold. We propose two

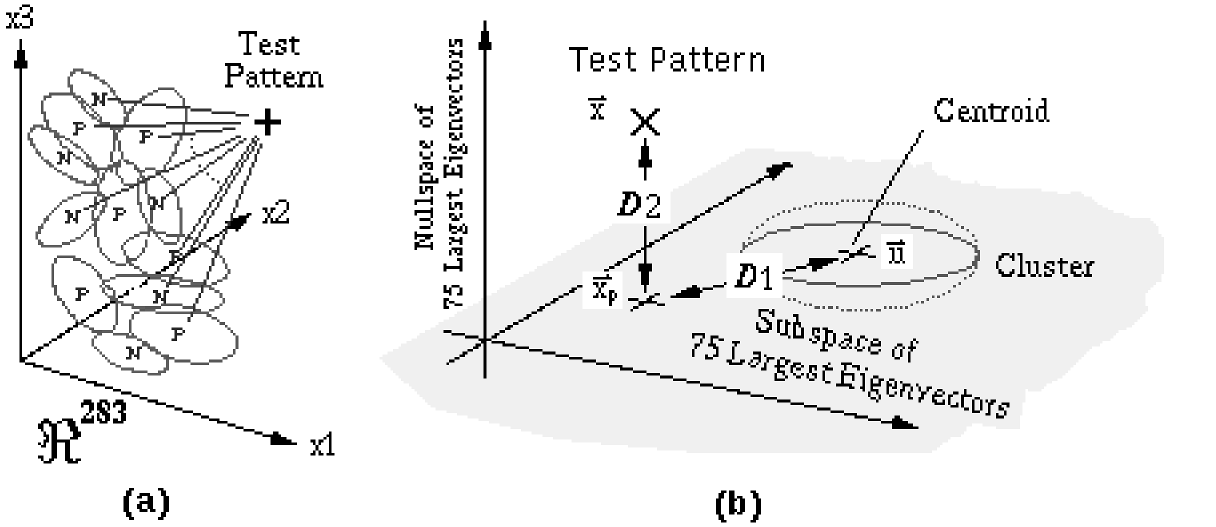

Figure 6: The measurements returned from matching a test pattern against our distribution-based model. **(a):** Each set of measurements is a vector of 12 distances between the test pattern and the model's 12 cluster centroids. **(b):** Each distance measurement between the test pattern and a cluster centroid is a 2-value distance metric. The first component 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. The second component is the Euclidean distance between the test pattern and its projection in the subspace. So, the entire set of 12 distance measurements is a vector of 24 values.

possibly related ways of reasoning about the 6 "non-face" pattern clusters:

1. The 6 "non-face" clusters explicitly model the distribution of face-like patterns that are not actual faces — i.e. "near miss" patterns that the face detection system should classify as "non-faces". They define and represent a separate "near-miss" pattern class in the multi-dimensional vector space, just like how the 6 "face" clusters encode a "canonical face" pattern class. Test patterns that fall under this "near-miss" class are classified as non-faces.
2. The 6 "non-face" clusters carve out regions in and around the "face" clusters that should not correspond to face patterns. They help shape the boundaries in our Gaussian mixture model of the canonical face manifold. More interestingly, they can also help one model concavities in the canonical face manifold with fewer Gaussian basis functions (see Figure 5). This advantage of using "non-face" clusters becomes especially critical when one has too few face data samples to model a concave distribution region with a large number of Gaussian "face" clusters.

4 Matching Patterns with the Model

To detect faces, our system first matches each candidate window pattern in an input image against our distribution-based canonical face model. Each match returns a set of measurements that captures a notion of "similarity" or "difference" between the test pattern and the face model. At a later stage, a trained classifier determines, based on the set of "match" measurements, whether or not the test pattern is a frontal face view.

This section describes the set of "match" measurements we compute for each new window pattern. Each set of measurements is a vector of 12 distances between the test window pattern and the model's 12 cluster centroids in our multi-dimensional image vector space (see Figure 6(a)). One way of interpreting our vector of distances is to treat each distance measurement as the test pattern's actual distance from some key reference location on or near the canonical face pattern manifold. The set of all 12 distances can then be viewed as a crude "difference" notion between the test pattern and the entire "canonical face" pattern class.

4.1 A 2-Value Distance Metric

We now define a new metric for measuring distance between a test pattern and a prototype cluster centroid. Ideally, we want a metric that returns small distances between face patterns and the "face" prototypes, and either large distances between non-face patterns and the "face" prototypes, or small distances between non-face patterns and the "non-face" centers. We propose one such measure that normalizes distances by taking into account both the local data distribution around a prototype centroid, and the reliability of the local distribution estimate. The measure scales up distances orthogonal to a cluster's major axes of elongation, and scales down distances along the cluster's major elongation directions. We argue that such a distance measure should meet our ideal goals reasonably well, because we expect most face patterns on the face manifold to lie along the major axes of one or more "face" clusters, and hence register small distances with one or more "face" prototypes. Similarly, we expect most non-face patterns to lie either far away from all the "face" clusters or along the major axes of

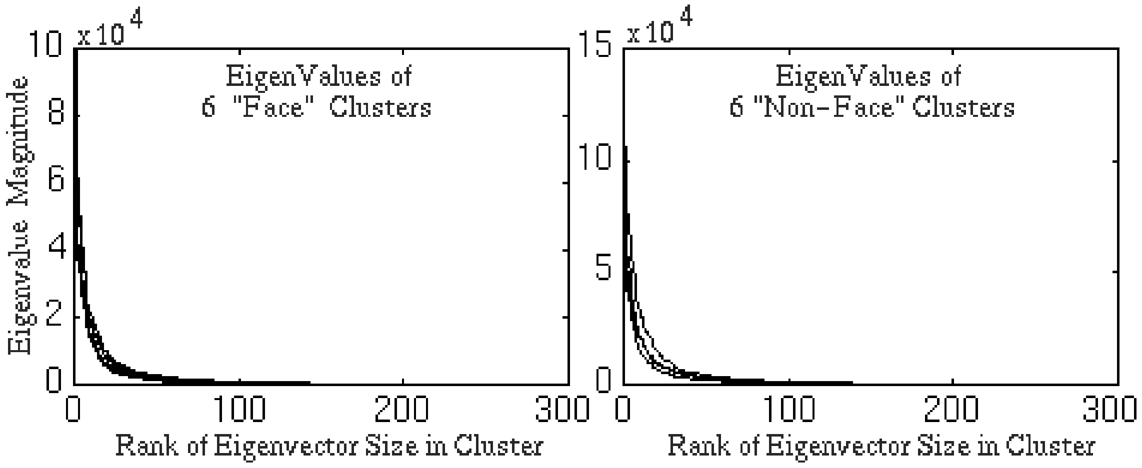

Figure 7: Graphs of eigenvalues in decreasing order of magnitude. **Left:** Decreasing eigenvalues for 6 "Face" clusters. Right : Decreasing eigenvalues for 6 "Non-Face" clusters.

one or more "non-face" clusters, and hence either register large distances with all the "face" prototypes or small distances with some "non-face" prototypes.

Our proposed distance measure consists of two output components (see Figure 6(b)). The first value is a *Mahalanobis-like* distance between the test pattern and the prototype centroid, defined within a lower-dimensional sub-space of our masked 19×19 pixel image vector space. The sub-space is spanned by the 75 largest eigenvectors of the prototype cluster. This distance component is directionally weighted to reflect the test pattern's location relative to the major elongation directions in the local data distribution around the prototype center. The second value is a normalized Euclidean distance between the test pattern and its projection in the lower-dimensional sub-space. 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 *Mahalanobis* distance. Let $\vec{x}$ be a column vector test pattern, $\vec{\mu}$ be a column vector prototype centroid, and Σ be the covariance matrix describing the local data distribution near the centroid. The *Mahalanobis distance* between the test pattern and the prototype centroid is given by:

$$\mathcal{M}(\vec{x},\vec{\mu}) = (\vec{x}-\vec{\mu})^T \Sigma^{-1} (\vec{x}-\vec{\mu}).$$

Geometrically, the *Mahalanobis distance* can be interpreted as follows. If one models the prototype cluster with a best-fit multi-dimensional Gaussian distribution, then one gets a best-fit Gaussian distribution centered at $\vec{\mu}$ with covariance matrix Σ. All points at a given *Mahalanobis distance* from $\vec{\mu}$ occupy a constant density surface in this multi-dimensional vector space. This interpretation of the *Mahalanobis distance* leads to a closely related distance measure, which we call the *normalized Mahalanobis distance*, given by:

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

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

The *normalized Mahalanobis* distance is simply the negative natural logarithm of the best-fit Gaussian distribution described above. It is normalized in the sense that it originates directly from a probability density distribution that integrates to unity. Our modified *k-means* clustering algorithm in Section 3 uses the *normalized Mahalanobis* distance metric instead of the standard form because of stability reasons. Notice that for a given spatial displacement between a test pattern and a prototype centroid, the standard *Mahalanobis* distance tends to be smaller for long clusters with large covariance matrices than for small clusters. So, if one uses a standard *Mahalanobis* distance metric to perform clustering, the larger clusters will constantly increase in size and eventually overwhelm all the smaller clusters.

As a distance metric for establishing the class identity of test patterns, the *Mahalanobis distance* and its normalized form are both intuitively pleasing for the following reason: They both measure pattern differences in a distribution dependent manner, unlike the Euclidean distance which measures pattern differences in an absolute sense. More specifically, the distances they measure are indicative of how the test pattern ranks relative to the overall location of other known patterns in the pattern class. In this sense, they capture very well the notion of "similarity" or "dis-similarity" between a test pattern and a pattern class.

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

As mentioned earlier, our distance metric consists of 2 output values as shown in Figure 6(b). The first value, $\mathcal{D}_1$, is a *Mahalanobis-like* distance between the

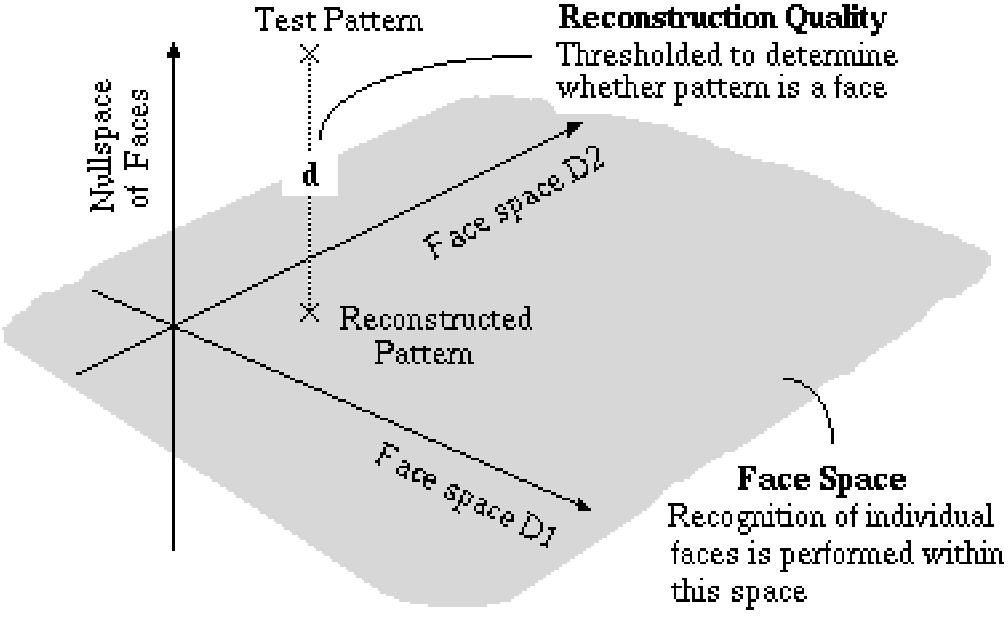

Figure 8: The eigen-image model of faces by Turk and Pentland [19] [12]. The approach represents the space of all face views (i.e. the "Face Space") as a linear combination of several orthonormal eigen-images. The eigen-images are computed by performing PCA on an example database of face patterns. The modeling scheme assumes that all face views lie within the "face space", and each face view corresponds to a set of coefficients that describes its appearance as a linear combination of the eigen-images. One can use the coefficients as features to recognize faces of different individuals. One can also determine whether or not a test pattern is a face by computing and thresholding d, a measure of how badly the eigen-images reconstruct the test pattern.

test pattern and the prototype center. This distance is defined within a 75-dimensional sub-space of our masked 19×19 pixel image vector space, spanned by the 75 largest eigenvectors of the current prototype cluster. Geometrically, we can interpret the first distance value as follows: The test pattern is first projected onto the 75 dimensional vector sub-space. We then compute the normalized Mahalanobis distance between the test pattern's projection and the prototype center as the first output value. Like the standard Mahalanobis distance, this first value locates and characterizes the test pattern relative to the cluster's major directions of data distribution.

Mathematically, the first value is computed as follows: Let $\vec{x}$ be the column vector test pattern, $\vec{\mu}$ be the prototype pattern, E_{75} be a matrix with 75 columns, where column i is a unit vector in the direction of the cluster's i^{th} 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 $\Sigma_{75} = (E_{75} W_{75} E_{75}^T)$, and the first distance value is:

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

We emphasize again that this first value is not a "complete" distance measure in our masked 19×19 pixel image vector space, but rather a "partial" distance measure in a subspace of the current cluster's 75 most significant eigenvectors. The measure is not "complete" in the sense that it does not account for pattern differences in certain image vector space directions, namely differences orthogonal in direction to the cluster's 75 most significant eigenvectors. We omit the smaller eigenvectors in this directionally dependent distance measure because we believe that their corresponding eigenvalues may be significantly inaccurate due to the small number of data samples available to approximate each cluster. For instance, we have, on the average, fewer than 700 data points to approximate each "face" cluster in a 283 dimensional masked image vector space. Using the smaller eigenvectors and eigenvalues to compute a distribution dependent distance can therefore easily lead to meaningless results.

Figure 7 plots the eigenvalues of our 12 prototype clusters in decreasing order of magnitude. Notice that for all 12 curves, only a small number of eigenvectors are significantly larger than the rest. We use the following criterion to arrive at 75 "significant" eigenvectors for each cluster: We eliminate from each cluster the maximum possible number of trailing eigenvectors, such that the sum of all eliminated eigenvalues is still smaller than the largest eigenvalue in the cluster. This criterion leaves us with approximately 75 eigenvectors for each cluster, which we standardize at 75 for the sake of simplicity.

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

The second output value of our distance metric is a standard Euclidean distance between the test pattern and its projection in the 75 dimensional sub-space. This distance component accounts for pattern differences not

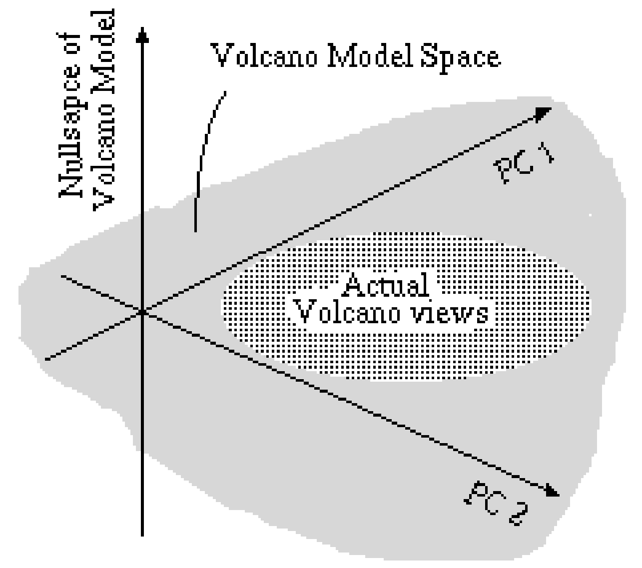

Figure 9: The view-based model of volcanos by Burl et. al. [4]. The approach assumes that the set of all volcano patterns lies within a low dimensional model space, spanned by a small number of the largest principal component vectors from a volcano image database. Each pattern in the model space corresponds to a set of coefficients that describes the pattern's appearance as a linear combination of the principal component images. Not all linear combinations of principal component images are volcano patterns, so one can detect volcanos by learning from examples the range of coefficient values that corresponds to volcanos.

captured by the first component, namely pattern differences in the smaller eigenvector directions. Because we may not have a reasonable estimate of the smaller eigenvalues, we simply assume an isotropic Gaussian sample data distribution in the smaller eigenvector directions, and hence a Euclidean distance measure.

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

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

Notice that on its own, the second distance value is also not a "complete" distance measure in our masked 19×19 pixel image vector space. Specifically, it does not account for pattern differences in the subspace of the cluster's 75 most significant eigenvectors. It complements the first output value to form a "complete" distance measure that captures pattern differences in all directions of our masked 19×19 pixel image vector space.

4.5 Relationship between our 2-Value Distance and the Mahalanobis Distance

There is an interesting relationship between our 2-Value distance metric and the "complete" normalized Mahalanobis distance involving all 283 eigenvectors of a prototype cluster [1]. Recall from Section 4.2 that the Mahalanobis distance and its normalized form arise from fitting a multi-dimensional full-covariance Gaussian to a sample data distribution. For high dimensional vector spaces, modeling a distribution with a full-covariance Gaussian is often not feasible because one usually has too few data samples to recover the covariance matrix accurately. In a d dimensional vector space, each full-covariance Gaussian requires d parameters to define its centroid and another $d(d+1)/2$ more parameters to define its covariance matrix. For $d = 283$, this amounts to 40469 parameters!

One way of reducing the number of model parameters is to use a restricted Gaussian model with a diagonal covariance matrix, i.e., a multi-dimensional 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, but whose pair-wise 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 a d dimensional Gaussian with h principal components has a covariance matrix of only $h(2d - h + 1)/2$ free parameters. This can easily be a tractable number of model parameters if h is small. Geometrically, the operation corresponds to projecting the original d dimensional Gaussian cluster onto an h dimensional subspace, spanned by the cluster's h most "significant" elongation directions. The h dimensional subspace projection preserves the most prominent correlations between pixels in the target pattern class. To exploit these pixel-wise 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 2-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. The isotropic Gaussian distribution requires only 1 free parameter to describe its variance. To measure distances in this subspace of isotropic data distribution, we use a directionally independent Euclidean distance — $\mathcal{D}_2$ of our 2-Value distance metric.

We can thus view our 2-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 Gaussian model. The approximation degrades gracefully by using its limited degrees of freedom to capture the most prominent pixel correlations in the data distribution. Notice

[1] See [8] for a similar interpretation and presentation. Our explanation here is adapted from theirs.

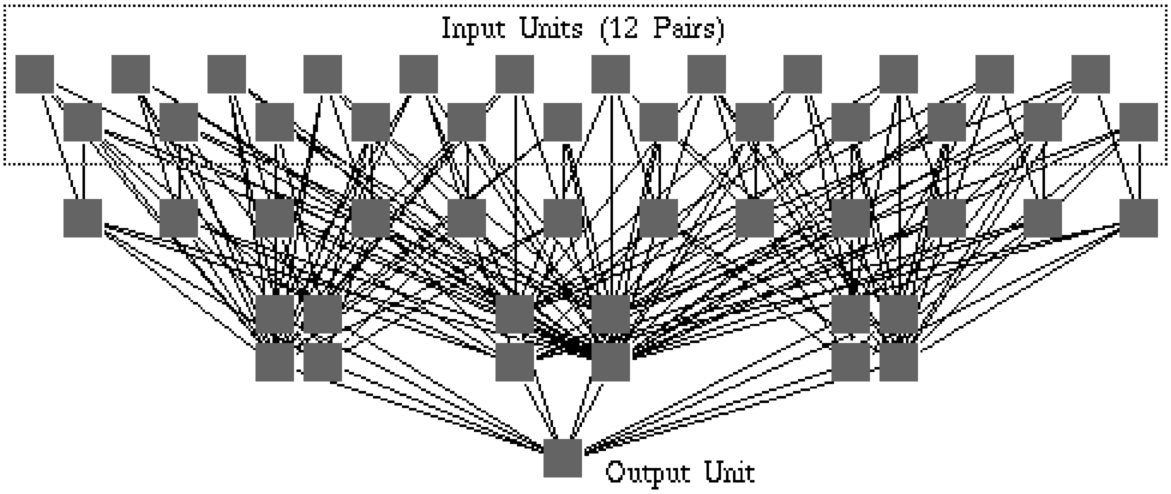

Figure 10: One of the multi-layer perceptron (MLP) net architectures we trained to identify face patterns from our vector of distance measurements. The net has 12 pairs of input units to accept the 12 pairs of distance values from our matching stage. When trained, the output unit returns a 1 for face patterns and a 0 otherwise. Notice that this net has one special layer of hidden units that combines each 2-Value distance pair into a single distance value. Our experiments show that the detailed net architecture is really not crucial in this application.

that 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.

4.6 Relationship between our 2-Value Distance and some PCA-based Classical Distance Measures

Our 2-Value distance metric is also closely related to the classical distance measures used by *principal components analysis* (PCA, also called Karhunen-Loeve expansion) based classification schemes [5] [6] [18]. We examine two such distance measures below.

A standard way of modeling a 3D object for pattern recognition is to represent its space of 2D views using a few basis images, obtained by performing *principal components analysis* on a comprehensive set of 2D views. The modeling approach assumes that all possible 2D views of the target object lie within a low dimensional sub-space of the original high dimensional image space. The low dimensional sub-space is linearly spanned by the principal components. Each new view of the target object can be represented by a set of coefficients that describes the view as a linearly weighted superposition of the principal components.

Kirby and Sirovich [16] [9] have used principal components analysis methods to build low-dimensional models for characterizing the space of human faces. As a representative recent example, let us consider Turk and Pentland's application of PCA techniques to face recognition [19] and facial feature detection [12]. To optimally represent their "face space", Turk and Pentland compute an orthonormal set of eigen-images from an example database of face patterns to span the space. New face views are represented by projecting their image patterns onto the set of eigen-images to obtain their linear combination coefficients. Because the technique assumes that all face patterns actually lie within the "face space", one can use the set of linear combination coefficients as features to parameterize and recognize faces of different individuals (see Figure 8). Similarly, one can determine whether or not a given pattern is a face by measuring how well or poorly the eigen-images reconstruct the pattern — i.e., the pattern's distance (d in Figure 8) from the "face space" [12].

In another recent application of classical techniques, Burl et. al. [4] have used a different PCA based approach for a terrain classification task on finding volcanos in SAR images of Venus. Their system builds a low dimensional view-based volcano model, parameterized by a small number of the largest principal component vectors from an example set of volcano images (see Figure 9). Within this low-dimensional model space, Burl et. al. further observe that not all reconstructible patterns are actual volcano views; i.e., not all linear combinations of principal component images are volcano patterns. Thus, to detect volcanos, they use the set of linear combination coefficients as classification features, and learn from examples the range of coefficient values, i.e. distances within the volcano model space, that actually correspond to volcano patterns.

One can relate our 2-Value distance metric to the above PCA based classification techniques as follows: Suppose we treat each Gaussian cluster in our distribution-based model as locally representing a sub-class of "face" or "non-face" patterns, then the cluster's 75 largest eigenvectors can be viewed as a set of basis images that linearly span the sub-class. Each new pattern in the sub-class corresponds to a set of 75 coefficients in the model space.

Our distance component $\mathcal{D}_2$ is the Euclidean distance between a test pattern and the cluster's 75 largest eigenvector subspace. Assuming that the 75 eigenvectors are

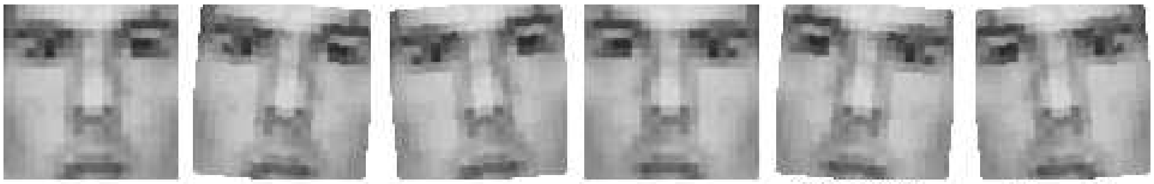

Figure 11: Artificially generating *virtual examples* of face patterns. For each original face pattern in our training database, we can generate a few new face patterns using some simple image transformations. We artificially enlarge our face database in this fashion to obtain a more comprehensive sample of face patterns.

the "basis views" that span our target sub-class, this distance component measures how poorly the basis views reconstruct the test pattern, and hence how "different" the test pattern is relative to the entire target sub-class. This distance is analogous to Turk and Pentland's *distance to "face space"* metric (i.e. d in Figure 8), which quantifies the "difference" between a test pattern and the class of face views.

The distance component $\mathcal{D}_1$ is a Mahalanobis distance between the cluster centroid and a test pattern's projection in the 75 largest eigenvector subspace. Here, we implicitly assume, like Burl et. al., that not all reconstructible patterns in our 75 eigenvector model space are views of the target sub-class, and that all views of the target sub-class are located within a certain Mahalanobis distance from the cluster centroid. This constrains the range of model coefficients that correspond to actual target patterns, just like how Burl et. al. constrain by learning their range of model parameters that correspond to actual volcano views.

Thus, our $\mathcal{D}_1$ and $\mathcal{D}_2$ distance measurements correspond to different classical classification criteria used also recently. As far as we know, this paper presents the first attempt to combine these two measures for pattern classification.

5 The Classifier

The classifier's task is to identify "face" test patterns from "non-face" patterns based on their match feature vectors of 12 distance measurements. Our approach treats the classification stage as one of learning a functional mapping from input feature distance vectors to output classes using a representative set of training examples.

5.1 A Multi-Layer Perceptron Classifier

We use a *Multi-Layer Perceptron* (MLP) net to perform the desired classification task. The net has 12 pairs of input terminals, 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 prototype centers. Each input terminal pair receives the distance values for a designated prototype pattern. The output unit returns a '1' if the input distance vector arises from a "face" pattern, and a '0' otherwise.

In our current system, the hidden units are partially connected to the input terminals and output unit as shown in Figure 10. The connections exploit some prior knowledge of the problem domain. Notice in particular that there is one layer of hidden units that combines each 2-Value distance pair into a single distance value. Our experiments in the next section will show that the number of hidden units and network connectivity structure do not significantly affect the classifier's performance.

We train our multi-layer perceptron classifier on feature distance vectors from a database of 47316 window patterns. There are 4150 positive examples of "face" patterns in the database and the rest are "non-face" patterns. The net is trained with a standard backpropagation learning algorithm [14] until the output error stabilizes at a very small value. For the particular multi-layer perceptron net architecture in Figure 10, we actually get a training output error of 0.

5.2 Generating and Selecting Training Examples

In many example-based learning applications, how well a learner eventually performs on its task depends heavily on the quality of examples it receives during training. An ideal learning scenario would be to give the learner as large a set of training examples as possible, in order to attain a comprehensive sampling of the input space. Unfortunately, there are some 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 "non-face"" patterns? For "face" patterns, the task at hand seems rather straight forward. We simply collect all the frontal views of faces we can find in mugshot databases and other image sources. Because we do not have access to many mugshot databases and the size of our mugshot databases are all fairly small, we do not encounter the problem of having to deal with an unmanageably large number of "face" patterns. In fact, to make our set of "face" patterns more comprehensive, we even artificially enlarged our data set by adding *virtual examples* [13] of faces to the "face" database. These virtual examples are mirror images and slightly rotated versions of the original face patterns, as shown in Fig-

Figure 12: 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 re-train our system in a "boot-strap" fashion.

ure 11.

For "non-face" patterns, the task we have seems more tricky. In essence, every square non-canonical face window pattern of any size in any image is a valid "non-face" pattern. Clearly, even with a few source images, our set of "non-face" patterns can grow intractably large if we are to include all valid "non-face" patterns in our training database.

To constrain the number of "non-face" examples in our database, we use a "boot-strap" strategy that incrementally selects only those "non-face" patterns with high information value. The idea works as follows:

1. Start with a small and possibly incomplete set of "non-face" examples in the training database.
2. Train the multi-layer perceptron classifier with the current database of examples.
3. Run the face detector on a sequence of random images. Collect all the "non-face" patterns that the current system wrongly classifies as "faces" (see Figure 12). Add these "non-face" patterns to the training database as new negative examples.
4. Return to Step 2.

At the end of each iteration, the "boot-strap" strategy enlarges the current set of "non-face" patterns with new "non-face" patterns that the current system classifies wrongly. We argue that this strategy of collecting wrongly classified patterns as new training examples is reasonable, because we expect these new examples to improve the classifier's performance by steering it away from the mistakes it currently commits.

Notice that if necessary, we can use the same "boot-strap" technique to enlarge the set of positive "face" patterns in our training database. Also, notice that at the end of each iteration, we can re-cluster our "face" and "non-face" databases to generate new prototype patterns that might model the distribution of face patterns more accurately.

6 Results and Performance Analysis

We implemented and tested our face detection system on a wide variety of images. Figure 13 shows some sample results. The system detects faces over a fairly large range of scales, beginning with a window size of 19×19 pixels and ending with a window size of 100×100 pixels. Between successive scales, the window width is enlarged by a factor of 1.2.

The system writes its face detection results to an output image. Each time a "face" window pattern is found in the input, an appropriately sized dotted box is drawn at the corresponding window location in the output image. Notice that many of the face patterns in Figure 13 are enclosed by multiple dotted boxes. This is because the system has detected those face patterns either at a few different scales or at a few slightly offset window positions or both.

The upper left and bottom right images of Figure 13 show that our system works reliably without making many false positive errors (none in this case), even for fairly complex scenes. Notice that the current system does not detect Geordi's face. This is because Geordi's face differs significantly from the notion of a "typical" face pattern in our training database of faces — his eyes are totally occluded by an opaque metallic visor. The upper right input-output image pair demonstrates that the same system detects real faces and hand-drawn faces equally well, while the bottom left image shows that the system finds faces successfully at two very different

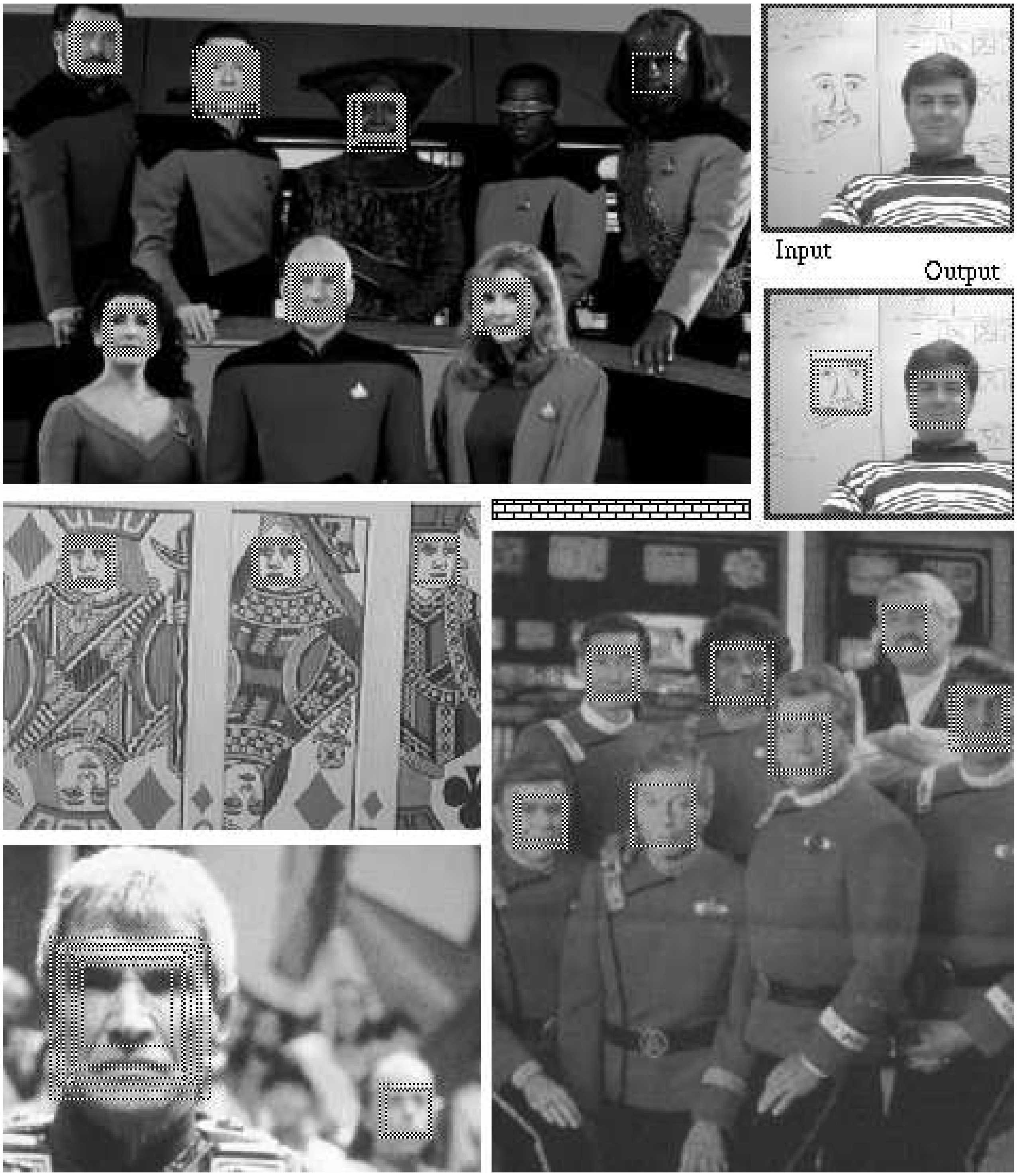

Figure 13: Some face detection results by our system. The upper left and bottom right images show that the system finds faces in complex scenes without making many false detects. The upper right image pair shows that the same system detects real faces and hand drawn faces equally well. The bottom left image shows the system working successfully at two very different scales. Notice that in the cards image, the system detects only the vertically oriented face patterns as it is meant to at this time.

scales.

6.1 Measuring the System's Performance

To quantitatively measure our system's performance, we ran our system on two test databases and counted the number of correct detections versus false alarms. All the face patterns in both test databases are new patterns not found in the training data set. 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 taken by a CCD camera in a laboratory environment. There is a fair amount of lighting variation among images in this database, with about 10 images having very strong lighting shadows. We use this database to obtain a "best case" detection rate for our system on high quality input patterns.

The second database contains 23 images with a total of 149 face patterns. There is a wide variation in quality among the 23 images, 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 on a more representative sample of input images.

For the first database, our system correctly finds 96.3% of all the face patterns and makes only 3 *false detects*. All the face patterns that it misses have either strong illumination shadows or fairly large off-plane rotation components or both. Even though this high detection rate applies only to high quality input images, we still find the result encouraging because often, one can easily replace poorer sensors with better ones to obtain comparable results. For the second database, our system achieves a 79.9% detection rate with 5 *false positives*. The face patterns it misses are mostly either from low quality newspaper scans or hand drawn pictures. We consider this behavior acceptable because the system is merely degrading gracefully with poorer image quality.

6.2 Analyzing the System's Components

We conducted the following additional experiments to identify the key components of our face detection algorithm. Specifically, we examined how the following three aspects of our system affects its performance in terms of face detection rate versus false alarm ratio: (1) The classifier architecture, (2) our 2-Value distance metric versus other distance measures for computing feature distance vectors, and (3) the importance of "non-face" prototypes in our distribution-based face model.

Classifier Architecture The first experiment investigates how varying the classifier's architecture affects our system's overall 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 multi-layer perceptron net in Figure 10 with a single perceptron unit connected directly to the 12 pairs of input terminals. The single perceptron unit computes a sigmoidally thresholded weighted sum of its input feature distance vector. It represents the simplest possible architecture in the family of multi-layer perceptron net classifiers, and its purpose here is to provide an "extreme case" performance figure for multi-layer perceptron net classifiers in this problem domain.
2. **A nearest neighbor classifier.** We perform nearest neighbor classification on feature distance vectors to identify new test patterns. The nearest neighbor classifier works as follows: For each training pattern, we compute and store its 24 value feature distance vector and output class at compile time. When classifying a new test pattern, we compute its 24 value feature distance 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.

The Distance Metric The second experiment investigates how using a different distance metric for computing feature distance vectors in the matching stage affects the system's performance. We compare our 2-Value distance metric with three other distance measures: (1) the normalized Mahalanobis distance within a 75 dimensional vector subspace spanned by the prototype cluster's 75 largest eigenvectors — i.e. the first component only ($\mathcal{D}_1$) of our 2-Value distance metric, (2) the Euclidean distance between the test pattern and its projection in the 75 dimensional subspace — i.e. the second component only ($\mathcal{D}_2$) of our 2-Value distance metric, and (3) the standard normalized Mahalanobis distance ($\mathcal{M}_n$) between the test pattern and the prototype 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. Each new set of experiments differs from the original set as follows: During the matching stage, we use one of the three distance measures above instead of the original 2-Value distance metric to compute feature distance vectors between new test patterns and the 12 prototype centroids. Notice that because the three new distance measures are all single-value measurements, our new feature distance vectors have only 12 values instead of 24 values. This means that we have to modify the classifier architectures accordingly by reducing the number of input terminals from 24 to 12.

"Non-Face" Prototypes This experiment looks at how differently the system performs with and without "non-face" prototypes in the distribution-based model. We compare results from our original system, whose face model contains both "face" and "non-face" prototypes, against results from two new systems whose internal models contain only "face" prototypes. The two new systems are:

1. **A system with 12 "face" prototypes and no "non-face" prototypes.** In this system, we fix the total number of pattern prototypes in the

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

Table 1: Summary of performance figures from Experiments 1 (Classifier Architecture) and 2 (Distance Metric). Detection rates versus number of false positives for different classifier architectures and distance metrics. 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. Notice that we did not test the *nearest neighbor* architecture on the second database. This is because the test results from the first database already show that the *nearest neighbor* classifier is significantly inferior to the other 2 classifiers in this problem domain.

canonical face model, and hence the dimensionality of the feature distance vector the matching stage computes. We obtain the 12 "face" prototypes by performing elliptical *k-means* clustering with 12 centers on our enlarged canonical face database of 4150 patterns (see Section 3.3). The matching stage computes the same 2-Value distance metric that our original system uses; i.e., for each prototype cluster, $\mathcal{D}_1$ is the normalized Mahalanobis distance in a subspace of the 75 largest eigenvectors, and $\mathcal{D}_2$ is the Euclidean distance between the test pattern and the subspace.

2. **A system with only 6 "face" prototypes.** In this system, we preserve only the "face" prototypes from the original system. The matching stage computes the same 2-Value distances between each test pattern and the 6 "face" centroids. Notice that the resulting feature distance vectors have only 6 pairs of values.

We generate two sets of performance statistics for each the two new systems above. For the first set, we use a trained multi-layer perceptron net with 12 hidden units to classify new patterns from their feature distance vectors. For the second set, we replace the multi-layer perceptron net classifier with a trained single perceptron unit classifier.

6.3 Performance Figures and Interpretation

Table 1 summarizes the performance statistics for both the *classifier architecture* and *distance metric* experiments, while Table 2 shows how the system performs with and without "Non-Face" model prototypes.

Classifier Architecture The horizontal rows of Table 1 show how different classifier architectures affect the system's face detection rate versus false alarm instances. Quantitatively, the two network-based classifiers have very similar performance figures, while the *nearest neighbor* classifier produces rather different performance statistics. 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 other two classifiers. Because we do not have a reasonable measure of relative importance between face detection rate and number of false alarms, we empirically rank the performance figures by visually examining their corresponding output images. In doing so, we find that the two network-based classifiers produce results that are a lot more visually appealing than the nearest neighbor classifier.

It is interesting that the two network-based classifiers we tested produce very similar performance statistics, especially on the first test database. The similarity here suggests that the system's performance depends most critically on the classifier type, i.e. a perceptron net, and not on the specific net architecture. It is also somewhat surprising that one can get very encouraging performance figures even with an extremely simple single perceptron unit classifier, especially when using our 2-Value distance metric to compute feature distance vectors. This suggests that the distance vectors we get with our 2-Value distance metric are a linearly very separable set of features for "face" and "non-face" patterns.

Why does the *nearest neighbor* classifier have a much lower face detection rate than the other two network-based classifiers when used with our 2-Value distance metric? We believe this has to do with the much larger number of non-face patterns in our training database than face patterns (43166 non-face patterns versus 4150 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 feature distance vector space. Assuming that roughly the same fraction of face and non-face patterns lie along the linear class boundary, we get a boundary population with 10 times more non-face samples than face samples. A new face pattern near the class boundary will therefore have a very high chance of lying nearer a non-face sample than a face sample, and hence be wrongly classified by a nearest neighbor scheme. Notice that despite the huge difference in number of face and non-face train-

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

Table 2: Summary of performance figures for Experiment 3 ("Non-Face" Prototypes). Detection rates versus number of false positives for different classifier architectures and composition of prototypes in distribution-based model. 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.

ing patterns, a perceptron classifier can still locate the face vs. non-face class boundary accurately if the two pattern classes are indeed linearly highly separable, and hence still produce good classification results.

The Distance Metric The vertical columns of Table 1 show how the system's performance changes with different distance metrics in the pattern matching stage. Both the multi-layer perceptron net and single perceptron classifier systems produce better performance statistics with our 2-Value distance metric than with the other three distance measures, especially on images from the second test database. The observation should not at all be surprising. Our 2-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 2-Value distance should therefore produce performance figures that are at least as good as a similar system that uses either $\mathcal{D}_1$ or $\mathcal{D}_2$ only. In fact, since our 2-Value distance systems actually produce better classification results than the systems using only $\mathcal{D}_1$ or $\mathcal{D}_2$, one can further affirm that the two components $\mathcal{D}_1$ and $\mathcal{D}_2$ do not mutually contain totally redundant information.

Our 2-Value distance metric should also produce classification results that are at least as good as those obtained with a Mahalanobis distance metric. This is because both metrics treat and quantify *distance* in essentially the same way — as a "difference" notion between a test pattern and a local data distribution. Furthermore, the "difference" notions they use are based on two very similar Gaussian generative models of the local data distribution. In our experiments with perceptron-based classifiers, the 2-Value distance metric actually out-performs the Mahalanobis distance metric consistently. We suggest two possible reasons for this difference in performance: (1) As discussed in Section 4.5, we have too few sample points in our local data distribution to accurately recover a full Gaussian covariance matrix for computing Mahalanobis distances. By naively trying to do so, we get a distance measure that poorly reflects the "difference" notion we want to capture. (2) The local data distribution we are trying to model may not be truly Gaussian. Our 2-Value distance metric provides an additional degree of freedom that better describes a test pattern's location relative to the local non-Gaussian data distribution.

It is interesting that even a network-based classifier with a "partial" $\mathcal{D}_2$ distance metric almost always out-performs a similar classifier with a "complete" Mahalanobis distance metric. At first glance, the observation can be surprising because unlike the "complete" Mahalanobis distance, the $\mathcal{D}_2$ metric alone does not account for pattern differences in certain image vector space directions. More specifically, the $\mathcal{D}_2$ metric only measures a pattern's Euclidean distance from a cluster's 75 most significant eigenvector subspace, and does not account for pattern differences within the subspace. We believe this comparison truly reveals that without sufficient data to accurately recover a full Gaussian covariance matrix, the resulting Mahalanobis distance can be a very poor notion of "pattern difference".

"Non-Face" Prototypes Table 2 summarizes the performance statistics for comparing systems with and without "non-face" prototypes. As expected, the systems with "non-face" prototypes clearly out-perform those without "non-face" prototypes. Our results suggest that not only are the "non-face" prototypes an additional set of features for face detection, they are in fact a very discriminative set of additional features.

7 Conclusion

We have successfully developed a system for finding unoccluded vertical frontal views of human faces in images. The approach is view based. It models the distribution of face patterns by means of a few prototype clusters, and learns from examples a set of distance parameters for distinguishing between "face" and "non-face" test patterns. We stress again, however, that our ultimate goal is to develop our face detection approach into a general methodology for taking on feature detection and pattern recognition tasks in multiple domains.

We plan to further our work in the following two directions. First, we would like to demonstrate the full power of our face detection approach by building a more comprehensive face detection system. One obvious extension would be to have the system detect faces over a wider range of poses instead of just near-frontal views. We believe that our current approach can in fact be used without modification to perform this new task. All we need is a means of obtaining or artificially generating a sufficiently large example database of human faces at the different poses [1].

Second, we would like to demonstrate the versatility and generality of our approach by building a few more feature and pattern detection applications in other problem domains. Some possibilities include industrial inspection applications for detecting defects in manufactured products and terrain feature classification applications for SAR imagery. We believe that the key to making this work would be to find appropriate transformation spaces for each new task, wherein the target pattern classes would be reasonably stable.

References

[1] D. Beymer, A. Shashua, and T. Poggio. Example Based Image Analysis and Synthesis. A.I. Memo No. 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 Transactions on Pattern Analysis and Machine Intelligence*, 15(10):1042–1052, 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. Technical Report CNS TR 34, California Institute of Technology, October 1993.

[5] Richard O. Duda and Peter E. Hart. *Pattern Classification and Scene Analysis.* John Wiley and Sons Inc., New York, 1973.

[6] K. Fukunaga. *Introduction to Statistical Pattern Recognition.* Academic Press, 1990.

[7] W. E. L. Grimson and T. Lozano-Perez. Model-Based Recognition and Localization from Sparse Range Data. In A. Rosenfeld, editor, *Techniques for 3-D Machine Perception.* North-Holland, Amsterdam, 1985.

[8] G. Hinton, M. Revow, and P. Dayan. Recognizing Handwritten Digits using Mixture of Linear Models. In D. Touretzky G. Tesauro and J. Alspector, editors, *Advances in Neural Information Processings Systems 7*, San Mateo, CA, 1995. Morgan Kaufman.

[9] M. Kirby and L. Sirovich. Applications of the Karhunen-Loeve Procedure for the Characterization of Human Faces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(1):103–108, 1990.

[10] B. Kumar, D. Casasent, and H. Murakami. Principal Component Imagery for Statistical Pattern Recognition Correlators. *Optical Engineering*, 21(1), Jan/Feb 1982.

[11] A. Mahalanobis, A. Forman, N. Day, M. Bower, and R. Cherry. Multi-Class SAR ATR using Shift-Invariant Correlation Filters. *Pattern Recognition*, 27(4):619–626, April 1994.

[12] A. Pentland, B. Moghaddam, and T. Starner. View-based and Modular Eigenspaces for Face Recognition. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition*, pages 84–91, June 1994.

[13] T. Poggio and T. Vetter. Recognition and Structure from One (2D) Model View: Observations on Prototypes, Object Classes, and Symmetries. A.I. Memo No. 1347, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 1992.

[14] D. Rumelhart and J. McClelland. *Parallel Distributed Processing*, volume 1. MIT Press, Cambridge, Massachusetts, 1986.

[15] P. Sinha. Object Recognition via Image Invariants: A Case Study. In *Investigative Ophthalmology and Visual Science*, volume 35, pages 1735–1740, Sarasota, Florida, May 1994.

[16] L. Sirovich and M. Kirby. Low-dimensional Procedure for the Characterization of Human Faces. *Journal of the Optical Society of America*, 4(3):519–524, March 1987.

[17] K. Sung and T. Poggio. Example-based Learning for View-based Human Face Detection. In *Proceedings Image Understanding Workshop*, volume II, pages 843–850, Monterey, CA, November 1994.

[18] C. Therrien. *Decision, Estimation and Classification.* John Wiley and Sons, Inc., 1989.

[19] M. Turk and A. Pentland. Eigenfaces for Recognition. *Journal of Cognitive Neuroscience*, 3(1):71–86, 1991.

[20] A. Yuille, P. Hallinan, and D. Cohen. Feature Extraction from Faces using Deformable Templates. *International Journal of Computer Vision*, 8(2):99–111, 1992.