Real-time Face Verification

by

Raquel Andrea Romano

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of Master of Science in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

August 1995

© Massachusetts Institute of Technology 1995. All rights reserved.
Real-time Face Verification

by

Raquel Andrea Romano

Submitted to the Department of Electrical Engineering and Computer Science on August 11, 1995, in partial fulfillment of the requirements for the degree of Master of Science in Computer Science and Engineering

Abstract

Automatic face identification is a promising tool for enhancing communication between humans and machines through applications such as automated security systems and personalized interaction with electronic devices. In this thesis, we present a real-time face verification system that uses normalized cross-correlation in conjunction with template-based strategies to quickly and reliably verify the identity of a face in a novel image from a single reference image. We analyze the design trade-offs between our methods and existing approaches in terms of the requirements of practical tasks and test the system on two sets of data which include images acquired interactively by the real-time system in realistic conditions.

Thesis Supervisor: Tomaso Poggio
Title: Uncas and Helen Whitaker Professor of Vision Sciences and Biophysics
Acknowledgments

I would like to thank my research advisor, Tomaso Poggio, for his support and advice. I am especially grateful to Mike Bolotski, Tina Kapur, Jose Luis Robles, and Mike Wessler for reading drafts of the thesis and offering extremely helpful suggestions. Thanks also to David Beymer, Tina Kapur, Lily Lee, and Jose Luis Robles for enlightening discussions, and to David Beymer and Tony Ezzat for sharing portions of their code. Finally, thanks to the members of the AI Lab who have kept me entertained and, of course, to my close friends and family.
## Contents

1 Introduction ........................................ 9
   1.1 The Problem .................................. 10
   1.2 Previous Work ................................ 10
   1.3 Difficulty ..................................... 11
   1.4 System Requirements .......................... 13
   1.5 The Face Verification Algorithm .............. 13
   1.6 Thesis Overview ................................. 14

2 Related Work ........................................ 17
   2.1 Three-dimensional representations ............... 18
   2.2 Sparse two-dimensional representations ........... 18
   2.3 Dense two-dimensional representations ............. 19
       2.3.1 Two approaches ............................ 19
       2.3.2 Detection ................................... 21
       2.3.3 Discrimination ............................. 22
   2.4 Classifiers ...................................... 25
   2.5 Summary .......................................... 26

3 The Face Verification System ....................... 27
   3.1 The Model ...................................... 27
   3.2 Normalization .................................. 29
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.2.1</td>
<td>Hierarchical correlation</td>
<td>31</td>
</tr>
<tr>
<td>3.2.2</td>
<td>Face detection</td>
<td>31</td>
</tr>
<tr>
<td>3.2.3</td>
<td>Eye detection</td>
<td>34</td>
</tr>
<tr>
<td>3.2.4</td>
<td>Refinement</td>
<td>35</td>
</tr>
<tr>
<td>3.2.5</td>
<td>Geometric registration</td>
<td>36</td>
</tr>
<tr>
<td>3.3</td>
<td>Verification</td>
<td>37</td>
</tr>
<tr>
<td>3.3.1</td>
<td>Correlation</td>
<td>37</td>
</tr>
<tr>
<td>3.3.2</td>
<td>Classification</td>
<td>39</td>
</tr>
<tr>
<td>4</td>
<td>Experiments and Results</td>
<td>45</td>
</tr>
<tr>
<td>4.1</td>
<td>Images</td>
<td>46</td>
</tr>
<tr>
<td>4.2</td>
<td>Training</td>
<td>48</td>
</tr>
<tr>
<td>4.3</td>
<td>Testing</td>
<td>50</td>
</tr>
<tr>
<td>4.4</td>
<td>Performance and Reliability</td>
<td>51</td>
</tr>
<tr>
<td>4.5</td>
<td>Working System</td>
<td>52</td>
</tr>
<tr>
<td>5</td>
<td>Conclusions</td>
<td>55</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

The field of computer vision is ripe with ideas ready to be applied to real-world problems. Applications that facilitate human-machine interaction and improve the interface between users and computers are in particularly high demand due to the increasing presence of electronic tools in daily life. Automatic face recognition is a means toward improving the mode of communication between people and machines. It is an especially appropriate solution to the problems of automating security and of customizing the behavior of microprocessor-controlled devices. The underlying task in such applications is to automatically distinguish between images of human faces.

We have built a real-time face verification system that operates in realistic conditions and is directly applicable to the task of automated security. This thesis describes the design and implementation of the system and presents an evaluation of its performance.
1.1 The Problem

The face verification problem is to determine whether the face of a given person appears in a novel image. It is closely related to face recognition, the problem of deciding the identity of a person depicted in a given image. In the former, the proposed identity of the individual is known, and the task of the system is to confirm or deny it. In the latter, the system has no information about the face’s suggested identity, and the objective is to choose the correct name from a list of possible identities. Often, a face recognition algorithm is implemented by performing a verification of the image with each list entry and returning the identity confirmed with most certainty by the verifier. In this context, face verification may be viewed as a special case of the face recognition problem.

Both problems are examples of object recognition, the problem of recognizing an imaged object whose appearance may vary due to the object’s geometric and photometric properties, non-rigid deformations, and the complexity and illumination of the scene. These same challenges confront any general solution to face verification or face recognition.

1.2 Previous Work

Existing object recognition strategies represent an object either with a two-dimensional model based on properties of the object’s two-dimensional image or with a three-dimensional model based on properties of the object’s three-dimensional structure. These approaches can be further classified as sparse representations and dense representations. Two-dimensional sparse representations describe an object according to the spatial relationships among salient characteristics in the image, such as edges or corners. Dense representations retain to some extent the structure and organization of the two-dimensional array of raw
image data depicting the object. The same distinctions can be made between

Our system models a face with a two-dimensional template-based representa-

and compares face models and images using template matching. Template

matching as applied to computer vision is a technique for evaluating the similarity

between a pair of two-dimensional signals by directly comparing the raw data with

a distance metric such as the sum of squared differences or the cross-correlation

between the two arrays of pixel values. In object recognition, the technique may

be used to search for instances of an object by performing comparisons between

a template depicting the object and an image of a scene. Template matching

can be enhanced by modifying either the distance measure or the actual grey-

level image data to normalize the means and standard deviations of the template

and subimages. When used for face recognition, a template often represents a

particular face or facial feature to be localized in a novel image.

The choice of representation often has an immense impact on the ability of

a system to distinguish between face images, since the process of distilling raw

data into a compact model forces implicit assumptions about which information

is essential for discrimination, which information is irrelevant, and what computa-
tions may be performed on the remaining data. Therefore, it makes sense to

allow the assumptions and requirements of the task to guide the choice of repre-
sentation. Chapter 2 discusses existing systems that adopt various face modeling

techniques in the specific context of the face verification task and its assumptions

and requirements.

1.3 Difficulty

The difficulty of distinguishing between images of human faces depends on the

generality of the task to be performed. In a fully unconstrained environment,
faces may be viewed under different illuminations, from varying distances and
directions, in cluttered backgrounds, and with unpredictable facial expressions.
The applications at which our face verification system is aimed may operate under
more constrained conditions since the individual is aware of the authentication
process and is willing to cooperate to some extent by following simple guidelines
that improve the chances of gaining access to the desired facility.

The following assumptions are made about the images presented to the sys-
tem:

- The user provides a consistent facial expression when interacting with the
  system.
- The user presents a frontal face view which may rotate slightly within the
  image plane.
- The background scene may be arbitrarily complex, provided that the face
to be verified is not occluded.
- The scene illumination may vary uniformly among images.

Under these assumptions, the remaining difficulty in achieving high accuracy
is the persistent variability in images of a single human face. Even when the
conditions are intended to be constrained, practical applications demand some
flexibility with respect to head orientation, facial expression, and scene illumi-
nation. Even slight variations of these sorts are capable of producing significant
changes in a person’s appearance.

While different images of one face may vary in appearance, images of dif-
ferent faces are often very similar since human faces share a common spatial
configuration. When recognizing people, humans typically use not only the vi-
ual information confined to the facial area, but the context in which the face
lies, including properties of the individual’s hair and clothing. Since this context
changes, it is not necessarily a suitable source of information for representing faces. The combined effects of similarity across images of different faces and variability across images of the same face pose a difficult challenge for automatic discrimination of faces.

1.4 System Requirements

Before outlining our approach to the problem, we list the basic requirements that should be met by a face verification system intended for practical applications. These demands guide the choices in designing the system. The system must comply with the following requirements:

- Fast, real-time execution.
- Inexpensive hardware.
- Convenience and ease of use.
- Low rate of false entries.
- Flexible security level.

With these in mind, we demonstrate the feasibility of applying correlation based template matching strategies to the problem of real-time face verification using a single image.

1.5 The Face Verification Algorithm

Our face verification system is based on the work of Beymer [1] and Brunelli and Poggio [2]. They compare two face images by applying normalized correlation to pairs of subimages depicting salient facial features. Our algorithm takes the same approach, which can be summarized as follows:
1. Capture a novel image and a proposed identity from the environment.

2. Normalize the face in the image to a standard position, orientation, and scale.

3. Verify the match between the face and its identity by matching templates of facial features.

4. Return either a confirmation or a denial of the face’s identity.

Figure 1-1 sketches the components of the face verification system.

1.6 Thesis Overview

The following chapters present the design, implementation, and testing of the real-time face verification system. Chapter 2 explores alternative methods of representing and comparing face images in the context of a practical solution to face verification. Chapter 3 presents the implementation details of the system. Chapter 4 demonstrates experiments and results of testing the system. Chapter 5 discusses conclusions and future work.
Figure 1-1: Sketch of the face verification system.
Chapter 2

Related Work

The success of a face verification system hinges largely on how well its internal representation captures information necessary to discriminate between face images belonging to different people. Since complex representations are time-consuming to compute, the representation must also be simple enough to be usable in a real-time system. This chapter examines existing approaches to modeling human faces in the context of building a face verification system that satisfies the requirements outlined in Chapter 1. We compare the correlation-based template matching approach of our system in terms of applicability to practical tasks. Many of the techniques for modeling faces are suitable for both feature detection and face comparison. We are interested in both, since feature detection is a useful first step in face image normalization. The following sections survey systems which adopt three-dimensional, two-dimensional, sparse, and dense face representations. In our discussion of dense models, we pay particular attention to comparing correlation strategies with the eigenvector decomposition approach to comparing images.
2.1 Three-dimensional representations

Both Terzopolous and Waters [17] and Essa and Pentland [5] model faces with physical 3D models that incorporate information about facial muscle structure. Though these models are useful for solving the problem of facial expression recognition and potentially for recognition under varying expression, we chose to work with less complex models requiring minimal computation in order to meet the needs of practical face verification applications.

In the domain of recognition, Gordon [7] adopts a 3D approach by modeling a face with depth and curvature features extracted from range data. Models that require special purpose hardware are less practical for our applications than methods that operate on grey-level intensities.

2.2 Sparse two-dimensional representations

Face models built by extracting salient characteristics from an image to build abstract representations may be thought of as sparse representations, as opposed to dense representations which share much of the structure of the intensity data.

Feature-based models locate a collection of facial characteristics in a face image and build a model from the spatial configurations of these feature points [10]. One potential problem with such representations is that inaccuracies in the feature locations may propagate to errors in the geometric measurements. The result may be that the variation among measurements taken from images of a single person is of the same order as the variation among measurements taken from images of different people [2]. Higher accuracy in feature detection could improve the robustness of feature-based approaches.

Parameterized representations are used by Yuille et.al. [19], and Xie et.al. [18], to model facial features as parameterized curves with energy functions that de-
scribe how well the curves match portions of an image. In order to locate facial features, gradient descent techniques find parameters that minimize the energy functions by translating, rotating, and deforming the feature models until they best fit the image. These flexible models and the more general active contour models, or snakes, are useful for locating features in faces with varying expressions.

2.3 Dense two-dimensional representations

Information stored in dense face representations corresponds directly to the image’s grey-level intensity data. The most common approaches of this type are template-based models and eigenvector decompositions. This section contrasts the two strategies for comparing face images, noting both their similarities and their applicability to real-time face verification.

2.3.1 Two approaches

Template matching

Our face verification system uses the template matching approach of Beymer [1] and Brunelli and Poggio [2], in which a facial feature is found in an image by comparing subimages at every point in the image to a stored template of the feature using normalized correlation as a distance measure.

The normalized cross-correlation coefficient between a template $T$ and a sub-image $S$ of identical dimensions is defined as

$$r_n(T, S) = \frac{1}{\sigma_T \sigma_S} \left( \sum_{i=1}^{N} (t_i s_i) - N \mu_T \mu_S \right),$$
where

$$T = \begin{pmatrix} t_1 \\ t_2 \\ \vdots \\ t_N \end{pmatrix} \quad \text{and} \quad S = \begin{pmatrix} s_1 \\ s_2 \\ \vdots \\ s_N \end{pmatrix}$$

are the $N$-pixel template and subimage treated as vectors of grey-level values, $\sigma_t$ and $\sigma_s$ are the respective standard deviations of the template and the subimage, and $\mu_t = \frac{1}{N} \sum_{j=1}^{N} t_j$ and $\mu_s = \frac{1}{N} \sum_{j=1}^{N} s_j$ are their respective means.

The normalized cross-correlation coefficient reduces the influence of ambient scene illumination on the image by effectively adjusting the pixel values to have zero mean and unit variance. This can be seen by normalizing $T$ to $T'$ and $S$ to $S'$ by subtracting the mean grey level from each pixel and dividing by the standard deviation. Then by computing the standard unnormalized cross-correlation coefficient,

$$r_c(T', S') = \sum_{i=1}^{N} t'_i s'_i,$$

where $s'_i = \frac{1}{\sigma_s} (s_j - \mu_s)$ and $t'_i = \frac{1}{\sigma_t} (t_i - \mu_t)$, we find that

$$r_c(T', S') = \sum_{i=1}^{N} \left( \frac{t_i - \frac{1}{N} \sum_{j=1}^{N} t_j}{\sigma_t} \right) \left( \frac{s_i - \frac{1}{N} \sum_{j=1}^{N} s_j}{\sigma_s} \right)$$

$$= \frac{1}{\sigma_t \sigma_s} \sum_{i=1}^{N} \left( t_i s_i - \frac{1}{N} t_i \sum_{j=1}^{N} s_j - \frac{1}{N} s_i \sum_{j=1}^{N} t_j + \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} t_j \sum_{j=1}^{N} s_j \right)$$

$$= \frac{1}{\sigma_t \sigma_s} \left( \sum_{i=1}^{N} t_i s_i - \frac{1}{N} \sum_{i=1}^{N} t_i \sum_{i=1}^{N} s_i \right)$$

$$= r_n(T, S).$$

If we assume $T$ and $S$ have been normalized in this manner, maximizing the cross-correlation $\sum t_i s_i$ is equivalent to minimizing the Euclidean distance $\sum (t_i - s_i)^2$ between two images, since $\sum s_i^2 = 1$. Hereafter, we discuss correlation-based template matching in terms of the Euclidean distance in image space, keeping in
mind its equivalence to cross-correlation. This will facilitate the comparison of
the correlation-based template matching approach and the eigenvector decompo-
sition approach.

**Eigenvector decomposition**

The eigenvector decomposition method adopted by Turk and Pentland [16, 11]
also treats an image of a feature containing \( N \) pixels as an \( N \)-dimensional vector
of grey-level values. Given a set of images depicting example faces or features,
the eigenvector decomposition of a new image is its projection onto the subspace
spanned by the eigenvectors of the image set’s covariance matrix that have the
highest eigenvalues. This subspace is termed the *eigenspace*. According to prin-
cipal component analysis, the \( k \) eigenvectors with highest eigenvalues represent
the \( k \) directions in the \( N \)-dimensional image space along which the set of example
images varies most. The eigenvectors with lowest eigenvalues represent directions
of low variation.

### 2.3.2 Detection

To compare the applicability of template matching and eigenspaces to feature
detection, assume an eigenspace has been constructed from a set of eye templates.
Since eye images do not vary significantly along directions orthogonal to the
eigenspace, the distance between a novel eye image and its projection onto the
eigenspace should be small. Likewise, an image not depicting an eye is likely to
vary along directions orthogonal to the eigenspace, and consequently the distance
between the image and its eigenspace projection should be large. Hence, the
cluster of eye images are approximated by a \( k \)-dimensional subspace of the \( N \)-
dimensional image space.

In contrast, a template matching strategy approximates the set of eye images
with a set of templates, or single points in the image space. The same Euclidean distance measure is used, but rather than measuring the perpendicular distance to a subspace, correlation with a single template measures the isotropic distance to a point. Depending on where the representative points lie in the image space, the template set may be a good or a poor approximation to the cluster of eye images. If only one template is used, the approximation will be weaker than an eigenspace spanned by only one eigenvector, since a single point fits a cluster of points more poorly than a line does. The same comparisons can be made between face detection strategies that use template matching and eigenspaces.

In a face verification system, a feature detector searches specifically for features belonging to the person to be verified. Consequently, in the above example it is only necessary to approximate the eye images belonging to that single individual, rather than the entire set of eye images. If the set of eye images belonging to the individual does not vary significantly, a single template can approximate the image sufficiently well for detection. As a result, template matching is preferable because it avoids the extra computation required for projection onto the eigenspace.

2.3.3 Discrimination

As a strategy for discrimination between images of faces, the eigenspace approach is a low-dimensional approximation to correlation. The theory behind projecting images into the eigenspace is that the directions of highest variation are also the directions of maximum separability between classes of images representing different individuals. If so, discarding the directions of low variation will facilitate discrimination in the eigenspace. However, this is not necessarily the case. It is possible that a direction of high variation is also a direction along which classes overlap, or that a direction of low variation is a direction along which classes
are separable. In either situation, projection onto a lower-dimensional subspace results in a loss of information useful for discrimination [15].

Let us assume that this is not the case and that the separability of points is as good in the eigenspace as in the full image space. Then the only difference between measuring distance in the image space and in the eigenspace is a reduction in dimensionality and a consequent reduction in computation. Examining the difference in computation time required by the two approaches provides a basis for comparing them in the context of the face verification problem.

The recognition algorithm of Turk and Pentland [16] projects an entire database of images onto the eigenspace and computes the mean point in the space for each person, using all available images of the individual. A novel image of a face or feature is compared to an individual in the database by projecting it onto the eigenspace and computing its Euclidean distance to the individual’s mean point in the eigenspace. Thus, it is a low-dimensional approximation to correlation between the novel image and the mean image of the person.

To compare the computational demands of the two methods, consider the comparison of a novel image $X$ to the mean image $M$ of the images of an individual. Let

$$X = \begin{pmatrix} X_1 \\ X_2 \\ \vdots \\ X_N \end{pmatrix} \quad \text{and} \quad M = \begin{pmatrix} M_1 \\ M_2 \\ \vdots \\ M_N \end{pmatrix}$$

be the vector representations of the original grey-level $N$-pixel images. Comparing $X$ and $M$ with the correlation coefficient is equivalent to computing the sum of squared differences,

$$r_c = \sum_{i=1}^{N} (X_i - M_i)^2,$$

in that maximizing one is equivalent to minimizing the other. Whereas computing
\( r_e \) requires \( O(N) \) operations, computing the Euclidean distance,

\[
    r_e = \sum_{j=1}^{k} (x_j - m_j)^2,
\]

in a subspace spanned by \( k < N \) eigenvectors requires only \( O(k) \) operations, where \( x_j = X^T e_j \) and \( m_j = M^T e_j \) are the respective projections of \( X \) and \( M \) onto the \( j \)-th eigenvector \( e_j \). However, computing the \( N \)-dimensional dot product

\[
    x_j = X^T e_j
\]

requires \( O(N) \) operations for each of the \( k \) eigenvectors, so the total computation requirement for a single eigenspace comparison is \( O(k + kN) \).

In a general face recognition system using a library of \( M \) individuals, each of these comparisons would be performed \( M \) times, so generating the correlation coefficients takes \( O(MN) \) time while the eigenspace method requires \( O(kM + kN) = O(kN) \) time if \( M \ll N \). The idea behind eigenvector decomposition is to use a small number of eigenvectors, so that when \( k \ll M \), the technique offers an improvement of a large constant factor.

In the context of face verification, in which only one comparison is required, the computation for correlation remains fixed at \( O(N) \) operations as the number of individuals increases. In contrast, as the set of images increases, more variability is introduced, so the number \( k \) of eigenvectors required to approximate the data well is likely to increase. Therefore, given the complexity \( O(kN) \) of computing the eigenspace distance, the number of operations will increase linearly with \( k \).
2.4 Classifiers

Many of the existing approaches to recognizing human faces involve computing a set of measurements from a given face image and treating these measurements as vectors to be classified into categories corresponding to their identities.

We have seen that Turk and Pentland represent a face image as a projection onto a lower-dimensional subspace and classify it by minimizing and thresholding its distance to the mean vector of each class in the eigenspace. Kurita, et.al. [9], extract second order local autocorrelation features and use closed form linear classifiers to separate the feature vectors into classes. Brunelli and Poggio [2] use the template-based approach adopted by our system and classify an image by maximizing and thresholding the sum of normalized correlation coefficients between the image and the templates. Gilbert and Yang [6] also use correlation coefficients, but they find independent acceptance and rejection thresholds for each coefficient and apply a rule-based scheme for accepting a list of coefficients.

The above works address strategies for solving both multi-class pattern recognition problems and two-class pattern recognition problems of the kind required by a verification system. The difference is that in the multi-class case, a given vector represents a single face image and must be classified as belonging to one of many possible people, while in the two-class case, the vector obtained in the verification system describes not a single face image, but the similarity between a face image and a single face model. Both the binary classifiers described here and the one used in our system may be extended to multi-class strategies by applying the input image to every library image and choosing the one entry accepted with highest confidence.
2.5 Summary

In this chapter we have investigated the design trade-offs of various approaches to modeling faces in the context of the requirements of the real-time face verification task described in Chapter 1. Three-dimensional representations seem unnecessarily complex for the problem and may require expensive hardware. Sparse two-dimensional representations have not proven to be reliable enough for high-precision verification. Among the dense two-dimensional representations, we compare the two most promising alternatives: template-based correlation and eigenvector decomposition. The analysis shows that for verification, correlation has lower computational requirements since increasing the size of the database may incur additional costs for eigenspace approaches but does not necessarily affect a template-based approach. In addition, template matching may well be sufficient for person-dependent feature detection because the need to approximate the entire database of feature images is replaced by the need to approximate a less variable cluster of images belonging to a single individual.
Chapter 3

The Face Verification System

This chapter presents in detail the implementation of the face verification system. The system expects as input a novel image with a proposed identity and returns confirmation or denial of the face's identity. The first section describes the internal face model stored for each individual in the system’s library. The second section describes the normalization stage of the system, and the third section describes the verification stage.

3.1 The Model

This section presents the model used by the system to represent faces. The system stores a library of one face model per person. Under the assumption that the face of a given person will have roughly the same pose and expression in all input images, only a single reference image is required to build a face model. When a new user is submitted to the system, a single frame of the person’s face is captured and six points demarcating prominent facial features, the left and right eye centers, the left and right nose lobes, and the left and right mouth corners, are manually labeled, as shown in Figure 3-1. The system uses the left and right
eye centers to automatically normalize the new library entry to ensure that every model is scaled, rotated, and translated to a standard width, orientation, and position. The six feature points also guide the automatic extraction of subimages of each eye, the nose, the mouth, and the whole face, as shown in Figure 3-2. Though it would be possible to use a feature detector to automate the labeling of models, it is crucial that these labels be accurate in order to build a model that is representative of the class of images of a given person’s face.

The normalization and verification components use slightly different variations of the same template-based representation. The normalization stage uses only two types of templates: eye templates and whole face templates. The hierarchical search strategy outlined in the next section requires storage of several
low-resolution versions of these templates, sampled at \( \frac{1}{4} \), \( \frac{1}{16} \), and \( \frac{1}{64} \) of the original sampling density. Each template is also stored at three scales in order to compensate for the variable size of the face in the image. Figure 3-3 illustrates the organization of the template set.

The verification component relies on only one resolution level, but requires four templates depicting both eyes, the nose, and the mouth. In addition to the set of grey level templates, each model includes templates extracted from the result of filtering the reference image with the following differential operators: the horizontal and vertical components of discrete approximations of the image gradient, the magnitude of the gradient image, and a discrete approximation of the Laplacian of the image. Figure 3-4 displays the full template set referenced by the verification stage.

The total storage requirement of the full template set used by both the normalization and verification stages is less than 40 kilobytes.

### 3.2 Normalization

This section explains the system’s method for normalizing images to achieve invariance with respect to the distance, location, and rotation of the head relative
The two-dimensional transformations between a novel image and a normalized model may be computed using correspondences between feature points. We use the two points found by the eye detector to perform translation, rotation, and scaling of a new image to a standard position, orientation, and size.

The normalization process is composed of four stages: the face detection stage, the eye detection stage, the refinement stage, and the geometric registration stage. We adopt a coarse-to-fine search strategy to use the first stage's estimate for the face location to guide the second stage's search for rough eye locations.

---

### Table: Grey level and Gradient Components

<table>
<thead>
<tr>
<th>Grey level</th>
<th>Horizontal gradient component</th>
<th>Vertical gradient component</th>
<th>Gradient magnitude</th>
<th>Laplacian</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Image]</td>
<td>[Image]</td>
<td>[Image]</td>
<td>[Image]</td>
<td>[Image]</td>
</tr>
</tbody>
</table>

Figure 3-4: Organization of the template model referred to at the verification stage.

to the camera. Recall that the novel image must satisfy the size, pose, and illumination assumptions outlined in Section 1.3 of Chapter 1. In addition, the number of scales at which templates are stored determines the range of distances from the camera that the normalization can handle. This range may be set to handle more or fewer distances falling within a smaller or larger range by building more templates at varying scales into the library models. Similarly, the normalization procedure handles only slight image plane rotations, but the limit may be increased by adding sets of rotated templates to the model.
The final stage then uses the optical flow between these eye location estimates and stored eye templates to perform fine adjustments to the eye positions. The following section discusses the strategy of hierarchical correlation that guides the eye detector.

### 3.2.1 Hierarchical correlation

Burt [3] suggests the use of multiresolution image pyramids in conjunction with coarse-to-fine search strategies to perform object recognition tasks in a computationally efficient manner. One can construct a Gaussian image pyramid from an original image by smoothing the image with a Gaussian kernel and then subsampling the image by a factor of two. Repeated application of the smoothing and subsampling steps creates a sequence of images decreasing in resolution. A coarse-to-fine algorithm for finding features begins by searching for coarse features in the lowest resolution representation of the image to obtain approximate results that are then used to limit the search for finer features at higher resolutions. The low sampling density of the coarse search reduces the time required to find less detailed features, and the restriction of the search space to limited regions at higher resolutions reduces the computation necessary to detect more detailed features. Because of its efficiency, we use this search strategy in our algorithm for locating first the face position and then the eye positions. The next four sections describe the four stages of the normalization process.

### 3.2.2 Face detection

The first step of the face detection process is to build the coarsest level of the image pyramid by performing a sequence of reductions of the input image. For each reduction, the image is convolved with a low pass filter that discretely approximates a Gaussian kernel and then subsampled by a factor of two by eliminating
Figure 3-5: An input image reduced by a factor of $2^6$ for comparison with low resolution face templates at the standard template size and at scales of 0.8 and 1.2 relative to the standard template size.

alternating rows and columns. As indicated in Figure 3-3, the face model of the person to be verified contains three low resolution face templates constructed in the same way, except that the face model has previously been normalized to have a standard interocular distance and scaled slightly to compensate for variation in camera distance. Figure 3-5 displays a low resolution input image and the model's three scaled low resolution face templates. As we shall see, processing at reduced resolutions significantly lowers the computational demands of computing correlation coefficients. Consequently, increasing the number of scales in a model and hence the number of templates to be compared with the image does not incur too much additional computational cost.

Each of the face templates is compared to a subimage centered at every pixel in the low resolution image using the normalized cross-correlation coefficient defined in Chapter 2. Exhaustive search returns a correlation coefficient corresponding to each pixel in the low resolution input image. Whereas an exhaustive search on the original input image of size $N$ requires $N$ computations of correlation coefficients, the search at level $k$ of the image pyramid requires the computation
of only $\frac{N}{2^k}$ coefficients, a significant reduction.

For each scaled face template, a map of correlation coefficients indicates how well each image location matches the face template. The map attaining the highest correlation coefficient indicates the scale that most closely approximates the face size in the input image. The other two sets of face candidates are discarded, and future processing uses only the templates at the chosen scale. Pixels that reach local maxima in the correlation map for the chosen scale are thresholded and returned as potential face candidates.

Hence, the face detector does not necessarily return a single face location, but may select several image locations, of which one should mark the center of the face. For each possible face center, there exist regions above the face center that are likely to contain the eyes. These regions are delimited and passed to the eye finder to restrict the search area to be explored at higher resolutions.

Although there exist sophisticated face detection techniques that more reliably detect faces of varying sizes in complex scenes [14, 13, 11], the simplicity of our face detector lends itself well to face verification applications because the face to be verified is likely to be either the only face or the most salient face in the image. This expectation combined with the fact that the face template used for correlation is taken from an image of the person being identified improves the likelihood that the face template will correlate most highly with the correct face center. The face detector is more prone to failure when a user claims a false identity because the face template is taken from an image of a different person and therefore may correlate poorly with the user's face. In this case, since the regions passed to the eye finder do not contain the actual eye locations, the eye finder fails and the intruder is automatically rejected. The face detector thus acts as an initial filter to immediately discard faces that bear little holistic resemblance to the model's reference image.
3.2.3 Eye detection

The eye detection component processes the input image at a finer resolution level than that of the face detection component since the eye is a more detailed feature than the face and requires more precise localization. The eye search uses only the eye templates at the scale chosen by the face finder as most closely approximating the scale of the face in the input image. For each eye, the eye template at the current resolution level is correlated with the image only at those points lying within the left or right eye regions selected by the face finder. Together, the left and right eye regions span only about $\frac{1}{4}$ of the low resolution image, so computation is reduced by limiting search to the regions. For each eye, correlation at pixels in the selected regions returns a map of correlation coefficients. The system thresholds the pixels at the local maxima of each map, and returns the remaining pixels as candidate eye locations to be considered at the next finest resolution level. The finder may be set to repeat this procedure at an arbitrary number of pyramid levels, each time using the points with highest correlation coefficients to define smaller regions delimiting the search for the eyes at a finer resolution level. At each successive level, pixels not centered at eye locations will return lower results because at higher resolutions the correlation coefficient is more sensitive to detail. Consequently, false eye locations will be eliminated and the highest remaining correlation score for each eye should correspond to the correct eye location.

In our system, the face detector operates on images at a resolution of $\frac{1}{64}$ the original sampling density, and the eye detector operates on images at $\frac{1}{16}$ resolution. In order to save computation time, the system stops at this level and returns the pixel with the highest correlation coefficient for each eye. Since the estimated locations are chosen at a low resolution, transforming the coordinates to the corresponding locations in the full resolution image may place the eye
positions several pixels away from the actual eye centers. The refinement stage described in the next section uses optical flow to perform fine adjustments to these locations.

3.2.4 Refinement

At the refinement step, it is assumed that the eye detector’s current estimates of eye locations lie within a neighborhood of several pixels from the true eye centers. To calculate the displacement of an eye center estimate from its actual location, we compute the optical flow field between the stored eye template and the subimage of identical dimensions centered at the current estimate. The optical flow algorithm determines the apparent motion of brightness patterns between two images [8]. This method requires that the estimated eye locations lie close to the actual ones, since the flow field relies heavily on local changes in brightness. If the current locations are not within several pixels of the actual positions, the refinement step will not improve the estimates, and the finder will fail to locate the eyes and therefore reject the user. This happens in a number of situations, most of them involving suboptimal imaging conditions such as varying expression, including closed eyes, varying illumination, out of plane head rotation, the presence of eye glasses, and extreme distance or proximity of the face to the camera. The finder may also fail when the user gives a false identity, in which case the system automatically rejects the user, as desired.

If the eye-finding stage returns estimates that are are close to correct, the refinement step recovers the direction and magnitude of the displacement of each current eye estimate with respect to its correct location. Each eye’s position is then adjusted by the flow vector at its estimated center. The flow computation and adjustment of eye labels may be iterated in order to further improve precision.

After refinement, the eye labels must pass a final confidence test before they
are used for normalization. The relative positions of the left and right eye estimates are evaluated using knowledge of expected configurations of eyes on the human face. The distance between a pair of candidates must lie between an upper and lower bound determined by the range of scales at which the system is operating. If the distance between two eye location candidates is too great, the pair of estimates is assumed to be erroneous and the finder fails to locate the eyes. The angle of the head's tilt is also bounded, and if the pair of eye locations violates this bound, the finder likewise fails. Although the bound on this angle may be varied, if the angle is allowed to be too large, the eye templates may not match well with the eyes in the image since the templates are taken from frontal upright views. Storing rotated templates as well as scaled templates would solve this problem with an added cost of the correlation computation required for each additional eye template.

3.2.5 Geometric registration

Once the eye locations have been found, the system brings the face into registration with the model face by performing a two-dimensional rigid transformation and uniform scaling using the eye centers as reference points. These two points determine the rotation of the face with respect to the camera in the plane parallel to the image plane. Rotation of the image by this angle fixes the line through the eye centers, called the interocular axis, at a horizontal orientation. The distance between the eye centers, called the interocular distance, reflects the distance between the camera and the face. Scaling the image to fix the interocular distance to the standardized interocular distance in the model brings the face to the standard template size. Once the image is rotated and scaled, the eye positions guide the extraction of a subimage containing only the face. Extraction of the face subimage effectively translates the face to a known position, thus completing the
normalization of the face's position, size, and orientation. The normalized image is now geometrically registered with the normalized model image, such as the one shown in Figure 3-2.

The precision of the eye finder directly determines the appearance of the face in the normalized image that is used in the verification stage. Inaccurate eye locations will result in poor scaling, rotation, and extraction of the face. The results from the verifier's template matching will then suffer because of improper registration between the input and the model, rather than because of inherent differences in the actual faces being imaged. Therefore, it is critical that the eye detector returns eye positions precise enough for accurate normalization.

3.3 Verification

This section describes the method for comparing a user's image to the model of the person to be verified. The verification stage receives a normalized image from the eye detection stage, computes the similarity between the image and the model, classifies the list similarity measures as a good match or a poor match, and makes the final decision to accept or reject the individual.

3.3.1 Correlation

After normalization, the eye locations in the input image have been spatially registered with the eye locations in the reference image. Under the assumption that the expressions in the two images are similar, the other facial features should also be well-registered when these two reference points are aligned. The positions of the four feature templates in the model guide the extraction of subimages around the corresponding features in the normalized input image. Each feature template of the model is correlated with the corresponding subimage of the input
image using the normalized cross-correlation form presented in Chapter 2.

For each feature, the template is correlated with each of several subimages shifted by several pixels horizontally and vertically in order to compensate for small variations in the feature locations in the input image. The shifted subimage that best encloses the feature is most likely to match well with the template, so the maximum correlation coefficient over all shifted subimages is returned as the measure of similarity between a template and its corresponding feature in the input image.

The fact that the eye positions in the input image are determined by the finder ensures that the eye subimages are accurately positioned at the centers of the eyes. However, since we have no explicit pixel coordinates for the nose and mouth corners, the subimages defined by their model templates may or may not entirely enclose the input image's actual nose and mouth. In fact, if the user enters a false identity, it is quite possible that the spatial relationships between the user's facial features will be significantly different from those of the reference face. In this case, the automatically defined subimages are likely to be misaligned to the extent that the horizontal and vertical shifts are not sufficient to find a subimage that contains the full feature. As a result, the maximum correlation coefficient over all shifted subimages will be low, and the match between model and image will be poor. Poor similarity measures are cause for rejection, precisely as desired when the individual's identity is false. Hence, the extraction of subimages not only defines the image regions to be correlated with model templates, but enforces spatial constraints on the positions of features in the input.

Each comparison between template and subimage returns a correlation coefficient. If there are $m$ templates of facial features and $n$ types of filters, a total of $mn$ correlation coefficients form a vector representing the similarity between the input image and model. Figure 3-7 shows the subimage comparison between the
Figure 3-6: Input and model subimage comparisons for each of 4 templates (left eye, right eye, nose, mouth) and 5 preprocessing types (grey level, horizontal gradient component, vertical gradient component, gradient magnitude, and Laplacian operator).

Figure 3-7: input and model for each of four templates and five preprocessing types. From this 20-dimensional vector, the system must decide whether the similarity scores indicate a match or a mismatch between the user and the suggested identity and accept or reject the user accordingly. The next section discusses strategies for classifying the vector of correlation coefficients as an acceptance or a rejection.

### 3.3.2 Classification

In the last stage of the verification system, the problem reduces to a pattern classification problem. Each list of $d$ correlation coefficients is a point $x = (x_1, \ldots, x_d)$ in $\mathbb{R}^d$, where $x_i$ lies in the interval $[-1, 1]$. A vector of correlation scores comparing a face model and an input image from the same user is called a positive example, and a vector of scores comparing faces of different people is called a negative example. Given a correlation vector returned by the correlation stage, the classifier must make a binary decision: acceptance if the correlation scores
indicate a match, and rejection if the scores suggest a false entry. Geometrically, an ideal classifier defines a surface in \( \mathbb{R}^d \) that separates the positive examples from the negative examples. Figure 3-8 shows a set of points representing 20-dimensional correlation vectors projected from \( \mathbb{R}^{20} \) onto \( \mathbb{R}^3 \).

The choice of classifier is guided heavily by a desire to minimize the rate of false acceptances since the primary applications of face verification demand careful control of security. In addition, it is desirable to allow the system’s security level to vary according to the application. In general, the higher the security, the fewer the false acceptances but the higher the number of retries requested of authentic users. Conversely, the lower the security, the fewer the false rejections but the more successful entries by users with false identities. Chapter 4 presents experiments with training and testing the classifier on sets of images and comparing it to a thresholded sum classifier.

Our system’s classifier is a variation of the standard nearest mean classifier.
The nearest mean procedure estimates the mean vector for each class \( C_i \) and classifies a new correlation vector \( x \in \mathbb{R}^d \) according to which class mean \( m_i \) is nearest:

\[
C = \arg \min_{C_i} d(m_i, x)
\]

In our case, there are only two classes, a positive class and a negative class, so if \( m_p \) and \( m_n \) are the positive and negative class means respectively, then the classifier’s decision rule is as follows:

If \( d(m_p, x) > d(m_n, x) \), accept.
If \( d(m_p, x) < d(m_n, x) \), reject.

When \( d \) is the Euclidean metric, the decision boundary, \( d(m_p, x) = d(m_n, x) \), is linear and the decision rule is equivalent to thresholding a weighted sum of the correlation vector components. To see this, square the distances and simplify:

\[
0 = d^2(m_p, x) - d^2(m_n, x) \\
= (m_p - x)^t(m_p - x) - (m_n - x)^t(m_n - x) \\
= m_p^t m_p - m_n^t m_n - 2(m_p - m_n)^t x.
\]

Consequently,

\[
(m_p - m_n)^t x = \frac{1}{2} (m_p^t m_p - m_n^t m_n).
\]

By letting \( w = m_p - m_n \) and \( T = \frac{1}{2} (m_p^t m_p - m_n^t m_n) \), we can write the classifier as a linear decision rule that thresholds a weighted sum of the correlation vector components:
If $w^T x > T$, accept.
If $w^T x < T$, reject.

The decision boundary is therefore a hyperplane in $\mathbb{R}^d$.

The sample means $m_p$ and $m_n$ clearly affect the weights and thresholds that determine where the decision boundary lies in $\mathbb{R}^d$. Ideally, the input and model are highly correlated for positive examples and poorly correlated for negative examples. Consequently, the mean positive vector of normalized correlation coefficients should lie close to the point $e = (1, 1, \ldots, 1)^T$, and the mean negative vector should lie far from $e$. However, in practical applications, image variability due to slight changes in facial expression, head orientation, and scene illumination may cause certain correlation coefficients between corresponding subimages of faces belonging to the same person to fall well below 1, making discrimination between low-scoring positives and high-scoring negatives difficult. It is therefore desirable to determine the classifier’s parameters during a training stage that takes into account the distributions of positive and negative examples for a given image database.

The negative examples with low coefficients are easy to identify as mismatches, but those with higher coefficients are difficult to distinguish from positive examples. Therefore, we concentrate our training strategy on those negative examples with correlation scores high enough to be potentially mistaken for positive examples. We call such negative examples near misses, and define them to be the set of negative examples falling within a fixed spherical neighborhood around the positive mean. Only the near misses are considered in the computation of the negative class mean.

\footnote{In the case that the mean correlation coefficients along each dimension are equal, that is, if $m_p = k_p e^T$ and $m_n = k_n e^T$, where $k_p$ and $k_n$ are the mean correlation coefficients for positive and negative examples and $e = (1, 1, \ldots, 1)^T$, the classifier reduces to a simple thresholded sum without weights since $(m_p - m_n)^T x = (k_p - k_n)e^T x = T'$ implies that $e^T x = \sum_{i=1}^d x_i = T'$, where $T' = \frac{T}{k_p - k_n}$.}
The choice of neighborhood radius determines the set of near misses and therefore the negative mean. The system’s training stage allows the radius of the neighborhood defining the near misses to vary and examines the effect of the varying negative mean on the accuracy of the classifier by applying it to a set of testing images. For a positive and negative mean, the classifier’s performance on the testing set may be evaluated with a scoring function that takes into consideration the specific requirements of the application. For example, enforcing a low false positive rate will tighten the security of a system while focusing on a low false negative rate will diminish its tendency to reject authentic entries. In Chapter 4, we illustrate these tradeoffs by examining the ROC function of several data sets.

The training stage estimates the class means by running the verifier on a set of known input/model pairs and recording the resulting correlation vectors. Given this training set of positive and negative examples, the training algorithm proceeds as follows:

1. Initialize the radius $r$ of the neighborhood around the positive mean to the smallest possible value for which the set of enclosed negative examples is non-empty.

2. Let $\mathbf{m}_p$ be the mean of the positive training examples.

3. Let $\mathbf{m}_n$ be the mean of all negative examples within a distance $r$ from $\mathbf{m}_p$.

4. Apply the nearest mean decision rule to every example in the testing set and record the number of false positives and false negatives.

5. Evaluate the error rates with a scoring function $S$ that takes into consideration the system’s requirements.

6. If $S$ is greater than any score seen so far, record $r$ as the best radius.
7. Increment $r$.

8. Repeat steps 3 through 7 until all negative examples have been included in the neighborhood of near misses.

The choice the radius $r$ that defines the set of near misses will determine the value of $m_n$, which in turn dictates the weights $w$ and the threshold $T$ of the classifier. The linear search for the ideal radius is therefore a method for estimating the weights and threshold of a linear classifier and is comparable to using a gradient descent procedure to find the weights and threshold that optimize a criterion function. The difference is that instead of allowing the weights $w$ to vary freely, they are constrained by the values of the positive and negative means.

After training is complete, the system applies the linear decision rule to the new correlation vector returned by the correlation stage and accepts or rejects the input image according to the classifier’s decision.
Chapter 4

Experiments and Results

This chapter describes the experiments that test the performance and reliability of the verification system.

Results gathered from the testing of face recognition systems depend critically on the specific constraints placed on the images used for testing. Even when such constraints are explicitly declared, they are often subject to interpretation, so that different image sets allegedly satisfying the same set of constraints may have noticeably different properties. For example, image sets of supposedly expressionless faces may in actuality allow some flexibility in expression, but the extent of variability may vary widely among image sets. Similarly, images captured from cluttered scenes may have differences in appearance significant enough to affect a system’s ability to segment the face from the background. The implications of such disparities are that it is not meaningful to compare error rates of recognition systems tested on different data sets. For this reason, we test our face verification system both on a publicly available image set and on a set of images acquired interactively by the real-time verification system.
4.1 Images

The first set of images used for testing belongs to the University of Essex face database [12]. Each 180 x 200 pixel JPEG compressed 24 bit color image has been decompressed and converted to an 8 bit grey scale image. Images of faces with glasses have been eliminated because the specularities introduced by the lenses often saturate the image resulting in a loss of information that impedes recognition. The remaining image set contains 20 images per person for 81 people, totaling 1620 images of frontal views under constant illumination in uncluttered background scenes. Though the verification system is intended for faces with fixed expressions, the Essex image set contains images with expressions that vary. Figure 4-1 shows an example set of images for a single person in the Essex database.

The second set of images used for testing was taken at the MIT Artificial Intelligence Laboratory by the real-time verification system. The system runs on an SGI Indy workstation and captures images with the IndyCam digital color video camera bundled with the workstation as a standard peripheral device. For each image, the view is frontal, the eyes are open, visible, and free of glasses, and the expression is fixed across all images of a given individual. The images are partitioned into two classes depending on the quality of scene illumination in a typical office environment. In one class the scene is illuminated by overhead lighting and in the other class by ambient sunlight from a window. Since the differences in lighting conditions created by the two types of scene illumination are extreme, only images from the same illumination class are compared. The set consists of 158 360 x 240 pixel images of 48 people, and the number of images per person ranges from 1 to 8. Figure 4-2 shows several example images from the AI Lab data set.
Figure 4-1: Example set of images from University of Essex database.
4.2 Training

For each data set, the images are divided into non-overlapping sets of training and testing images. The correlation vectors used for training are computed from images drawn exclusively from the training set, and the correlation vectors used for testing are computed from images drawn only from the testing set.

As described in Chapter 3, the primary purpose of training is to compute the positive and negative means needed to perform the nearest mean classification. At the same time that the training stage computes the positive and negative class means, it can tune the system to provide a particular level of reliability for a given database. Just as varying the threshold of a linear classifier effects a change in the decision boundary, so does varying the neighborhood from which negative examples are drawn to compute the negative class mean.

A larger set of near misses generally pulls the negative mean away from the positive class mean which increases the system’s leniency, since with a mean vector with low-valued components, the verifier is more likely to misclassify a high-scoring negative example. A smaller set of near misses pulls the negative
class mean closer to the positive examples and increases the system's level of security, since with a mean vector with high-valued components, the verifier is more likely to misclassify a low-scoring positive example. In addition, variation in the negative class mean causes the effective weight vector to change, thus allowing the orientation of the separating hyperplane to vary freely. However, given a training set in which most of the examples fall close to the line in the direction \( \mathbf{e} = (1, 1, \ldots, 1)^T \), the weight vector is expected to be close to \( \mathbf{e} \).

To evaluate the success rate of the system and examine the trade-off between leniency and security, we use the Receiver Operating Characteristic methodology. The ROC function compares a binary decision strategy’s ability to accept true positive examples with its ability to reject true negative examples. For example, a classifier that blindly accepts all examples has a 100% success rate on positive examples but a 0% success rate on negative examples. Ideally, a system classifies all examples correctly, that is, it simultaneously achieves 100% success rates for both positive and negative examples. When this is not the case, it is beneficial to observe how varying a parameter or a threshold affects the relative error rates of positive and negative examples.

The ROC curve is simply a plot of the fraction of positive examples that are correctly classified against the fraction of negative examples that are correctly classified [4]. The closer the curve lies to the left and upper boundaries of the plot, the more reliable the classifier is, since the true positive rate does not suffer when the true negative rate increases, and vice versa. Each point on the curve represents a certain parameter setting for which the classifier achieved the corresponding true positive and negative rates. ROC data allows such a parameter to be tuned to achieve a desired balance between the true positive and true negative success rates.
4.3 Testing

We make the assumption that only a limited number of images may be available for training and therefore only use one image per person to compute the positive and negative class means. For each set of images belonging to a person in the Essex database, one image is a reference image for building the face model, one image is placed in the training set, and the remaining 18 images are placed in the testing set. The training set therefore consists of 81 positive examples, one for each person, and $80 \times 81 = 6480$ negative examples. The testing set contains $18 \times 81 = 1458$ positive examples and $18 \times 80 \times 81 = 116,640$ negative examples. Figure 4-3 plots the ROC curve for the Essex set to illustrate how varying the set of negative examples used in computing the negative mean affects the recognition rate.

The varying parameter for the nearest mean classifier is the radius of the neighborhood around the positive mean from which near misses are drawn to compute the negative mean. The varying parameter for the thresholded unweighted sum of features is simply the value of the threshold. To obtain a low false positive rate of less than 0.5%, the maximum attainable true positive rate is 94.86%, meaning that authentic users may have to retry verification 5% of the time. This false negative error rate is due to the variable expressions in the Essex database, which keep the true positive rate from approaching 100% as quickly as the true negative rate does. Therefore only users that vary their facial expressions are likely to have to retry, and a higher true positive rate can be expected from a data set with images that satisfy the fixed expression requirement.

To test the AI Lab image set, we again use one image per person for building the model, one image for computing the class means, and the remaining images for testing. The data set contains 35 positive training examples, 445 negative testing
4.4 Performance and Reliability

To evaluate the system's overall reliability, we maximize the verification rate, defined as the total percentage of correctly classified examples, both positive and negative. Table 4.1 displays the maximum verification rates attained by the two

---

1When only one image per person is available, the model is built from that image and no positive examples are available for that person.
classifiers on both image sets. Since the total number of examples is dominated by negative examples, this accuracy rate is skewed in favor of the true negative rate.

4.5 Working System

The working system developed at the Artificial Intelligence Laboratory has been built into a screen lock utility to demonstrate the feasibility of an example application. The real-time system meets the requirements outlined in Chapter 1 as follows:

- **Speed:** The average execution time of the real-time verification system alone is 2.7 seconds on an Indy workstation with a MIPS R4400 processor. The entire system with a built-in user interface that displays video and image windows runs in an average of 5 seconds.

- **Cost:** the system may be installed on any SGI Indy workstation with video input from the IndyCam or alternate video source.
<table>
<thead>
<tr>
<th>Classifier</th>
<th>AI Lab</th>
<th>Essex</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nearest Mean</td>
<td>99.5%</td>
<td>99.85%</td>
</tr>
<tr>
<td>Thresholded Sum</td>
<td>99.5%</td>
<td>99.80%</td>
</tr>
</tbody>
</table>

Table 4.1: Comparison of verification accuracy rates between two classifiers on both image sets.

- **Convenience**: The system provides easy entrance of a new person into the model library in several seconds. After manual labeling of six points, the templates of the face model are immediately and automatically built and stored, requiring no user assistance.

- **Reliability**: The system yields an estimated false entry rate of less than .5%

- **Flexibility**: The system may be tuned to be more tolerant or less tolerant depending on the security level demanded by the application.
Chapter 5

Conclusions

In this thesis we have shown how by exploiting the person-dependent nature of the face verification problem, it is possible to use fast, simple template-based correlation strategies to perform real-time face verification. The real-time system presented in this thesis uses a normalized correlation-based template matching strategy to reliably verify the identity of a face in a novel image from a single model view.

Future improvements include continually updating the library model of a person by saving a copy of each input image taken while the system is running. This is a simple way of gathering multiple images across an extended period of time without inconveniencing the user. When enough images are available, the variation among images of a person can be estimated and the corresponding template model can be improved by re-centering it to the mean of the distribution. If the images presented to the system consistently meet the criteria outlined in Chapter 1, namely fixed expression, frontal views, and uniform illumination changes, then the variation in images of a person may be small enough that a single-template approximation is a sufficient representation. In addition, we intend to perform further testing to determine to what extent the system can handle more
extreme changes in lighting, a largely unsolved issue that challenges most existing face recognition systems.

We have analyzed our method for fast normalization and verification with respect to alternative approaches to modeling and recognizing faces. This discussion has guided our choices in designing the system in terms of the primary requirements of high speed, low cost, convenience, reliability, and flexibility.

The system has been incorporated into a screen locking application to demonstrate its suitability for human-machine interaction tasks. The working system demonstrates the feasibility of integrating a real-time face verification system into usable, practical applications.
Bibliography


[12] University of Essex.


