To appear in Proceedings Image Understanding Workshop 1994

Example-based Learning for View-based Human Face Detection

Kah-Kay Sung
sung@ai.mit.edu

Tomaso Poggio
poggio@ai.mit.edu

Artificial Intelligence Laboratory
Massachusetts Institute of Technology
Cambridge, MA 02139

Abstract

Finding human faces automatically in an image is a difficult yet important first step to a fully automatic face recognition system. This paper presents an example-based learning approach for locating unoccluded frontal views of human faces in complex scenes. The technique represents the space of human faces by means of a few view-based "face" and "non-face" pattern prototypes. At each image location, a 2-value distance measure is computed between the local image pattern and each prototype. A trained classifier determines, based on the set of distance measurements, whether a human face exists at the current image location. We show empirically that our distance metric is critical for the success of our system.

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 [1] [6] [7] [8]. 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 system that detects vertically oriented and unoccluded frontal views of human faces in grey-level images. The system handles faces within a range of scales as specified by the user, and works under different lighting conditions, even on faces 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 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.

[0]This report describes research done at the MIT Artificial Intelligence Laboratory and within the Center for Biological and Computational Learning. This research is sponsored by grants from the Office of Naval Research under contracts N00014-91-J-1270 and N00014-92-J-1879; by a grant from the National Science Foundation under contract ASC-9217041. Support for the A.I. Laboratory's artificial intelligence research is provided by ONR contract N00014-91-J-4038.

1.1 The Face Detection problem

We begin by describing face detection problem. Given as input an arbitrary image, return an encoding of the location and spatial extent of each human face in the image. 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 commercial applications.

Why is automatic face *detection* an interesting problem? Application wise, face detection has direct relevance to the problem of face recognition, because it is usually the first important step of a fully automatic face recognizer. So far, the focus of research in face recognition has been mainly on distinguishing individual faces from others in a database [3]. The task of finding faces in an arbitrary background has often been avoided by either hand segmenting the input image, or by capturing faces against a known uniform background.

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', tissue 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.

1.2 Previous Work

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 two main approaches for dealing with allowable pattern variations, namely: (1) the use of correlation templates, and (2) spatial image invariants.

Correlation Templates Correlation templates compute a similarity measurement between a fixed target pattern and candidate image locations. The output is then thresholded for a matching decision. 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 [4] [5]. At a later stage, the technique infers the presence of faces from the spatial relationships between the detected subfeatures.

A very closely related approach to correlation templates is that of *view-based eigen-spaces* [9]. 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. Typically, the approach approximates the sub-space of face patterns using data clusters and their principal components from one or more example sets of face images. An image pattern is classified as "a face" if its distance to the clusters is below a certain threshold, according to an appropriate distance metric. So far, this approach has only been demonstrated on images with not-so-cluttered backgrounds.

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 the local ordinal structure of brightness distribution between different parts of a human face [11].

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 derive a view-based model for the distribution of face patterns in the image domain. 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 scheme does not depend on domain specific knowledge or special hand-crafting techniques to build face models. Instead, it models faces directly by representing them with the distribution of face patterns it receives. This immediately eliminates one potential source of modeling error — that due to inaccurate or incomplete knowledge. Furthermore, because we are modeling with real data, we can expect our face models to be more accurate and more comprehensive than the manually synthesized ones if we use a sufficiently wide sample of example face patterns.

Second, unlike most non learning-based approaches that typically obtain their operating parameters and thresholds manually from a few trial cases, our 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) The thresholds and parameters it arrives at are statistically more reliable because they come from a larger and wider sample of training data. (2) Because our scheme automatically learns thresholds and parameters, it can be easily made to learn a larger and more complex set of thresholds and parameters that may be too tedious for human observers to discover manually.

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

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 can also be easily extended to detect human faces over a wider range of poses by providing it with the relevant training examples.

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 pattern in the image domain for the purpose of pattern matching. Figure 1(a) shows the canonical face pattern 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.

Our system detects faces by scanning the image for these face-like window patterns at all possible scales. At each scale, the image is divided into multiple, possibly overlapping sub-images of the current window size. For each window, the system attempts to classify the enclosed image pattern as being either "a face" or "not a face". Each time a "face" 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 working with 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 used for classification.

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. The approach appropriately models the distribution of canonical face patterns in some high dimensional image window vector space, and learns 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. All window patterns of different dimensions must first be re-scaled to this size 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.
(2) In the 19×19 dimensional image window vector space, we use a few "face" and "non-face" window pattern prototypes to piece-wise approximate the distribution of canonical face patterns. These pattern prototypes serve as a view-based model for the class of canonical face patterns. The pattern prototypes are synthesized off-line from an example database of "face" window patterns and a similar database of "non-face" window patterns. They are hard-wired into the system at compile time. Each prototype is encoded as a multi-dimensional 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 prototype window patterns in the image window vector space. We define a new 2-value distance metric for measuring distances between the input window pattern and the prototype centers. The distance metric takes into account the shape of each prototype cluster, in order to penalize distances orthogonal to the local data distribution.
(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.

The following sections describe our window pattern classification algorithm in greater detail. Section 3 describes 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 Synthesizing Pattern Prototypes

Our approach uses 6 "face" and 6 "non-face" prototype clusters to piece-wise approximate the distribution of canonical face patterns in the image window vector space. We choose a piece-wise continuous modeling scheme because face patterns appear to occupy a smoothly varying and continuous region in this 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. Figure 2 shows the 12 pattern prototypes we sythesized for our scheme. The 6 "face" prototypes are synthesized from a database of canonical face window patterns, while the 6 "non-face" prototypes are derived from a similar database of non-face patterns.

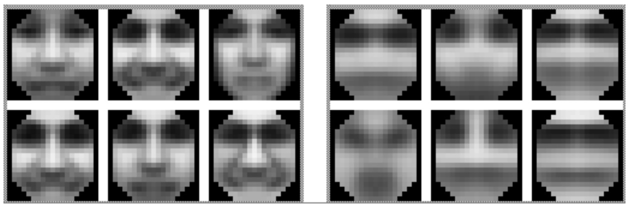

Figure 2: The 12 prototype patterns for approximating the distribution of face patterns. The 6 patterns on the left are "face" prototypes. The 6 on the right are "non-face" prototypes.

3.1 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 manageable small, but also large enough to visually 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.
(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.

3.2 Clustering for "Face" Prototypes

We use a database of 4150 normalized canonical "face" patterns to synthesize 6 "face" pattern prototypes. The database contains 1067 real face patterns, obtained from several different image sources. We artificially enlarge the original database from 1067 patterns to 4150 patterns by adding to it slightly rotated versions of the original face patterns and their mirror images.

We use a modified version of the *k-means* clustering algorithm to compute 6 representative face patterns and their cluster covariance matrices from the enlarged database. Our clustering algorithm differs from the traditional *k-means* algorithm in that it uses an adaptively changing *normalized Mahalanobis* distance metric instead of a standard *Euclidean* distance metric to partition the data samples into clusters. The rationale is that the actual face data clusters may in fact be more elongated along certain directions of the image window vector space than others. So, by using a normalized Mahalanobis distance metric, we can appropriately reduce the penalty of pattern differences along a cluster's major axes of elongation. Section 4.1 explains the *normalized Mahalanobis* distance metric in greater detail.

The following is a crude outline of our 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, go to **Step 5**. Otherwise, go 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* (i.e. Steps 3 to 6) iterations has been exceeded, go to **Step 7**. Otherwise, go 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.

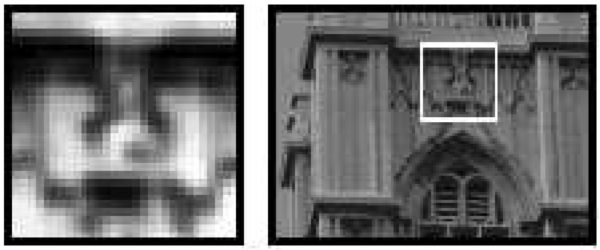

Figure 3: An example of a naturally occurring "non-face" pattern that resembles a face. **Left:** The pattern viewed in isolation. **Right:** The pattern viewed in the context of its environment.

3.3 Clustering for "Non-Face" Prototypes

There are many naturally occurring "non-face" patterns in the real world that look like faces when viewed in isolation. Figure 3 shows one such example. In our 19×19 image window vector space, some of these face-like patterns may even be located nearer the "face" pattern prototypes than some real "face" patterns. This may give rise to misclassification problems if we assume that only real face patterns may lie near the "face" pattern prototypes.

In order to avoid possible misclassification, we explicitly model the distribution of these face-like patterns using 6 "non-face" prototypes. These "non-face" prototypes 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.

We use our modified *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.

4 A 2-Value Distance Metric

Our system classifies a new window pattern based on its distance to each of the 12 prototype centers. In this section, we define a new metric for measuring distances between test patterns and prototype patterns. 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. Although it is difficult, in general, to systematically derive such a metric, we found by experimentation a 2-value distance measure that does reasonably well in terms of meeting these requirements.

Our 2-value distance measure takes into account both the detailed shape of the prototype cluster and the reliability of the shape estimate. The first value is a *Mahalanobis-like* distance between the test pattern and the prototype center, defined within a lower-dimensional sub-space of the original 19×19 dimensional image window vector space. The sub-space is spanned by the larger 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.

4.1 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 prototype center (also a column vector) and Σ be the covariance matrix of its local data distribution. The *Mahalanobis distance* between the test pattern and the prototype center 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 as a best-fit multi-dimensional Gaussian distribution centered at $\vec{\mu}$ with covariance matrix Σ, then all points at a given *Mahalanobis distance* from $\vec{\mu}$ will 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 distribution function that integrates to unity. Our modified *k-means* clustering algorithm in Section 3 uses the *normalized Mahalanobis* distance metric instead of the unnormalized form for stability reasons.

As a distance metric for establishing the class identity of test patterns, the *Mahalanobis distance* and its normalized form are both intuitively pleasing because they 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 the test pattern's location 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 "dissimilarity" between a test pattern and a pattern class.

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

As mentioned earlier, our distance metric consists of a pair of output values. The first value is a *Mahalanobis-like* distance between the test pattern and the prototype center. This distance is defined within a 75-dimensional sub-space of the original image window vector space, spanned by the 75 largest eigenvectors of the current prototype cluster. 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 sub-space is given by $\Sigma_{75} = (E_{75} W_{75} E_{75}^T)$, and the first distance value is:

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

Notice that this first value is not a standard *Mahalanobis* distance measure in the full image window vector space, and in particular, the expression does not include many of the prototype cluster's smaller eigenvectors. We use a measure independent of the smaller eigenvectors because we believe that their corresponding eigenvalue estimates 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, which has 283 unmasked pixel dimensions. Using these smaller eigenvectors and eigenvalues to compute a distribution dependent distance can therefore easily lead to meaningless results.

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

The second component 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 captured by the first component, namely pattern differences within the sub-space spanned by the smaller eigenvectors. Because we may not have a reasonable estimate of the smaller eigenvalues, we assume a radially uniform sample data distribution in this smaller eigenvector sub-space, 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 $\vec{x_p}$:

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

5 The Classifier

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

5.1 A Multi-Layer Perceptron Classifier

Our approach uses 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 in a way that exploits some a-priori knowledge of the problem domain. Our experiments in the next section will show that the number of hidden units and network connectivity structure do not significantly affect the classsifier's performance.

We train our multi-layer perceptron classifier on distance measurements 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 until the output error stabilizes at a very small value.

5.2 Generating and Selecting Training Examples

In many example-based learning applications, how well a learner eventually performs 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, for example the availability 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* [10] of faces to the "face" database. These virtual examples are mirror images and slightly rotated versions of the original face patterns.

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". 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. Figures 4 and 5 show 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 Figures 4 and 5 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 left image in Figure 4 shows 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 right input-output image pair demonstrates that the same system detects real faces and hand-drawn faces equally well, while figure 5 shows that the system finds faces successfully at two very different scales.

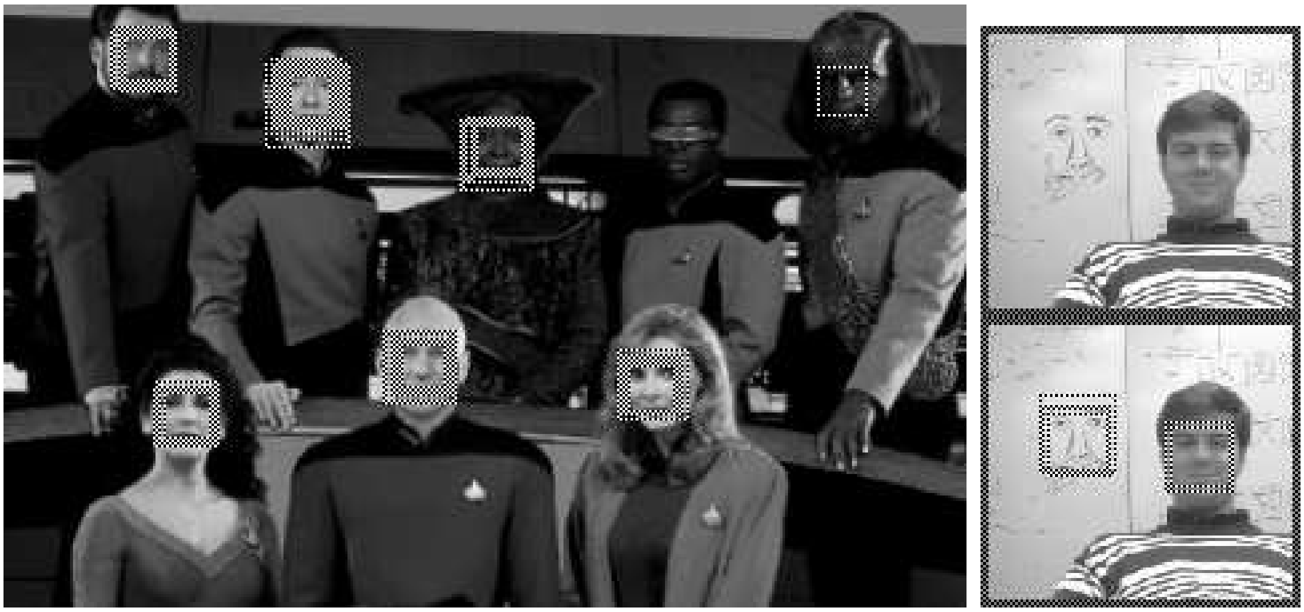

Figure 4: Some face detection results by our system. See text for details.

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.

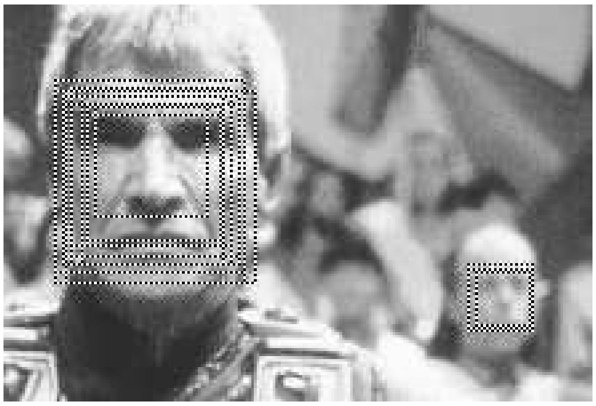

Figure 5: More face detection results by our system. See text for details.

6.2 Analyzing the System's Components

We conducted the following additional experiments to identify the key components of our face detection algorithm. The first experiment investigates how the classifier's architecture affects our system's overall performance. To do this, we create a similar system with a different classifier architecture, and collect relevant statistics by running the system on the two image databases above. Instead of using the original moderately complex *multi-layer perceptron* net as a classifier, our modified system uses the simplest possible network classifier — a single perceptron unit that computes a sigmoidally thresholded weighted sum of the input distances.

The second experiment compares the performance of 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 pat-

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

Table 1: 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.

tern 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 ($\mathcal{M}_n$) within the full image window vector space. To conduct this experiment, we configure and generate statistics for three new systems, each using one of the three distance measures above to compute distances between test patterns and the prototype centers. Notice that because the three new distance measures are all single-value measurements, we also have to modify the classifier architecture accordingly by reducing the number of input terminals from 24 to 12.

Table 1 summarizes the statistics that our two experiments generate. Empirically, the figures suggest that while the classifier's network architecture does not significantly affect the system's performance, our 2-value distance metric noticeably out-performs the other three distance measures in terms of achieving both high detection rates and few false positive errors simultaneously.

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 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 [2].

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

[2] D. Beymer, A. Shashua, and T. Poggio. Example based Image Analysis and Synthesis. A.I. Memo No. 1431, Artificial Intelligence Laboratory, MIT, 1993.

[3] David J. Beymer. Face Recognition under Varying Pose. A.I. Memo No. 1461, Artificial Intelligence Laboratory, MIT, 1993.

[4] M. Bichsel. *Strategies of Robust Objects Recognition for Automatic Identification of Human Faces.* PhD thesis, ETH, Zurich, 1991.

[5] R. Brunelli and T. Poggio. Face Recognition: Features versus Templates. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(10):1042–1052, 1993.

[6] 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.

[7] W. Eric L. Grimson and Tomas 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] 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.

[9] 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.

[10] 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, MIT, 1992.

[11] 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.