Neural Network-Based Face Detection

Henry A. Rowley

May 1999

CMU-CS-99-117

School of Computer Science
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:

Takeo Kanade, Carnegie Mellon, Chair
Manuela Veloso, Carnegie Mellon
Shumeet Baluja, Lycos Inc.
Tomaso Poggio, MIT AI Lab
Dean Pomerleau, AssistWare Technology

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy.

This research is sponsored by the Hewlett-Packard Corporation, the Siemens Corporate Research, the National Science Foundation, the Army Research Office under Grant No. DAAH04-94-G-0006, and the Office of Naval Research under Grant No. N00014-95-1-0591. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies or endorsements, either expressed or implied, of the sponsors or the U.S. government.

Keywords: Face detection, Pattern recognition, Computer vision, Artificial neural networks, Machine learning, Pattern classification, Multilayer perceptrons, Statistical classification

for Huang Ning

Abstract

Object detection is a fundamental problem in computer vision. For such applications as image indexing, simply knowing the presence or absence of an object is useful. Detection of faces, in particular, is a critical part of face recognition and, and critical for systems which interact with users visually.

Techniques for addressing the object detection problem include those matching a two- and three-dimensional geometric models to images, and those using a collection of two-dimensional images of the object for matching. This dissertation will show that the latter view-based approach can be effectively implemented using artificial neural networks, allowing the detection of upright, tilted, and non-frontal faces in cluttered images. In developing a view-based object detector using machine learning, three main subproblems arise. First, images of objects such as faces vary considerably with lighting, occlusion, pose, facial expression, and identity. When possible, the detection algorithm should explicitly compensate for these sources of variation, leaving as little as possible unmodelled variation to be learned. Second, one or more neural networks must be trained to deal with all remaining variation in distinguishing objects from non-objects. Third, the outputs from multiple detectors must be combined into a single decision about the presence of an object.

This thesis introduces some solutions to these subproblems for the face detection domain. A neural network first estimates the orientation of any potential face. The image is then rotated to an upright orientation and preprocessed to improve contrast, reducing its variability. Next, the image is fed to a frontal, half profile, or full profile face detection network. Supervised training of these networks requires examples of faces and nonfaces. Face examples are generated by automatically aligning labelled face images to one another. Nonfaces are collected by an active learning algorithm, which adds false detections into the training set as training progresses. Arbitration between multiple networks and heuristics, such as the fact that faces rarely overlap in images, improve the accuracy. Use of fast candidate face selection, skin color detection, and change detection allows the upright and tilted detectors to run fast enough for interactive demonstrations, at the cost of slightly lower detection rates.

The system has been evaluated on several large sets of grayscale test images, which contain faces of different orientations against cluttered backgrounds. On their respective test sets, the upright frontal detector finds 86.0% of 507 faces, the tilted frontal detector finds 85.7% of 223 faces, and the non-frontal detector finds 56.2% of 96 faces. The differing detection rates reflect the relative difficulty of these problems. Comparisons with several other state-of-the-art upright frontal face detection systems will be presented, showing that our system has comparable accuracy. The system has been used successfully in the Informedia video indexing and retrieval system, the Minerva robotic museum tour-guide, the WebSeer image search engine for the WWW, and the Magic Morphin' Mirror interactive video system.

Acknowledgements

I arrived at CMU in the fall of 1992, nearly seven years ago. I had a lot to learn: how to find my way around a new school and a new city, and how to do computer science research. Fortunately, my family, friends and colleagues made this easy, teaching me about life and research in too many ways to list them all. Let me just mention a few examples.

Thanks to my advisor, Takeo Kanade, for teaching me about science and computer vision; and to my committee members for their encouragement and advice. Thanks especially to Shumeet Baluja, my coauthor for much of the work presented in Chapters 3 and 4, for teaching me about neural networks and providing constant amusement.

Thanks to my family: my parents, for asking "Have you found a thesis topic yet?" until I could answer "Yes"; and my brother Tim, for constant teasing over the last seven years.

Thanks to my officemates over the years: Manuela Veloso, for showing me that professors can be friends too; Xue-Mei Wang, for introducing me to yoga; Puneet Kumar, for delicious Indian dinners; Alicia Pérez, for showing compassion in every action; James Thomas, for the margaritas; Bwolen Yang, for characterizing this thesis as "How to automatically shoot people in the head"; Yirng-An Chen, for teasing me about my international phone bills; and Sanjit Seshia, for the reminding me of the enthusiasm I had as a first year student. Thanks also to my friends and virtual officemates: Claudson Bornstein, for his apple pies and his unique driving technique; Chi Wong, for many conversations; Tammy Carter, for Star Trek and pizza; and Eugene Fink, for mathematical puzzles. Thanks to my colleagues in the Computer Science Department, the Robotics Institute, and the Electrical and Computer Engineering Department, who taught me so much.

And finally, thanks to my wife Huang Ning. We were married last year just before I began writing this document. She is studying in Japan, and now that this document is complete, I can finally join her there. Her constant love, support, and encouragement made finishing this impossible task possible.

Henry A. Rowley
May 6, 1999

Contents

List of Figures and Tables

Chapter 1

Introduction

The goal of my thesis is to show that the face detection problem can be solved efficiently and accurately using a view-based approach implemented with artificial neural networks. Specifically, I will demonstrate how to detect upright, tilted, and non-frontal faces in cluttered grayscale images, using multiple neural networks whose outputs are arbitrated to give the final output.

Object detection is an important and fundamental problem in computer vision, and there have been many attempts to address it. The techniques which have been applied can be broadly classified into one of two approaches: matching two- or three-dimensional geometric models to images [Seutens *et al.*, 1992, Chin and Dyer, 1986, Besl and Jain, 1985], or matching view-specific image-based models to images. Previous work has shown that view-based methods can effectively detect upright frontal faces and eyes in cluttered backgrounds [Sung, 1996, Vaillant *et al.*, 1994, Burel and Carel, 1994]. This thesis implements the view-based approach to object using neural networks, and evaluates this approach in the face detection domain.

In developing a view-based object detector that uses machine learning, three main subproblems arise. First, images of objects such as faces vary considerably, depending on lighting, occlusion, pose, facial expression, and identity. The detection algorithm should explicitly deal with as many of these sources of variation as possible, leaving little unmodelled variation to be learned. Second, one or more neural-networks must be trained to deal with all remaining variation in distinguishing objects from non-objects. Third, the outputs from multiple detectors must be combined into a single decision about the presence of an object.

The problems of object detection and object recognition are closely related. An object recognition system can be built out of a set of object detectors, each of which detects one object of interest. Similarly, an object detector can be built out of an object recognition system; this object recognizer would either need to be able to distinguish the desired object from all other objects that might appear in its context, or have an *unknown object* class. Thus the two problems are in a sense identical, although in practice most object recognition systems are rarely tuned to deal with arbitrary backgrounds, and object detection systems are rarely trained on a sufficient variety of objects to build an interesting recognition system. The different focuses of these problems lead to different representations and algorithms.

Often, face recognition systems work by first applying a face detector to locate the face, then applying a separate recognition algorithm to identify the face. Other object recognition system sometimes use the hypothesize and verify technique, in which they first generate a hypothesis of which object is present (recognition), then use a more precise algorithm to verify whether that object is actually present (detection).

The work in this thesis concentrates on the face detection problem, but incorporates recognition techniques to deal with the changes in the pose of the face, using the hypothesize and verify technique.

1.1 Challenges in Face Detection

Object detection is the problem of determining whether or not a sub-window of an image belongs to the set of images of an object of interest. Thus, anything that increases the complexity of the decision boundary for the set of images of the object will increase the difficulty of the problem, and possibly increase the number of errors the detector will make.

Suppose we want to detect faces that are tilted in the image plane, in addition to upright faces. Adding tilted faces into the set of images we want to detect increases the set's variability, and may increase the complexity of the boundary of the set. Such complexity makes the detection problem harder. Note that it is possible that adding new images to the set of images of the object will make the decision boundary becomes simpler and easier to learn. One way to imagine this happening is that the decision boundary is smoothed by adding more images into the set. However, the conservative assumption is that increasing the variability of the set will make the decision boundary more complex, and thus make the detection problem harder.

There are many sources of variability in the object detection problem, and specifically in the problem of face detection. These sources are outlined below.

Variation in the Image Plane: The simplest type of variability of images of a face can be expressed independently of the face itself, by rotating, translating, scaling, and mirroring its image. Also included in this category are changes in the overall brightness and contrast of the image, and occlusion by other objects. Examples of such variations are shown in Figure 1.1.

Pose Variation: Some aspects of the pose of a face are included in image plane variations, such as rotation and translation. Rotations of the face that are not in the image plane can have a larger impact on its appearance. Another source of variation is the distance of the face

Figure 1.1: Examples of how face images between poses and between different individuals.

from the camera, changes in which can result in perspective distortion. Examples of such variations are shown in Figure 1.1.

Lighting and Texture Variation: Up to now, I have described variations due to the position and orientation of the object with respect to the camera. Now we come to variation caused by the object and its environment, specifically the object's surface properties and the light sources. Changes in the light source in particular can radically change a face's appearance. Examples of such variations are shown in Figure 1.2.

Figure 1.2: Examples of how images of faces change under extreme lighting conditions.

Background Variation: In his thesis, Sung suggested that with current pattern recognition techniques, the view-based approach to object detection is only applicable for objects that have "highly predictable image boundaries" [Sung, 1996]. When an object has a predictable shape, it is possible to extract a window which contains only pixels within the object, and to ignore the background. However, for profile faces, the border of the face itself is the most important feature, and its shape varies from person to person. Thus the boundary is not predictable, so the background cannot be simply masked off and ignored. A variety of different backgrounds can be seen in the example images of Figures 1.1 and 1.2.

Shape Variation: A final source of variation is the shape of the object itself. For faces, this type of variation includes facial expressions, whether the mouth and eyes are open or closed, and the shape of the individual's face, as shown in some of the examples of Figure 1.1.

The next section will describe my approach to the face detection problem, and show how each of the above sources of variation can be addressed.

1.2 A View-Based Approach using Neural Networks

The face detection systems in this thesis work are based on four main steps:

1. **Localization and Pose Estimation:** Use of a machine learning approach, specifically an artificial neural network, requires training examples. To reduce the amount of variability in the positive training images, they are aligned with one another to minimize the variation in the positions of various facial features.

   At runtime, we do not know the precise facial feature locations, and so we cannot use them to locate potential face candidates. Instead, we use exhaustive search over location and scale to find all candidate locations. Improvements over this exhaustive search will be described that yield faster algorithms, at the expense of a 10% to 30% penalty in the detection rates.

   It is at this stage rotations of the face, both in- and out-of-plane, are handled. A neural network analyzes the potential face region, and determines the pose of the face. This allows the face to be rotated to an upright position (in-plane) and selects the appropriate detector network for the particular out-of-plane orientation.

2. **Preprocessing:** To further reduce variation caused by lighting or camera differences, the images are preprocessed with standard algorithms such as histogram equalization to improve the overall brightness and contrast in the images. I also examine the possibility of lighting compensation algorithms that use knowledge of the structure of faces to perform lighting correction.

3. **Detection:** The potential faces which are already normalized in position, pose, and lighting in the first two steps are examined to determine whether they are really faces. This decision

is made by neural networks trained with many face and nonface example images. This stage handles all sources of variation in face images not accounted for the in the previous two steps. Separate networks are trained for frontal, partial profile, and full profile faces.

4. **Arbitration:** In addition to using three separate detector, one for each class of poses of the face, multiple networks are also used within each pose. Each network learns different things from the training data, and makes different mistakes. Their decisions can be combined using some simple heuristics, resulting in reinforcement of correct face detections and suppression of false alarms. The thesis will present results for using one, two, and three networks within an individual pose.

Together these steps attempt to account for the sources of variability described in the previous section. These steps are illustrated schematically by Figure 1.3.

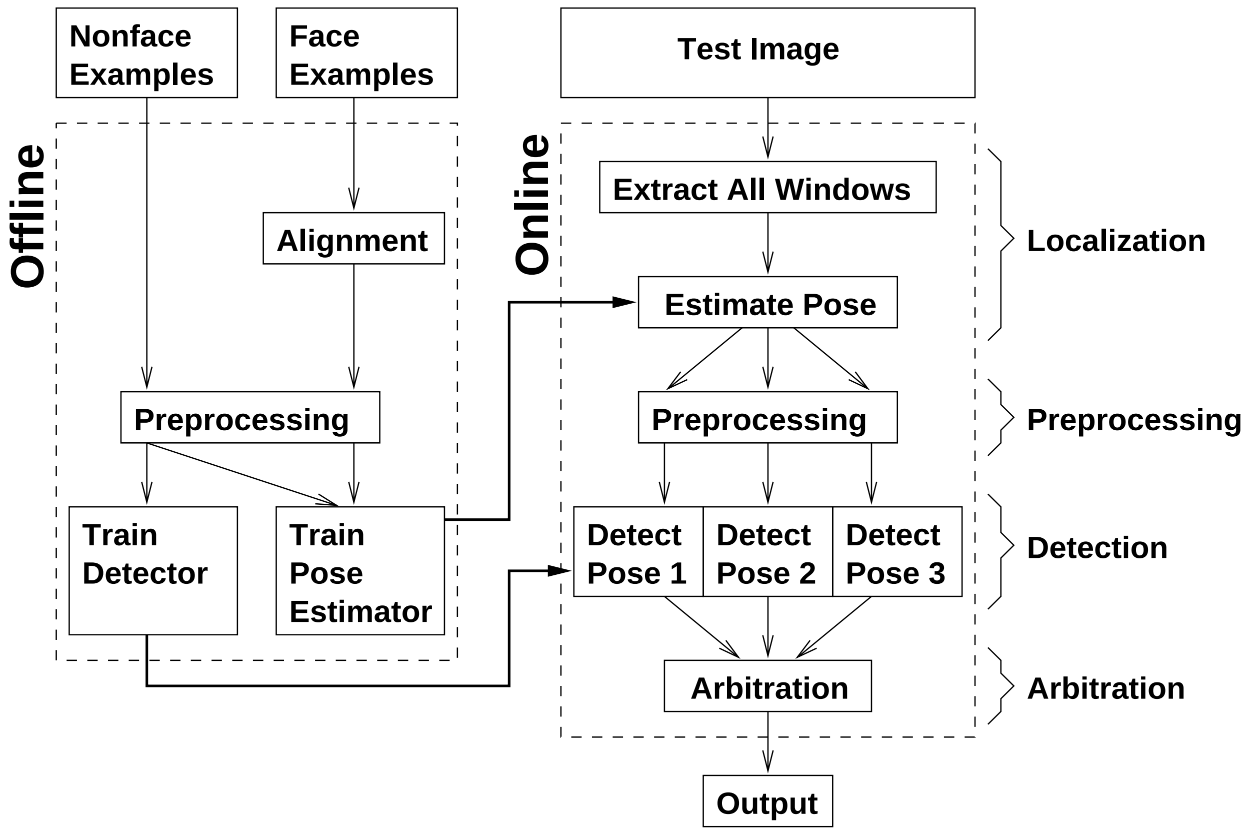


Figure 1.3: Schematic diagram of the main steps of the face detection algorithms developed in this thesis.

1.3 Evaluation

This thesis provides a rigorous analysis of the accuracy of the algorithms developed. A number of test sets were used, with images collected from a variety of sources, including the World Wide Web, scanned photographs and newspaper clippings, and digitized video images.

Each test set is designed to test one aspect of the algorithm, including the ability to detect faces in cluttered backgrounds, the ability to detect a wide variety of faces of different people, and the detection of faces of different poses. An overview of the results is given in Table 1.4. We will see that the upright detector is able to detect 86.0% of faces on a test set containing mostly upright faces, while the tilted face detector has comparable detection rates. Both of these systems have fairly low false alarm rates. The detection rate for the non-frontal detector is significantly lower, reflecting the relative difficulty of these problems. Comparisons with several other state-of-the-art face upright frontal detection systems will be presented, showing that our system has comparable performance in terms of detection and false-positive rates in this simpler domain. Although there are a few other detectors designed to handle tilted or non-frontal faces, they have not been evaluated on large public datasets, so performance comparisons are not possible.

Table 1.4: Overview of the results from the systems described in this thesis.

System	Test Set	Detection Rate	False Alarms
Upright Detector (Chapter 3)	*Upright Test Set* (130 images, 507 upright faces)	86.0%	31
Tilted Detector (Chapter 4)	*Tilted Test Set* (50 images, 223 frontal faces)	85.7%	15
Non-Frontal Detector (Chapter 5)	*Non-Frontal Test Set* (53 images, 96 faces)	56.2%	118

The test sets which contain publically available images have been placed on the World Wide Web at `http://www.cs.cmu.edu/~har/faces.html` as references for the development and evaluation of future face detection techniques.

1.4 Outline of the Dissertation

The remainder of the dissertation is organized as follows.

Chapter 2 will discuss some basic methods for normalizing images of potential faces before they are passed to a detector. These techniques include simple image processing operations, such as histogram equalization and linear brightness normalization, as well some knowledge-based methods for correcting the overall lighting of a face. An important section of this chapter describes how to align example face images with one another; this method, along with the preprocessing algorithms, is used throughout the rest of the dissertation.

Chapter 3 describes the first face detection system of the thesis, which is limited to upright, frontal faces. The system uses two neural networks trained on example faces and nonfaces. To simplify training, the training algorithm for these networks selects nonface examples from images of scenery, instead of using a hand-picked set of representative nonfaces. The outputs from the networks are arbitrated using some simple heuristics to produce the final results. The system is evaluated over several large test sets.

Chapter 4 presents some extensions to this algorithm for the detection of tilted faces, that is faces which are rotated in the image plane. The main change needed is in the input normalization stage of the algorithm, where not only is the contrast normalized, but also the orientation. This is accomplished by a neural network which estimates the orientation of potential faces, allowing them to be rotated to an upright orientation. The resulting system is evaluated over the same test sets as the upright detector, as well as a new test set specifically for tilted faces.

Chapter 5 further extends the face detection domain to include non-frontal faces. One important subproblem is aligning faces in three dimensions, whereas the previous chapters only need two-dimensional alignment. Also, the detection problem is distributed among several view-specific networks, rather than lumping the entire set of face examples into a single class. Although the results are not as accurate as those of the upright and tilted face detectors, they are promising and may be good enough for applications requiring the detection of non-frontal faces.

Chapter 6 examines some techniques for speeding up the face detection algorithms. These include using a fast but inaccurate candidate selection network along with fast skin color and motion detection algorithms to prune out uninteresting portions of the image.

Chapter 7 describes some applications in which the upright frontal face detector described in Chapter 3 (incorporating the speed-up techniques from Chapter 6 has been used by other researchers, ranging image and video indexing systems to systems that interact with people.

Chapter 8 describes related work in the face and object detection domains, and presents comparisons of the accuracy of the algorithms when they have been applied to the same test sets.

Chapter 9 summarizes the contributions of the thesis and points out directions for future work.

Chapter 2

Data Preparation

2.1 Introduction

This thesis will utilize a view-based approach to face detection, and will use a statistical model (an artificial neural network) to represent each view. A view-based face detector must determine whether or not a given sub-window of an image belongs to the set of images of faces. Variability in the images of the face may increase the complexity of the decision boundary to distinguish faces from nonfaces, making the detection task more difficult. This section presents techniques to reduce the amount of variability in face images.

Section 2.2 begins with a brief description of the training images that were collected for this work. Section 2.3 describes how faces are aligned with one another; this removes variation in the two dimensional position, scale, and orientation of the face. It also gives a way to specify the location at which the detector should find a face in a test image. Section 2.4 describes how to separate the foreground face from the background in a set of images called the FERET database [Phillips *et al.*, 1996, Phillips *et al.*, 1997, Phillips *et al.*, 1998], which we used for training and testing various parts of our system. Section 2.5 describes how to preprocess the images to remove some differences in facial appearance due to poor lighting or contrast.

The techniques presented in this chapter are quite general. Later chapters describing specific face detectors which use these tools will require small changes for the particular application in that chapter.

2.2 Training Face Images

Before describing how the training images are processed, I will first list their sources. They come from three large databases:

CMU Face Files Many students, faculty, and staff in the School of Computer Science at Carnegie Mellon University have digital images online. These images were acquired using a standard camcorder, either recording onto video tape for later digitization, or digitizing directly from the camera. Face images from the Vision and Autonomous Systems Center were obtained from this WWW page: `http://www.ius.cs.cmu.edu/cgi-bin/vface`. Face images from the Computer Science department are typically available from personal homepages: `http://www.cs.cmu.edu/scs/directory/index.html`. A few examples are shown in Figure 2.1. Since this time, better quality face images for the department have been made available at `http://sulfuric.graphics.cs.cmu.edu/~photos/`.

Figure 2.1: Example CMU face files.

Harvard Images Dr. Woodward Yang at Harvard University provided a set of over 400 mug-shot images which are part of the training set. These are high quality gray-scale images with a resolution of approximately 640 by 480 pixels, originally collected for face recognition research. Parts of this database were used as the "high-quality" test images in [Sung, 1996]. The images are similar to the ones shown in Figure 2.2.

Figure 2.2: Images representative of the size and quality of the images in the Harvard mugshot database (the actual images cannot be shown here for privacy reasons).

Picons Face Files Another set of face images was collected from the Picons database available on the WWW at `http://www.cs.indiana.edu/picons/ftp/index.html`. This database consists of small (48×48 pixel) images with 16 gray levels or colors, collected at Usenix conferences. A few examples are shown in Figure 2.3. The database has grown significantly in size since I made a local copy for training; it now contains several thousand images which may be appropriate for training.

Figure 2.3: Example Picons face files.

2.3 Facial Feature Labelling and Alignment

The first step in reducing the amount of variation between images of faces is to align the faces with one another. This alignment should reduce the variation in the two-dimensional position, orientation, and scale of the faces. Ideally, the alignment would be computed directly from the images, using image registration techniques. This would give the most compact space of images of faces. However, the image intensities of faces can differ quite dramatically, which would make some faces hard to align with each other, but we want every face to be aligned with every other face.

The solution used for this thesis is manual labelling of the face examples. For each face, a number of feature points are labelled, depending on the three-dimensional pose of the head, as listed in Figure 2.4.

The next step is to use this information to align the faces with one another. First, we must define what is meant by alignment between two sets of feature points. We define it as the rotation, scaling, and translation which minimizes the sum of squared distances between pairs of corresponding features. In two dimensions, such a coordinate transformation can be written in the following form:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} s\cos\theta & -s\sin\theta \\ s\sin\theta & s\cos\theta \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \end{pmatrix} = \begin{pmatrix} a & -b & t_x \\ b & a & t_y \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

Figure 2.4: Features points manually labelled on the face, depending on the three-dimensional pose of the face. The left profile views are mirrors of the right profiles.

If we have several corresponding sets of coordinates, this can be further rewritten as follows:

$$\begin{pmatrix} x_1 & -y_1 & 1 & 0 \\ y_1 & x_1 & 0 & 1 \\ x_2 & -y_2 & 1 & 0 \\ y_2 & x_2 & 0 & 1 \\ & \vdots & & \end{pmatrix} \cdot \begin{pmatrix} a \\ b \\ t_x \\ t_y \end{pmatrix} = \begin{pmatrix} x'_1 \\ y'_1 \\ x'_2 \\ y'_2 \\ \vdots \end{pmatrix}$$

When there are two or more pairs of distinct feature points, this system of linear equations can be solved by the pseudo-inverse method. Renaming the matrix on the left hand side as A, the vector of variables $(a, b, t_x, t_y)^T$ as $\mathbf{T}$, and the right hand side as $\mathbf{B}$, the pseudo-inverse solution to these equations is:

$$\mathbf{T} = (A^T A)^{-1}(A^T \mathbf{B})$$

The pseudo-inverse solution yields the transformation $\mathbf{T}$ which minimizes the sum of squared differences between the sets of coordinates x'_i, y'_i and the transformed versions of x_i, y_i, which was our goal initially.

Now that we have seen how to align two sets of labelled feature points, we can move on to aligning sets of feature points. The procedure is described in Figure 2.5.

Empirically, this algorithm converges within five iterations, yielding for each face the transformation which maps it to close to a standard position, and aligned with all the other faces. Once the

1. Initialize $\bar{\mathbf{F}}$, a vector which will be the average positions of each labelled feature over all the faces, with some initial feature locations. In the case of aligning frontal faces, these features might be the desired positions of the two eyes in the input window. For faces of another pose, these positions might be derived from a 3D model of an average head.

2. For each face i, use the alignment procedure to compute the best rotation, translation, and scaling to align the face's features $\mathbf{F}_i$ with the average feature locations $\bar{F}$. Call the aligned feature locations $\mathbf{F}'_i$.

3. Update $\bar{\mathbf{F}}$ by averaging the aligned feature locations $\mathbf{F}'_i$ for each face i.

4. The feature coordinates in $\bar{\mathbf{F}}$ are rotated, translated, and scaled (using the alignment procedure described earlier) to best match some standardized coordinates. These standard coordinates are the ones used as initial values used for $\bar{\mathbf{F}}$.

5. Go to step 2.

Figure 2.5: Algorithm for aligning manually labelled face images.

parameters needed to align the faces are known, the image can be resampled using bilinear interpolation to produce an cropped and aligned image. The averages and distributions of the feature locations for frontal faces are shown in Figure 2.6, and examples of images that have been aligned using this technique are shown in Figure 2.7.

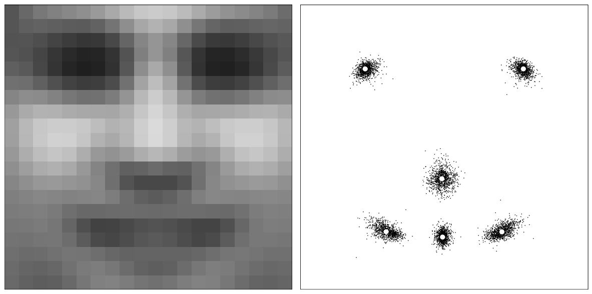


Figure 2.6: Left: Average of upright face examples. Right: Positions of average facial feature locations (white circles), and the distribution of the actual feature locations (after alignment) from all the examples (black dots).

In training a detector, obtaining a sufficient number of examples is an important problem. One commonly used technique is that of virtual views, in which new example images are created from real images. In my work, this has taken the form of randomly rotating, translating, and scaling

Figure 2.7: Example upright frontal face images aligned to one another.

example images by small amounts. [Pomerleau, 1992] made extensive use of this approach in training a neural network for autonomous driving. The network learns from watching an experienced driver staying on the road, but needs examples of what to do should the vehicle start to leave the road. Examples for this latter case are generated synthetically.

Figure 2.8: Example upright frontal face images, randomly mirrored, rotated, translated, and scaled by small amounts.

Once the faces are aligned to have a known size, position, and orientation, the amount of variation in the training data can be controlled. The detector to be made more or less robust to particular variations in a desired degree. Some example images in which random amounts rotation (up to $10°$), random translations of up to half a pixel, and random scalings up to 10% are shown in Figure 2.8.

2.4 Background Segmentation

To train the non-frontal face detector, this thesis work used the FERET image database. One characteristic of this database is that the images have fairly uniform backgrounds. Because the detector itself will be trained with regions of the image which include the background, we need to make sure that the detector does not learn to simply look for the uniform background. For this purpose, we must segment the background, so that it can be replaced with a more realistic (and less easily detectable) background. Much of the complexity of what follows is due to the fact that the original images are grayscale. When color information is available, standard "blue screening" techniques can separate the foreground and background in a more straightforward manner [Smith, 1996].

Since the backgrounds of the images tend to be bright but not entirely uniform, I modelled the backgrounds as varying linearly across the image. Each background pixel's value is assumed to have a Gaussian distribution, with a mean centered on the model intensity, and a fixed standard deviation across the background. Formally, the intensity I of a background pixel (x, y) is:

$$\begin{aligned} I(x,y) &= a \cdot x + b \cdot y + c + N(0,\sigma) \\ &= N(a \cdot x + b \cdot y + c, \sigma) \end{aligned}$$

where $N(0, \sigma)$ is a Gaussian distributed noise with mean zero and standard deviation σ. An alternative representation for the background is to treat the affine function as the mean of the Gaussian distribution. The background model is adapted using the Expectation-Maximization (EM) procedure.

The initial model for the background is uniform across the image (so a and b are both 0), with the intensity c set by the top half of the pixels in the left-most and/or right-most columns of the image, as shown in Figure 2.9. The standard deviation of these intensities is also measured, and used as the σ for the Gaussian distribution. The σ value will be held constant, while the remaining parameters a, b, c will be adjusted to match the image.

Given this initial model, we can iterate the following two steps:

1. **Expectation Step (E-Step):** Label each pixel (x, y) with a probability of belonging to the background. We assume that a pixel's intensity $I(x, y)$ is distributed according to a Gaussian distribution, with mean $a \cdot x + b \cdot y + c$ and variance σ specified by the initial background

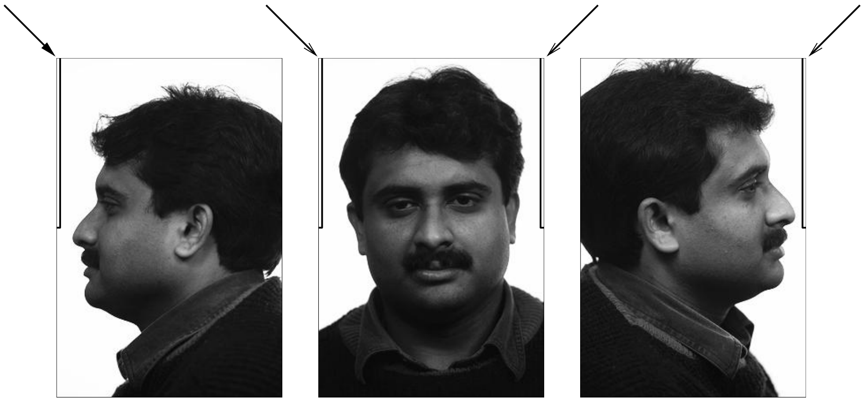


Figure 2.9: The portions of the image used to initialize the background model depend on the pose of the face, because occasionally the back of the head covers some useful pixels.

model, so the probability that it was generated by the model is:

$$P(I(x,y)|\text{background}) \propto e^{-\left((I(x,y)-(a\cdot x+b\cdot y+c))^2/\sigma^2\right)}$$

Since the foreground cannot be estimated, I set the probability of generating a pixel arbitrarily to: $P(I(x,y)|\text{foreground}) \propto e^{-1}$. We can combine these two with Bayes' formula to get the following formula for the probability of the background model explaining a particular observed intensity:

$$P(\text{background}|I(x,y)) = \frac{P(I(x,y)|\text{background})}{P(I(x,y)|\text{background}) + P(I(x,y)|\text{foreground})}$$

2. **Maximization Step (M-Step):** We then compute an updated version of the background model parameters, using the probabilities computed in the E-Step as weights for the contribution of each pixel. This is done by solving the following over-constrained linear system for all pixels x, y:

$$P(\text{background}|I(x,y)) \cdot I(x,y) = P(\text{background}|I(x,y)) \cdot \begin{pmatrix} x & y & 1 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \end{pmatrix}$$

where $P(\text{background}|I(x,y))$ is the probability that pixel x, y belongs to the background. This set of equations can be solved for the model parameters a, b, c by the pseudo-inverse method.

We run 15 iterations of the above algorithm, although empirically it usually converges in fewer iterations. Some examples of the intermediate output for one image are shown in Figure 2.10.

To find the final segmentation, the probability map $P(\text{background}|I(x,y), x, y)$ is thresholded at 0.5. A connected components algorithm is applied, and any background colored components

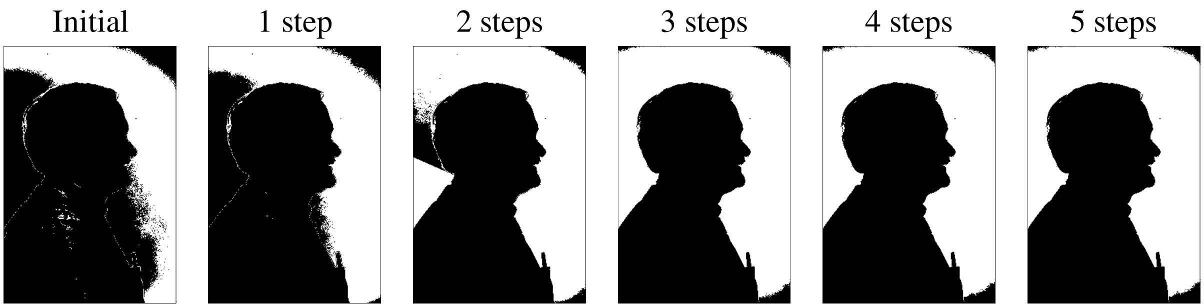


Figure 2.10: The segmentation of the background as the EM-based algorithm progresses. The images show the probability that a pixel belongs to the background, where white is zero probability, and black is one. Note that this is a particularly difficult case for the algorithm; usually the initial background is quite accurate, and the result converges immediately.

which touch the border of the image are merged into a single component. This allows for things like strands of hair which might otherwise separate the background components. Applying the resulting mask (like that in Figure 2.11a) to the original image gives a result like the one shown in Figure 2.11b.

A bright white border is visible around the face. These pixels are actually a mixture of intensities from the foreground and background, and thus their intensities are no longer close enough to the background to be classified as such. One solution to this is to apply a median filter to the inside border of the masked region, using sample intensities from a small neighborhood around the pixel. The result is shown in Figure 2.11c.

Although the E-M segmentation algorithm worked well in the example of Figure 2.11, it sometimes fails when the background is non-uniform, or when the person's skin or clothing is close in intensity to the background. In such cases, the initial model usually gives better results. Since the segmentation is used to produce training data, complete automation is not necessary. Thus rather than trying to make the algorithm work perfectly, I examined the segmentation results for each image using the initial and final background models, and selected the one with the best segmentation.

Once the background mask is computed, we can replace the background with something more realistic. I developed four random background generators:

1. Constant background, with an intensity selected uniformly from the range 0 to 255.

2. Static noise background. Each pixel will have an intensity of either 0 or 255. The probability of pixel being 255 is first picked randomly (from 0% to 100%), and then the intensity of each pixel is chosen randomly according to that probability.

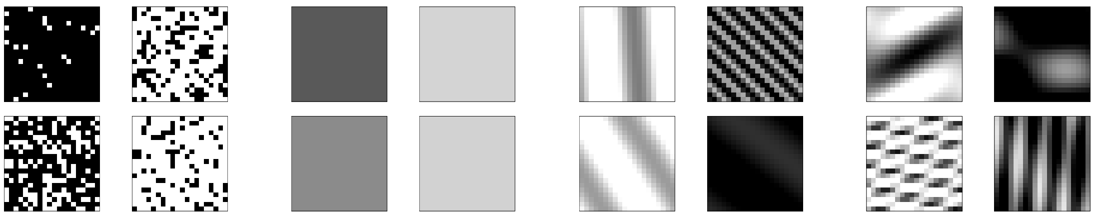
3. Sinusoidal background, with a random mean (between 0 and 255), amplitude (between 0 and

(a) (b) (c)

Figure 2.11: a) The background mask generated by the EM-based algorithm. b) The masked image, in which the masked area is replaced by black. Note the bright border around the face. c) Removing the bright border using a 5×5 median filter on pixels near the border of the mask.

128), orientation ($0°$ to $180°$), initial phase, and wavelength. The wavelength was chosen to be at least one pixel, in the scale of the face image to be generated, usually 20×20 or 30×30 pixels, and less than the window size. The intensity values were clipped to the range 0 to 255.

4. Two sinusoids added like those described above added together.



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

Figure 2.12: Examples of randomly generated backgrounds: (a) constant, (b) static noise, (c) single sinusoid, (d) two sinusoids.

These background generators are illustrated in Figure 2.12. For each face example to be generated, one of these four generators was chosen at random. I do not make any particular claims about the realism of the backgrounds generated by this process, but rather that they are more varied than the uniform images present in the FERET database. I considered using backgrounds extracted from real images which contain no faces, but decided against it for this work. Choosing an appropriate subset of the backgrounds is difficult, because there are so many, and randomly changing these backgrounds at runtime to give a complete sample would be computationally expensive.

2.5 Preprocessing for Brightness and Contrast

After aligning the faces and replacing the background pixels with more realistic values, there is one remaining major source of variation (apart from intrinsic differences between faces). This variation is caused by lighting and camera characteristics, which can result in brightly or poorly lit images, or images with poor contrast.

We first address these problems by using a simple image processing approach, which was also used in [Sung, 1996]. This preprocessing technique first attempts to equalize the intensity values in across the window. We fit a function which varies linearly across the window to the intensity values in an oval region inside the window (shown in Figure 2.13a). Pixels outside the oval may represent the background, so those intensity values are ignored in computing the lighting variation across the face. If the intensity of a pixel x, y is $I(x, y)$, then we want to fit this linear model parameterized by a, b, c to the image:

$$\begin{pmatrix} x & y & 1 \end{pmatrix} \cdot \begin{pmatrix} a \\ b \\ c \end{pmatrix} = I(x, y)$$

The choice of this particular model is somewhat arbitrary. It is useful to be able to represent brightness differences across the image, so a non-constant model is useful. The variation is limited to linear to keep the number of parameters low and allow them to be fit quickly. Collecting together the contributions for all the pixels in the oval window gives an over-constrained matrix equation, which is solved by the pseudo-inverse method, like the one used to model the background. This linear function will approximate the overall brightness of each part of the window, and can be subtracted from the window to compensate for a variety of lighting conditions.

Next, histogram equalization is performed, which non-linearly maps the intensity values to expand the range of intensities in the window. The histogram is computed for pixels inside an oval region in the window. This compensates for differences in camera input gains, as well as improving contrast in some cases. Some sample results from each of the preprocessing steps are shown in Figure 2.13. The algorithm for this step is as follows. We first compute the intensity histogram of the window, where each intensity level is given its own bin. This histogram is then converted to

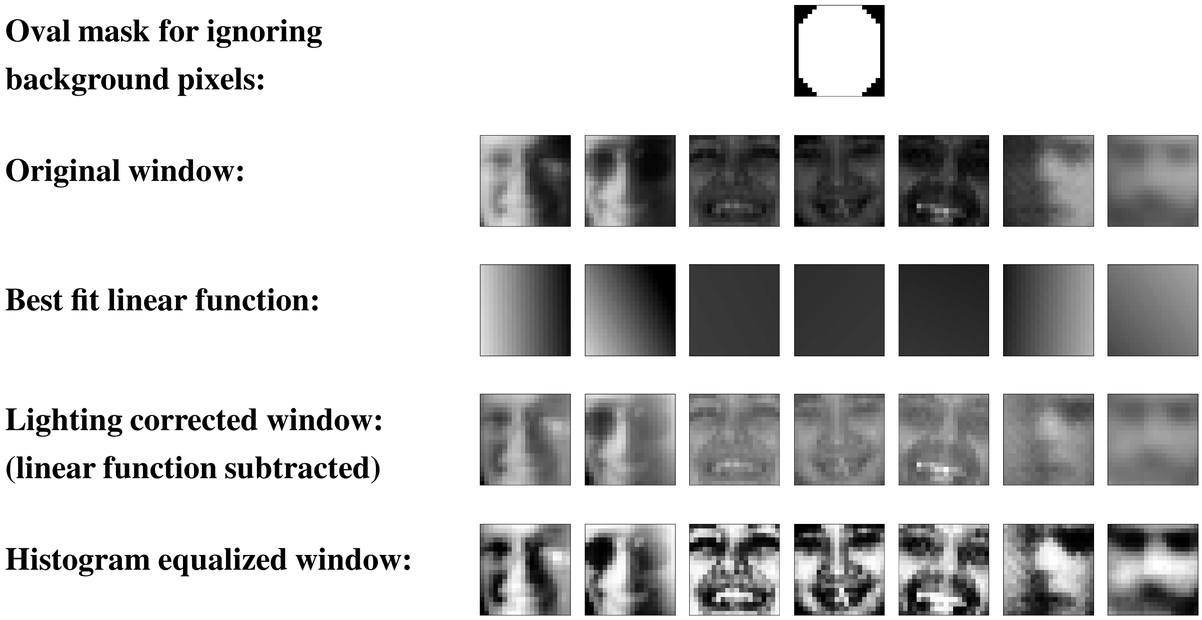


Figure 2.13: The steps in preprocessing a window. First, a linear function is fit to the intensity values in the window, and then subtracted out, correcting for some extreme lighting conditions. Then, histogram equalization is applied, to correct for different camera gains and to improve contrast. For each of these steps, the mapping is computed based on pixels inside the oval mask, and then applied to the entire window.

a cumulative histogram, in which the value at each bin says how many pixels have intensities less than or equal to the intensity of the bin.

The goal is to produce a flat histogram, that is an image in which each pixel intensity occurs an equal number of times. The cumulative histogram of such an image will have that property that the number of pixels with an intensity less than or equal to a given intensity is proportional to that intensity.

Given an arbitrary image, we can produce an image with a linear cumulative histogram by adjusting the pixel intensities. Each intensity will be mapped to the value of the cumulative histogram for that bin. This guarantees that the number of pixels matches the intensity, which is the property we want. In practice, it is impossible to get a perfectly flat histogram (for example, the input image might have a constant intensity), so the result is only an approximately flat intensity histogram. To see how the histograms change with each step of the algorithm, see Figure 2.14.

In some parts of this thesis, only histogram equalization with subtracting the linear model is used. This is done when we do not know which pixels in the input window are likely to be foreground or background, and cannot apply the linear correction to just the face. Instead, we just apply the histogram equalization to the whole window, hoping that it will reduce the variability

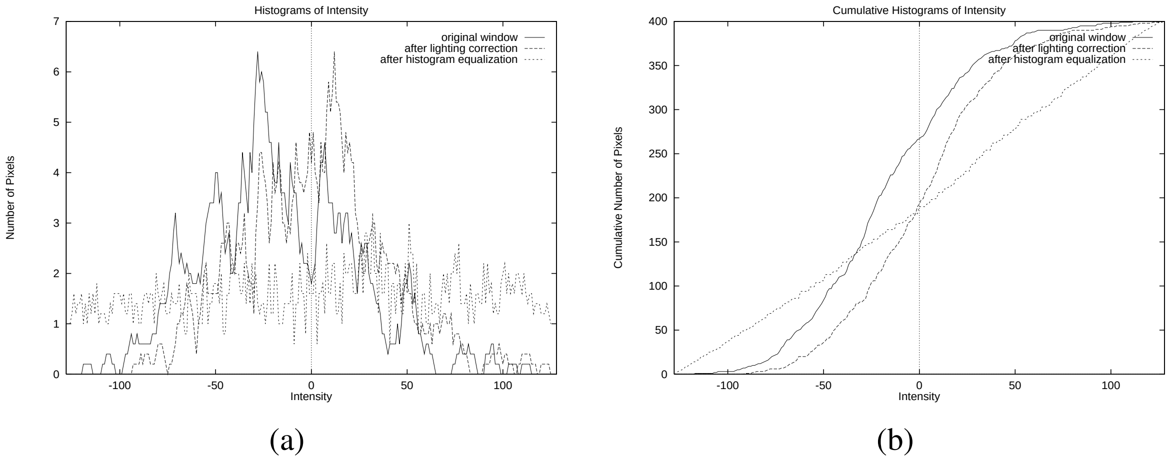


Figure 2.14: (a) Smoothed histograms of pixel intensities in a 20×20 window as it is passed through the preprocessing steps. Note that the lighting correction centers the peak of intensities at zero, while the histogram equalization step flattens the histogram. (b) The same three steps shown with cumulative histograms. The cumulative histogram of the result of lighting correction is used as a mapping function, to map old intensities to new ones.

somewhat, without the background pixels having too much effect on the appearance of the face in the foreground.

2.6 Face-Specific Lighting Compensation

Part of the motivation of the preprocessing steps in the previous section is to have robustness to variations in the lighting conditions, for instance lighting from the side of the face which changes its overall appearance. However, there are limits to what "dumb" corrections, with no knowledge of the structure of faces, can accomplish. In this section, I will present some preliminary ideas on how to intelligently correct for lighting variation.

2.6.1 Linear Lighting Models

The ideas in this section are based on the illumination models in the work of [Belhumeur and Kriegman, 1996], in which they explored the range of appearances an object can take on under differently lighting conditions. One assumption they use is that adding light sources to a scene results in an image which is a sum of the images for each individual light source. The authors further use the assumption that the object obeys a Lambertian lighting model for each individual light source, in which light is scattered from the object's surface equally in all directions. This means that the brightness of a point on the object depends only on the reflectivity of the object (its albedo) and the angle between the object's surface and the direction to the light source, according to the following formula (assuming there are no shadows):

$$I(x, y) = A(x, y)\mathbf{N}(x, y) \cdot \mathbf{L}$$

where $I(x, y)$ is the intensity of pixel x, y, $A(x, y)$ is the albedo of the corresponding point on the object, $\mathbf{N}(x, y)$ is the normal vector of the object's surface (relative to a vector pointing toward the camera) and $\mathbf{L}$ is a vector from the object to the light source, which is assumed to cast parallel rays on the object.

As the light source direction $\mathbf{L}$ is varied, $I(x, y)$ also varies, but the surface shape and albedo are fixed. Since the equation is linear, and L has three parameters, the space of images of the object (without shadows) is a three dimensional subspace. This subspace can be determined from (at least) example images of the object, by using principal components analysis (PCA). This subspace is related by a linear transformation to the set of normal vectors $\mathbf{N}(x, y)$. If we want to recover the true normal vectors, we need to know the actual light source directions. If these directions are available, the system can be treated as an over-constrained set of equations and solved directly for $\mathbf{N}(x, y)$ without performing principal components analysis. Actually, we will solve for the product $A(x, y)\mathbf{N}(x, y)$, but since $\mathbf{N}(x, y)$ have unit length, it is possible to separate the albedo $A(x, y)$. An example result is shown in Figure 2.15.

With $A(x, y)$ and $\mathbf{N}(x, y)$ in hand, which are essentially the color and shape of the face, we can then generate new images of the face under any desired lighting conditions. Some examples of images which can be generated are shown in Figure 2.16.

Such images can be used directly for training a face detector, and such experiments will be reported on in the next chapter. It is however quite time consuming to capture images of faces under multiple lighting conditions, and this limits the amount of training data. Ideally, we would like to learn about how images of faces change with different lighting, and apply that to new images of faces, for which we only have single images. The next two subsections describe some approaches for this.

2.6.2 Neural Networks for Compensation

Given a new input window to be classified as a face or nonface, we would like to apply a lighting correction which will remove any artifacts caused by extreme lighting conditions. This lighting correction must not change faces to nonfaces and vice-versa. The architecture we tried is shown in Figure 2.17.

The architecture feeds the input window to a neural network, which has been trained to produce a lighting correction, that is an image to add to the input which will make the lighting appear

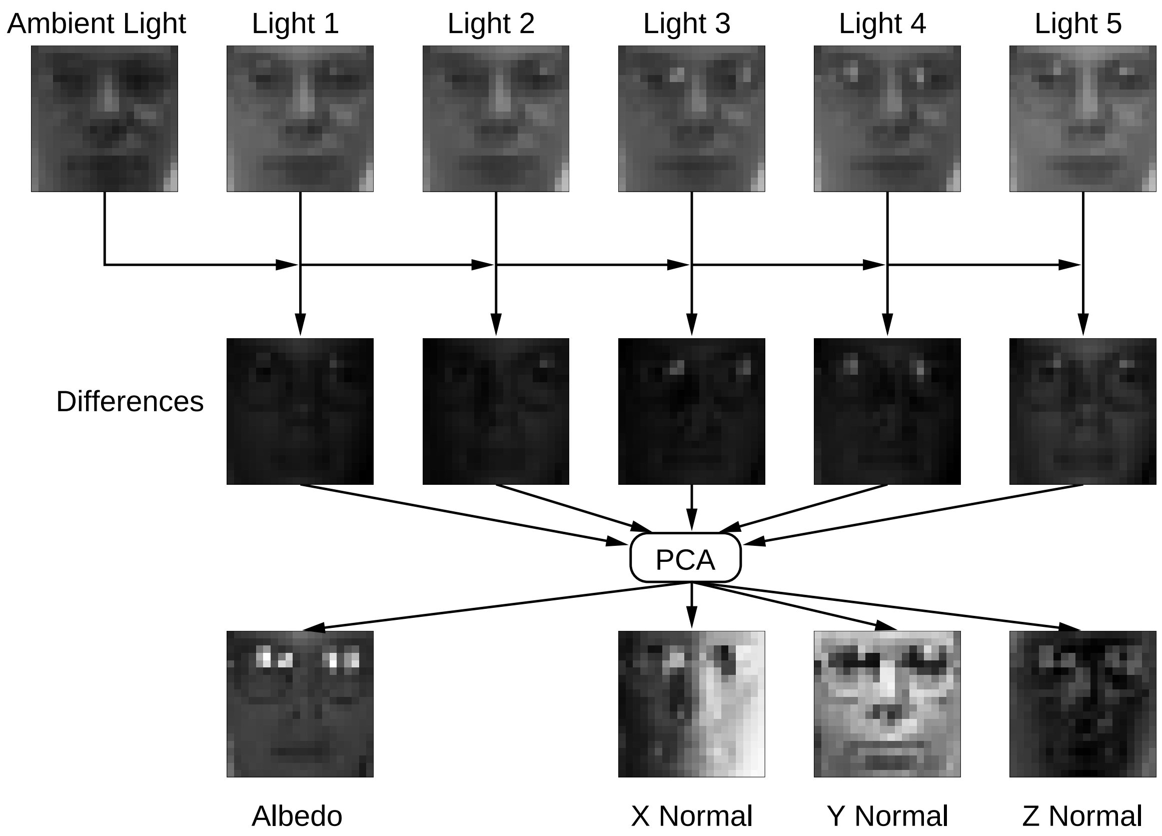


Figure 2.15: Example images under different lighting conditions, such as these, allow us to solve for the normal vectors on a face and its albedo.

that it is coming from the front of the face. Some example training data is shown in Figure 2.18. This data was prepared using the lighting models described above. This lighting correction is then added back into the original input window to get the corrected window. To prevent the neural network from applying arbitrary corrections (which could turn any nonface into a face), the network architecture contains a bottleneck, forcing the network to parameterize the correction using only four activation levels. The output layer essentially computes a linear combination of four images based on these activations.

Some results from this system for faces and nonfaces are shown in Figure 2.19. As can be seen, most of the results for faces are quite good (one exception is the fifth face from the left). Most of the strong shadows are removed, and the brightness levels of all parts of the face are similar. However, the results for nonfaces are troubling. Many of the nonfaces now look very face-like. The reason for this can be seen by considering the types of corrections that must be performed. When the lighting is very extreme, say from the left side of the face, the right side of the face will have intensity values of zero. Thus the corrector must "construct" the entire right half of the face. This construction capability makes it create faces when given relatively uniform nonfaces as input.

One potential solution to this problem would be to measure how much work the lighting cor-

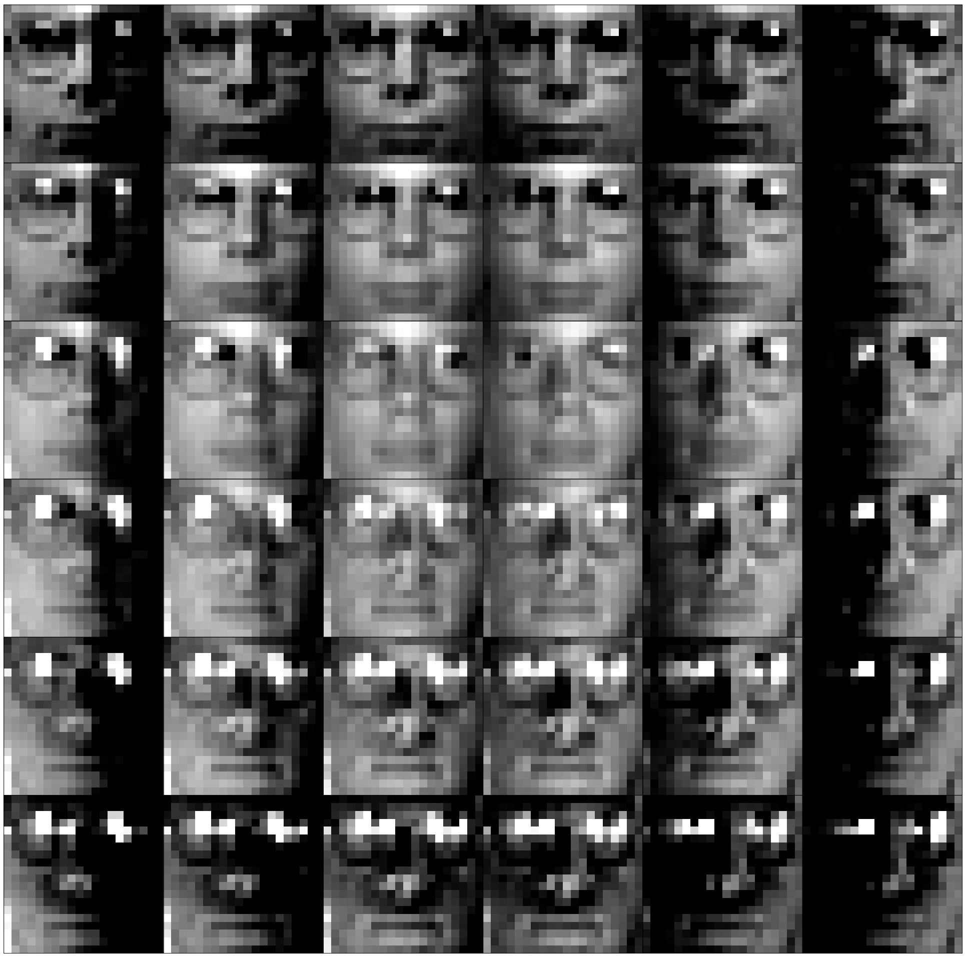


Figure 2.16: Generating example images under variable lighting conditions.

rection network had to do. If it made large changes in the image, then the result of the face detector applied to that window should be more suspect. This has not yet been explored.

2.6.3 Quotient Images for Compensation

Another approach to intelligently correcting the lighting in an image is presented in [Riklin-Raviv and Shashua, 1998]. The idea in this work is again to use linear lighting models. They present a technique where an input image can be simultaneously projected into the linear lighting spaces of a set of linear models. The simulatenous projection finds the $\mathbf{L}$ which minimizes the following quantity:

$$\sum_{i=1}^{n} \sum_{x,y} (I(x,y) - A_i(x,y)(\mathbf{N}_i(x,y) \cdot \mathbf{L}))^2$$

Where $I(x,y)$ is the input image, i is summed over all n lighting models, and, and $A_i(x,y)$ and $\mathbf{N}_i(x,y)$ are the corresponding albedo and normal vectors for lighting model i at pixel (x,y). The

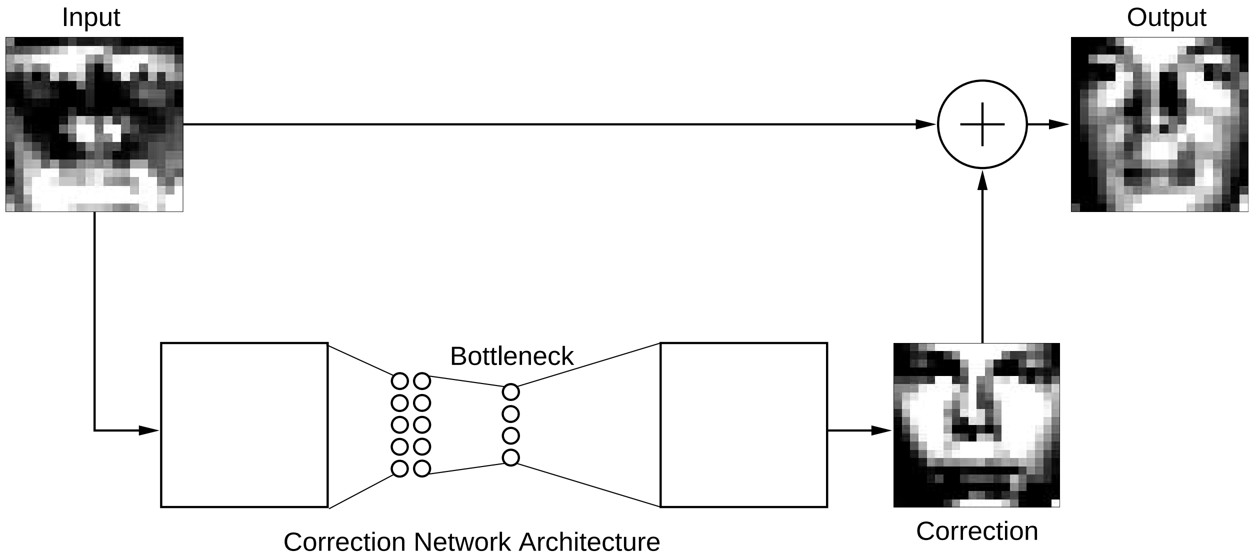


Figure 2.17: Architecture for correcting the lighting of a window. The window is given to a neural network, which has a severe bottleneck before its output. The output is a correction to be added to the original image.

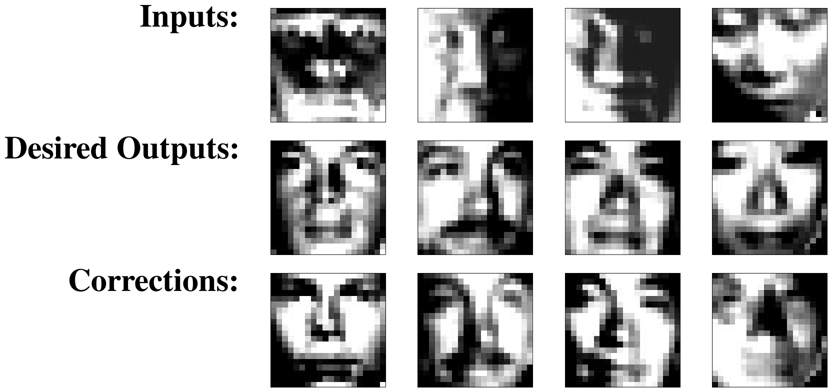


Figure 2.18: Training data for the lighting correction neural network.

result of this optimization is a vector **L** representing the lighting conditions for the face in the input image. Using a set of linear models allows for some robustness to differences in the albedos and shapes of individual faces. Using the collection of face lighting models, they then compute an image of the average face under the same conditions using the following equation:

$$\frac{1}{n}\sum_{i=1}^{n} A_i(x,y)(\mathbf{N}_i(x,y)\cdot\mathbf{L})$$

The input image is divided by this synthetic image, yielding a so called “quotient image”. Mathematically, the quotient image contains only the ratio of the albedos of the new face and the average face, assuming that the faces have similar shapes.

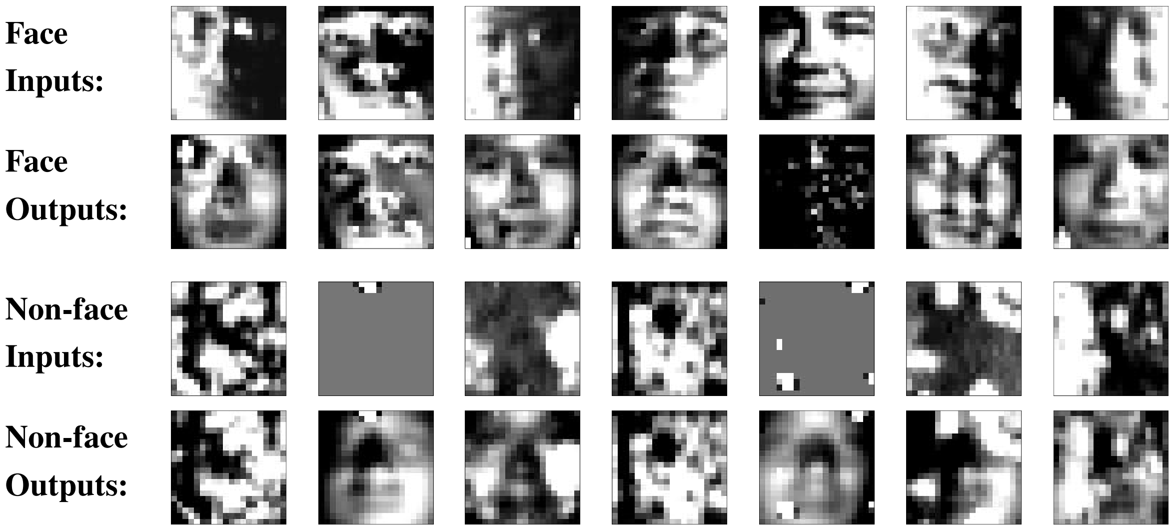


Figure 2.19: Result of lighting correction system. The lighting correction results for most of the faces are quite good, but some of the nonfaces have been changed into faces.

The original work on this technique used the quotient image for face recognition, because it removes the effects of lighting and allows recognition with fewer example images [Riklin-Raviv and Shashua, 1998]. The same approach can be used to normalize the lighting of input windows for face detection. Instead of just dividing by the average face under the estimated lighting conditions, we can go a step further, multiplying by the average face under frontal lighting:

$$I'(x,y) = I(x,y)\frac{\sum_{i=1} nA_i(x,y)(\mathbf{N}_i(x,y)\cdot\mathbf{L})}{\sum_{i=1} nA_i(x,y)(\mathbf{N}_i(x,y)\cdot\begin{pmatrix} 1 & 0 & 0 \end{pmatrix})}$$

This should ideally give an image of the original face but with frontal lighting. Some examples are shown in Figure 2.20. It is not clear that this approach will work well for face detection. As can be seen, while the overall intensity has been roughly normalized, the brightness differences across the face have not been improved. In some cases, bright spots have been introduced into the output image, probably because of the specular reflections in the images used to build the basis for the face images. Finally, since the lighting model does not incorporate shadows, the shadows cast by the nose or brow will cause problems.

2.7 Summary

The first part of this chapter described the training and test databases used throughout this thesis. The major focus however was some methods for segmenting face regions from training images, aligning faces with one another, and preprocessing them to improve contrast. The chapter ended

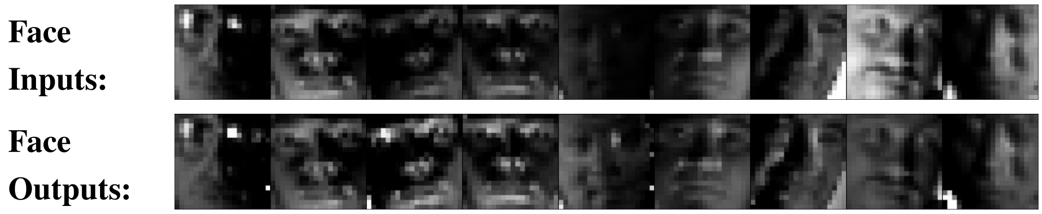


Figure 2.20: Result of using quotient images to correct lighting.

with some speculative results on how to intelligently correct for extreme lighting conditions in images. Together these techniques will be used to generate training data for the detectors to be described later.

The next chapter will begin the discussion of face detection itself, by examining the problem of detecting upright faces in images.

Chapter 3

Upright Face Detection

3.1 Introduction

In this chapter, I will present a neural network-based algorithm to detect upright, frontal views of faces in gray-scale images. The algorithm works by applying one or more neural networks directly to portions of the input image, and arbitrating their results. Each network is trained to output the presence or absence of a face.

Training a neural network for the face detection task is challenging because of the difficulty in characterizing prototypical "nonface" images. Unlike face *recognition*, in which the classes to be discriminated are different faces, the two classes to be discriminated in face *detection* are "images containing faces" and "images not containing faces". It is easy to get a representative sample of images which contain faces, but much harder to get a representative sample of those which do not. We avoid the problem of using a huge training set for nonfaces by selectively adding images to the training set as training progresses [Sung, 1996]. This "bootstrap" method reduces the size of the training set needed. The use of arbitration between multiple networks and heuristics to clean up the results significantly improves the accuracy of the detector.

The architecture of the system and training methods for the individual neural networks which make up the detector are presented in Section 3.2. Section 3.3 examines how these individual networks behave, by measuring their sensitivity to different parts of the input image, and measuring their performance on some test images. Methods to clean up the results and to arbitrate among multiple networks are presented in Section 3.4. The results in Section 3.5 show that the system is able to detect 90.5% of the faces over a test set of 130 complex images, with an acceptable number of false positives.

3.2 Individual Face Detection Networks

The system operates in two stages: it first applies a set of neural network-based detectors to an image, and then uses an arbitrator to combine the outputs. The individual detectors examine each location in the image at several scales, looking for locations that might contain a face. The arbitrator then merges detections from individual networks and eliminates overlapping detections.

The first component of our system is a neural network that receives as input a 20×20 pixel region of the image, and generates an output ranging from 1 to -1, signifying the presence or absence of a face, respectively. To detect faces anywhere in the input, the network is applied at every location in the image. To detect faces larger than the window size, the input image is repeatedly reduced in size (by subsampling), and the detector is applied at each size. This network must have some invariance to position and scale. The amount of invariance determines the number of scales and positions at which it must be applied. For the work presented here, we apply the network at every pixel position in the image, and scale the image down by a factor of 1.2 for each step in the pyramid. This image pyramid is shown at the left of Figure 3.1.

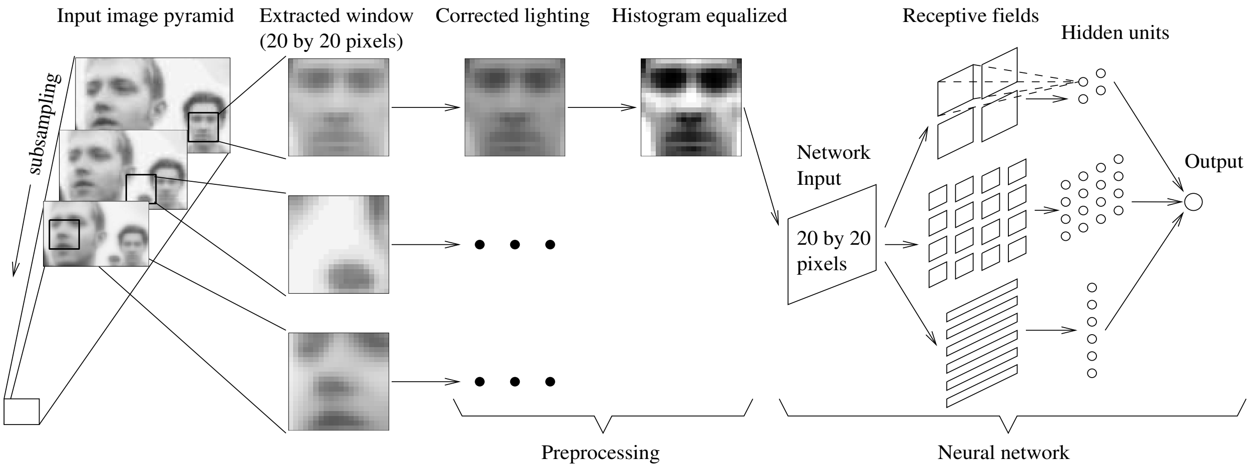


Figure 3.1: The basic algorithm used for face detection.

After a 20×20 pixel window is extracted from a particular location and scale of the input image pyramid, it is preprocessed using the affine lighting correction and histogram equalization steps described in Section 2.5. The preprocessed window is then passed to a neural network. The network has retinal connections to its input layer; the receptive fields of hidden units are shown in Figure 3.1. The input window is broken down into smaller pieces, of four 10×10 pixel regions, sixteen 5×5 pixel regions, and six overlapping 20×5 pixel regions. Each of these regions will have complete connections to a hidden unit, as shown in the figure. Although the figure shows a single hidden unit for each subregion of the input, these units can be replicated. For the experiments which are described later, we use networks with two and three sets of these hidden units. The shapes of these subregions were chosen to allow the hidden units to detect local features that might be important for face detection. In particular, the horizontal stripes allow the hidden units to detect such features as mouths or pairs of eyes, while the hidden units with square receptive fields might detect features such as individual eyes, the nose, or corners of the mouth. Other experiments have shown that the exact shapes of these regions do not matter; however it is important that the input is broken into smaller pieces instead of using complete connections to the entire input. Similar input connection patterns are commonly used in speech and character recognition tasks [Waibel *et al.*, 1989, Le Cun *et al.*, 1989]. The network has a single, real-valued output, which indicates whether or not the window contains a face.

3.2.1 Face Training Images

In order to use a neural network to classify windows as faces or nonfaces, we need training examples for each set. For positive examples, we use the techniques presented in Section 2.3 to align example face images in which some feature points have been manually labelled. After alignment, the faces are scaled to a uniform size, position, and orientation within a 20×20 pixel window. The images are scaled by a random factor between $1/\sqrt{1.2}$ to $\sqrt{1.2}$, and translated by a random amount up to 0.5 pixels. This allows the detector to be applied at each pixel location and at each scale in the image pyramid, and still detect faces at intermediate locations or scales. In addition, to give the detector some robustness to slight variations in the faces, they are rotated by a random amount (up to 10° degrees). In our experiments, using larger amounts of rotation to train the detector network yielded too many false positive to be usable. There are a total of 1046 training examples in our training set, and 15 of these randomized training examples are generated for each original face. The next sections describe methods for collecting negative examples and training.

3.2.2 Non-Face Training Images

We needed a large number of nonface images to train the face detector, because the variety of nonface images is much greater than the variety of face images. One large class of images which do not contain any faces are pictures of scenery, such as trees, mountains, and buildings. There is a large collection of images located at `http://wuarchive.wustl.edu/multimedia/images/gif/`. We selected the images with the keyword "Scenery" in their descriptions from the index, and downloaded those images. This, along with a couple other images from other sources, formed our collection of 120 nonface "scenery" images.

Collecting a "representative" set of nonfaces is difficult. Practically any image can serve as a nonface example; the space of nonface images is much larger than the space of face images. The statistical approach to machine learning suggests that we should train the neural networks on precisely the same distribution of images which it will see at runtime. For our face detector, the number of face examples is 15,000, which is a practical number. However, our representative set of scenery images contains approximately 150,000,000 windows, and training on a database of this size is very difficult. The next two sections describe two approaches to training with this amount of data.

3.2.3 Active Learning

Because of the difficult of training with every possible negative example, we utilized an algorithm described in [Sung, 1996]. Instead of collecting the images before training is started, the images are collected during training, in the following manner:

1. Create an initial set of nonface images by generating 1000 random images. Apply the preprocessing steps to each of these images.

2. Train a neural network to produce an output of 1 for the face examples, and -1 for the nonface examples. On the first iteration of this loop, the network's weights are initialized randomly. After the first iteration, we use the weights computed by training in the previous iteration as the starting point.

3. Run the system on an image of scenery *which contains no faces*. Collect subimages in which the network incorrectly identifies a face (an output activation > 0).

4. Select up to 250 of these subimages at random, apply the preprocessing steps, and add them into the training set as negative examples. Go to Step 2.

The training algorithm used in Step 2 is the standard error backpropogation algorithm with a momentum term [Hertz *et al.*, 1991]. The neurons use the $\tanh$ activation function, which gives an output ranging from -1 to 1, hence the threshold of 0 for the detection of a face. Since we are not training with all the negative examples, the probabilistic arguments of the previous section do not apply for setting the detection threshold.

Since the number of negative examples is much larger than the number of positive examples, uniformly sampled batches of training examples would often contain only negative examples, which would be inappropriate for neural network training. Instead, each batch of 100 positive and negative examples is drawn randomly from the entire training sets, and passed to the backpropogation algorithm as a batch. We choose the training batches such that they have 50% positive examples and 50% negative examples. This ensures that initially, when we have a much larger set of positive examples than negative examples, the network will actually learn something from both sets. Note that this changes the distribution of faces and nonfaces in the training sets compared with what the network will see at run time. Although theoretically the wrong thing to do, [Lawrence *et al.*, 1998] observes such techniques often work well in practice.

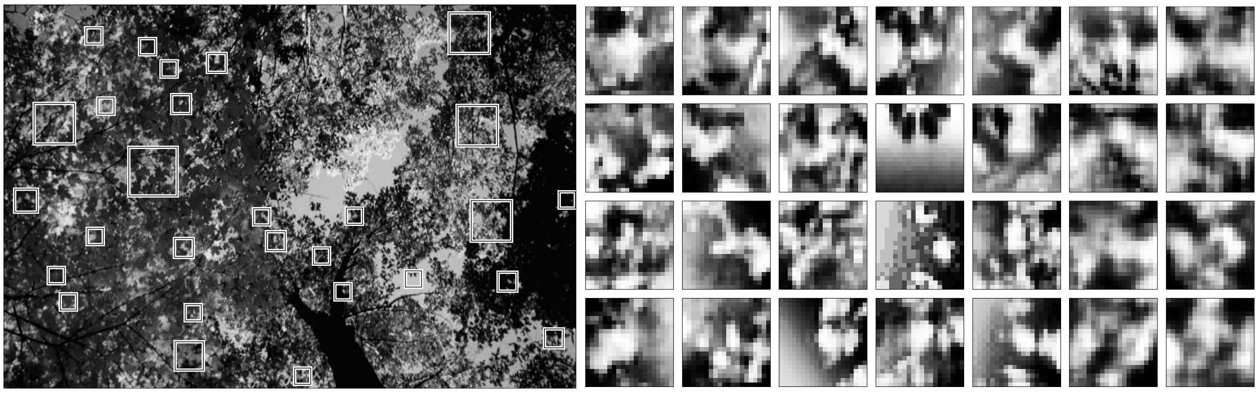


Figure 3.2: During training, the partially-trained system is applied to images of scenery which do not contain faces (like the one on the left). Any regions in the image detected as faces (which are expanded and shown on the right) are errors, which can be added into the set of negative training examples.

Some examples of nonfaces that are collected during training are shown in Figure 3.2. Note that some of the examples resemble faces, although they are not very close to the positive examples shown in Figure 2.7. The presence of these examples forces the neural network to learn the precise boundary between face and nonface images. We used 120 images of scenery for collecting negative examples in the bootstrap manner described above. A typical training run selects approximately 8000 nonface images from the 146,212,178 subimages that are available at all locations and scales in the training scenery images. A similar training algorithm was described in [Drucker *et al.*, 1993], where at each iteration an entirely new network was trained with the examples on which the previous networks had made mistakes.

3.2.4 Exhaustive Training

Neural network training usually requires training the network many times on its training images; a single pass through 150,000,000 scenery windows not only requires a huge amount of storage, but also takes nearly a day on a four processor SGI supercomputer. Additionally, a network usually trains on images in batches of about 100 images; by the time we reach the end of 150,000,000 examples, it will have forgotten the characteristics of first images.

As in the previous section, to insure that the the neural network learns about both faces and nonfaces, we select the batches of negative examples to have approximately equal numbers of positive and negative examples. However, this changes the apparent distribution of positive and negative examples, so that it no longer matches the real distribution.

It is possible to compensate for this using Bayes' Theorem, though (see also the discussion in [Lawrence *et al.*, 1998]). If we denote $P(\text{face}|\text{window})$ as the probability that a given window is a face, and $P'(\text{face})$ and $P'(\text{nonface})$ as the prior probability of faces and nonfaces in the training sets (both 0.5), then Bayes' Theorem says:

$$\text{NN Output} = P'(\text{face}|\text{window}) = \frac{P(\text{window}|\text{face}) \cdot P'(\text{face})}{P(\text{window}|\text{face}) \cdot P'(\text{face}) + P(\text{window}|\text{nonface}) \cdot P'(\text{nonface})}$$

Neural networks will learn to estimate the left hand side of this equation, and since we know $P'(\text{face})$, $P'(\text{nonface})$, and that $P(window|nonface) = 1 - P(window|face)$, this equation simplifies dramatically, giving:

$$P(\text{window}|\text{face}) = \text{NN Output}$$

Let us denoted the true probability of faces is $P(\text{face})$, and nonfaces is $P(\text{nonface})$. Then we can use Bayes' Theorem in the forward direction to get the true probability of a face given the image:

$$P(\text{face}|\text{window}) = \frac{\text{NN Output} * P(\text{face})}{\text{NN Output} \cdot P(\text{face}) + (1 - \text{NN Output}) \cdot P(\text{nonface})}$$

We would like to classify a window as a face if $P(\text{face}|\text{window}) > 0.5$, which is equivalent to setting a threshold of:

$$\text{NN Output} > 1 - P(\text{face})$$

Since we are using neural networks with $\tanh$ activation functions, the output range is -1 to 1, so this threshold is adjusted as follows:

$$\text{NN Output} > 1 - 2P(\text{face})$$

Thus we need to determine the prior probability of faces, which will be discussed in Section 3.3.2.

3.3 Analysis of Individual Networks

This section presents some analysis of the performance of the networks described above, beginning with a sensitivity analysis, then examining the performance on the *Upright Test Set*.

3.3.1 Sensitivity Analysis

In order to determine which part of its input image the network uses to decide whether the input is a face, we performed a sensitivity analysis using the method of [Baluja, 1996]. We collected a positive test set based on the training database of face images, but with different randomized scales, translations, and rotations than were used for training. The negative test set was built from a set of negative examples collected during the training of other networks. Each of the 20×20 pixel input images was divided into 100 2×2 pixel subimages. For each subimage in turn, we went through the test set, replacing that subimage with random noise, and tested the neural network. The resulting root mean square error of the network on the test set is an indication of how important that portion of the image is for the detection task. Plots of the error rates for two networks we trained are shown in Figure 3.3. Network 1 uses two sets of the hidden units illustrated in Figure 3.1, while Network 2 uses three sets.

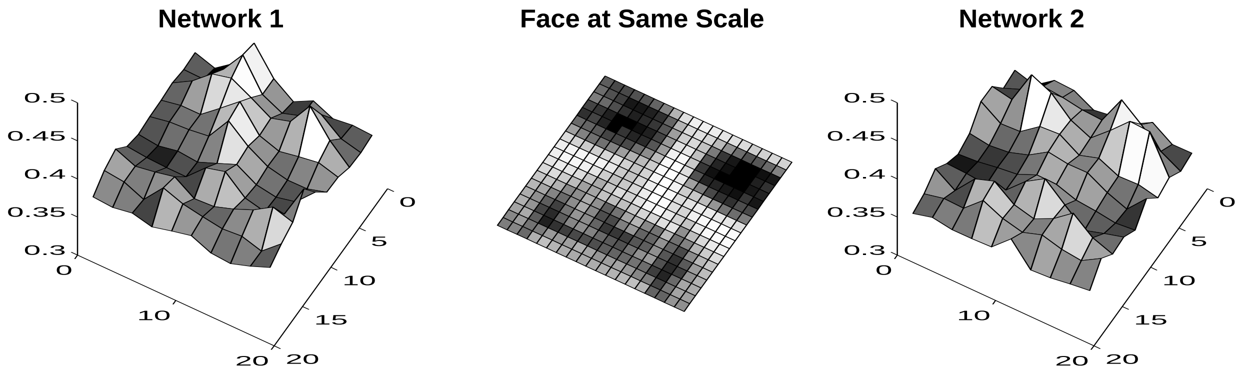


Figure 3.3: Error rates (vertical axis) on a test created by adding noise to various portions of the input image (horizontal plane), for two networks. Network 1 has two copies of the hidden units shown in Figure 3.1 (a total of 58 hidden units and 2905 connections), while Network 2 has three copies (a total of 78 hidden units and 4357 connections).

The networks rely most heavily on the eyes, then on the nose, and then on the mouth (Figure 3.3). Anecdotally, we have seen this behavior on several real test images. In cases in which only one eye is visible, detection of a face is possible, though less reliable, than when the entire face is visible. The system is less sensitive to the occlusion of the nose or mouth.

3.3.2 ROC (Receiver Operator Characteristic) Curves

The outputs from our face detection networks are not binary. The neural networks produce real values between 1 and -1, indicating whether or not the input contains a face. A threshold value of zero is used during *training* to select the negative examples (if the network outputs a value of greater than zero for any input from a scenery image, it is considered a mistake). Although this value is intuitively reasonable, by changing this value during *testing*, we can vary how conservative the system is. To examine the effect of this threshold value during testing, we measured the detection and false positive rates as the threshold was varied from 1 to -1. At a threshold of 1, the false detection rate is zero, but no faces are detected. As the threshold is decreased, the number of correct detections will increase, but so will the number of false detections.

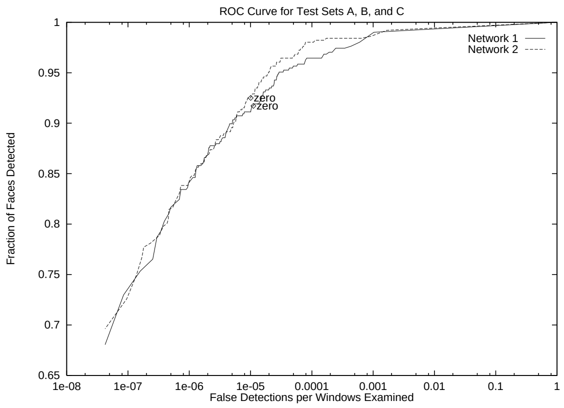


Figure 3.4: The detection rate plotted against false positive rates as the detection threshold is varied from -1 to 1, for the same networks as Figure 3.3. The performance was measured over all images from the *Upright Test Set*. The points labelled "zero" are the zero threshold points which are used for all other experiments.

This tradeoff is presented in Figure 3.4, which shows the detection rate plotted against the number of false positives as the threshold is varied, for the two networks presented in the previous section. This is measured for the images in the *Upright Test Set*, which consists 130 images with 507 faces (plus 4 upside-down faces not considered in this chapter), and requires the networks to process 83,099,211 windows. The false positive rate is in terms of the number of 20×20 pixel windows that must be examined. This number can be approximated from the number of pixels in the image and the scale factor between different resolutions in the image pyramid (1.2):

$$\text{number of windows} \approx \text{width} \cdot \text{height} \cdot \left(\sum_{l=0}^{\infty} (1.2 \cdot 1.2)^{-l} \right) = \frac{\text{width} \cdot \text{height}}{1 - 1.2^{-2}} \approx 3.27 \cdot \text{width} \cdot \text{height}$$

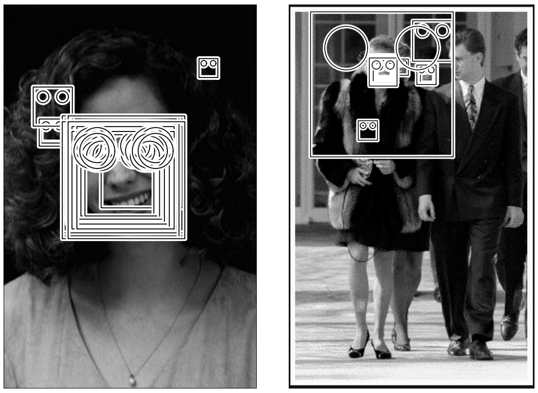
Since the zero threshold locations are close to the "knees" of the curves, as can be seen from the figure, we used a zero threshold value throughout testing.

To give an intuitive idea about the meaning of the numbers in Figure 3.4 (with a zero threshold), some examples of the output on the two images in Figure 3.5 are shown in Figure 3.6. In the figure, each box represents the position and size of a window to which Network 1 gave a positive response. The network has some invariance to position and scale, which results in multiple boxes around some faces. Note also that there are quite a few false detections; the next section presents some methods to reduce them.

The above analysis can be used with the probabilistic analysis in Section 3.2.4 to determine the threshold for detecting faces in that scheme. Suppose that for a true face, windows one pixel either

(a) (b)

Figure 3.5: Example images on to test the output of the upright detector.



(a) (b)

Figure 3.6: Images from Figure 3.5 with all the above threshold detections indicated by boxes. Note that the circles are drawn for illustration only, they do not represent detected eye locations.

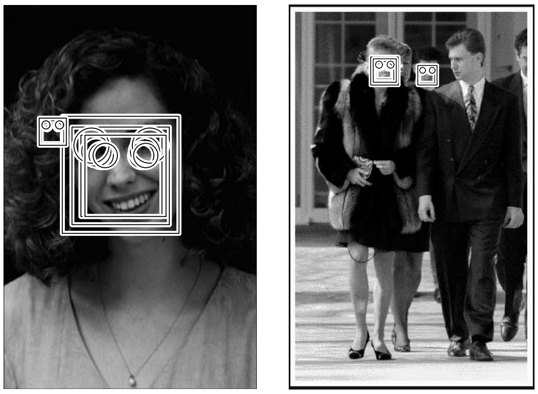
side of its location, and windows either side of its scale can be detected, then each face contributes about $3 \cdot 3 \cdot 3 = 27$ face windows. In the training database, there are 1046 faces ($27 \times 1046 = 28242$ face windows) and 592,624,845 20×20 windows, giving a probability of faces equal to $1/20984$. This is the value that will be used later in testing.

3.4 Refinements

The examples in Figure 3.6 showed that the raw output from a single network will contain a number of false detections. In this section, we present two strategies to improve the reliability of the detector: cleaning-up the outputs from an individual network, and arbitrating among multiple networks.

3.4.1 Clean-Up Heuristics

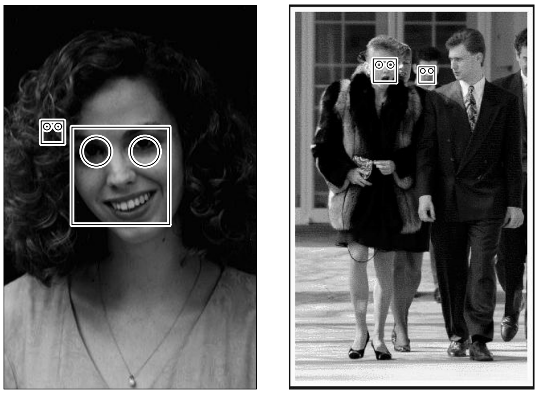
Note that in Figure 3.6a, the face is detected at multiple nearby positions or scales, while false detections often occur with less consistency. The same is true of Figure 3.6b, but since the faces are smaller the overlapping detections are not visible. These observation lead to a heuristic which can eliminate many false detections. For each detection, the number of other detections within a specified neighborhood of that detection can be counted. If the number is above a threshold, then that location is classified as a face. The centroid of the nearby detections defines the location of the detection result, thereby collapsing multiple detections. In the experiments section, this heuristic will be referred to as *thresholding(size,level)*, where *size* is the size of the neighborhood, in both pixels and pyramid steps, on either side of the detection in question, and *level* is the total number of detections which must appear in that neighborhood. The result of applying *threshold(4,2)* to the images in Figure 3.6 is shown in Figure 3.7.



(a) (b)

Figure 3.7: Result of applying *threshold(4,2)* to the images in Figure 3.6.

If a particular location is correctly identified as a face, then all other detection locations which overlap it are likely to be errors, and can therefore be eliminated. Based on the above heuristic regarding nearby detections, we preserve the location with the higher number of detections within a small neighborhood, and eliminate locations with fewer detections. In the discussion of the experiments, this heuristic is called *overlap*. There are relatively few cases in which this heuristic fails; however, one such case is illustrated by the left two faces in Figure 3.8b, where one face partially occludes another, and so is lost when this heuristic is applied. These arbitration heuristics are very similar to, but computationally less expensive than, those presented in my previous paper [Rowley *et al.*, 1998].



(a) (b)

Figure 3.8: Result of applying *overlap* to the images in Figure 3.7.

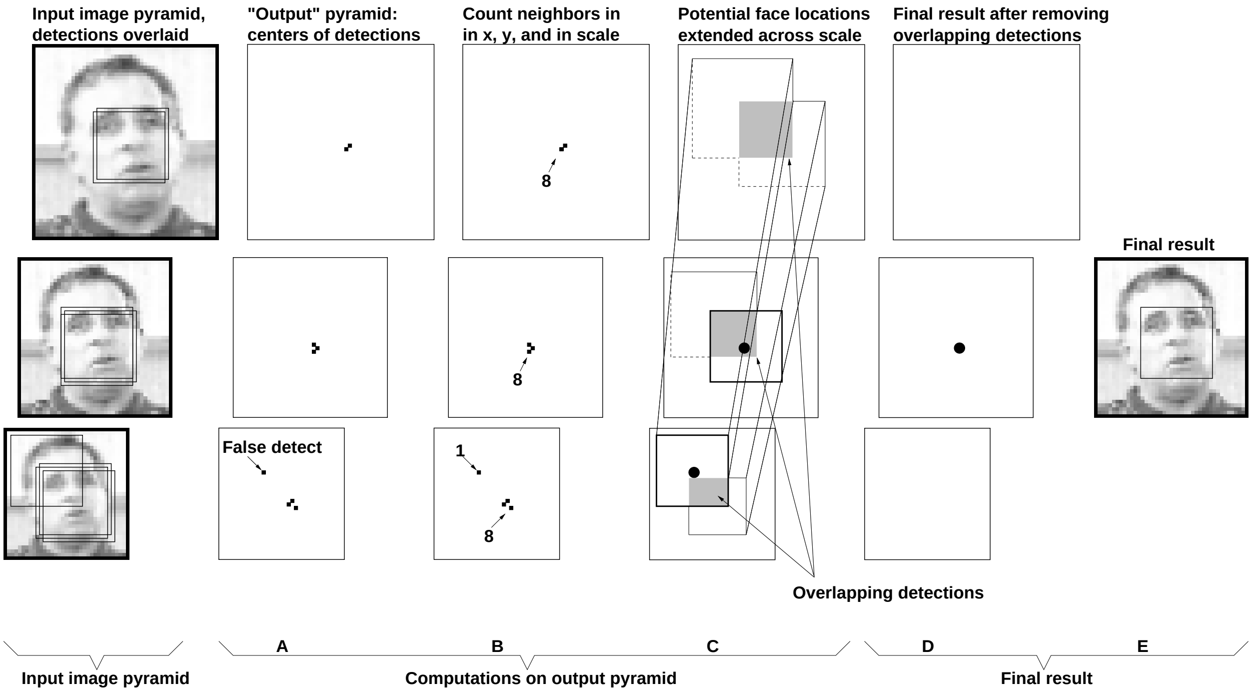


Figure 3.9: The framework for merging multiple detections from a single network: A) The detections are recorded in an "output" pyramid. B) The number of detections in the neighborhood of each detection are computed. C) The final step is to check the proposed face locations for overlaps, and D) to remove overlapping detections if they exist. In this example, removing the overlapping detection eliminates what would otherwise be a false positive.

The implementation of these two heuristics is illustrated in Figure 3.9. Each detection at a particular location and scale is marked in an image pyramid, called the "output" pyramid. Then, each detection is replaced by the number of detections within its neighborhood. A threshold is applied to these values, and the centroids (in both position and scale) of all above threshold detections are computed (this step is omitted in Figure 3.9. Each centroid is then examined in order, starting from the ones which had the highest number of detections within the specified neighborhood. If any other centroid locations represent a face overlapping with the current centroid, they are removed from the output pyramid. All remaining centroid locations constitute the final detection result. In the face detection work described in [Burel and Carel, 1994], similar observations about the nature of the outputs were made, resulting in the development of heuristics similar to those described above.

3.4.2 Arbitration among Multiple Networks

[Sung, 1996] provided some formalization of how a set of identically trained detectors can be used together to improve accuracy. He argued that if the errors made by a detector are independent, then by having a set of networks vote on the result, the number of overall errors will be reduced. [Baker and Nayar, 1996] used the converse idea, that of pattern rejectors, for recognition. Each classifier eliminates a set of potential classifications of an example, until only the example's class is left.

To further reduce the number of false positives, we can apply multiple networks, and arbitrate between their outputs to produce the final decision. Each network is trained using the same algorithm with the same set of face examples, but with different random initial weights, random initial nonface images, and permutations of the order of presentation of the scenery images. As will be seen in the next section, the detection and false positive rates of the individual networks will be quite close. However, because of different training conditions and because of self-selection of negative training examples, the networks will have different biases and will make different errors.

The arbitration algorithm is illustrated in Figure 3.10. Each detection at a particular position and scale is recorded in an image pyramid, as was done with the previous heuristics. One way to combine two such pyramids is by ANDing them. This strategy signals a detection only if both networks detect a face at precisely the same scale and position. Due to the different biases of the individual networks, they will rarely agree on a false detection of a face. This allows ANDing to eliminate most false detections. Unfortunately, this heuristic can decrease the detection rate because a face detected by only one network will be thrown out. However, we will see later that individual networks can all detect roughly the same set of faces, so that the number of faces lost due to ANDing is small.

Similar heuristics, such as ORing the outputs of two networks, or voting among three networks, were also tried. In practice, these arbitration heuristics can all be implemented with variants of the *threshold* algorithm described above. For instance, ANDing can be implemented by combining the results of the two networks, and applying *threshold(0,2)*, ORing with *threshold(0,1)*, and voting by applying *threshold(0,2)* to the results of three networks.

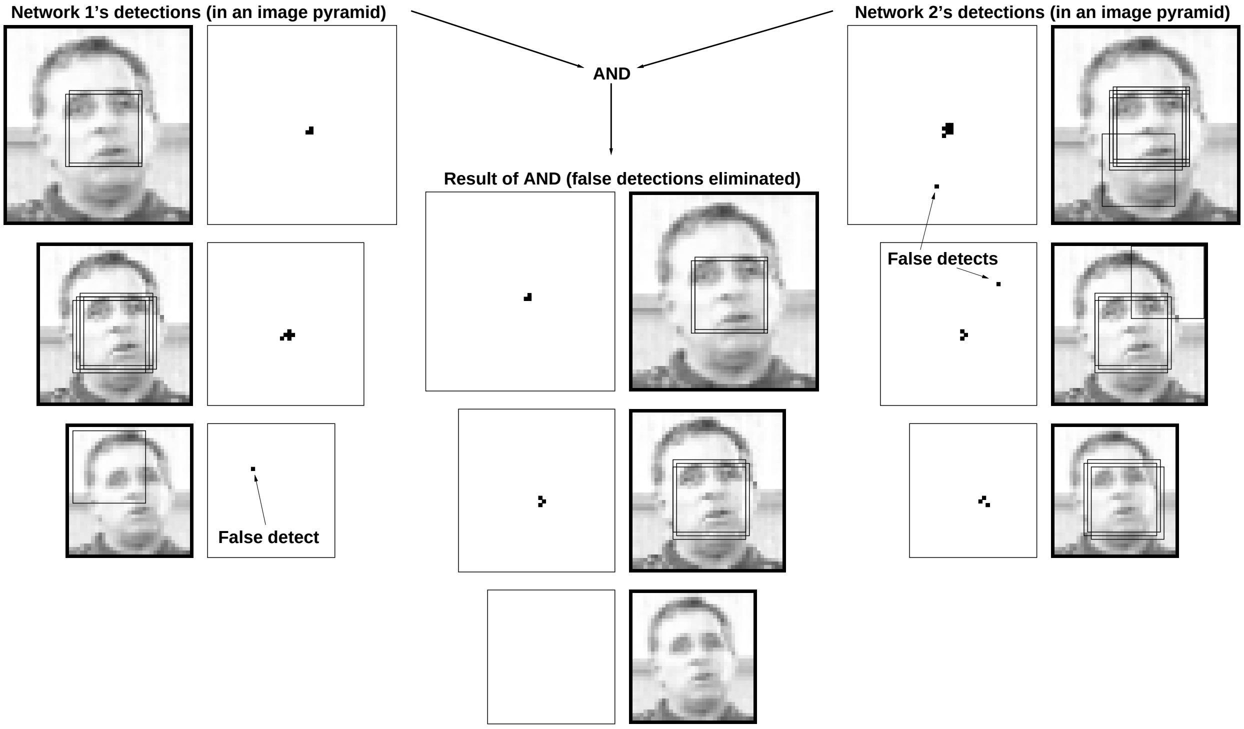


Figure 3.10: ANDing together the outputs from two networks over different positions and scales can improve detection accuracy.

Each of these arbitration methods can be applied before or after the clean-up heuristics. If applied afterwards, we combine the centroid locations rather than actual detection locations, and require them to be within some neighborhood of one another rather than precisely aligned, by setting the *size* parameter of the *threshold* which implements the arbitration to a 4 rather than 0. These are denoted *AND(4)* and *AND(0)* in the experiments.

Arbitration strategies such as ANDing, ORing, or voting seem intuitively reasonable, but perhaps there are some less obvious heuristics that could perform better. To test this hypothesis, we applied a separate neural network to arbitrate among multiple detection networks, as illustrated in Figure 3.11. For every location, the arbitration network examines a small neighborhood surrounding that location in the output pyramid of each individual network. For each pyramid, we count the number of detections in a 3×3 pixel region at each of three scales around the location of interest, resulting in three numbers for each detector, which are fed to the arbitration network. The arbitration network is trained (using the images from which the positive face examples were extracted) to produce a positive output for a given set of inputs only if that location contains a face, and to produce a negative output for locations without a face. As will be seen in the next section, using an arbitration network in this fashion produced results comparable to (and in some cases, slightly better than) those produced by the heuristics presented earlier, at the expense of extra complexity.

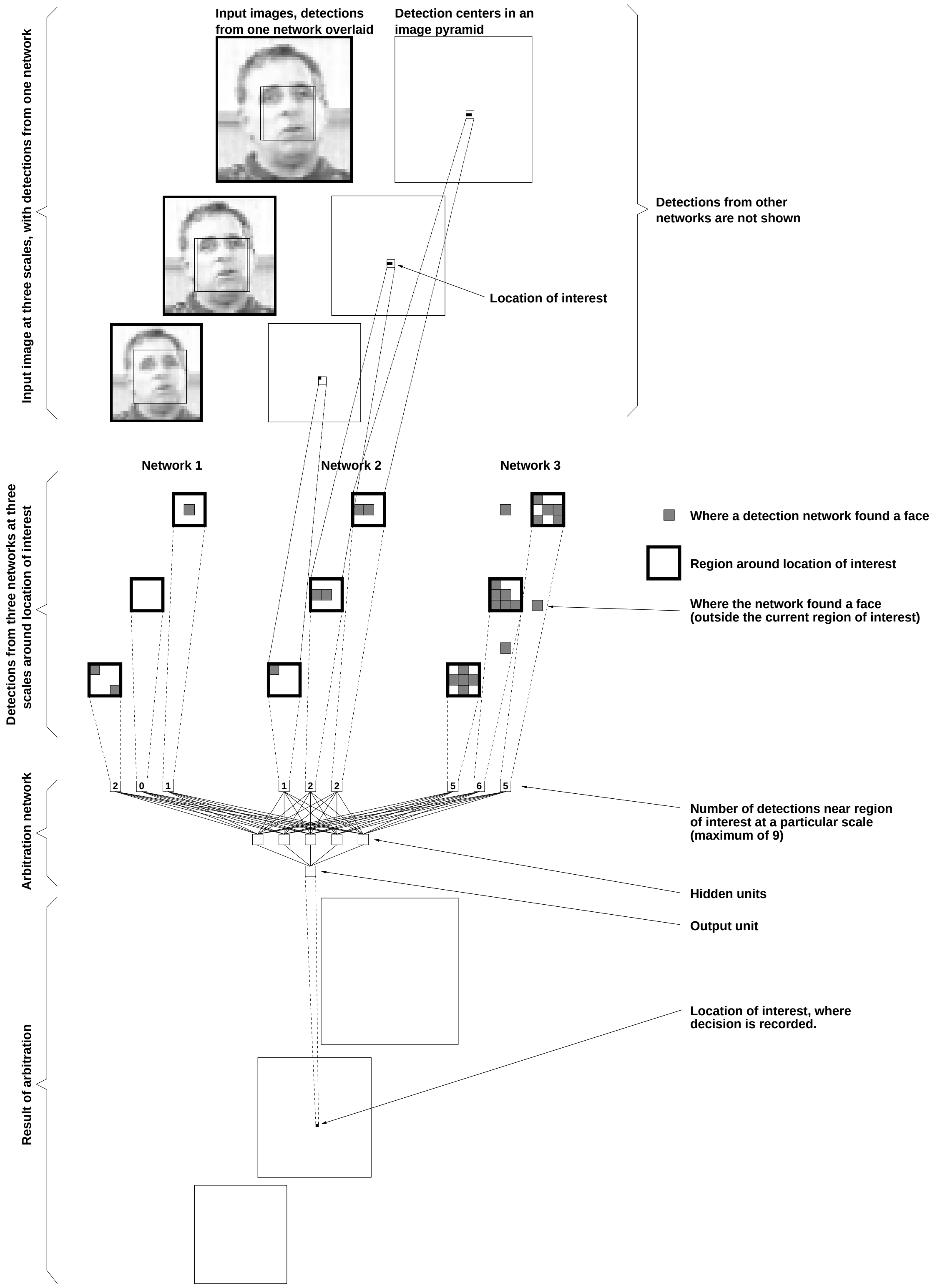


Figure 3.11: The inputs and architecture of the arbitration network which arbitrates among multiple face detection networks.

3.5 Evaluation

A number of experiments were performed to evaluate the system. We first show an analysis of which features the neural network is using to detect faces, then present the error rates of the system over two large test sets, and finally show some example output.

3.5.1 Upright Test Set

The first set of test images is for testing the capabilities of the upright face detector.

One of the first face detection systems with high accuracy in cluttered images was developed at the MIT Media Lab by Kah-Kay Sung and Tomaso Poggio [Sung, 1996]. To evaluate the accuracy of their system, they collected a test database of 23 images from various sources, which we also use for testing purposes.

In addition to these images, we collected 107 images containing upright faces locally. These images were scanned from newspapers, magazines, and photographs, found on the WWW, or captured with CCD cameras attached to digitizers, or digitized from broadcast television. The latter images were provided by Michael Smith from the Informedia project at CMU.

A number of these images were chosen specifically to test the tolerance to clutter in images, and did not contain any faces. Others contained large numbers of upright, frontal faces, to test the detector's tolerance of different types of faces. A few example images are shown in Figure 3.12. In the following, this test set will be called the *Upright Test Set*.

Figure 3.12: Example images from the *Upright Test Set*, used for testing the upright face detector.

Table 3.13 shows the performance of different versions of the detector on the *Upright Test Set*. The four columns show the number of faces missed (out of 507), the detection rate, the total number of false detections, and the false detection rate (compared with the number of 20×20 windows examined.

Table 3.13: Detection and error rates for the *Upright Test Set*, which consists of 130 images and contains 507 frontal faces. It requires the system to examine a total of 83,099,211 20×20 pixel windows.

Type	System	Missed faces	Detect rate	False detects	False detect rate
One network, no heuristics	1) Network 1 (2 copies of hidden units (52 total), 2905 connections)	44	91.3%	928	1/89546
	2) Network 2 (3 copies of hidden units (78 total), 4357 connections)	37	92.7%	853	1/97419
	3) Network 3 (2 copies of hidden units (52 total), 2905 connections)	47	90.7%	759	1/109485
	4) Network 4 (3 copies of hidden units (78 total), 4357 connections)	40	92.1%	820	1/101340
One network, with heuristics	5) Network 1 → threshold(4,1) → overlap	50	90.1%	516	1/161044
	6) Network 2 → threshold(4,1) → overlap	44	91.3%	453	1/183441
	7) Network 3 → threshold(4,1) → overlap	51	89.9%	422	1/196917
	8) Network 4 → threshold(4,1) → overlap	42	91.7%	452	1/183847
Arbitrating among two networks	9) Networks 1 and 2 → AND(0)	66	87.0%	156	1/532687
	10) Networks 1 and 2 → AND(0) → threshold(4,3) → overlap	92	81.9%	8	1/10387401
	11) Networks 1 and 2 → threshold(4,2) → overlap → AND(4)	71	86.0%	31	1/2680619
	12) Networks 1 and 2 → threshold(4,2) → overlap → OR(4) → threshold(4,1) → overlap	50	90.1%	167	1/497600
Arbitrating among three networks	13) Networks 1, 2, 3 → voting(0) → overlap	55	89.2%	95	1/874728
	14) Networks 1, 2, 3 → network arbitration (5 hidden units) → threshold(4,1) → overlap	85	83.2%	10	1/8309921
	15) Networks 1, 2, 3 → network arbitration (10 hidden units) → threshold(4,1) → overlap	86	83.0%	10	1/8309921
	16) Networks 1, 2, 3 → network arbitration (perceptron) → threshold(4,1) → overlap	89	82.4%	9	1/9233245

The table begins by showing the results for four individual networks. Networks 1 and 2 are the same as those used in Sections 3.3.1 and 3.3.2. Networks 3 and 4 are identical to Networks 1 and 2, respectively, except that the negative example images were presented in a different order during training. The results for ANDing and ORing networks were based on Networks 1 and 2, while voting and network arbitration were based on Networks 1, 2, and 3. The neural network arbitrators were trained using the images from which the face examples were extracted. Three different architectures for the network arbitrator were used. The first used 5 hidden units, as shown in Figure 3.11. The second used two hidden layers of 5 units each, with complete connections between each layer, and additional connections between the first hidden layer and the output. The last architecture was a simple perceptron, with no hidden units.

As discussed earlier, the *threshold* heuristic for merging detections requires two parameters, which specify the size of the neighborhood used in searching for nearby detections, and the threshold on the number of detections that must be found in that neighborhood. In the table, these two parameters are shown in parentheses after the word *threshold*. Similarly, the ANDing, ORing, and voting arbitration methods have a parameter specifying how close two detections (or detection centroids) must be in order to be counted as identical.

Systems 1 through 4 in the table show the raw performance of the networks. Systems 5 through 8 use the same networks, but include the *threshold* and *overlap* steps which decrease the number of false detections significantly, at the expense of a small decrease in the detection rate. The remaining systems all use arbitration among multiple networks. Using arbitration further reduces the false positive rate, and in some cases increases the detection rate slightly. Note that for systems using arbitration, the ratio of false detections to windows examined is extremely low, ranging from 1 false detection per $497,600$ windows to down to 1 in $10,387,401$, depending on the type of arbitration used. Systems 10, 11, and 12 show that the detector can be tuned to make it more or less conservative. System 10, which uses ANDing, gives an extremely small number of false positives, and has a detection rate of about 81.9%. On the other hand, System 12, which is based on ORing, has a higher detection rate of 90.1% but also has a larger number of false detections. System 11 provides a compromise between the two. The differences in performance of these systems can be understood by considering the arbitration strategy. When using ANDing, a false detection made by only one network is suppressed, leading to a lower false positive rate. On the other hand, when ORing is used, faces detected correctly by only one network will be preserved, improving the detection rate.

Systems 14, 15, and 16, all of which use neural network-based arbitration among three networks, yield detection and false alarm rates between those of Systems 10 and 11. System 13, which uses voting among three networks, has an accuracy between that of Systems 11 and 12.

3.5.2 FERET Test Set

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

Figure 3.14: Examples of nearly frontal FERET images: (a) frontal (group labels `fa` and `fb`, (b) $15°$ from frontal (group labels `rb` and `rc`), and (c) $22.5°$ from frontal (group labels `ql` and `qr`).

The second test set we used was the portion of the FERET database [Phillips *et al.*, 1996, Phillips *et al.*, 1997, Phillips *et al.*, 1998] containing roughly frontal faces. The FERET project was run by the Army Research Lab to perform an uniform comparison of several face recognition algorithms. As part of this work, the researchers collected a large database of face images. For each person, they collected several images in different sessions, from with different angles of the face relative to the camera. The images were taken as photographs, using studio lighting conditions, and digitized later. The backgrounds were typically uniform or uncluttered, as can be seen in Figure 3.14. There are a wide variety of faces in the database, which are taken at a variety of angles. Thus these images are more useful for checking the angular sensitivity of the detector, and less useful for measuring the false alarm rate.

We partitioned the images into three groups, based on the nominal angle of the face with respect to the camera: frontal faces, faces at an angle $15°$ from the camera, and faces at an angle of $22.5°$. The direction of the face varies significantly within these groups. As can be seen from Table 3.15, the detection rate for systems arbitrating two networks ranges between 98.1% and 100.0% for frontal and $15°$ faces, while for $22.5°$ faces, the detection rate is between 93.1% and 97.1%. This difference is because the training set contains mostly frontal faces. It is interesting to note that the systems generally have a higher detection rate for faces at an angle of $15°$ than for frontal faces. The majority of people whose frontal faces are missed are wearing glasses which are reflecting light into the camera. The detector is not trained on such images, and expects the eyes to be darker than the rest of the face. Thus the detection rate for such faces is lower.

Table 3.15: Detection and error rates for the *FERET Test Set*.

Type	System	Frontal Faces		15° Angle		22.5° Angle	
	Number of Images		1001		241		378
	Number of Faces		1001		241		378
	Number of Windows		255129875		61424875		96342750
		# miss /	Detect rate	# miss /	Detect rate	# miss /	Detect rate
Type	System	False detects /	Rate	False detects /	Rate	False detects /	Rate
One network, no heuristics	1) Net 1 (2 copies of hidden units, 2905 connections)	5	99.5%	1	99.6%	8	97.9%
		1743	1/146373	446	1/137723	812	1/118648
	2) Net 2 (3 copies of hidden units, 4357 connections)	5	99.5%	0	100.0%	11	97.1%
		1466	1/174031	489	1/125613	614	1/156910
	3) Net 3 (2 copies of hidden units, 2905 connections)	4	99.6%	1	99.6%	8	97.9%
		1209	1/211025	365	1/168287	604	1/159507
	4) Net 4 (3 copies of hidden units, 4357 connections)	6	99.4%	0	100.0%	15	96.0%
		1618	1/157682	471	1/130413	733	1/131436
One network, with heuristics	5) Network 1 → threshold(4,1) → overlap	5	99.5%	1	99.6%	11	97.1%
		572	1/446031	127	1/483660	247	1/390051
	6) Network 2 → threshold(4,1) → overlap	5	99.5%	0	100.0%	12	96.8%
		433	1/589214	117	1/524998	131	1/735440
	7) Network 3 → threshold(4,1) → overlap	5	99.5%	1	99.6%	10	97.4%
		379	1/673165	75	1/818998	135	1/713650
	8) Network 4 → threshold(4,1) → overlap	7	99.3%	0	100.0%	16	95.8%
		514	1/496361	107	1/574064	193	1/499185
Arbitrating among two networks	9) Nets 1 and 2 → AND(0)	13	98.7%	1	99.6%	20	94.7%
		290	1/879758	102	1/602204	162	1/594708
	10) Nets 1 and 2 → AND(0) → threshold(4,3) → overlap	19	98.1%	1	99.6%	26	93.1%
		2	1/127564937	1	1/61424875	2	1/48171375
	11) Nets 1 and 2 → threshold(4,2) → overlap → AND(2)	8	99.2%	1	99.6%	20	94.7%
		9	1/28347763	2	1/30712437	3	1/32114250
	12) Nets1,2→threshold(4,2)→overlap →OR(4)→threshold(4,1)→overlap	3	99.7%	0	100.0%	11	97.1%
		125	1/2041039	36	1/1706246	55	1/1751686
Arbitrating among three networks	13) Nets 1,2,3 → voting(0) → overlap	7	99.3%	2	99.2%	14	96.3%
		46	1/5546301	10	1/6142487	20	1/4817137
	14) Nets 1,2,3 → NN (5 hidden units) → threshold(4,1) → overlap	13	98.7%	1	99.6%	20	94.7%
		4	1/63782468	2	1/30712437	2	1/48171375
	15) Nets 1,2,3 → NN (10 hidden units) → threshold(4,1) → overlap	16	98.4%	1	99.6%	21	94.4%
		4	1/63782468	1	1/61424875	2	1/48171375
	16) Nets 1,2,3 → NN (perceptron) → threshold(4,1) → overlap	16	98.4%	1	99.6%	23	93.9%
		3	1/85043291	1	1/61424875	2	1/48171375

3.5.3 Example Output

Based on the results shown in Tables 3.13 and 3.15, both Systems 11 and 15 make acceptable tradeoffs between the number of false detections and the detection rate. Because System 11 is less complex than System 15 (using only two networks rather than a total of four), it is preferable. System 11 detects on average 86.0% of the faces, with an average of one false detection per $2,680,619$ 20×20 pixel windows examined in the *Upright Test Set*. Figs. 3.16, 3.17, and 3.18 show example output images from System 11 on images from the *Upright Test Set*[1].

3.5.4 Effect of Exhaustive Training

All of the experiments presented so far have used the active training algorithm of Section 3.2.3. In this section, we examine the performance of exhaustively training the algorithm on all available nonface images, as described in Section 3.2.4. As before, I trained two networks, and tested them independently and with arbitration on the *Upright Test Set*. The results are shown in Table 3.19. As can be seen, the results are not as good as those of the active learning algorithm (in Table 3.13. The false alarm rate is significantly lower, but it is unable to detect as many faces. This may be due in part to a poor estimate of the prior probability of faces in images.

An alternative to combining the two outputs using arbitration heuristics is to averages the two probability estimates. Assuming that the two algorithms which produced the estimates are independent and that the two algorithms are unbiased, the average estimator will have a lower variance; in other words it should be more accurate. The result of averaging the two networks of Table 3.19 is shown in Table 3.20. As can be seen from this table, the accuracy is comparable with the arbitration heuristics used earlier.

The results from doing exhaustive training on the scenery data look promising, but are not yet as good as the active learning method. This may be in part due to insufficient training of the networks, caused by the large memory and computational requirements of exhaustive training. Throughout the rest of this thesis, only the active learning scheme will be used.

3.5.5 Effect of Lighting Variation

Section 2.6 discussed methods to use linear lighting models of faces to explicitly compensate for variations in lighting conditions before attempting to detect a face. These models can also be used

[1] After painstakingly trying to arrange these images compactly by hand, we decided to use a more systematic approach. These images were laid out automatically by the PBIL optimization algorithm [Baluja, 1994]. The objective function tries to pack images as closely as possible, by maximizing the amount of space left over at the bottom of each page.

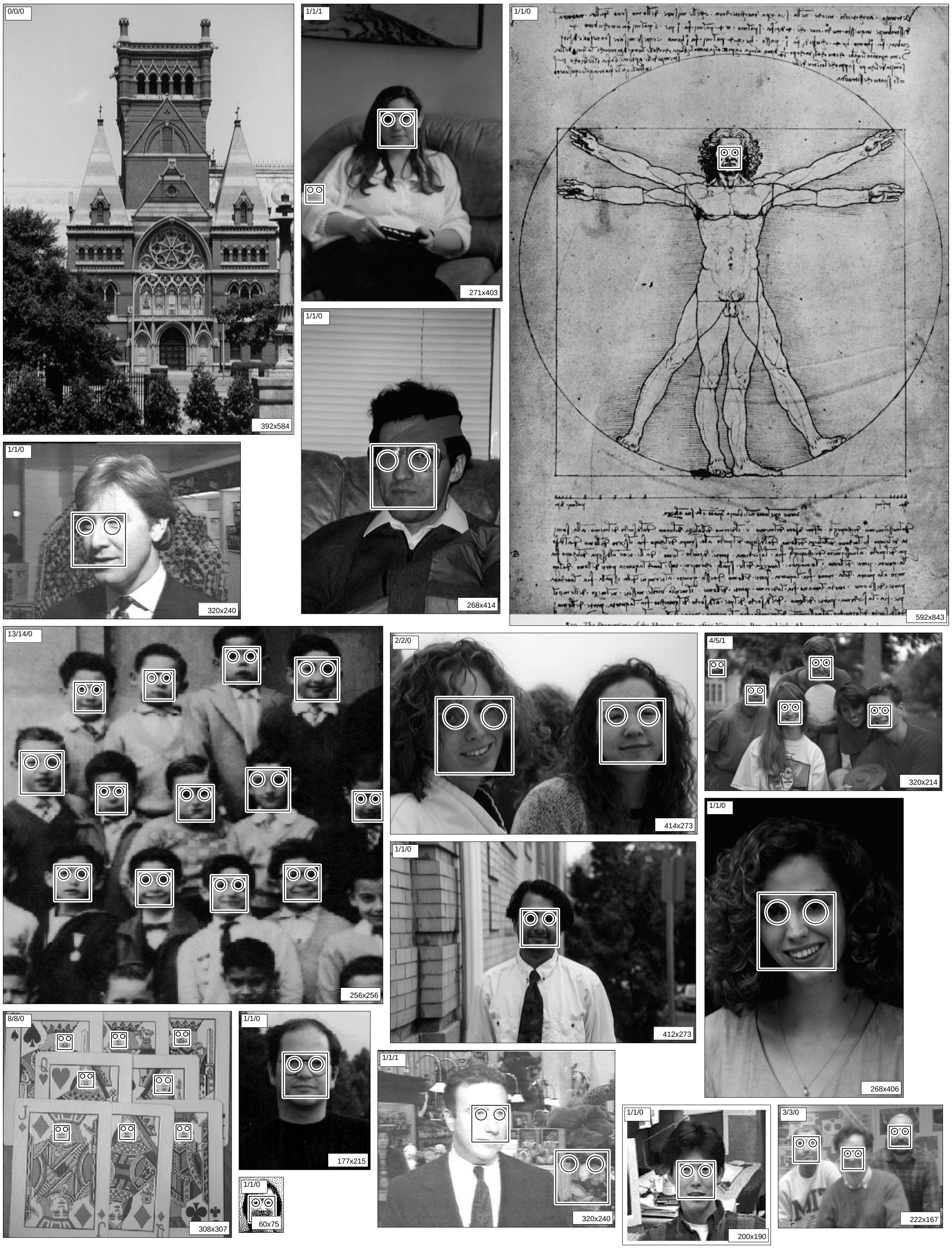


Figure 3.16: Output from System 11 in Table 3.13. The label in the upper left corner of each image (D/T/F) gives the number of faces detected (D), the total number of faces in the image (T), and the number of false detections (F). The label in the lower right corner of each image gives its size in pixels.

Figure 3.17: Output obtained in the same manner as the examples in Figure 3.16.

Figure 3.18: Output obtained in the same manner as the examples in Figure 3.16.

Table 3.19: Detection and error rates two networks trained exhaustively on all the scenery data, for the *Upright Test Set*.

Type	System	Missed faces	Detect rate	False detects	False detect rate
One network, no heuristics	1) Network 1 (2 copies of hidden units (52 total), 2905 connections)	86	83.0%	703	1/118206
	2) Network 2 (3 copies of hidden units (78 total), 4357 connections)	97	80.9%	120	1/692493
One network, with heuristics	5) Network 1 → threshold(4,1) → overlap	93	81.7%	312	1/266343
	6) Network 2 → threshold(4,1) → overlap	100	80.3%	68	1/1222047
Arbitrating among two networks	9) Networks 1 and 2 → AND(0)	129	74.6%	80	1/1038740
	10) Networks 1 and 2 → AND(0) → threshold(4,3) → overlap	166	67.3%	4	1/20774802
	11) Networks 1 and 2 → threshold(4,2) → overlap → AND(4)	147	71.0%	5	1/16619842
	12) Networks 1 and 2 → threshold(4,2) → overlap → OR(4) → threshold(4,1) → overlap	99	80.5%	103	1/806788

Table 3.20: Detection and error rates resulting from averaging the outputs of two networks trained exhaustively on all the scenery data, for the *Upright Test Set*.

System	Missed faces	Detect rate	False detects	False detect rate
1) Network 1 (2 copies of hidden units (52 total), 2905 connections)	122	75.9%	52	1/1598061
5) Network 1 → threshold(4,1) → overlap	123	75.7%	25	1/3323968

to generate training data for a face detector, so that the neural network can implicitly learn to handle lighting variation.

Using lighting models of a total of 27 faces collected at CMU, I generated a training database containing 100 examples of each face, with random lighting conditions, in addition to the usual small variations in the scale, angle, and center location of the face. The result of training two networks on these images using the active learning scheme, and testing on the *Upright Test Set*, are shown in Table 3.21. Given the small number of lighting models available, we would expect that the performance would not be comparable with the networks trained on a large number of faces (as in Table 3.13). The fact that this network is able to detect approximately 50% of the faces is quite surprising; it suggests that much of the variation in the appearance of faces can be accounted for by lighting conditions. Note that the *Upright Test Set* was not selected specifically to test the tolerance of lighting variation.

Table 3.21: Detection and error rates two networks trained with images generated from lighting models, for the *Upright Test Set*.

Type	System	Missed faces	Detect rate	False detects	False detect rate
One network, no heuristics	1) Network 1 (2 copies of hidden units (52 total), 2905 connections)	142	72.0%	2656	1/31287
	2) Network 2 (3 copies of hidden units (78 total), 4357 connections)	156	69.2%	1278	1/65022
One network, with heuristics	5) Network 1 → threshold(4,1) → overlap	156	69.2%	1521	1/54634
	6) Network 2 → threshold(4,1) → overlap	165	67.5%	845	1/98342
Arbitrating among two networks	9) Networks 1 and 2 → AND(0)	242	52.3%	116	1/716372
	10) Networks 1 and 2 → AND(0) → threshold(4,3) → overlap	296	41.6%	4	1/20774802
	11) Networks 1 and 2 → threshold(4,2) → overlap → AND(4)	251	50.5%	20	1/4154960
	12) Networks 1 and 2 → threshold(4,2) → overlap → OR(4) → threshold(4,1) → overlap	160	68.4%	374	1/222190

For completeness, I also trained two neural networks on the 27 lighting model faces, but this time only with frontal lighting for each model. Again, 100 variations of each face were generated, with slightly randomized translation, scale, and orientation. The results on the *Upright Test Set* are shown in Table 3.22. As can be seen, the accuracy is much smaller than the networks trained with lighting variation, again suggesting the importance of lighting variation in the face detection problem.

3.6 Summary

The algorithm presented in this chapter can detect between 81.9% and 90.1% of faces in a set of 130 test images with cluttered backgrounds, with an acceptable number of false detections. Depending on the application, the system can be made more or less conservative by varying the arbitration heuristics or thresholds used. The system has been tested on a wide variety of images, with many

Table 3.22: Detection and error rates two networks trained with images with frontal lighting only, for the *Upright Test Set*.

Type	System	Missed faces	Detect rate	False detects	False detect rate
One network, no heuristics	1) Network 1 (2 copies of hidden units (52 total), 2905 connections)	408	19.5%	226	1/367695
	2) Network 2 (3 copies of hidden units (78 total), 4357 connections)	430	15.2%	161	1/516144
One network, with heuristics	5) Network 1 → threshold(4,1) → overlap	408	19.5%	195	1/426149
	6) Network 2 → threshold(4,1) → overlap	433	14.6%	134	1/620143
Arbitrating among two networks	9) Networks 1 and 2 → AND(0)	463	8.7%	10	1/8309921
	10) Networks 1 and 2 → AND(0) → threshold(4,3) → overlap	487	3.9%	1	1/83099211
	11) Networks 1 and 2 → threshold(4,2) → overlap → AND(4)	481	5.1%	2	1/41549605
	12) Networks 1 and 2 → threshold(4,2) → overlap → OR(4) → threshold(4,1) → overlap	439	13.4%	28	1/2967828

faces and unconstrained backgrounds. I have also shown the effects of using an exhaustive training algorithm (for negative examples) and the effect of using a lighting model to generate synthetic positive examples. In the next chapter, this technique is extended to faces which are tilted in the image plane. Chapter 6 will return to the algorithm of this chapter, and present techniques on how to make it run faster.

Chapter 4

Tilted Face Detection

4.1 Introduction

In demonstrating the system described in the previous chapter, the people watching the demonstration would expect faces to be detected at any angle, as shown in Figure 4.1. In this chapter, we present some modifications to the upright face detection algorithm to detect such tilted faces. This system efficiently detects frontal faces which can be arbitrarily rotated within the image plane.

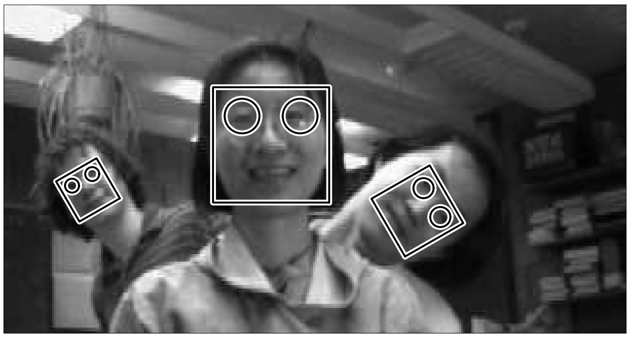


Figure 4.1: People expect face detection systems to detect rotated faces. Overlaid is the output of the system to be presented in this chapter.

There are many ways to use neural networks for rotated-face detection. The simplest would be to employ the upright face detection, by repeatedly rotating the input image in small increments and applying the detector to each rotated image. However, this would be an extremely computationally expensive procedure. The system described in the previous chapter is invariant to approximately 10° of tilt from upright (both clockwise and counterclockwise). Therefore, the entire detection procedure would need to be applied *at least* 18 times to each image, with the image rotated in increments of 20°.

An alternate, significantly faster procedure is described in this chapter, extending some early results in [Baluja, 1997]. This procedure uses a separate neural network, termed a "derotation network", to analyze the input window before it is processed by the face detector. The derotation network's input is the same region that the detector network will receive as input. If the input contains a face, the derotation network returns the angle of the face. The window can then be "derotated" to make the face upright. Note that the derotation network *does not* require a face as input. If a nonface is encountered, the derotator will return a meaningless rotation. However, since a rotation of a nonface will yield another nonface, the detector network will still not detect a face. On the other hand, a rotated face, which would not have been detected by the detector network alone, will be rotated to an upright position, and subsequently detected as a face. Because the detector network is only applied once at each image location, this approach is significantly faster than exhaustively trying all orientations.

Detailed descriptions of the algorithm are given in Section 4.2. We then analyze the performance of each part of the system separately in Section 4.3, and test the complete system on three large test sets in Section 4.4. We will see that the system is able to detect 79.6% of the faces over the *Upright Test Set* and *Tilted Test Set*, with a very small number of false positives.

4.2 Algorithm

The overall structure of the algorithm, shown in Figure 4.2, is quite similar to the one presented in the previous chapter. Starting from the input image, an image pyramid is built, with scaling steps of 1.2. 20×20 pixel windows are extracted from every position and scale in this input pyramid, and passed to a classifier.

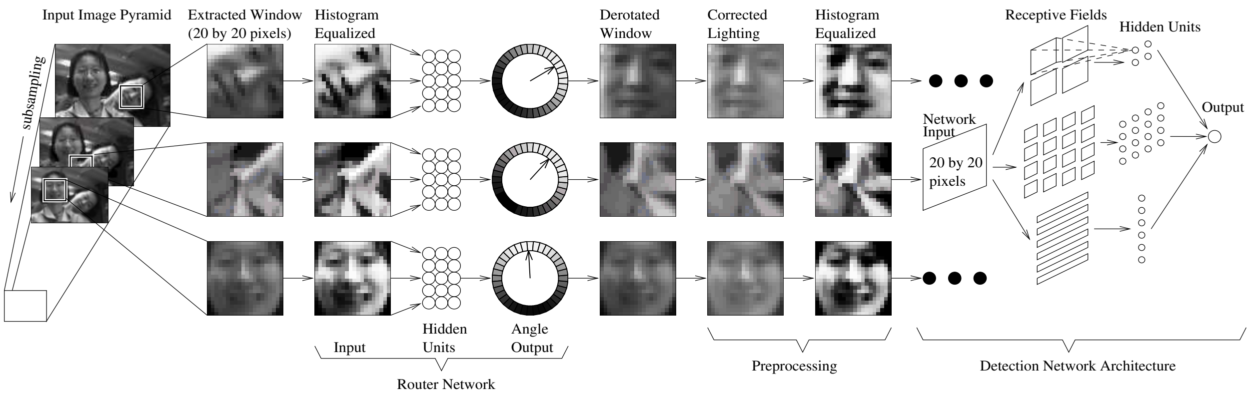


Figure 4.2: Overview of the algorithm.

First, the window is preprocessed using histogram equalization (Section 2.5), and then given to a *derotation network*. The tilt angle returned by the derotation network is then used to rotate the window with the potential face to an upright position. Finally, the *derotated window* is preprocessed with linear lighting correction and histogram equalization, and then passed to one or more upright face detection network, like those in the previous chapter, which decide whether or not the window contains a face.

The system as presented so far could easily signal that there are two faces of very different orientations at adjacent pixel locations in the image. To counter such anomalies, and to reinforce correct detections, clean up heuristics and multiple detection networks are employed. The design of the derotation network and the heuristic arbitration scheme are presented in the following subsections.

4.2.1 Derotation Network

The first step in processing a window of the input image is to apply the derotation network. This network assumes that its input window contains a face, and is trained to estimate its orientation. The inputs to the network are the intensity values in a 20×20 pixel window of the image (which have been preprocessed by histogram equalization, Section 2.5). The output angle of rotation is represented by an array of 36 output units, in which each unit i represents an angle of $i * 10°$. To signal that a face is at an angle of θ, each output is trained to have a value of $\cos(\theta - i * 10°)$. This approach is closely related to the Gaussian weighted outputs used in the autonomous driving domain [Pomerleau, 1992]. Examples of the training data are given in Figure 4.3.

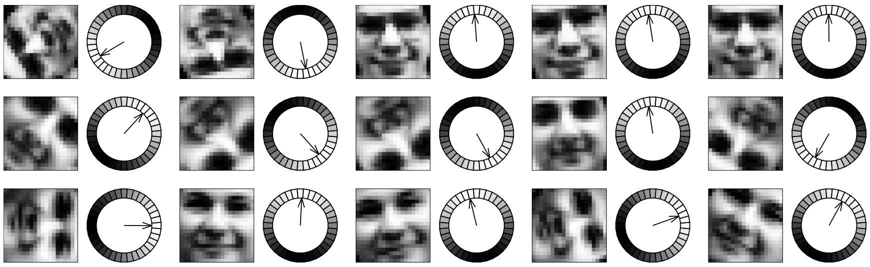


Figure 4.3: Example inputs and outputs for training the derotation network.

Previous algorithms using Gaussian weighted outputs inferred a single value from them by computing an average of the positions of the outputs, weighted by their activations. For angles, which have a periodic domain, a weighted sum of angles is insufficient. Instead, we interpret each output as a weight for a vector in the direction indicated by the output number i, and compute a weighted sum as follows:

$$\left(\sum_{i=0}^{35} \text{output}_i * \cos(i * 10°), \sum_{i=0}^{35} \text{output}_i * \sin(i * 10°)\right)$$

The direction of this average vector is interpreted as the angle of the face.

As with the upright face detector, the training examples are generated from a set of manually labelled example images containing 1048 faces. After each face is aligned to the same position, orientation, and scale, they are rotated to a random known orientation to generate the training example. Note that the training examples for the upright detector had small random variations in scale and position for robustness; the derotation network performed better without these variations.

.

The architecture for the derotation network consists of four layers: an input layer of 400 units, two hidden layers of 15 units each, and an output layer of 36 units. Each layer is fully connected to the next. Each unit uses a hyperbolic tangent activation function, and the network is trained using the standard error backpropogation algorithm.

4.2.2 Detector Network

After the derotation network has been applied to a window of the input, the window is derotated to make any face that may be present upright. Because the input window for the derotation network and detection network are both 20×20 square windows, and are an angle with respect to one another, their edges may not overlap. Thus the derotation must resample the original input image.

The remaining task is to decide whether or not the window contains an upright face. For this step, we used the algorithm presented in the previous chapter.The resampled image, is preprocessed using the linear lighting correction and histogram equalization procedures described in Section 2.5. The window is then passed to the detector, which is trained to produce 1 for faces, and -1 for nonfaces. The detector has two sets of training examples: images which are faces, and images which are not. The positive examples are generated in a manner similar to that of the derotation network; however, the amount of rotation of the training images is limited to the range $-10°$ to $10°$.

Some examples of nonfaces that are collected during training were shown in Figure 3.2. At runtime, the detector network will be applied to images which have been derotated, so it may be advantageous to collect negative training examples from the set of derotated nonface images, rather than only nonface images in their original orientations. In Section 4.4, both possibilities are explored.

4.2.3 Arbitration Scheme

As mentioned earlier, it is possible for the system described so far to signal faces of very different orientations at adjacent pixel locations. As with the upright detector, we use some simple clean-up and arbitration heuristics to improve the results. These heuristics are restated below, with the changes necessary for handling rotation angles in addition to positions and scales. Each detection is first placed in a 4-dimensional space, where the dimensions are the x and y positions of the center of the face, the scale in the image pyramid at which the face was detected, and the angle of the face, quantized in increments of $10°$. For each detection, we count the number of detections within 4 units along each dimension (4 pixels, 4 pyramid scales, or $40°$). This number can be interpreted as a confidence measure, and a threshold is applied. As before, this heuristic is denoted *threshold(distance,level)*. Once a detection passes the threshold, any other detections in the 4-dimensional space which would overlap it are discarded. This step is called *overlap* in the experiments section.

To further reduce the number of false detections, and reinforce correct detections, we arbitrate between two independently trained detector networks. To use the outputs of these two networks, the postprocessing heuristics of the previous paragraph are applied to the outputs of each individual network, and then the detections from the two networks are ANDed. The specific preprocessing thresholds used in the experiments will be given in Sections 4.4.

4.3 Analysis of the Networks

In order for the system described above to be accurate, the derotator and detector must perform robustly and compatibly. Because the output of the derotator network is used to normalize the input for the detector, the angular accuracy of the derotator must be compatible with the angular invariance of the detector. To measure the accuracy of the derotator, we generated test example images based on the training images, with angles between $-30°$ and $30°$ at $1°$ increments. These images were given to the derotation network, and the resulting histogram of angular errors is given in Figure 4.4 (left). As can be seen, 92% of the errors are within $\pm 10°$.

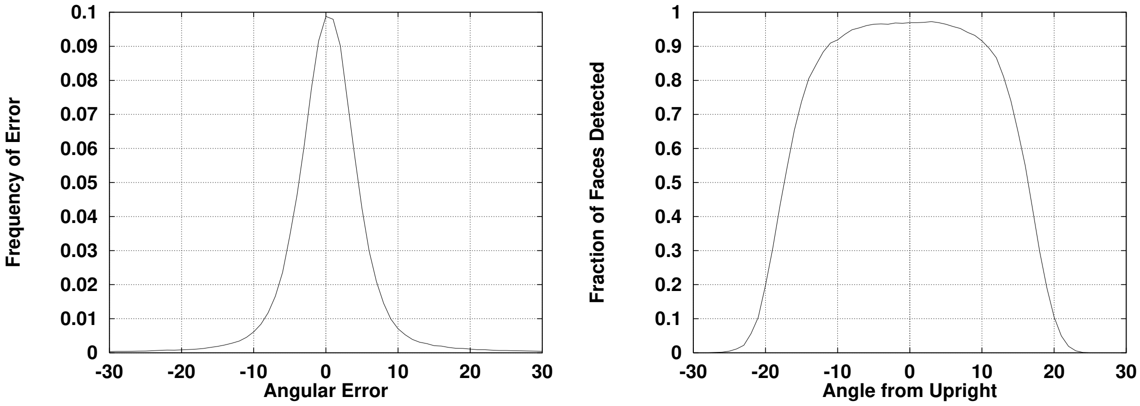


Figure 4.4: Left: Frequency of errors in the derotation network with respect to the angular error (in degrees). Right: Fraction of faces that are detected by a detection network, as a function of the angle of the face from upright.

The detector network was trained with example images having orientations between -10° and 10°. It is important to determine whether the detector is in fact invariant to rotations within this range. We applied the detector to the same set of test images as the derotation network, and measured the fraction of faces which were correctly classified as a function of the angle of the face. Figure 4.4 (right) shows that the detector detects over 90% of the faces that are within 10° of upright, but the accuracy falls with larger angles.

Since the derotation network's angular errors are usually within 10°, and since the detector can detect most faces which are rotated up to 10°, the two networks should be compatible.

Just as we noted in the previous section that the detector network is applied only to nonfaces which have been derotated, the same observation can be made about faces. The derotation network does make some mistakes, but those mistakes may be *systematic*; in this case the detector may be able to exploit this to produce more accurate results. This idea will be tested in the experiments section.

4.4 Evaluation

4.4.1 Tilted Test Set

In this section, we integrate the pieces of the system, and test it on three sets of images. The first set is the *Upright Test Set* used in the previous chapter. It contains many images with faces against complex backgrounds and many images without any faces. There are a total of 130 images, with 511 frontal faces (of which 469 are within 10° of upright), and 83,099,211 windows to be processed. The second test set is the *FERET Test Set*, partitioned into three classes based on how far the face is from frontal.

Figure 4.5: Example images in the *Tilted Test Set* for testing the tilted face detector.

To evaluate a version of the system which could detect faces that are tilted in the image, we collected a third set of images to exercise this part of the detector. These were collected from the same variety of sources as the *Upright Test Set*. A few examples are shown in Figure 4.5. The test set contains 50 images, and requires the networks to examine 34,064,635 20×20 pixel windows. Of the 223 faces in this set, 210 are at angles of more than $10°$ from upright. In the following sections, this test set will be called the *Tilted Test Set*.

The *Upright Test Set* and *FERET Test Set* are used as a baseline for comparison with the previous chapter. They will ensure that the modifications for rotated faces do not hamper the ability to detect upright faces. The *Tilted Test Set* will demonstrate the new capabilities of our system. Figure 4.6 shows the distributions of the angles of faces in each test set; as can be seen, most of the faces in the first two sets are very close to upright, while the last has more tilted faces. The peak for the tilted test set at $30°$ is due to a large image with 135 upright faces that was rotated to an angle of $30°$, as can be seen in Figure 4.9.

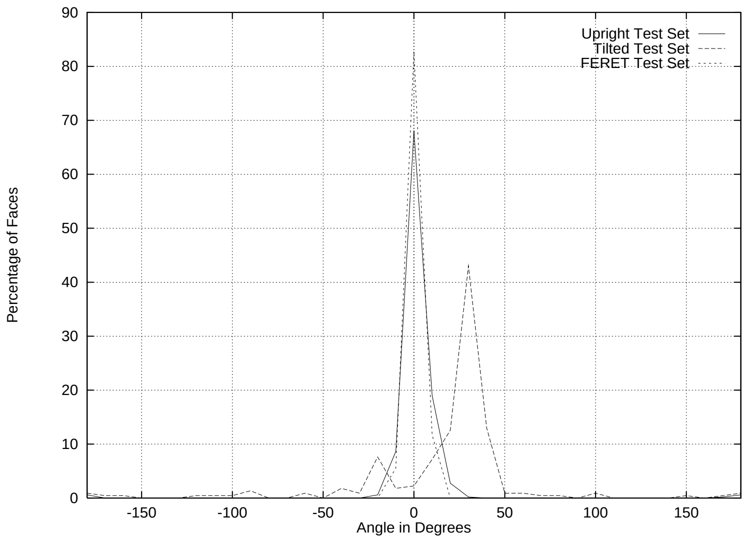


Figure 4.6: Histograms of the angles of the faces in the three test sets used to evaluate the tilted face detector. The peak for the tilted test set at $30°$ is due to a large image with 135 upright faces that was rotated to an angle of $30°$, as can be seen in Figure 4.9.

Knowledge of the distribution of faces in particular applications may allow the detector to be simplified. In particular, faces rotated more than $45°$ may be quite rare in images on the WWW, so the derotation can be customized to a smaller range of angles, and possibly be more accurate. On the other hand, a digital photograph manager might use face angles to determine whether a photograph was taken with the camera in a horizontal or vertical orientation. For this application, the detector must locate faces at any angle.

4.4.2 Derotation Network with Upright Face Detectors

The first system we test employs the derotation network to determine the orientation of any potential face, and then applies two upright face detection networks from the previous chapter, Networks 1 and 2. Table 4.7 shows the number of faces detected and the number of false alarms generated on the three test sets. We first give the results of the individual detection networks, and then give the results of the post-processing heuristics (using a threshold of one detection). The last row of the table reports the result of arbitrating the outputs of the two networks, using an AND heuristic. This is implemented by first post-processing the outputs of each individual network, followed by requiring that both networks signal a detection at the same location, scale, and orientation. As can be seen in the table, the post-processing heuristics significantly reduce the number of false detections, and arbitration helps further. Note that the detection rate for the *Tilted Test Set* is higher than that for the *Upright Test Set*, due to differences in the overall difficulty of the two test sets.

Table 4.7: Results of first applying the derotation network, then applying the standard upright detector networks.

	Upright Test Set		Tilted Test Set	
System	Detect %	# False	Detect %	# False
Network 1	89.6%	4835	91.5%	2174
Network 2	87.5%	4111	90.6%	1842
Net 1 → threshold(4,1) → overlap	85.7%	2024	89.2%	854
Net 2 → threshold(4,1) → overlap	84.1%	1728	87.0%	745
Nets 1,2 → threshold(4,1) → overlap → AND(4) → overlap	81.6%	293	85.7%	119

4.4.3 Proposed System

Table 4.7 shows a significant number of false detections. This is in part because the detector networks were applied to a different distribution of images than they were trained on. In particular, at runtime, the networks only saw images that were derotated. We would like to match this distribution as closely as possible during training. The positive examples used in training are already in upright positions, and barring any systematic errors in the derotator network, have an approximately correct distribution. During training, we can also run the scenery images from which negative examples are collected through the derotator. We trained two new detector networks using this scheme, and their performance is summarized in Table 4.8. As can be seen, the use of these new networks reduces the number of false detections by at least a factor of 4. The detect rate has also dropped, because now the detector networks must deal with nonfaces derotated to look as much like faces as possible. This makes the detection problem harder, and the detection networks more conservative. Of the systems presented in this chapter, this one has the best trade-off between the detection rate and the number of false detections. Images with the detections resulting from arbitrating between the networks are given in Figure 4.9.

Table 4.8: Results of the proposed tilted face detection system, which first applies the derotator network, then applies detector networks trained with derotated negative examples.

	Upright Test Set		Tilted Test Set	
System	Detect %	# False	Detect %	# False
Network 1	81.0%	1012	90.1%	303
Network 2	83.2%	1093	89.2%	386
Network 1 → threshold(4,1) → overlap	80.2%	710	89.2%	221
Network 2 → threshold(4,1) → overlap	82.4%	747	88.8%	252
Networks 1 and 2 → threshold(4,1) → overlap → AND(4)	76.9%	44	85.7%	15

This system was also applied to the *FERET Test Set*, used to evaluate the upright face detector in the previous chapter. The results are shown in Table 4.10. The general pattern of the results is similar to that of the upright detector in Table 3.15, although the detection rates are slightly lower.

This idea can be carried a step further, to training the detection networks on face examples which have been derotated by the derotation network. If there are any systematic errors made by the derotation network (for example, faces looking slightly to one side might have a consistent error in their angles), the detection network might be able to take advantage of this, and produce better detection results. The results of this training procedure are shown in Figure 4.11. As can be seen, the detection rates are somewhat lower, and the false alarm rates are significantly lower.

One hypothesis for why this happens is as follows: For robustness, the previous detector networks were trained with face images including small amounts of rotation, translation and scaling. However, since the derotation network was more accurate without such variations, it was trained without them. In this experiment, the positive examples had these sources of variation removed. The scale and translation was removed when the randomly rotated faces are created, while the rotation variation is removed the the derotation network. This may have made the detector somewhat brittle to small variations in the faces. However, at the same time it makes the set of face images that must be accepted smaller, making it easier to discard nonfaces.

An alternative hypothesis is that the errors made by the derotation network are not systematic enough to be useful. Instead, perhaps they introduce more variability into the face images. Because

Figure 4.9: Result of arbitrating between two networks trained with derotated negative examples. The label in the upper left corner of each image (D/T/F) gives the number of faces detected (D), the total number of faces in the image (T), and the number of false detections (F). The label in the lower right corner of each image gives its size in pixels.

Table 4.10: Results of the proposed tilted face detection system, which first applies the derotator network, then applies detector networks trained with derotated negative examples. These results are for the *FERET Test Set*.

System	FERET Frontal		FERET 15°		FERET 22.5°	
	Detect %	# False	Detect %	# False	Detect %	# False
Network 1	97.7%	1567	99.2%	388	95.2%	620
Network 2	97.7%	1616	99.2%	413	94.1%	671
Net 1 → threshold(4,1) → overlap	97.6%	898	99.2%	209	94.9%	383
Net 2 → threshold(4,1) → overlap	97.7%	867	99.2%	234	93.0%	373
Nets 1,2 → threshold(4,1) → overlap → AND(4) → overlap	97.2%	17	99.2%	3	92.0%	12

of the random error in the recovery of the angle, important facial features are no longer at consistent locations in the input window, making the detection problem itself harder. This hypothesis does not explain the lower false alarm rate, however. Both of these hypotheses deserve further exploration.

Table 4.11: Result of training the detector network on both derotated faces and nonfaces.

System	Upright Test Set		Tilted Test Set	
	Detect %	# False	Detect %	# False
Network 1	67.3%	294	75.8%	109
Network 2	69.9%	341	79.4%	102
Network 1 → threshold(4,1) → overlap	66.9%	245	75.3%	89
Network 2 → threshold(4,1) → overlap	69.1%	278	79.4%	88
Networks 1 and 2 → threshold(4,1) → overlap → AND(4)	61.1%	10	71.7%	6

4.4.4 Exhaustive Search of Orientations

To demonstrate the effectiveness of the derotation network for rotation invariant detection, we applied the two sets of detector networks described above without the derotation network. The detectors were instead applied at 18 different orientations (in increments of 20°) for each image location. We expect such systems to detect most rotated faces. However, assuming that errors occur independently, we may also expect many more false detections than the systems presented above. Table 4.12 shows the results using the upright face detection networks from the previous chapter, and Table 4.13 shows the results using the detection networks trained with derotated negative examples.

Table 4.12: Results of applying the upright detector networks from the previous chapter at 18 different image orientations.

	Upright Test Set		Tilted Test Set	
System	Detect %	# False	Detect %	# False
Network 1	93.7%	17848	96.9%	7872
Network 2	94.7%	15828	95.1%	7328
Network 1 → threshold(4,1) → overlap	87.5%	4828	94.6%	1928
Network 2 → threshold(4,1) → overlap	89.8%	4207	91.5%	1719
Networks 1 and 2 → threshold(4,1) → overlap → AND(4)	85.5%	559	90.6%	259

Table 4.13: Networks trained with derotated examples, but applied at all 18 orientations.

	Upright Test Set		Tilted Test Set	
System	Detect %	# False	Detect %	# False
Network 1	90.6%	9140	97.3%	3252
Network 2	93.7%	7186	95.1%	2348
Network 1 → threshold(4,1) → overlap	86.9%	3998	96.0%	1345
Network 2 → threshold(4,1) → overlap	91.8%	3480	94.2%	1147
Networks 1 and 2 → threshold(4,1) → overlap → AND(4)	85.3%	195	92.4%	67

Recall that Table 4.7 showed a larger number of false positives compared with Table 4.8, due to differences in the training and testing distributions. In Table 4.7, the detection networks were trained with false-positives in their original orientations, but were tested on images that were rotated from their original orientations. Similarly, if we apply these detector networks to images at all 18 orientations, we should expect an increase in the number of false positives because of the differences in the training and testing distributions (see Tables 4.12 and 4.13). The detection rates are higher than for systems using the derotation network. This is because any error by the derotator network will lead to a face being missed, whereas an exhaustive search of all orientations may find it. Thus, the differences in accuracy can be viewed as a tradeoff between the detection and false detection rates, in which better detection rates come at the expense of much more computation.

4.4.5 Upright Detection Accuracy

Finally, to check that adding the capability of detecting rotated faces has not come at the expense of accuracy in detecting upright faces, in Table 4.14 we present the result of applying the original detector networks and arbitration method from Chapter 3 to the three test sets used in this chapter. The results for the *Upright Test Set* are slightly different from those presented in the previous chapter because we now check for the detection of 4 upside-down faces, which were present, but ignored, in the previous chapter. As expected, the upright detector does well on the *Upright* and FERET Test Sets, but has a poor detection rate on the *Tilted Test Set*.

Table 4.14: Results of applying the upright algorithm and arbitration method from the previous chapter to the test sets.

	Upright Test Set		Tilted Test Set	
System	Detect %	# False	Detect %	# False
Network 1	90.6%	928	20.6%	380
Network 2	92.0%	853	19.3%	316
Network 1 → threshold(4,1) → overlap	89.4%	516	20.2%	259
Network 2 → threshold(4,1) → overlap	90.6%	453	17.9%	202
Networks 1 and 2 → threshold(4,2) → overlap → AND(4)	85.3%	31	13.0%	11

Table 4.15 shows a breakdown of the detection rates of the above systems on faces that are rotated less or more than 10° from upright, in the *Upright Test Set* and *Tilted Test Set*. As expected, the upright face detector trained exclusively on upright faces and negative examples in their original orientations gives a high detection rate on upright faces. The tilted face detection system has a slightly lower detection rate on upright faces for two reasons. First, the detector networks cannot recover from all the errors made by the derotation network. Second, the detector networks which are trained with derotated negative examples are more conservative in signalling detections; this is because the derotation process makes the negative examples look more like faces, which makes the classification problem harder.

Another way to breakdown the results of the tilted face detector is to look at how each of the two stages, the derotation stage and the detection stage, contribute to the detection rate. To measure this, we extract the 20×20 windows in the test sets which contain a face, and compute the derotation angle using two methods: the neural network, and the alignment method used to prepare the training data for this network. By comparing the results of these two methods, we can see how accurate the derotation network is on an independent test set. Next, the faces derotated by these two methods can be passed to the detection network, whose detection rates can be measured

Table 4.15: Breakdown of detection rates for upright and rotated faces from the test sets.

System	All Faces	Upright Faces ($\leq 10°$)	Rotated Faces ($> 10°$)
Tilted detector (Table 4.8)	79.6%	77.2%	84.1%
Upright detector (Chapter 3)	63.4%	88.0%	16.3%

for these two cases. The results of these comparisons, for the *Upright* and *Tilted Test Sets* are shown in Table 4.16.

Table 4.16: Breakdown of the accuracy of the derotation network and the detector networks for the tilted face detector.

		Test Sets	
Stage	Statistic	Upright	Tilted
Derotation network output	Angle within $10°$	69.3%	76.7%
Detector output	Manual derotation	61.4%	58.7%
	Automatic derotation	49.3%	56.5%
Complete system	Detection rate	76.9%	85.7%

As can be seen from the table, there is a between 2% and 12% penalty for using the neural network to derotate the images, relative to derotating them by hand. This penalty partly explains the decrease in the detection rate compared with the upright detector. The table shows only the detection rates when applying a single detector network at a single pixel location and scale in the image. In practice, the detectors are applied at every pixel location and scale, giving them more opportunities to find each face. This explains the higher detection rates of the complete system (the last line in Table 4.16 relative to the earlier lines.

4.5 Summary

This chapter has demonstrated the effectiveness of detecting faces rotated in the image plane by using a derotation network in combination with an upright face detector. The system is able to detect 79.6% of faces over several large test sets, with a small number of false positives. The technique is applicable to other template-based object detection schemes. The next chapter will examine some techniques for detecting faces that are rotated out of the image plane.

Chapter 5

Non-Frontal Face Detection

5.1 Introduction

The previous chapter presented a two stage face detection algorithm, in which first analyzes the angle of a potential face, then uses this information to geometrically normalize that part of the image for the detector itself. The same idea can be applied in the more general context of detecting faces rotated out of the image plan. There are two ways in which this could be approached. The first is directly analogous to the approach for tilted faces: by using knowledge of the shape and symmetry of human faces, image processing operations may be able to convert to a profile or half-profile view of a face to a frontal view. A second approach, and the one we have explored in more detail, is to partition the views of the face, and to train separate detector networks for each view. We used five views: left profile, left semi-profile, frontal, right semi-profile, and right profile. A pose estimator is responsible for directing the input window to one of these view-specific detectors, similar to the idea presented in [Zhang and Fulcher, 1996]. The work presented here only handles two degrees of freedom of rotation: rotation in the image plane, and rotation to the left or right out of the plane. Extending the algorithm to faces rotated up or down should be straightforward.

This chapter in Section 5.2 begins with a discussion of how to estimate the three dimensional pose of a face given an input image, and how to use this pose information to geometrically distort the image to synthesize an upright, frontal view. We will see that the results of the procedure are not good enough for use in a face detector. Section 5.3 uses pose information to select a detector customized to a particular view of the face. Section 5.4 shows some evaluation of the method.

5.2 Geometric Distortion to a Frontal Face

To detect faces which are tilted in the image place, we simply applied some image processing operators to rotate each window to an upright orientation before applying the detector. If we have a 3D model of the head, and information about the orientation of the head in the image, we can use texture mapping to generate an upright, frontal image of the head. Similar techniques have been used in the past of generate frontal images of the face from partial profile views for the face recognition task [Vetter *et al.*, 1997, Beymer *et al.*, 1993]. These techniques work quite well, but are quite computationally expensive, requiring an iterative optimization procedure to align each face with the model.

Our algorithm must compute the orientation of a potential face for every window of the image before running the detector. There is a fairly large literature on this problem, including approaches using eigenspaces [Nayar *et al.*, 1996, Pentland *et al.*, 1994], and those using three dimensional geometric models of the face [Horprasert *et al.*, 1997]. Because this algorithm must be applied so many times, a computationally less expensive technique such as a neural network is more appropriate. The following sections describe the training data for this network, how it was trained, and finally describe how texture mapping was used to synthesize an upright, frontal view of the face.

5.2.1 Training Images

To train the pose invariant face detector, I used two additional training databases beyond the frontal face images used in the previous two chapters. These two databases are described before.

FERET Images: The FERET image database was used in testing the upright face detector. However, the database also contains images from a wide variety of out of place rotations, and so it is useful for training this detector in this chapter.

NIST Images: The second database used to train the profile face detector is the NIST mugshot database. These images frontal and full profile mugshots against fairly uniform backgrounds. The database can be ordered from NIST at this location on the WWW: `http://www.nist.gov/srd/nistsd18.htm`. Henry Schneiderman hand segmented the faces in these images from their backgrounds for use in his work [Schneiderman and Kanade, 1998], and kindly allowed me to use his segmentation masks to generate training data.

5.2.2 Labelling the 3D Pose of the Training Images

There are several new issues that arise in creating the training data for this problem. As before, we begin by manually labelling important feature points in images. In the previous chapters, we aligned the faces with one another by performing a two-dimensional alignment, using translation, rotation, and uniform scaling to minimize the sum of squared distances between the labelled feature locations.

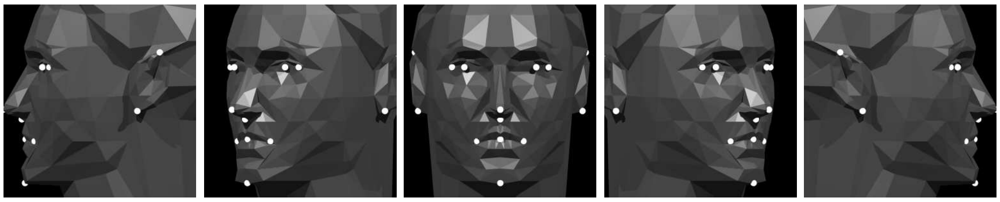


Figure 5.1: Generic three-dimensional head model used for alignment. The model itself is based on a 3D head model file, `head1.3ds` found on the WWW. The white dots are the labelled 3D feature locations.

Since we are now considering out-of-plane rotations, the faces must be aligned with one another in three-dimensions. However, we are given only a two-dimensional representation of each face, and two-dimensional feature locations. We begin with a three-dimensional model of a generic face, shown in Figure 5.1. The feature locations used to label the face images are labelled in 3D on the model. Then, we attempt to find the best three-dimensional rotation, scaling, and translation of the model which, under an orthographic projection, best matches each face. A perspective projection could also be used, but since it has more parameters, their estimates will be less robust. This is similar to the alignment strategy presented in Chapter 2, but using a three-dimensional model. Unlike two-dimensional alignment, this least-squares optimization no longer has a closed form solution in terms of an over-constrained linear system. If we denote the locations of feature i of the face as x'_i and y'_i, and the feature locations of the 3D model as x_i, y_i, z_i, then optimization problem is to minimize E in the following equation:

$$E(S, T_x, T_y, q) = \sum_{i=1}^{n} \left| \begin{pmatrix} x'_i \\ y'_i \end{pmatrix} - \begin{pmatrix} S & 0 & 0 & T_x \\ 0 & S & 0 & T_y \end{pmatrix} \cdot R(q) \cdot \begin{pmatrix} x_i \\ y_i \\ z_i \\ 1 \end{pmatrix} \right|^2$$

where S is the scaling factor, $R(q)$ is a 4×4 rotation matrix parameterized by a four dimensional quaternion q, and T_x, T_y are the translation parameters. Each x'_i and y'_i gives rise to an term which contributes to the summation, and depends nonlinearly on the parameters (particularly the quaternion q). A standard method to optimize such a system is the iterative Levenberg-Marquardt method [Marquardt, 1963, Press *et al.*, 1993]. For simplicity, the summation above can be rewritten as a matrix equation, as follows:

$$E = \left| \begin{pmatrix} x'_1 \\ y'_1 \\ x'_2 \\ y'_2 \\ \vdots \\ x'_n \\ y'_n \end{pmatrix} - f(P) \right|^2 = |X' - f(P)|^2$$

where P is the vector of parameters S, T_x, T_y, q, and f is a vector function generating the coordinatoes of the 3D head model from the pose specified by those parameters.

The Levenberg-Marquardt method approximates the function $f(P)$ using a first-order Taylor expansion, as follows:

$$f(P_0 + \Delta P) \approx f(P_0) + \left(\frac{\partial f(P_0)}{\partial P}\right) \Delta P = f(P_0) + J\Delta P$$

Since the error function is quadratic, it can be minimized by setting its derivative to zero using the above Taylor approximation, as follows:

$$\begin{aligned} \frac{\partial E(P_0 + \Delta P)}{\partial \Delta P} &\approx \frac{\partial}{\partial \Delta P} |X' - f(P_0 + \Delta P)|^2 \\ &= \frac{\partial}{\partial \Delta P} |X' - (f(P_0) + J\Delta P)|^2 \\ &= 2J^T(X' - (f(P_0) + J\Delta P)) \\ &= 0 \end{aligned}$$

This is an over-constrained linear system, whose approximate solution is:

$$\Delta P \approx (J^T J)^{-1} J(X' - f(P_0))$$

This solution can only be computed when the initial parameters P_0 are near the minimum, and the Taylor approximation is accurate. Under these conditions, the update to the parameters ΔP can be very accurate.

However, dependences between the parameters may make the inversion of $J^T J$ numerically unstable. In some cases, a better approach is that of gradient descent, in which ΔP is computed as follows:

$$\Delta P \propto \frac{\partial E(P_0)}{\partial P} \approx 2 \left(\frac{\partial f(P_0)}{\partial P}\right)^T (X' - f(P_0)) = 2J^T(X' - f(P_0))$$

The Levenberg-Marquardt combines these two methods, as follows:

$$\Delta P = (J^T J + \lambda I) J^T (X' - f(P_0))$$

λ is a weight for the contributions of the two methods. When λ is zero, the method follows the Taylor expansion method. When λ is large, ΔP's value is dominated by the gradient descent value. The actual update to the parameters is computed from $P' = P_0 + k\Delta P$, and the method is iterated until the parameters converge. Although there are methods to adapt the value of λ [Marquardt, 1963, Press *et al.*, 1993], in this work setting λ to a fixed value of 1 worked well.

To apply the Levenberg-Marquardt method to matching a face in three-dimensions, we need to know the $f(P)$ and J functions. The $f(P)$ function is given by the above equation relating x'_i, y'_i and x_i, y_i, z_i. The only non-linear portion of this equation is in the rotation matrix $R(q)$ which has a non-linear dependence on the four quaternion parameters q. This relationship is shown in [Gleicher and Witkin, 1992], which also describes how to compute the J matrix efficiently.

Using the Levenberg-Marquardt method, we can rotate, translate, and scale the 3D face model in three-dimensions to align it with each of the training faces. In this chapter, we use the FERET database for training, because it contains faces at a variety of angles. The face images are roughly categorized according to their angle from frontal, but the actual angles of the faces can vary significantly within each category. The approximate angle for each category is used to initialize the Levenberg-Marquardt optimization, which is then run until the parameters converge.

As with the alignment procedure in Chapter 2, once the faces are all aligned with the 3D model, the positions of the features on the aligned faces can be averaged to update the feature positions in the 3D model. However, since each facial feature contains only two-dimensional information, they cannot be directly averaged.

Instead, we return to the equation relating x'_i, y'_i and x_i, y_i, z_i, and write it in the following form:

$$\begin{pmatrix} S & 0 & 0 & T_x \\ 0 & S & 0 & T_y \end{pmatrix} \cdot R(q) \cdot \begin{pmatrix} x_i \\ y_i \\ z_i \\ 1 \end{pmatrix} = \begin{pmatrix} x'_i \\ y'_i \end{pmatrix}$$

We can see that when the rotation, translation, and scaling parameters of the face are known, this equation describes a linear relationship between the feature locations in the 3D model and the feature locations on each example face. We can make a larger matrix equation which includes all the features of all the faces. This equation will be over constrained, and can be solved by the least-squares method to find the vector of feature locations of the 3D model.

With an updated 3D model, we can go back and update the alignment parameters aligning that model with each face, and iterate the two steps several times until convergence. The resulting 3D model feature locations are shown in Figure 5.2 and the results of the alignment, illustrated by rendering the original 3D model together with the face, are shown in Figure 5.3.

Figure 5.2: Refined feature locations (gray) with the original 3D model features (white).

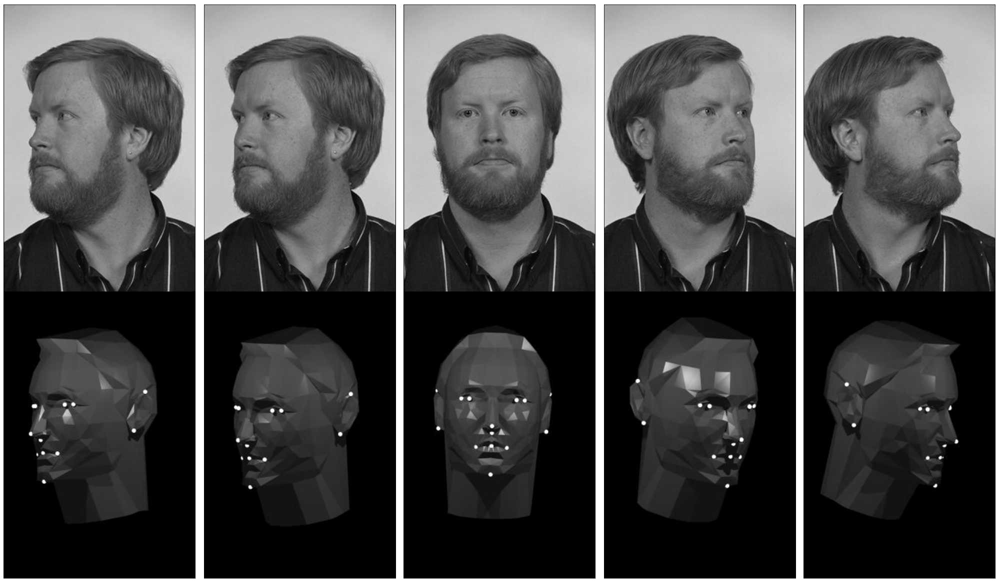


Figure 5.3: Rendered 3D model after alignment with several example faces.

5.2.3 Representation of Pose

In the previous section, I used quaternions to represent the angles of the 3D model with respect to the face. Quaternions have several nice properties which make them attractive for the type of optimization used to align the models, because they have no singularities. Continuous changes in the four-dimensional unit-quaternion space result in continuous changes in the rotation matrix. The expense of this representation is redundancy; rather than the minimal three parameters needed to describe an orientation, four are used.

One result of this redundancy in the quaternion representation is that a quaternion and its corresponding negative quaternion both represent the same angle. Any attempt to restrict the representation to one of these pairs (say, by restricting the first component to always be positive) will lead to singularities in the representation. The redundancy of this form is not a problem for an optimization procedure using gradient descent, but it is a problem if you want to build a mapping from an input directly to a quaternion using a neural network.

Ideally, we need a representation which has no singularities, and also gives a unique representation for each rotation. One simple representation is to use two orthogonal 3D unit vectors, one pointing from the center of the 3D model to the right ear, the other pointing along its nose. This representation is clearly unique (any change in the unit vectors changes the orientation of the head), and also clearly continuous (any small change in the unit vectors gives a small change of orientation). Again, this improvement of the representation comes at the cost of redundancy, because we now need to use six parameters to represent the orientation.

5.2.4 Training the Pose Estimator

From the previous two sections, we have example face images which are aligned with one another in three dimensions, and a continuous representation of the angle of each face. The scale and translation components of the alignment are used to normalize the size and position of each angle. The image is rotated by a random amount in-plane, and the orientation parameters are adjusted appropriately. This gives example images and outputs like those shown in Figure 5.4. As before, a number of random variations of each example face are used to increase the robustness of the system. It is also important to balance the number of faces at each orientation. For this purpose, we quantize the angle of each face from frontal into increments of 10° and count the number of faces in each category. The number of random examples for each face is inversely proportional to the number of faces in the category, which equalizes the distribution of this angle.

Figure 5.4: Example input images (left) and output orientations for the pose estimation neural network. The pose is represent by six vectors of output units (bottom), collectively representing 6 real values, which are unit vectors pointing from the center of the head to the nose and the right ear. Together these two vectors define the three dimensional orientation of the face. The pose is also illustrated by rendering the 3D model at what same orientation as the input face (right).

The outputs from the pose estimation neural network should be two unit vectors, representing the orientation of the head. Each component of these vectors is represented by an array of output units. Their values are computed by a weighted sum of the positions of the outputs within the array, weighted by the activation of the output.

With the training examples in hand, a neural network can be trained. The network has an input retina of 20×20 pixels, connected to six sets of hidden units, each of which looks at a 5×5 sub-window. These hidden units are completely connected to a layer of 40 hidden units, which is then completely connected to the output layer. The output consists of six arrays of 31 units, each representing a real value between -1 and 1. The results of this network on some test images are illustrated in Figure 5.5. When applying the network, the output vectors' magnitudes are normalized, and the second unit vector is forced to be perpendicular to the first.

Figure 5.5: The input images and output orientation from the neural network, represented by a rendering of the 3D model at the orientation generated by the network.

5.2.5 Geometric Distortion

The main difficulty in producing a frontal image of a face from a partial profile is that parts of the face in the original image will be occluded. However, if we assume that the left and right sides of the face are symmetric with one another, we can replace the half of the upright, frontal view that contains partial occlusions with a mirror image of the other half.

Some example results are shown in Figure 5.6, for faces which are limited to angles of $45°$ from frontal. As can be seen, when the pose estimation network is accurate and the face is similar in overall shape to the 3D model, the resulting upright, frontal view of the face can be quite realistic. However, small errors in the pose estimation result in larger artifacts in the frontal faces, and large errors in pose estimation give results that are unrecognizable as faces. Additionally, even if the pose estimation is perfect, errors in the 3D model of the face (which is just a generic model) will lead to errors. Given these potential problems, we decided to use another approach.

Figure 5.6: Input windows (left), the estimated orientation of the head (center), and geometrically distorted versions of the input windows intended to look like upright frontal faces (right).

5.3 View-Based Detector

Instead of trying to geometrically correct the image of the face, we will try to detect the faces rotated out-of-plane directly. However, we cannot expect a single neural network to be able to detect all views of the face by itself. As mentioned in Section 3.2.1, even increasing the amount of in-plane rotation for frontal face images dramatically increase the error rate. To minimize the amount of variation in the images the neural network must learn, we partition the views of the face into several categories according to their approximate angle from frontal.

As with the tilted face detection in Chapter 4, the idea is to use a pose estimation network to first compute the in-plane angle of the face and the category, then rotate the image in-plane to an upright orientation, and finally to apply the appropriate detector network. Note that the detectors for the faces looking to the right are simply mirror images of the networks for the faces looking to the left. This algorithm is illustrated in Figure 5.7.

5.3.1 View Categorization and Derotation

We could apply the same pose estimation network as was used in Section 5.2, however the results will not be good. As was found in Chapter 3, the detector networks work by looking for particular features (mostly the eyes) at particular locations in the input window. Imagine a head rotating in the input window. As it rotates, all the feature locations will shift within the input window, meaning that none of the feature locations are stable. This would make the detection problem much harder. It would be better to apply the two-dimensional alignment procedure to the faces within each category. This would allow each view-specific face detector to concentrate on specific

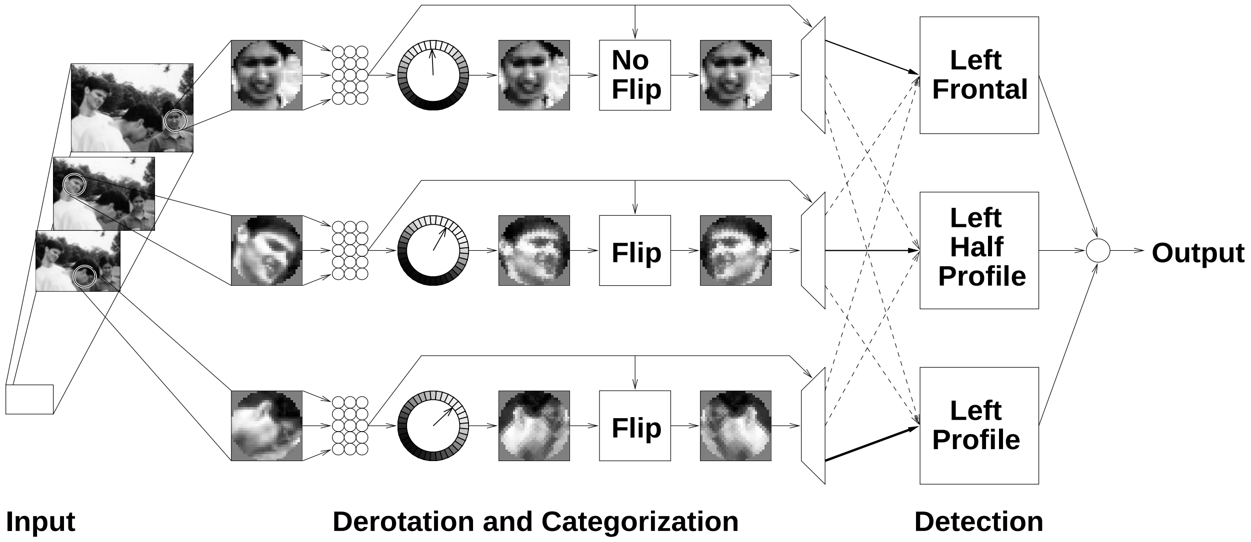


Figure 5.7: View-based algorithm for detecting non-frontal and tilted faces.

features in specific locations.

We are still left with the question of how to assign faces to specific categories. In light of the observation that the two-dimensional alignment of feature locations is important, we chose to use a criterion based on how closely the feature locations align with a prototypical example of the category. This prototype is constructed from the three-dimensional model, which is rotated to several angles from frontal, as shown in Figure 5.8. Each of the face examples is aligned as well as possible with all of the category prototypes, and assigned to the category it matches closest.

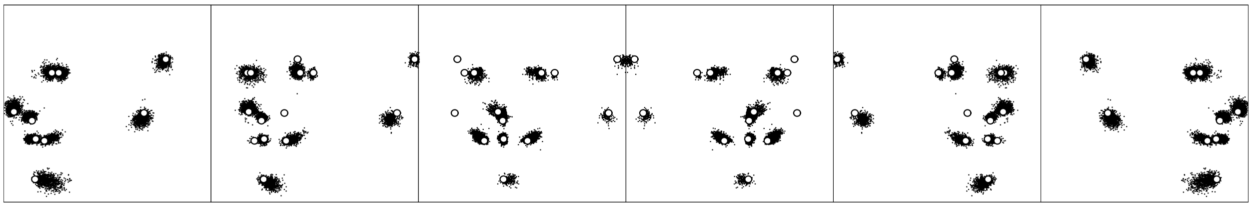


Figure 5.8: Feature locations of the six category prototypes (white), and the cloud of feature locations for the faces in each category (black dots).

As before, the actual alignment could be an iterative process, first aligning all the faces in the category with the category prototype, then updating the prototype with the average of all the aligned faces in the category, and repeating. However, in my experiments I found that simply aligning each faces with the original category prototype allowed the category estimation network to work better. This may be because the original prototypes have geometric relations with one another (such as the eyes always falling on the same scan line) which are disrupted if the prototypes are adjusted to the training examples.

Once the faces are aligned with one another, they are rotated to random in-plane orientations, and the resulting images are recorded as the training examples, as shown in Figure 5.9. Associated with each face example is its in-plane orientation and the category label. As before, we produce several random variations of each training face, and the number of variations is chosen to balance the number of examples in each category.

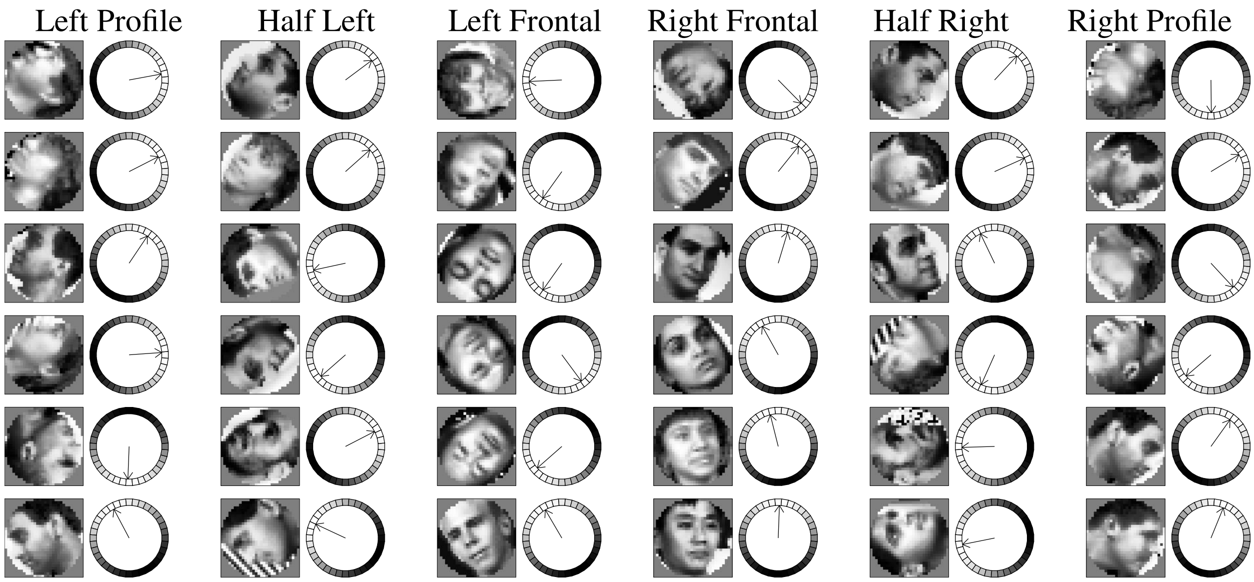


Figure 5.9: Training examples for each category, and their orientation labels, for the categorization network. Each column of images represents one category.

Next we can train a neural network to produce the categorization and in-plane orientation of an input window. The architecture used consists of four layers. The input layer consists of units which receive intensities from the input window, a circle of radius 15 pixels. The first hidden layer has localized connections to this circular input. The second hidden layer has 40 units, with complete connections to the first hidden layer and to the output layer. As was done in Chapter 4, the output angle is represented by a circle of output units, each representing a particular angle. Each category label has an individual output, and the category with the highest output is considered to be the classification of the window. Note that one difference with the previous work is that the input window is circular; this makes it possible to rotate the input window in-plane without having to recompute the preprocessing steps. With a square window, the derotated window covers different pixels than the original window, invalidating the histogram equalization that is done. As a further optimization, the rotation is done by sampling pixels at integer coordinators rather than bilinear interpolation. This causes a slightly pixelated appearance in the output images of Figure 5.10.

Figure 5.10: Training examples for the category-specific detection networks, as produced by the categorization network. The images on the left are images of random in-plane angles and out-of-plane orientations. The six columns on the right are the results of categorizing each of these images into one of the six categories, and rotating them to an upright orientation. Note that only three category-specific networks will be trained, because the left and right categories are symmetric with one another.

5.3.2 View-Specific Face Detection

We can apply this network to all of its training data, which will categorize and derotate the images into the appropriate categories, as shown in Figure 5.10. These images can then be used to train a set of detection networks. Note that we could have created training data based directly on the original images and their categorizations, but by using the view-categorization network to label the training data, we hope to capitalize on any systematic errors that it may make. The training networks are trained in the same way as those used for the tilted face detector. In particular, the negative examples are also run through the view categorization network, to make sure the detectors are trained on the same type of images they will see at runtime.

As in the previous chapters, two networks are trained for each of three categories, and their results are arbitrated to improve the overall accuracy of the system. Recall that from the discussion of the arbitration method in Section 3.4, the main technique is to examine all the detections within a small neighborhood of a given detection, and the total number is interpreted as a confidence measure for that particular detection. For the upright face detection, the neighborhood was defined in terms of the position and scale of the detection. For the tilted detector, and extra dimension, the in-plane orientation of the head, was added. Small changes in each of these dimensions result in small changes in the image seen by the detector, so the neighborhood of a detection is easily defined in terms of a neighborhood along each dimension.

For the non-frontal detector, yet another dimension is needed, that of the out-of-plane oriention, or category, of the head. Unlike the previous dimensions, this dimension has discrete values. The categories place facial features at different locations; see for example the location of the point between the two eyes for each category prototype in Figure 5.8. To decide whether two detections are in the same neighborhood, the offset between the facial feature locations in the two categories must be taken into account. For each pair of categories, the shift of the point between the two eyes is computed. The following procedure is then used to decide whether detection D_2 is in the neighborhood of detection D_1:

1. If D_1 and D_2's categories differ by more than one, return `false`.
2. If the scales of D_1 and D_2 more than 4 apart, return `false`.
3. Traslate the location of D_2 by the offset for the two categories. The translation is along the direction indicated by the in-plane orientation of D_2.
4. Scale the location of D_2 according to the difference in pyramid levels between the two detections.
5. If the adjusted location of D_2 further than 4 pixels in x and y from D_1, then return `false`.
6. Return `true`.

Using this concept of neighborhood, the actual arbitration procedure used in this chapter is the same as that used for the tilted face detector: First, each individual netowrk's output is filtered to remove spatially overlapping detections, keeping those with a higher confidence as measured by the number of detections within the small neighborhood. Then, the cleaned results of the two networks are ANDed together, again using the neighborhood concept to locate cooresponding detections in the outputs of the two networks.

The results of the individual networks and the complete system are reported in the next section.

5.4 Evaluation of the View-Based Detector

This chapter will use the *Upright Test Set* and *Rotated Test Set* from the previous chapters to evaluate the detector. Since the system is now expected to locate profile and partial profile faces, an additional 10 faces (for a total of 521) have been labelled in the *Upright Test Set*, and one additional face (for a total of 224) has been labelled in the *Tilted Test Set*. Note that the FERET

Test Set cannot be reused here, because it was used for training. In addition, to evaluate the non-frontal detection capabilities, three new test sets were used. The test new sets are described briefly below, before going into the results of the system.

5.4.1 Non-Frontal Test Set

The first set of images consists of pictures collected from the World Wide Web and locally at CMU. These images are intended to have a variety of poses, from frontal to profile, as well as a variety of in-plane angles. A typical image is shown in Figure 5.11. The set contains 53 images with 96 faces, and requires the network to examine 16,208,022 windows. In the experiments section, this set is referred to as the *Non-Frontal Test Set*.

Figure 5.11: An example image from the *Non-Frontal Test Set*, used for testing the pose invariant face detector.

5.4.2 Kodak Test Sets

Kodak provided a large set of images which we intend to use for testing purposes, consisting of typical family snapshots, with poor lighting, poor focus, partial occlusion, and other problems which challenge the capabilities of a face detector. Although the actual images cannot be shown here, the image shown in Figure 5.11 is typical of the types of images. The subset of the database used in this chapter, which contains mostly partial and full profile faces, has 46 faces in 17 images, which contain a total of 15,365,395 windows. This will be referred to as the *Kodak Test Set*.

The second database, known as the *Kodak Research Image Database*, consists of studio mugshots of 89 people from 25 angles for each face. The database contains 2225 images in all, with 2222 faces (3 images are blank), and requires processing of 111,893,025 windows. These photographs will be used in evaluating the profile face detector. They consist of images like those shown in Figure 5.12 (the actual images cannot be reproduced here).

Figure 5.12: Images similar to those in the *Kodak Research Image Database* mugshot database, used for testing the pose invariant face detector. Note that the actual images cannot be reproduced here.

5.4.3 Experiments

To evaluate the view detector network, I first applied it to the *Upright Test Set* and *Rotated Test Set* to test its capabilities compared with the previous two systems. The performance results are shown in Figure 5.13. As before, the accuracy of the system will depend on the types of arbitration used. As can be seen from the table, the system has significantly more false alarms than either of the two previous systems, and a slightly lower detection rate for these two test sets. This suggests that for applications needing the detection of only upright or tilted faces, one of the two previous detectors is a better choice.

Table 5.13: Results of the upright, tilted, and non-frontal detectors on the *Upright* and *Tilted Test Sets*.

	Upright Test Set		Tilted Test Set	
System	Detect %	# False	Detect %	# False
Network 1	80.0%	5499	85.7%	2555
Network 2	78.9%	5092	84.8%	2385
Net 1 → threshold(4,1) → overlap	73.1%	2273	77.7%	975
Net 2 → threshold(4,1) → overlap	73.3%	2134	76.8%	905
Nets 1,2 → threshold(4,1) → overlap → AND(4) → overlap	67.2%	365	69.6%	189
Upright (Chapter 3)	83.7%	31	12.9%	11
Tilted (Chapter 4)	75.4%	44	85.3%	15

I next tested the system on the two test sets designed specifically to measure its ability to detect profile faces. Table 5.14 shows the accuracy of the system using a variety of arbitration methods, along with the results of the other systems on the same data.

The performance of the upright, tilted, and non-frontal face detectors on the *Kodak Research Image Database* is shown in Figure 5.15. Next to each image are three pairs of numbers. The top pair gives the detection rate and number of false alarms for the upright face detector. As we can see, this detector performs best with frontal images, however it is quite robust to changes in the orientation of the head. The second pair of numbers gives the performance for the tilted face detector. As with the upright detector, it has a high detection rate for frontal images. However, its accuracy drops off more quickly than the upright detector as the face is rotated away from frontal. The last pair of numbers gives the accuracy of the non-frontal detector for these images. The detection rate of this detector is quite uniform over the test images, detecting both frontal and profile faces. The lowest detection rates are for faces looking above or below the camera, because this type of rotation is not covered by the non-frontal detector. As the faces turn towards profiles,

Table 5.14: Results of the upright, tilted, and non-frontal detectors on the *Non-Frontal* and *Kodak Test Sets*, and the *Kodak Research Image Database*.

	Non-Frontal Test Set		Kodak Test Set	
System	Detect %	# False	Detect %	# False
Network 1	75.0%	1313	58.7%	1347
Network 2	65.6%	1367	50.0%	1296
Net 1 → threshold(4,1) → overlap	61.5%	617	41.3%	639
Net 2 → threshold(4,1) → overlap	60.4%	627	41.3%	617
Nets 1,2→threshold(4,1)→overlap→AND(4)→overlap	56.2%	118	32.6%	136
Upright (Chapter 3)	21.9%	7	15.2%	6
Tilted (Chapter 4)	16.7%	5	13.0%	4

the detection rate improves. This is because up and down motion becomes rotation in the image plane, which the non-frontal detector is able to handle.

The systems do quite well on these images compared with the *Upright*, *Tilted*, and *Non-Frontal Test Sets*. We suspect that this is in part because of the studio conditions under which the *Kodak Research Image Database* was collected. The majority of the training data for this system came from the FERET database, whose pictures were taken under similar studio conditions. For further robustness, the system should be trained with profile and partial profile faces acquired under more natural conditions, including faces which are looking slightly up or down with respect to the camera.

To get a better understanding of why the detection rates are lower for the non-frontal face detector than the upright or tilted ones, the non-frontal detector can be broken into two stages, and the performance of each stage can be measured independently. The first stage is the derotation and categorization network. We applied this to 30×30 circular window in the test sets which contains a face, and the derotation angle and category of the face are computed using two methods: the neural network, and the method used to prepare the training data for this network. By comparing the results of these two methods, we can see how accurate the derotation and categorization network is on an independent test set. Next, the faces derotated and categorized by these two methods can be passed to the detection network, whose detection rates can be measured for these two cases. The results of these comparisons, for the *Upright*, *Tilted*, and *Non-Frontal Test Sets* are shown in Table 5.16.

It is reasonable to assume that the detection rate using the manual classification is the best possible, and that the detectors can only work when given the correct category, because a face in the wrong category would be aligned incorrectly for the erroneous category. Based on this

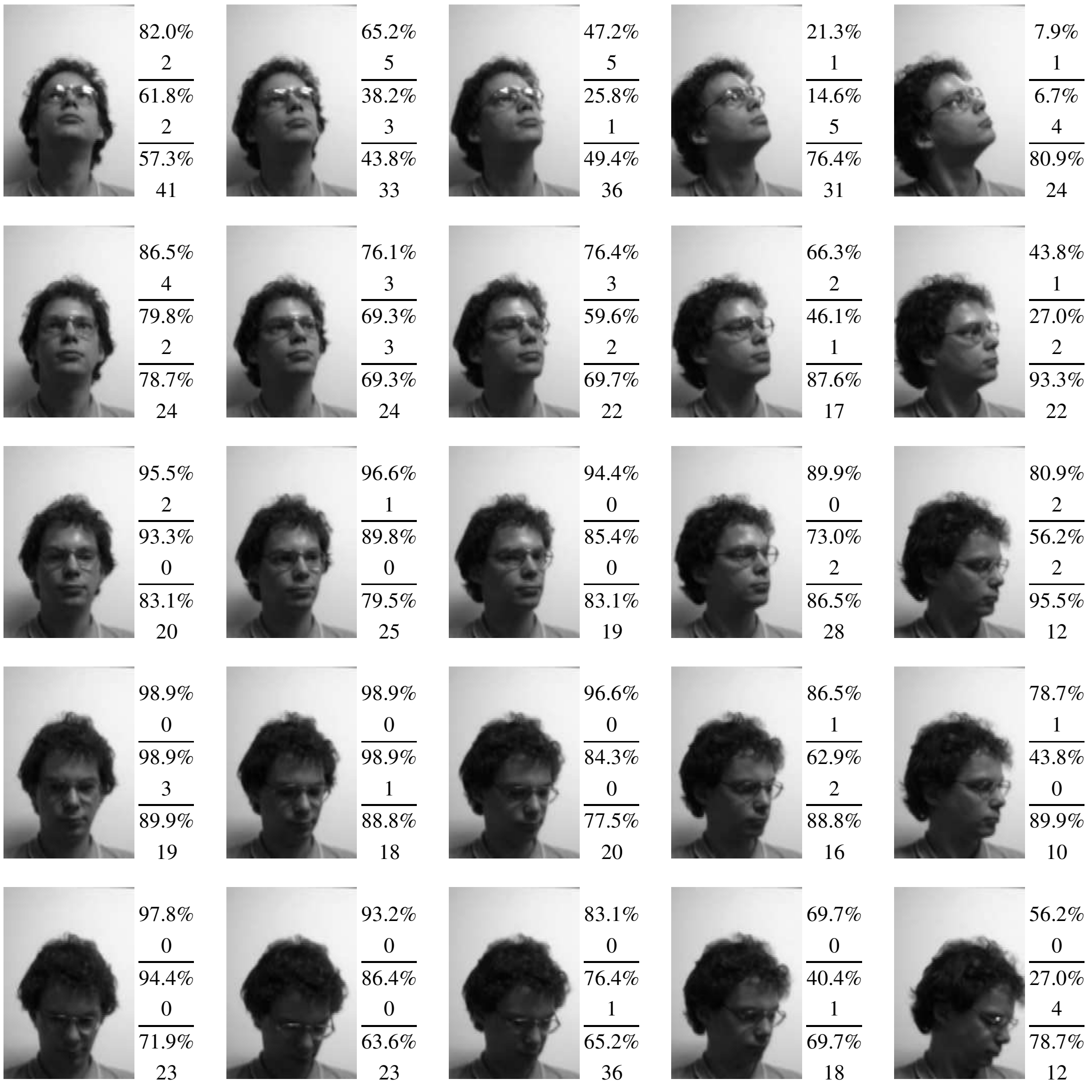


Figure 5.15: Images similar to those in the *Kodak Research Image Database* mugshot database, used for testing the pose invariant face detector. For each view shown here, there are 89 images in the database. Next to each representative image are three pairs of numbers. The top pair gives the detection rate and number of false alarms from the upright face detector of Chapter 3. The second pair gives the performance of the tilted face detector from Chapter 4, and the last pair contains the numbers from the system described in this chatper.

Table 5.16: Breakdown of the accuracy of the derotation and categorization network and the detector networks for the non-frontal face detector.

		Test Sets		
Stage	Statistic	Upright	Tilted	Non-Frontal
Derotation and categorization network output	Exact category	49.5%	48.7%	38.5%
	Not categorized	25.1%	20.5%	30.2%
	Angle within 10°	42.6%	42.4%	21.9%
Detector output	Manual categorization	65.3%	65.2%	39.6%
	Automatic categorization	34.9%	31.7%	16.7%
	Predicted	32.3%	31.8%	15.2%
Complete system	Detect rate	67.2%	69.6%	56.2%

assumption, we can predict that the detection rate using automatic categorization and derotation will be the product of the detection rate for manual categorization/derotation and the fraction of the faces for which the categorization/derotation network returns the rigth category. This prediction is shown in the "Predicted" line of Table 5.16. Since the prediction accurately matches the actual detection rate when using the neural network for categorization and derotation, we can see that improving the categorization performance will directly improve the overall detection rate.

The table shows only the detection rates when applying a single detector network at a single pixel location and scale in the image. In practice, the detectors are applied at every pixel location and scale, giving them more opportunities to find each face. This explains the higher detection rates of the complete system (the last line in Table 5.16 relative to the earlier lines.

Some example results from the *Upright*, *Tilted*, and *Non-Frontal* test sets are shown in Figure 5.17.

5.5 Summary

This chapter has presented an algorithm to detect faces which are rotated out of the image plane. First, by using geometric distortions, it may be possible to transform a face image from a partial profile to an upright frontal view, thereby enabling the use of an upright, frontal detector. However, in the experiments shown, it was found to be difficult to align a 3D model of the face precisely enough with the image to perform the transformation accurately; this made it unusable for detection.

The second approach partitions the out-of-plane rotations of the head into several views and uses separate detectors for each view. This approach is accurate enough at normalizing the views of

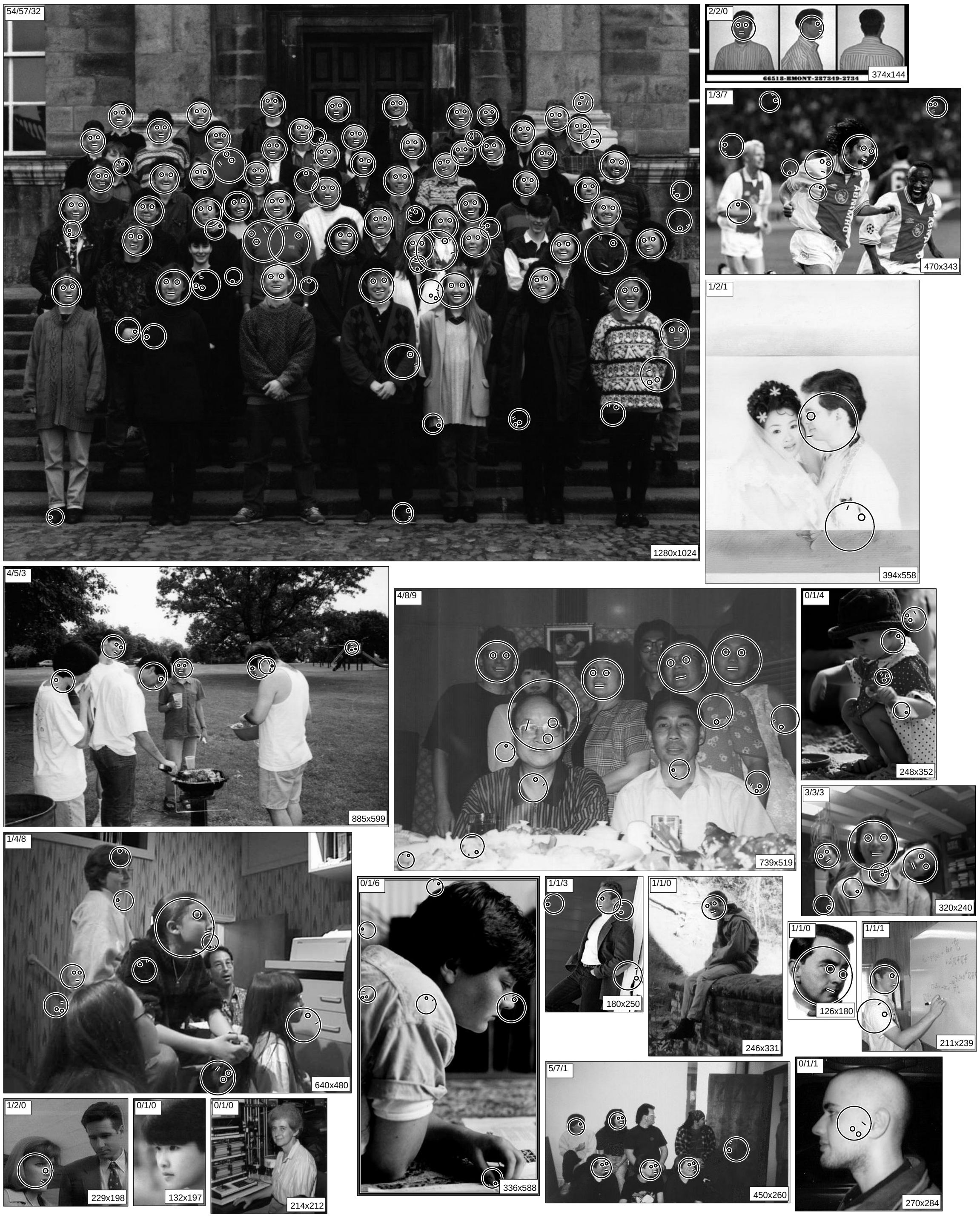


Figure 5.17: Example output images from the pose invariant system. The label in the upper left corner of each image (D/T/F) gives the number of faces detected (D), the total number of faces in the image (T), and the number of false detections (F). The label in the lower right corner of each image gives its size in pixels.

the faces to enable detection. The detection rate and false alarm rates are poorer than the systems in the previous two chapters, but when a particular application requires the detection of profile faces, it may be acceptable.

Chapter 6

Speedups

6.1 Introduction

In this chapter, we briefly discuss some methods to improve the speed of the face detectors presented in this thesis. This work is preliminary, and not intended to be an exhaustive exploration of methods to optimize the execution time.

6.2 Fast Candidate Selection

The dominant factor in the running time of the upright face detection system described thus far is the number of 20×20 pixel windows which the neural networks must process. Applying two networks to a 320×240 pixel image on a 175 MHz R10000 SGI O2 workstation takes approximately 140 seconds. The computational cost of the arbitration steps is negligible in comparison, taking less than one second to combine the results of the two networks over all positions in the image of this size.

6.2.1 Candidate Selection

Recall that the amount of position invariance in the detector networks determine how many windows must be processed. In the related task of license plate detection, this was exploited to decrease the number of windows that must be processed [Umezaki, 1995]. The idea was to make the neural network be invariant to translations of about 25% of the size of the license plate. Instead of a single number indicating the existence of a face in the window, the output of Umezaki's network is an image with a peak indicating the location of the license plate. These outputs are accumulated over the entire image, and peaks are extracted to give candidate locations for license plates.

The same idea can be applied to face detection. The original detector was trained to detect a 20×20 face centered in a 20×20 window. We can make the detector more flexible by allowing the same 20×20 face to be off-center by up to 5 pixels in any direction. To make sure the network can still see the whole face, the window size is increased to 30×30 pixels. Thus the center of the face will fall within a 10×10 pixel region at the center of the window, as shown in Figure 6.1. As before, the network has a single output, indicating the presence or absence of a face. This detector can be moved in steps of 10 pixels across the image, and still detect all faces that might be present. The scanning method is illustrated in Figure 6.2, for the upright face detection domain. The figure shows the input image pyramid and the 10×10 pixel regions that are classified as containing the centers of faces. An architecture with an image output was also tried, which yielded about the same detection accuracy, but required more computation. The network was trained using the same active learning procedure procedure described in Chapter 3. The windows are preprocessed with histogram equalization before they are passed to the candidate selector network.

As can be seen from the figure, the candidate selector has many more false alarms than the detectors described earlier. To improve the accuracy, we use the 20×20 detectors described earlier to verify it. Since the candidate faces are not precisely located, the verification network's 20×20 window must be scanned over the 10×10 pixel region potentially containing the center of the face. A simple arbitration strategy, ANDing, is used to combine the outputs of two verification networks. The heuristic that faces rarely overlap can also be used to reduce computation, by first scanning the image for large faces, and at smaller scales not processing locations which overlap with any detections found so far. The results of these verification steps are illustrated on the right side of Figure 6.2.

6.2.2 Candidate Localization

Scanning the 10×10 locations within the candidate for faces can still take a significant amount of time. This can be reduced by introducing another network, whose purpose is to localize the face more precisely. In the upright face detection domain, this network takes a 30×30 input window, and should produce two outputs, the x and y positions of the face. These outputs are represented using a standard representation. Each output has a range from -5 to 5, so the output is represented by a vector of 20 outputs, each having an associated value of in the range -10 to 10. To get the real valued output, a weighted sum of the values represented by each output is computed. The outputs are trained to produce a bell curve with a peak at the desired output. Given these values, the window centered on the face can be extracted and verified. This technique can be viewed as similar to the derotation network used in Chapter 4. As with the derotation network, we must check that the localization network is accurate enough, and the detection network is invariant enough, to

Figure 6.1: Example images used to train the candidate face detector. Each example window is 30×30 pixels, and the faces are as much as five pixels from being centered horizontally and vertically.

the location of the face. Figure 6.3 shows an error histogram for the localization network, and the sensitivity of the upright face detector networks to how far off-center the face is. Since the average error of the localization network is quite small, the verification networks need only be applied once, at the estimated location of the face.

6.2.3 Candidate Selection for Tilted Faces

The same idea can be applied in the tilted face detection domain. We begin by training a candidate detector with examples of faces at different locations and orientations in the input window. Like with the upright candidate selector, the idea is that this network should be able to eliminate portions

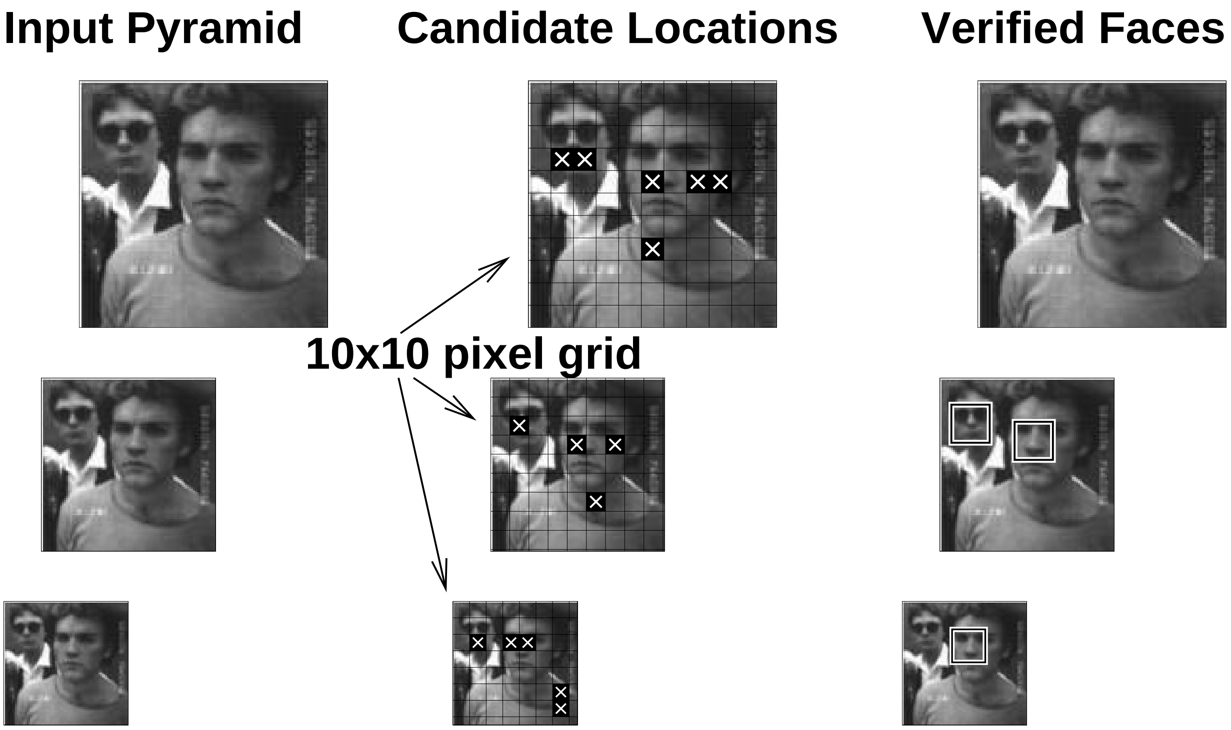


Figure 6.2: Illustration of the steps in the fast version of the face detector. On the left is the input image pyramid, which is scanned with a 30×30 detector that moves in steps of 10 pixels. The center of the figure shows the 10×10 pixel regions (at the center of the 30×30 detection windows) which the 20×20 detector believes contain the center of a face. These candidates are then verified by the detectors described in Chapter 3, and the final results are shown on the right.

of the input from consideration. However, since the set of allowed images is now much more variable, the candidate selector has a harder task, and it cannot eliminate as many areas, so the speedup will not be as large as in the upright case.

In addition to producing the x and y locations of the face, the tilted localization network must also produce the angle of the face. The angular output is represented in the same way as the derotation network used in Chapter 4. With these results, 20×20 windows for the detection networks can be generated. Again, we need to evaluate the accuracy of the localization network, as shown in Figure 6.4. As we know from earlier, the face needs to be centered within about 1 pixel (from Figure 6.3, and within an angle of about 10° (from Figure 4.4) to be detected accurately. Since the localization of this network is not as accurate as the upright detector, we need to apply the detectors to verify several candidate locations. Specifically, the detector is applied at 3 different x and y values and 5 angles around the estimated location and orientation of the face.

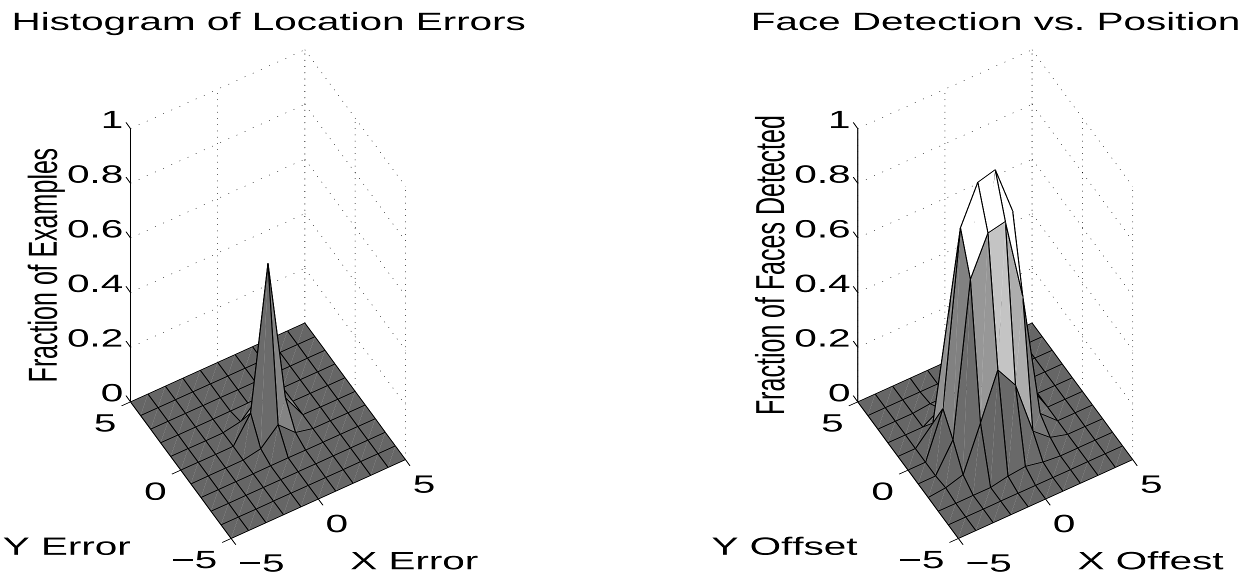


Figure 6.3: Left: Histogram of the errors of the localization network, relative to the correct location of the center of the face. Right: Detection rate of the upright face detection networks, as a function of how far off-center the face is. Both of these errors are measured over the training faces.

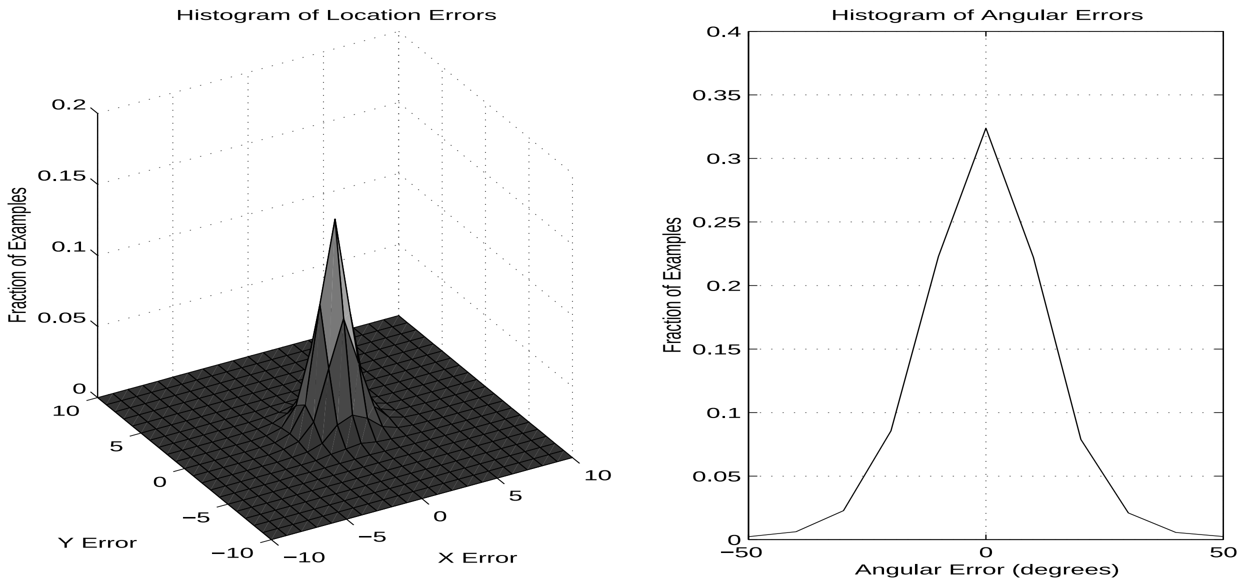


Figure 6.4: Left: Histogram of the translation errors of the localization network for the tilted face detector, relative to the correct location of the center of the face. Right: Histogram of the angular errors. These errors are measured over the training faces.

Figure 6.5: Examples of the input image, the background image which is a decaying average of the input, and the change detection mask, used to limit the amount of the image searched by the neural network. Note that because the person has been stationary in the image for some time, the background image is beginning to include his face.

6.3 Change Detection

Further performance improvements can be made if one is analyzing many pictures taken by a stationary camera. By taking a picture of the background scene, one can determine which portions of the picture have changed in a newly acquired image, and analyze only those portions of the image.

In practice, changes in the environment, lighting, or automatic adjustments of the iris or gain in the camera can change the intensity of the image. One way to deal with this is to use the differences between consecutive images. However, if a person remains relatively still between two frames, the face detector will lose track of the face. We need an intermediate between using a fixed background image, and using the previous image, to detect changes.

The intermediate I choose was to use a moving average of the input image as the “background”. The background model is initialized from the first input image. For subsequent frames, the background model is updated according to the rule:

$$\text{background}' = 0.95 \cdot \text{background} + 0.05 \cdot \text{input}$$

Then we compute the difference between the background and the input, and apply a threshold of 20. Before applying the candidate detector to a window, we first check if there is a certain number of pixel have changes above the given threshold. This pixel threshold is set quite low, because if a person stays fairly still in the image, the only changing pixels are at a border of their face. An example of the input image, background image, and change detection mask are shown in Figure 6.5.

6.4 Skin Color Detection

All of the work described thus far has used grayscale images. If color information is available, it may help the detector. Specifically, if there is a fast way to locate regions of the image containing skin color, then the search for faces can be restricted to those regions of the image.

A fast technique for locating skin color regions is described in [Hunke, 1994, Yang and Waibel, 1996]. It first converts the color information to a normalized color space, by dividing the red and green components by the intensity. These values are then classified by a simple Gaussian classifier, which has been trained with skin color samples. The research in [Hunke, 1994, Yang and Waibel, 1996] found that, perhaps surprisingly, for constant imaging and lighting conditions, skin colors for all races form a fairly tight cluster in normalize color space. Recent work described in [Jones and Rehg, 1998] using a very large number of images collected from the world wide web showed there are slightly differences between the races, but that a simple histogram-based model of skin color can accurately model the distribution of skin color.

For our work, however, we used the Gaussian model. This model has the advantage of not requiring many training images. We apply the skin color classifier to the average color of each 2×2 pixel region of the input image. The averaging reduces the noise that is typically present in the cheap single-CCD color cameras usually provided with workstations. Then, before applying the candidate selector to a window of the image, the number of skin color pixels in the region are computed, and if this number is above a threshold, the region is evaluated by the candidate selector.

Note that the observed color associated with skin will depend strongly on the camera and the lighting conditions. One way to make the detector tolerant of this fact is to train the skin color model with images from a wide variety of conditions. However, this tends to make the skin color model so broad that it cannot effectively reduce the search area. A better approach is to model the skin color as the system is running.

We begin by using a very broad color model. Effectively every region of the image will need to be processed, but the candidate selection and motion cues can speed this up. After a face has been detected, pixels from the center of the face (to avoid the eyes, glasses, and mouth, which are not skin colored) are extracted and used to update the Gaussian model. Over time, the Gaussian model will become more precise, allowing the detector to rule out larger areas of the image and run faster. This process is illustrated in Figure 6.6.

If the lighting conditions change, the skin color model may no longer be accurate, and faces will be missed. One technique to deal with this is to broaden the skin color model when no faces are detected. Eventually, the skin color model will accept skin regions, and can be narrowed down again once a face is detected.

This method has been used in a real-time demonstration system, and has been found quite

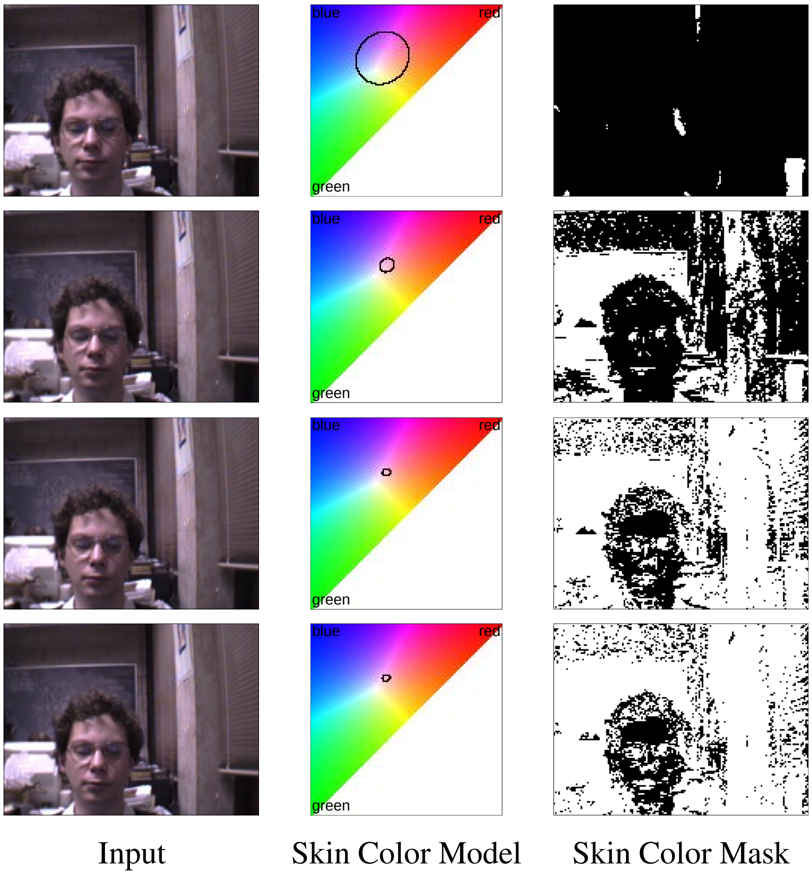


Figure 6.6: The input images, skin color models in the normalized color space (marked by the oval), and the resulting skin color masks to limit the potential face regions. Initially, the skin color model is quite broad, and classifies much of the background as skin colored. When the face is detected, skin color samples from the face are used to refine the model, so that is gradually focuses only on the face.

effective at adjusting the skin color model tightly enough to improve the speed of the detector, while able to adjust quickly to changes in the environment.

6.5 Evaluation of Optimized Systems

Using the candidate selection method, the upright face detection takes approximately 2 seconds to process a typical 320×240 pixel image on an SGI O2 workstation with a 175 MHz R10000 processor. Use of the candidate localization network can reduce this to 1.5 seconds. Incorporating the skin color and change detection algorithms brings the processing time to about 0.5 seconds. This time depends on the complexity of the scene, the number of candidate regions that are found, and the amount of skin color and motion present. Since the original processing time was 140 seconds, this represents a speedup of about 95 without skin color, and 280 with.

The tilted face detection method can also be made faster using the candidate selection technique, improving the processing time from 247 seconds for a 320×240 image to 14 seconds. Using skin color and motion can reduce this further, to about 1.5 seconds. Again, this times will depend on images themselves. The speedup in this case is about 17 without skin color, and 165 with skin color. These speedups are small than the upright face detector because the candidate selector network gives more false alarms, and because the localization network is not as accurate, requiring more effort to verify each face.

To examine the effect of these changes on accuracy, it was applied to the *Upright Test Set* and *Rotated Test Set* used the previous chapters. The results are listed in Table 6.7, along with the results for the unmodified systems. Note that since these test sets consist of static grayscale images, the skin color and change detection modules are not being used. The times reported are for a typical 320×240 pixel image, when color information is not used. With a color model well-tuned to the lighting conditions in the image, the fast upright detectors take about 0.5 seconds, while the tilted detector takes about 1.5 seconds. For comparison, the non-frontal detector requires 335 seconds on the same image.

As can be seen in the table, the systems utilizing the candidate selector method have somewhat lower detection rates than the original systems. This is to be expected; any false negatives by the candidate selector, or localization errors by the localization network, will result in more faces being missed. By the same logic, however, the number of false alarm will also decrease.

The use of the change and skin color detection techniques will exaggerate this effect further. This suggests that the detector itself could be made less conservative (perhaps by adjusting the threshold of the neural network outputs) to improve the detection rate while controlling the number of false alarms.

Table 6.7: The accuracy of the fast upright and tilted detectors compared with the original versions, for the *Upright* and *Tilted Test Sets*.

	Upright Test Set		Rotated Test Set		Time
System	Detect %	# False	Detect %	# False	(seconds)
Upright candidate selector → verification	74.0%	8	10.8%	6	2
Upright candidate selector → localization → verification	56.9%	0	8.1%	0	1.5
Tilted candidate selector → localization → verification	40.9%	0	54.7%	1	14
Upright (Chapter 3)	85.3%	31	13.0%	11	140
Tilted (Chapter 4)	76.9%	44	85.7%	15	247

6.6 Summary

This chapter has shown three techniques, candidate selection, change detection, and skin color detection, for speeding up a face detection algorithm. These techniques taken together have proven useful in building an almost real-time version of the system suitable for demonstration purposes. The speedups for upright face detector were 65 and for tilted face detection were 30, at the cost of a small decrease in the detection rate.

Chapter 7

Applications

7.1 Introduction

This chapter introduces a few applications of the face detection work presented in this thesis. These system actually use the algorithm and implementation of the upright face detector described in Chapters 3 and 6.

7.2 Image and Video Indexing

Every year, improved technology provides cheaper and more efficient ways of storing and retrieving visual information. However, automatic high-level classification of the information content is very limited. The first class of applications involves indexing of image and video databases, or providing convenient access to such databases.

7.2.1 Name-It (CMU and NACSIS)

The *Name-It* system [Satoh and Kanade, 1996, Satoh and Kanade, 1997], first developed at CMU as part of the *Informedia* project [Wactlar *et al.*, 1996], attempted to automate the labelling of faces in video sequences such as TV news stories. The face detector was applied to each image in the video, and the face images and times at which they appeared were recorded.

Automatic speech recognition was applied to the audio, and combined with closed caption information to produce a textual transcript of the video. Using a dictionary of English words, proper names could be extracted from the transcript, along with the times at which these names occurred.

When a name occurred at nearly the same time as a face, a co-occurrence score was computed for that pairing of the name and the face. By measuring the similarity of that face with all other detected faces, this co-occurrence score could be transferred to all other similar detected faces. In this work, the similarity was measured using the eigenface method [Pentland *et al.*, 1994], although any face recognition method could be used.

Pairings of faces and names which have high co-occurrences can then be used for queries. For instance, the user of the system could ask for "Clinton", and get faces that are commonly associated with the name "Clinton" in the news, perhaps including President Clinton or his family members. The system automatically handles special cases such as news show hosts, whose faces appear at the same time that many names appear in the transcript. It does this by detecting when one face is associated with too many names.

7.2.2 Skim Generation (CMU)

Another application of the face detector in the *Informedia* project was generating summaries of video [Smith and Kanade, 1996, Smith and Kanade, 1997]. To generate an understandable and useful summary of a video, one needs to select the parts of the video that are important. Important images might include those immediately after a scene break, those that contain text, those that are associated with important keywords in the transcript, or those that contain faces, as detected by the upright face detector. By combining these cues, the researchers were able to produce summary videos which effectively captured the main content of longer videos.

7.2.3 WebSeer (University of Chicago)

The *WebSeer* project attempted to build an index of images on the World Wide Web [Frankel *et al.*, 1996]. Each image that it collected from the web was associated with keywords selected from the Web page containing the image and from the file name itself. These keyword were used as the main search terms.

In addition, each image was classified according to a number of criteria. First, the image was classified as either black and white or color, as a real image or artificially generated with a painting program. Then, the face detector was applied, to measure the number and size of faces which appeared. This allows the image to be classified as a "crowd scene", or "head and shoulders portrait", "close up portrait". These classification criteria could then be used to limit the search results.

Although the *WebSeer* program did not try this, one could imagine applying the *Name-It* system to the face and associated text found on the World Wide Web, and automatically learn associations of names and faces there.

7.2.4 WWW Demo (CMU)

An interactive demonstration of the upright face detector described in Chapter 3 is available on the World Wide Web at `http://www.cs.cmu.edu/~har/faces.html`. This demonstration allows anyone to submit images for processing by the face detector, and to see the detection results for pictures submitted by other people.

7.3 User Interaction

The second class of applications involves allowing a computer to interact with a person in a more natural way. These include systems such as security cameras which record the people who enter a building without requiring such things as sign-in sheets, allowing robots to know when a person is present, or for amusement.

7.3.1 Security Cameras (JPRC and CMU)

A number of other researchers have obtained copies of the system for use as part of a face recognition project. One group which has actually built a system around the detector is at the Justsystem Pittsburgh Research Center [Sukthankar and Stockton, 1999, Sim *et al.*, 1999]. Researchers there set up a security camera at the door, and used the fast version of the upright detector presented in Chapter 6 to locate the faces before feeding them to the recognizer.

Another "security" camera was set up at the entrance of a building on the Carnegie Mellon Campus. It only ran the face detector, and showed the results in a monitor next to the camera, to entertain the people who visited the building. During nearly a year of operation, the system detected nearly 25,000 faces, of which approximately 10% were false alarms. The raw images were not recorded, so the false negative rate is hard to estimate.

7.3.2 Minerva Robot (CMU and the University of Bonn)

Researchers and companies have begun developing mobile robotic tour guides which among other technologies use computer vision to observe their environment and their audience. One of these systems, Minvera, was deployed at the Smithsonian Institution's National Museum of American History in Washington D.C. [Thrun *et al.*, 1999]. It applied the face detector to images from its vision system, and displayed pictures of the people it was talking to on its website. Although the output of the face detector was not used in controlling the robot, one could imagine using it to enhance the interaction between the audience and the robot in a number of ways, complementing the other sensors. The robot could first notice when people are present, and looking at the robot, to gauge interest, either in the robot or in the exhibits. If the robot has a "face", the face detector can be used to make the robot "look" at a particular person when explaining an exhibit.

7.3.3 Magic Morphin' Mirror (Interval Research)

Researchers at Interval Corporation built an interactive art exhibit called the Magic Morphin' Mirror'[Darrell *et al.*, 1998]. The goal of this project was to capture images, locate the faces, distort them in interesting ways, and present the image to the user all in real time, to give the appearance of a mirror which distorted faces. To achieve this goal, they used a variety of techniques to locate and track faces in real time, including skin color detection, change detection, and the upright face detector presented in this thesis. The exhibit was presented at the SIGGRAPH 1997 conference.

7.4 Summary

This chapter has given a summary of the published applications of the face detection techniques presented in this thesis. In general, the reactions from the authors of these systems has been quite positive.

However, some general problems have been noted, and should be addressed in future work. First, even with the techniques presented in Chapter 6, the system is sometimes not fast enough for real applications. In particular, security systems and robots must operate in near real time, and systems which process video or images from the WWW must deal with large quantities of images, so high speed is important.

Secondly, these projects have pointed out the limitations of being able to detect only upright frontal faces. Unfortunately, the current approaches for detecting tilted faces are somewhat slower, and for detecting profile faces are too inaccurate for use in other applications.

Chapter 8

Related Work

Compared with face recognition, the topic of face detection for its own sake was relatively unexplored until the early 1990s. Since then, however, partly due to the well known work by Sung [Sung, 1996], the field has become more developed. In this chapter, I will discuss some of this research. It can be broadly broken into two categories: The first category uses a template-based approach, in which statistics of the pixel intensities of a window of the image are used directly to detect the face. The second approach uses geometric information, by detecting the presence of particular geometric configurations of smaller features of the face. I will also briefly mention some commercial systems which include face detection components, although not much technical information is available about how they work.

In some cases, the authors of these systems have tested their algorithms on the same test sets I have used. Where possible, I will present performance comparisons among the algorithms. Most of the algorithms in the literature have focused on detection of upright faces, so most of the comparisons will use the *Upright Test Set* and the *FERET Test Set*.

In this thesis, a number of specific issues needed to be addressed, such as pose estimation, synthesizing face images, and feature selection, came up. The end of this chapter will describe some related approaches to these subproblems.

8.1 Geometric Methods for Face Detection

When computer vision was in its infancy, many researchers explored algorithms which extracted features from images, and used geometric constraints to understand the arrangements of these features. This was in part because of limited computational resources. The reduction in information from feature extraction made computer vision feasible on early computers.

It is natural then that some of the first face processing algorithms used this approach, and some of them are described in the following sections. Although this approach was initially motivated by limited computational resources, it remains valuable to this day.

8.1.1 Linear Features

One of the earliest projects on face recognition is described in [Kanade, 1973]. In order to detect and localize the face, the image (which was generally a mugshot) was converted to an edge image, and then projected onto the horizontal and vertical axes. By looking for particular patterns of peaks and valleys in these projections, the program could recognize the locations of the eyes, nose, mouth, borders of the face, and the hairline. If the patterns did not match the expected values, then the image was rejected as a nonface. Although the focus here was on recognition, this marks one of the first attempts to detect whether a face was present in an image.

The detector described in [Yang and Huang, 1994] uses an approach quite different from the ones presented above. Rather than having the computer learn the face patterns to be detected, the authors manually coded rules and feature detectors for face detection. Some parameters of the rules were then tuned based on a set of training images. Their algorithm proceeds in three phases. The first phase applies simple rules such as "the eyes should be darker than the rest of the face" to pixels in very low resolution windows ($4x4$ pixels) covering each potential faces. All windows that pass phase one are evaluted by phase two, which applies similar (but more detailed) rules to higher resolution 8×8 pixel windows. Finally, all surviving candidates are passed to phase three, which used edge-based features to classify the full-resolution window as either a face or a nonface. The test set consisted of 60 digitized television images and photographs, each containing one face. Their system was able to detect 50 of these faces, with 28 false detections.

8.1.2 Local Frequency or Template Features

Other researchers have located at the detection of more localized sub-features of the face. [Yow and Cippola, 1996] used elongated Gaussian filters (about 3 pixels wide) which were able to detect features like the corners of the eyes or the mouth. By looking for specific arrangements of these features, they were able to detect faces in an office environment. Since the corner features are so small, they are relatively invariant to the orientation of the face, making their detector reasonably robust to out of plane rotations.

The work presented in [Leung *et al.*, 1995, Burl and Perona, 1996] used correlation in the local frequency domain to locate sub-features of the face like the eyes, nose, and center of the mouth. Each of these features was not very robust, and gave a large number of false alarms. However, using a random graph matching technique, they were able to locate specific arrangements of these features which are present in faces. They use a statistical model of how the positions of each feature vary with respect to the others, built from real face images. Because the individual feature detectors and the graph matching algorithm are invariant to in-plane rotation, the overall detector is also invariant to in-plane rotation, like the work presented in Chapter 4.

8.2 Template-Based Face Detection

Many face detection systems are template-based; they encode facial images directly in terms of pixel intensities or colors. These images can be characterized by probabilistic models of the set of face images, or implicitly by neural networks or other mechanisms. The parameters for these models are adjusted either automatically from example images or by hand. The following subsections describe these methods in more detail.

8.2.1 Skin Color

One of the simplest methods to locate faces in images is to look for oval shaped regions of skin color. Many researchers have used this technique, including [Hunke, 1994, Yang and Waibel, 1996, Jones and Rehg, 1998, Choudhury *et al.*, 1999]. In this work, skin color was characterized in a normalized color space by a Gaussian distribution. This is the same model that I used for locating skin color regions in the Chapter 6 when speeding up my algorithm. Skin color based algorithms have two main advantages. First, they can be implemented to run at frame rate on normal workstations. Second, because they do not use specific facial features, they are robust to changes in the orientation of the head both in and out of plane.

8.2.2 Simple Templates

The disadvantages of skin color based methods is that if other skin color is present in the image (such as hands or arms), then these regions may be tracked by mistake. Some researchers have tried to use simple templates to complement the results of skin color matching [Birchfield, 1998]. These templates have ranged some ovals correlated with the edge image of the input, to correlation patterns for the skin-colored and non-skin-colored regions (like the eyes, hair, and lips). These techniques are able to improve the robustness of the color based detectors, but at the expense of speed. However, generally these trackers still need to be initialized with the starting location of the face.

8.2.3 Clustering of Faces and Non-Faces

Sung and Poggio developed a face detection system based on clustering techniques [Sung, 1996]. Their system, like ours, passes a small window over all portions of the image, and determines whether a face exists in each window. Their system uses a supervised clustering method with six face and six nonface clusters. Two distance metrics measure the distance of an input image to the prototype clusters, the first measuring the distance between the test pattern and the cluster's 75 most significant eigenvectors, and the second measuring the Euclidean distance between the test pattern and its projection in the 75 dimensional subspace. These distance measures have close ties with Principal Components Analysis (PCA), as described in [Sung, 1996]. The last step in their system is to use either a perceptron or a neural network with a hidden layer, trained to classify points using the two distances to each of the clusters. Their system is trained with 4000 positive examples and nearly 47500 negative examples collected in a bootstrap manner. In comparison, our system uses approximately 16000 positive examples and 9000 negative examples.

Table 8.2 shows the accuracy of their system on a set of 23 images (a portion of the *Upright Test Set*), along with the results of our system using the heuristics employed by Systems 10, 11, and 12 from Table 3.13. In [Sung, 1996], 149 faces were labelled in this test set, while we labelled 155 upright faces. Some of these faces are difficult for either system to detect. Assuming that Sung and Poggio were unable to detect any of the six additional faces we labelled, the number of faces missed by their system is six more than listed in their paper. Table 8.1 shows that for equal numbers of false detections, we can achieve slightly higher detection rates.

Table 8.1: Comparison of several face detectors on a subset of the *Upright Test Set*, which contains 23 images with 155 faces.

System	Detection Rate	False Alarms
10) Networks 1 and 2 → AND(0) → threshold(4,3) → overlap	81.3%	3
11) Networks 1 and 2 → threshold(4,2) → overlap → AND(4)	83.9%	8
12) Networks 1 and 2 → threshold(4,2) → overlap → OR(4) → threshold(4,1) → overlap	90.3%	38
Detector using a multi-layer network [Sung, 1996]	76.8%	5
Detector using perceptron [Sung, 1996]	81.9%	13
Support Vector Machine [Osuna *et al.*, 1997]	74.2%	20

The main computational cost for the system described in [Sung, 1996] is computing the two distance measures from each new window to the 12 clusters. We estimate that this computation requires fifty times as many floating point operations as are needed to classify a window in our system, where the main costs are in preprocessing and applying neural networks to the window.

8.2.4 Statistical Representations

In recent work, Colmenarez and Huang presented a statistically based method for face detection [Colmenarez and Huang, 1997]. Their system builds probabilistic models of the sets of faces and nonfaces, and compares how well each input window compares with these two categories. When applied to the *Upright Test Set*, their system achieves a detection rate between 86.8% and 98.0%, with between 6133 and 12758 false detections, respectively, depending on a threshold. These numbers should be compared to Systems 1 through 4 in Table 3.13, which have detection rates between 90.7% and 92.7%, with between 759 and 928 false detections. Although their false alarm rate is significantly higher, their system is quite fast. It would be interesting to use this system as a replacement for the candidate detector described in Chapter 6.

Another related system is described in [Pentland *et al.*, 1994]. This system uses PCA to describe face patterns (as well as smaller patterns like eyes) with a lower-dimensional space than the image space. Rather than detecting faces, the main goal of this work is to analyze images of faces, to determine head orientation or to recognize individual people. However, it is also possible to use this lower-dimensional space for detection. A window of the input image can be projected into the face space and then projected back into the image space. The difference between the original and reconstructed images is a measure of how close the image is to being a face. Although the results reported are quite good, it is unlikely that this system is as robust as [Sung, 1996], because Pentland's classifier is a special case of Sung and Poggio's system, using a single positive cluster rather than six positive and six negative clusters.

In more recent work, Moghoddam and Pentland's approach uses a two component distance measure (like [Sung, 1996]), but combines the two distances in a principled way based on the assumption that the distribution of each cluster is Gaussian [Moghaddam and Pentland, 1995b, Moghaddam and Pentland, 1995a]. The clusters are used together as a multi-modal Gaussian distribution, giving a probability distribution for all face images. Faces are detected by measuring how well each window of the input image fits the distribution, and setting a threshold. This detection technique has been applied to faces, and to the detection of smaller features like the eyes, nose, and mouth. In the latter case, the authors were able to get significantly improved performance by not requiring that all the smaller features be detected.

Moghaddam and Pentland's system, along with several others, was tested in the FERET evaluation of face recognition methods [Phillips *et al.*, 1996, Phillips *et al.*, 1997, Phillips *et al.*, 1998]. Although the actual detection error rates are not reported, an upper bound can be derived from the recognition error rates. The recognition error rate, averaged over all the tested systems, for frontal photographs taken in the same sitting is less than 2% (see the rank 50 results in Figure 4 of [Phillips *et al.*, 1997, Phillips *et al.*, 1998]). This means that the number of images containing detection errors, either false alarms or missing faces, was less than 2% of all images. Anecdotally, the actual error rate is significantly less than 2%. As shown in Table 3.15, our system using the configuration of System 11 achieves a 2% error rate on frontal faces. Given the large differences in performance of our system on *Upright Test Set* and the FERET images, it is clear that these two test sets exercise different portions of the system. The FERET images examine the coverage of a broad range of face types under good lighting with uncluttered backgrounds, while the *Upright Test Set* tests the robustness to variable lighting and cluttered backgrounds.

Another statistical representation was used by Schneiderman in his work on frontal and profile face detection [Schneiderman and Kanade, 1998]. Starting with the idea of building a complete model of the distribution of face images, he simplified the model by making a number of assumptions, until the model was tractable. These assumptions included quantizing smaller portions of the input image to fixed set of $\approx 10^6$ patterns, and modelling the frequency and location of those patterns in the image. Using a set of face images, a histogram of the face patterns and their locations could be built. The same general histogram model was used to model all the nonface images in a set of scenery images. Unlike the work presented in this thesis, his algorithm does not require iterative training; each example can be presented once. The results shown in Table 8.2 are quite good, in fact slightly more accurate than the results of Chapter 3. Currently, the main disadvantage of his system is the speed, in part because it uses 64×64 pixel windows to detect the faces, but ongoing work is addressing the issue of speed.

Table 8.2: Comparison of two face detectors on the *Upright Test Set*. The first three lines are results from Table 3.13 in Chapter 3, while the last three are from [Schneiderman and Kanade, 1998]. Note that the latter results exclude 5 images (24 faces) with hand-drawn faces from the complete set of 507 upright faces, because it uses more of the context like the head and shoulders which are missing from these faces.

System	Detection Rate	False Alarms
10) Networks 1 and 2 → AND(0) → threshold(4,3) → overlap	81.9%	8
11) Networks 1 and 2 → threshold(4,2) → overlap → AND(4)	86.0%	31
12) Networks 1 and 2 → threshold(4,2) → overlap → OR(4) → threshold(4,1) → overlap	90.1%	167
Histogram model [Schneiderman and Kanade, 1998]	77.0%	1
Histogram model [Schneiderman and Kanade, 1998]	90.5%	33
Histogram model [Schneiderman and Kanade, 1998]	93.0%	88

8.2.5 Neural Networks

The candidate verification process used to speed up our system, described in Chapter 6, is similar to the detection technique presented in [Vaillant *et al.*, 1994]. In that work, two networks were used. The first network has a single output, and like our system it is trained to produce a positive value for centered faces, and a negative value for nonfaces. Unlike our system, for faces that are not perfectly centered, the network is trained to produce an intermediate value related to how far off-center the face is. This network scans over the image to produce candidate face locations. The network must be applied at every pixel position, but it runs quickly because of the its architecture: using retinal connections and shared weights, much of the computation required for one application of the detector can be reused at the adjacent pixel position. This optimization requires the preprocessing to have a restricted form, such that it takes as input the entire image, and produces as output a new image. The nonlinear window-by-window preprocessing used in our system cannot be used. A second network is used for precise localization: it is trained to produce a positive response for an exactly centered face, and a negative response for faces which are not centered. It is not trained at all on nonfaces. All candidates which produce a positive response from the second network are output as detections. One possible problem with this work is that the negative training examples are selected manually from a small set of images (indoor scenes, similar to those used for testing the system). It may be possible to make the detectors more robust using the bootstrap training technique described here in this thesis and in [Sung, 1996].

8.2.6 Support Vector Machines

Osuna, Freund, and Girosi [Osuna *et al.*, 1997] have recently investigated face detection using a framework similar to that used in [Sung, 1996] and in our own work. However, they use a “support vector machine” to classify images, rather than a clustering-based method or a neural network. The support vector machine has a number of interesting properties, including the fact that it makes the boundary between face and nonface images more explicit [Burges, 1997]. The result of their system on the same 23 images used in [Sung, 1996] is given in Table 8.1; the accuracy is currently slightly poorer than the other two systems for this small test set.

8.3 Commercial Face Recognition Systems

In the last few years, a number of commercial face recognition systems have been developed, and most include some type of face detection capability [Velasco, 1998, Johnson, 1997, Wilson, 1996]. Not much has been published about how these systems work, apart from the articles above, although all of the companies claim to have excellent results.

8.3.1 Visionics

Visionics (`http://www.visionics.com/`) produces a face recognition toolkit called FaceIt. Anecdotally, this system has good recognition performance (much more robust than the basis eigenface system described in [Pentland *et al.*, 1994]). In the FERET evaluation, it achieved recognition rates of recognition scores of over 98% on images taken at the same sitting, putting an upper bound on the false negative error rate of about 2%.

The only information on the algorithm I could find published on the system were in a newspaper article [Velasco, 1998]. In that article, it is stated that the algorithm uses a variant of the eigenface technique, but one that concentrates on smaller features of the face (like the eyes, nose, and mouth) rather than the entire face. It is not clear whether this describes the face detection component as well, or only the recognition component.

Table 8.3: Results of the upright face detector from Chapter 3 and the FaceIt detectors for a subset of the *Upright Test Set* which contains 57 faces in 70 images.

System	Detection Rate	False Alarms
FaceIt	61.4%	0
Upright Face Detector	86.0%	6

Table 8.4: Results of the upright face detector from Chapter 3 and the FaceIt detectors on the *FERET Test Set*.

	Frontal Faces		15° Angle		22.5° Angle	
Number of Images		1001		241		378
Number of Faces		1001		241		378
System	Detection Rate	False Alarms	Detection Rate	False Alarms	Detection Rate	False Alarms
FaceIt Detector	98.7%	3	99.6%	0	96.0%	1
Upright Face Detector	99.2%	9	99.6%	2	94.7%	3

To better evaluate the accuracy of the system, the FaceIt system was applied to a subset of the images used to evaluate the upright face detector of Chapter 3. Although the FaceIt system has a mode which detects all faces in an image, the recommended mode for the highest accuracy is a single face-per-image mode. To perform a fair evaluation in this mode, only images with one (or zero) faces were used in the evaluation. The results for this subset of the *Upright Test Set* containing

70 images with 57 faces are presented in Table 8.3, and for the *FERET Test Set* in Table 8.4. As can be seen, the results of the two detectors are comparable for the *FERET Test Set*, while for the *Upright Test Set* FaceIt detects signicantly fewer faces. This difference in performances suggests that not only are the cleaenr faces images in the *FERET Test Set* easier to detect, but perhaps that the FaceIt detector is specifically tuned for such faces. It would be interesting to run the FaceIt software in the mode where it detects all faces, applied to images with more than one face. Detecting a single face in an image can be significantly easier than detecting all faces, one could simply detect all faces and only return the one with the highest confidence.

8.3.2 Miros

Miros produces another commercial face recognition system. According to [Velasco, 1998], it uses neural networks looking at smaller features of the face for recognition. No information is available about the recognition techniques or the system's accuracy.

8.3.3 Eyematic

The company Eyematic (`http://www.eyematic.com/`) is producing a face recognition product based on the work presented in [Wiskott *et al.*, 1996]. Although the work presented in that paper does not mention the problem of face detection, the measurement technique they describe should be able to distinguish faces from nonfaces. However, since the technique may be quite computationally expensive, they presumably use a simpler first pass to locate candidate faces. Their webpage states that the system uses color, depth, motion, and intensity patterns for detection, but does not say how it works.

8.3.4 Viisage

Viisage (`http://www.viisage.com/`) also produces face recognition software. Their website states that their recognition system is based on the eigenface work developed at MIT; presumably their face detection system also uses eigenfaces.

8.4 Related Algorithms

In this section, I will describe a number of algorithms related to problems seen in the this thesis, and justify the algorithms I chose to use.

8.4.1 Pose Estimation

Researchers have looked at the problem of how to recognize many poses of an object in an efficient way. One useful approach has been to compute eigenfeatures from the set of images of the object under different poses, thus reducing the dimensionality of the space so that a nearest-neighbor search can be effective [Nayar *et al.*, 1996, Neiberg *et al.*, 1996]. These techniques have been applied to object and pose recognition rather than object detection; in many cases, the object is already extracted from the background. Also, these systems have no explicit model of variation other than that caused by changes in object pose, making them brittle to such sources of variation.

Some work on eigenfaces has also included a simple pose estimation step [Pentland *et al.*, 1994]. In this work, training images from a number of poses were collected, and separate eigenspace were built for each pose. To classify a new image as a given pose, the distance reconstruction error between the input and each eigenspace was measured. The eigenspace with the smallest distance was considered the correct orientation. Although the accuracy and robustness of this technique might be appropriate for determining the overall pose, it is quite computationally expensive; the cost of projecting the image into a single eigenspace is almost the same as the neural network evaluation used in Chapters 4 and 5.

8.4.2 Synthesizing Face Images

The work of [Vetter *et al.*, 1992] suggests that for bilaterally symmetric objects (such as faces and cars), a large number of views of the object can be generated from a single view, given information about which points in the image are symmetric points on the object. This may provide a way to synthesize example images if real examples are scarce.

All of these techniques may provide ways to synthesize training images of different from a few examples. Although they might also be of use in synthesizing a frontal image from a potential face in a partial profile (like the approach in Section 5.2, these methods would require further development. They all require optical flow computation or other dense correspondences between the input image and the standard images in the database, which is quite computationally expensive to produce. This would be prohibitive for an algorithm must be run for windows at every pixel location in an image.

Chapter 9

Conclusions and Future Work

9.1 Conclusions

This thesis has demonstrated the effectiveness of detecting faces using a view based approach implemented with neural networks. Chapter 2 showed how to align training examples with one another, preprocess them to remove variation caused by lighting and camera parameters. To detect faces, each potential face region is classified according to its pose, and image plane normalizations are applied rotate the region to an upright orientation and improve contrast. The regions are passed through several neural networks which classify it as a face or nonface, and finally the results from these networks are arbitrated to give the final detection result.

The thesis has shown a series of face detectors, with varying degrees of sensitivity to the orientation of the faces. Chapter 3 presented an upright frontal face detector, which was able to detect between 77.9% and 90.3% of faces in a set of 130 test images, with an acceptable number of false detections. Depending on the application, the system can be made more or less conservative by varying the arbitration heuristics or thresholds used. The system has been tested on a wide variety of images, with many faces and unconstrained backgrounds. A fast version of the algorithm can process a 320x240 pixel image in about 1 second on an SGI O2 with a 175 MHz R10000 processor.

Chapter 4 described a version able to detect tilted faces, that is faces rotated in the image plane. The approach was to detect such faces by using a derotation network in combination with an upright face detector. This system is able to detect 85.7% of faces in a large test with tilted faces, with a small number of false positives. The capability of detecting tilted faces comes with a small expense in the detection rate. The technique is applicable to other template-based object detection schemes. A fast version of this system takes about 14 seconds to process a typical 320×240 pixel image on an SGI O2, and use of skin color and change detection provides an additional speedup.

Finally, Chapter 5 presented a generalization of the algorithm to handle faces rotated out of the image plane. This system is somewhat less accurate that the upright and tilted face detectors, but in some applications its additional capabilities may be useful. Chapter 7 demonstrated that these face detectors have been useful in practice, as evidenced by the number of other systems incorporating the upright face detector and applying it to real world data.

An additional contribution of this thesis is the methodology for building and training a face detector, beginning with suggestions for collecting and aligning training examples, through partitioning the examples into views, to training the classifier itself. In principle, the same techniques should apply to other objects.

9.2 Future Work

There are a number of directions for future exploration. The active learning algorithm introduced in Chapter 3 and used throughout this thesis could be refined in a number of ways, at the expense of more computation. Just as additional negative examples are chosen in an active manner during training, so could the positive examples. In particular, the small randomization of the position, orientation, and scale of the faces in the input windows could be performed during training, and new examples should be added only when the network misclassifies them. This may allow the system to be trained with a smaller number of face examples than is currently used.

In Chapter 5, it was stated that uniform backgrounds in example face images should replaced during training, to prevent the detector from learning to expect such backgrounds. In this thesis, the backgrounds were set randomly at the time the training set was built. Instead, they could be randomized during training. Also, the replacement backgrounds should be selected randomly from the scenery images rather than synthesized, which would improve their realism. Although these modifications are quite straightforward, they were not implemented in this thesis work simply because the additional computational expense would be too large.

Another training option to explore is to do away with active learning altogether. As was mentioned in Section 3.2.3, active learning is essentially a speed optimization. Since it changes the distribution of nonface examples seen by the network, it is not the "correct" thing to do, but it works well in practice. Training exhaustively on all the negative examples greatly increases the computational cost of training the system, but in the future it may be a feasible option. The preliminary experiment on this option in Section 3.5.4 gave slightly poorer results than active learning, perhaps due to the computationally limited amount of training. Training on the true distribution should improve the accuracy, assuming that enough training time and training examples available.

These points reveal that one of the limitations of the system is the iterative training required by the neural network used for pattern recognition. A number of statistical models mentioned in Chapter 8 on related work only require a single pass through the training data to build a histogram of the face and nonface images. Unfortunately, the fast statistical models are not particularly accurate [Colmenarez and Huang, 1997], and the accurate methods are (at the moment) quite slow [Schneiderman and Kanade, 1998]. Further work on such models may uncover a simple model of face images which can be trained quickly.

Another aspect of training that can be examined is the way that the training examples are aligned with one another. Throughout the work on this thesis, proper alignment of the features of the training examples was critical to the performance of the system. Currently, this is done by aligning manually labelled feature points. However, the neural networks do not see these feature locations directly; rather they see the intensity images. The right way to align them is to align the examples in image space. Given the amount of variability in the images themselves, this is a difficult problem, but one worthy of attention.

This work has focused as much as possible on using real example images, both faces and nonfaces, to train the detector. This approach requires large amounts of training data. There has been some research on building synthetic images of faces using three-dimensional or other types of models which may be applicable [Vetter and Blanz, 1998, Vetter *et al.*, 1997]. Recent work has also examined the problem of synthesizing realistic textures, which might provide a systematic way to generate background images [Bonet, 1997].

All of the work in this thesis, with the exception of a few experiments to speed up the system in Chapter 6, have used static grayscale images. When color or motion is available, there may be more information available for improving the accuracy of the detector. When an image sequence is available, temporal coherence can focus attention on particular portions of the images. As a face moves about, its location in one frame is a strong predictor of its location in next frame. Standard tracking methods, as well as expectation-based methods [Baluja, 1996], can be applied to focus the detector's attention. In addition, when a face cannot be detected in one frame, because of pose, lighting, or occlusion, it may be detectable in other frames.

Color information was used in Chapter 6 to speed up the algorithm, but it may also improve the accuracy. In particular, the detection network could be trained with color information. In this thesis, I avoided the use of color for two reasons. First, humans can easily locate faces in grayscale images, so it was interesting to see if a computer could do the same. A more pragmatic reason is that color would increase the number of inputs to the neural networks, making them slower, and requiring more training examples to train and generalize correctly. However, given appropriate training data, this additional data source might be valuable.

Although this thesis has concentrated on the detection of faces, there are other objects that one might want to detect. Most of the algorithm is general and could be applied to any type of object which has a relatively consistent appearance. Some preliminary work on detecting cars (specifically car tires) and eyes using the same techniques yielded promising results, but more domains should be explored.

Bibliography

[Baker and Nayar, 1996] S. Baker and S. K. Nayar. A theory of pattern rejection. In *ARPA Image Understanding Workshop*, Palm Springs, California, February 1996.

[Baluja, 1994] Shumeet Baluja. Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning. CMU-CS-94-163, Carnegie Mellon University, 1994. Also available at `ftp://reports.adm.cs.cmu.edu/usr/anon/1994/CMU-CS-94-163.ps`.

[Baluja, 1996] Shumeet Baluja. *Expectation-Based Selective Attention*. PhD thesis, Carnegie Mellon University Computer Science Department, October 1996. Available as CS Technical Report CMU-CS-96-182.

[Baluja, 1997] Shumeet Baluja. Face detection with in-plane rotation: Early concepts and preliminary results. JPRC-1997-001-1, Justsystem Pittsburgh Research Center, 1997. Also available at `http://www.cs.cmu.edu/~baluja/papers/baluja.face.in.plane.ps.gz`.

[Belhumeur and Kriegman, 1996] P. N. Belhumeur and D. J. Kriegman. What is the set of images of an object under all possible lighting conditions? In *Computer Vision and Pattern Recognition*, pages 270–277, San Francisco, California, 1996.

[Besl and Jain, 1985] Paul J. Besl and Ramesh C. Jain. Three-dimensional object recognition. *Computing Surveys*, 17(1):76–145, March 1985.

[Beymer *et al.*, 1993] David Beymer, Amnon Shashua, and Tomaso Poggio. Example based image analysis and synthesis. A.I. Memo 1431, MIT, November 1993.

[Birchfield, 1998] S. T. Birchfield. Elliptical head tracking using intensity gradients and color histograms. In *Computer Vision and Pattern Recognition*, pages 232–237, Santa Barbara, CA, June 1998.

[Bonet, 1997] J. S. De Bonet. Multiresolution sampling procedure for analysis and synthesis of texture images. In *Computer Graphics*, pages 361–368. ACM SIGGRAPH, August 1997.

[Burel and Carel, 1994] Gilles Burel and Dominique Carel. Detection and localization of faces on digital images. *Pattern Recognition Letters*, 15:963–967, October 1994.

[Burges, 1997] Christopher J. C. Burges. A tutorial on support vector machines for pattern recognition. To appear in *Data Mining and Knowledge Discovery*. Available at `http://svm.research.bell-labs.com/SVMdoc.html`, 1997.

[Burl and Perona, 1996] M. C. Burl and P. Perona. Recognition of planar object classes. In *Computer Vision and Pattern Recognition*, San Francisco, California, June 1996.

[Chin and Dyer, 1986] Roland T. Chin and Charles R. Dyer. Model-based recognition in robot vision. *Computing Surveys*, 18(1):67–108, March 1986.

[Choudhury *et al.*, 1999] T. Choudhury, B. Clarkson, T. Jebara, and A. Pentland. Multimodal person recognition using unconstrained audio and video. In *Second Conference on Audio- and Video-based Biometric Person Authentication*, 1999. To appear.

[Colmenarez and Huang, 1997] Antonio J. Colmenarez and Thomas S. Huang. Face detection with information-based maximum discrimination. In *Computer Vision and Pattern Recognition*, pages 782–787, San Juan, Puerto Rico, June 1997.

[Darrell *et al.*, 1998] T. Darrell, G. Gordon, M. Harville, and J. Woodfill. Integrated person tracking using stereo, color, and pattern detection. In *Computer Vision and Pattern Recognition*, pages 601–608, San Diego, California, June 1998.

[Drucker *et al.*, 1993] Harris Drucker, Robert Schapire, and Patrice Simard. Boosting performance in neural networks. *International Journal of Pattern Recognition and Artificial Intelligence*, 7(4):705–719, 1993.

[Frankel *et al.*, 1996] Charles Frankel, Michael J. Swain, and Vassilis Athitsos. WebSeer: An image search engine for the world wide web. TR-96-14, University of Chicago, August 1996.

[Gleicher and Witkin, 1992] Michael Gleicher and Andrew Witkin. Through-the-lens camera control. In *Computer Graphics*, pages 331–340. ACM SIGGRAPH, July 1992.

[Hertz *et al.*, 1991] John Hertz, Anders Krogh, and Richard G. Palmer. *Introduction to the Theory of Neural Computation*. Addison-Wesley Publishing Company, Reading, Massachusetts, 1991.

[Horprasert *et al.*, 1997] Thanarat Horprasert, Yaser Yacoob, and Larry S. Davis. An anthropometric shape model for estimating head. In *Third International Workshop on Visual Form*, Capri, Italy, May 1997.

[Hunke, 1994] H. Martin Hunke. Locating and tracking of human faces with neural networks. Master's thesis, University of Karlsruhe, 1994.

[Johnson, 1997] R. Colin Johnson. Face recognition provides security alternative. *Electronic Engineering Times*, page 36, July 1997.

[Jones and Rehg, 1998] Michael J. Jones and James M. Rehg. Stastical color models with apoplications to skin detection. Technical Report 98-11, Compaq Cambridge Research Laboratory, December 1998.

[Kanade, 1973] Takeo Kanade. *Picture Procesing System by Computer Complex and Recognition of Human Faces*. PhD thesis, Department of Information Science, Kyoto University, November 1973.

[Lawrence *et al.*, 1998] Steve Lawrence, I. Burns, A. D. Back, A. C. Tsoi, and C. Lee Giles. Neural network classification and prior class probabilities. In G. Orr, K.-R. Müller, and R. Caruana, editors, *Tricks of the Trade*, Lecture Notes in Computer Science State-of-the-Art Surveys, pages 299–314. Springer Verlag, 1998.

[Le Cun *et al.*, 1989] Y. Le Cun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropogation applied to handwritten zip code recognition. *Neural Computation*, 1:541–551, 1989.

[Leung *et al.*, 1995] T. K. Leung, M. C. Burl, and P. Perona. Finding faces in cluttered scenes using random labeled graph matching. In *Fifth International Conference on Computer Vision*, pages 637–644, Cambridge, Massachusetts, June 1995. IEEE Computer Society Press.

[Marquardt, 1963] Donald W. Marquardt. An algorithm for leant-squares estimation of nonlinear parameters. *Journal of the Society for Industrial and Applied Mathematics*, 11(2), June 1963.

[Moghaddam and Pentland, 1995a] Baback Moghaddam and Alex Pentland. Probabilistic visual learning for object detection. In *Fifth International Conference on Computer Vision*, pages 786–793, Cambridge, Massachusetts, June 1995. IEEE Computer Society Press.

[Moghaddam and Pentland, 1995b] Baback Moghaddam and Alex Pentland. A subspace method for maximum likelihood target detection. In *IEEE International Conference on Image Processing*, Washington, D.C., October 1995. Also available as MIT Media Laboratory Perceptual Computing Section Technical Report number 335.

[Nayar *et al.*, 1996] Shree K. Nayar, Sameer A. Nene, and Hiroshi Murase. Real-time 100 object recognition system. In *ARPA Image Understanding Workshop*, pages 1223–1227, Palm Springs, California, February 1996.

[Neiberg *et al.*, 1996] Leonard Neiberg, David Casasent, Robert Fontana, and Jeffrey E. Cade. Feature space trajectory neural net classifier: 8-class distortion-invarient tests. In *SPIE Applications and Science of Artificial Neural Networks*, volume 2760, pages 540–555, Orlando, FL, April 1996.

[Osuna *et al.*, 1997] Edgar Osuna, Robert Freund, and Federico Girosi. Training support vector machines: an application to face detection. In *Computer Vision and Pattern Recognition*, pages 130–136, San Juan, Puerto Rico, June 1997.

[Pentland *et al.*, 1994] Alex Pentland, Baback Moghaddam, and Thad Starner. View-based and modular eigenspaces for face recognition. In *Computer Vision and Pattern Recognition*, pages 84–91, June 1994.

[Phillips *et al.*, 1996] P. Jonathon Phillips, Patrick J. Rauss, and Sandor Z. Der. FERET (face recognition technology) recognition algorithm development and test results. ARL-TR-995, Army Research Laboratory, October 1996.

[Phillips *et al.*, 1997] P. Jonathon Phillips, Hyeonjoon Moon, Patrick Rauss, and Syed A. Rizvi. The FERET evaluation methodology for face-recognition algorithms. In *Computer Vision and Pattern Recognition*, pages 137–143, 1997.

[Phillips *et al.*, 1998] P. J. Phillips, H. Wechsler, J. Huang, and P. Rauss. The FERET database and evaluation procedure for face-recognition algorithms. *Image and Vision Computing*, 16(5):295–306, 1998.

[Pomerleau, 1992] Dean Pomerleau. *Neural Network Perception for Mobile Robot Guidance*. PhD thesis, Carnegie Mellon University, February 1992. Available as CS Technical Report CMU-CS-92-115.

[Press *et al.*, 1993] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, January 1993.

[Riklin-Raviv and Shashua, 1998] Tammy Riklin-Raviv and Amnon Shashua. The quotient image: Class bases recognition and synthesis under varying illumination conditions. Submitted for publication, 1998.

[Rowley *et al.*, 1998] Henry A. Rowley, Shumeet Baluja, and Takeo Kanade. Neural network-based face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(1):23–38, January 1998. Also available at `http://www.cs.cmu.edu/~har/faces.html`.

[Satoh and Kanade, 1996] Shin'ichi Satoh and Takeo Kanade. Name-it: Association of face and name in video. CMU-CS-96-205, Carnegie Mellon University, December 1996.

[Satoh and Kanade, 1997] Shin'ichi Satoh and Takeo Kanade. Name-it: Association of face and name in video. In *Computer Vision and Pattern Recognition*, pages 368–373, San Juan, Puerto Rico, June 1997.

[Schneiderman and Kanade, 1998] Henry Schneiderman and Takeo Kanade. Probabilistic modelling of local appearance and spatial relationships for object recognition. In *Computer Vision and Pattern Recognition*, Santa Barbara, CA, June 1998.

[Seutens *et al.*, 1992] Paul Seutens, Pascal Fua, and Andrew J. Hanson. Computational strategies for object recognition. *ACM Computing Surveys*, 24(1):5–61, March 1992.

[Sim *et al.*, 1999] T. Sim, R. Sukthankar, and S. Baluja. ARENA: A simple, high-performance baseline for frontal face recognition. Submitted to *Computer Vision and Pattern Recognition*, 1999.

[Smith and Kanade, 1996] M. Smith and T. Kanade. Skimming for quick browsing based on audio and image characterization. CMU-CS-96-186R, Carnegie Mellon University, May 1996.

[Smith and Kanade, 1997] Michael A. Smith and Takeo Kanade. Video skimming and characterization through the combination of image and language understanding techniques. In *Computer Vision and Pattern Recognition*, pages 775–781, San Juan, Puerto Rico, 1997.

[Smith, 1996] Alvy Ray Smith. Blue screen matting. In *Computer Graphics*, pages 259–268. ACM SIGGRAPH, August 1996.

[Sukthankar and Stockton, 1999] R. Sukthankar and R. Stockton. Argus: An automated multi-agent visitor identification system. Submitted to *Proceedings of the AAAI*, 1999.

[Sung, 1996] Kah-Kay Sung. *Learning and Example Selection for Object and Pattern Detection*. PhD thesis, MIT AI Lab, January 1996. Available as AI Technical Report 1572.

[Thrun *et al.*, 1999] S. Thrun, M. Bennewitz, W. Burgard, A.B. Cremers, F. Dellaert, D. Fox, D. Haehnel, C. Rosenberg, N. Roy, J. Schulte, and D. Schulz. MINERVA: A second generation mobile tour-guide robot. In *International Conference on Robotics and Automation*, 1999. In press.

[Umezaki, 1995] Tazio Umezaki. Personal communication, 1995.

[Vaillant *et al.*, 1994] R. Vaillant, C. Monrocq, and Y. Le Cun. Original approach for the localisation of objects in images. *IEE Proceedings on Vision, Image, and Signal Processing*, 141(4), August 1994.

[Velasco, 1998] Juan Velasco. Teaching the comptuer ot recognize a friendly face. *New York Times*, October 1998. Availabe at `http://www.miros.com/NY_Times_10-98.htm`.

[Vetter and Blanz, 1998] Thomas Vetter and Volker Blanz. Estimating coloured 3D face models from single images: An example based approach. In *European Conference on Computer Vision*, 1998.

[Vetter *et al.*, 1992] T. Vetter, T. Poggio, and H. Bülthoff. 3D object recognition: Symmetry and virtual views. A.I. Memo 1409, MIT, December 1992.

[Vetter *et al.*, 1997] Thomas Vetter, Michael J. Jones, and Tomaso Poggio. A bootstrapping algorithm for learning linear models of object classes. In *Computer Vision and Pattern Recognition*, pages 40–46, San Juan, Puerto Rico, June 1997.

[Wactlar *et al.*, 1996] H. Wactlar, T. Kanade, M. Smith, , and S. Stevens. Intelligent access to digital video: The informedia project. *IEEE Computer*, 29(5), May 1996. Special issue on the Digital Library Intiative.

[Waibel *et al.*, 1989] Alex Waibel, Toshiyuki Hanazawa, Geoffrey Hinton, Kiyohiro Shikano, and Kevin J. Lang. Phoneme recognition using time-delay neural networks. *Readings in Speech Recognition*, pages 393–404, 1989.

[Wilson, 1996] Dave Wilson. Neural nets help security systems face the facts. *Vision Systems Design*, pages 36–39, October 1996.

[Wiskott *et al.*, 1996] Laurenz Wiskott, Jean-Marc Fellous, Norbert Krüger, and Chistoph von der Malsburg. Face recognition by elastic bunch graph matching. 96-08, Institut für Neuroinformatik, Ruhr-Universität, Bochum, 1996.

[Yang and Huang, 1994] Gaungzheng Yang and Thomas S. Huang. Human face detection in a complex background. *Pattern Recognition*, 27(1):53–63, 1994.

[Yang and Waibel, 1996] J. Yang and A. Waibel. A real-time face tracker. In *Workshop on Applied Computer Vision*, pages 142–147, Sarasota, FL, 1996.

[Yow and Cippola, 1996] Kin Choong Yow and Roberto Cippola. Scale and orientation invariance in human face detection. In *British Machine Vision Conference*, 1996.

[Zhang and Fulcher, 1996] Ming Zhang and John Fulcher. Face recognition using artificial neural network group-based adaptive tolerance (GAT) trees. *IEEE Transactions on Neural Networks*, 7(3):555–567, 1996.

Index