Chapter 2

Eye Detection Using Support Vector Machines

2.1 Introduction

Software and hardware applications for human face image processing are becoming more ubiquitous. One form of biometrics uses human face recognition [WPB+98, GL02, LW02, CR02, Tit02, LY98] to identify or verify a person's identity. Such a system automatically extracts and recognises a person's face from any digitised image, usually obtained from a video or still camera. Its main uses are in surveillance, security and access control. Face processing is also being developed for non-security applications enhancing human computer interactions (HCI) [AMR02]. Systems have been developed that attempt to recognise human facial expressions and respond accordingly to the user's mood [CDCT+01].

In general, the first step of processing in any application that incorporates facial information is to locate the faces in the scene. This is also known as face detection. Interestingly, a recent study using magnetoencephalography (MEG) found evidence that a region in the human brain responds to faces in general whether or not the face is later recognised. This occurs at 100 ms after stimulus onset. Another region

of the brain seems to perform categorisation at a later stage to extract the identity of the face [LHK02].

Due to their importance in face processing applications many face detection algorithms have been researched and proposed. These include multiresolution [YH94], skin color [MGR98], texture [DN96], template [LTC95] and neural network based methods [RBK98a] to name a few. Support Vector Machines (SVM) have been used successfully for pattern classification and have been applied to face detection [ALX01, BKP01, SB02, SB98, SCB99, RTSB01, OFG97]. Related to face detection is eye detection. Naturally, once the eyes are located, the location of the face is also known. An eye detection approach using an infrared camera and based on the physiological properties of the eye is proposed by Haro et al. [HFE00]. Kalman trackers are used to model the dynamics of the eye/head and a probabilistic method is used to represent the appearance of the eye.

As pointed out by Yang et al. [YKA02] *face localisation* is a simplified form of *face detection* that assumes a single face in the input image. We propose an SVM based system for detecting left-right eye pairs of the face, which can then be used to implement face localisation or detection. Emphasis is placed on face localisation and its use in access control applications. In this situation we usually have some degree of control over environmental conditions and only one person attempts an access. For simplicity, we will disregard the possibility of background faces and photographs held up to the camera, but this does not imply that they are easy problems to solve. The XM2VTS face database (available at http://www.ee.surrey.ac.uk/Research/VSSP/xm2vtsdb) [MMK$^+$99, MHJ$^+$00] is used for our experiments because it contains one face per image taken in a controlled environment with variations in pose, facial expressions, facial hair and with and without eye glasses.

SVMs [CST00, SBS99] are learning machines that can be trained with respect to a higher dimensional feature space using learning algorithms from optimisation theory. It implements Structural Risk Minimisation, which minimises a bound on

the generalisation error by controlling the capacity of the machine. A non-linear feature space $\mathcal{A}$ is induced by the kernel k. Non-linear kernels are dot products on feature vectors $\mathbf{\Phi}(\mathbf{x})$ mapped from vectors $\mathbf{x}$ in the input space. The feature map $\mathbf{\Phi}$ is associated with k, and is normally defined in such a way that the feature vector resides in a higher dimensional space. Kernels with feature maps $\mathbf{\Phi} : \Re^N \rightarrow \mathcal{A}$ are defined as,

$$k(\mathbf{x}, \mathbf{y}) = \mathbf{\Phi}(\mathbf{x}) \cdot \mathbf{\Phi}(\mathbf{y}). \tag{2.1}$$

The map $\mathbf{\Phi}$ is non-linear for a non-linear kernel. The choice of $\mathbf{\Phi}$ can be explicit or implicit. Normally $\mathbf{\Phi}$ is defined implicitly by using a kernel function. A symmetric function $k(\mathbf{x}, \mathbf{y})$ on a finite input space $\mathcal{X}$ is a kernel function if, and only if, the matrix,

$$\mathbf{K} = (k(\mathbf{x}_i, \mathbf{x}_j))_{i,j=1}^{n} \tag{2.2}$$

is positive semi-definite for any finite subset $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \subset \mathcal{X}$.

The training algorithm finds the optimal decision hyperplane in $\mathcal{A}$ with a maximal margin by treating it as a linear problem in that space. The decision function associated with this hyperplane (for the two-class case) is given by

$$f(\mathbf{x}) = \sum_{i=1}^{l} y_i \alpha_i \mathbf{\Phi}(\mathbf{x}) \cdot \mathbf{\Phi}(\mathbf{x}_i) + b = \mathbf{w} \cdot \mathbf{\Phi}(\mathbf{x}) + b \tag{2.3}$$

where $\mathbf{x}_i$ are the support vectors, $y_i = \{-1, 1\}$ are the class labels, α_i are the parameters that solve the quadratic optimisation problem, $\mathbf{w} = \sum_{i=1}^{l} y_i \alpha_i \mathbf{\Phi}(\mathbf{x}_i)$ and b is a bias for the hyperplane. The value of $f(\mathbf{x})$ represents a decision on $\mathbf{x}$, where a positive value indicates that $\mathbf{x}$ is classified as class "1", and a negative value represents class "-1". The magnitude of $f(\mathbf{x})$ can be interpreted as the level of confidence in that classification.

Support vectors are subsets of the original set of training vectors. The number of support vectors are usually significantly less than the original number of vectors. However, the number of support vectors can still range in the thousands or more for some problems. With the run-time complexity proportional to the number of support vectors, non-linear SVMs are still too slow for many real-time applications.

Reduced set methods [SMB+99, OG99] have been proposed that successfully reduces the number of support vectors required to define the decision hyperplane.

Our system performs exhaustive searching of an input image using SVM classification for potential eye locations. Exhaustive searching has advantages over methods that use heuristics as there are fewer parameters to tinker with. An obvious disadvantage is the computational overhead. This is alleviated with our novel approach of combining the decision functions of an SVM with the Fast Fourier Transform (FFT) to perform correlation in the frequency domain, which ultimately creates a complete SVM classification map of the input image. The method requires simplified decision functions and we propose a new approach to approximating them for non-linear SVMs using only two vectors to reduce the number of required computations. This approach is along the lines of reduced set methods using two vectors. However, our vectors are defined differently. Our method produces a more accurate approximation of the decision hyperplane than the reduced set method, using the same number of vectors. This is demonstrated in experiments using the ORL database.

An SVM classification map of every point in the input image is generated, and a knowledge base is used to reduce the number of false eye locations. Knowing the location of the eyes permit faces to be normalised in terms of size and in-plane rotations with respect to the camera's view plane, which is useful for face processing applications such as face recognition.

In this chapter the word *image* has two different meanings. The first refers to an element of a range in relation to a function that assigns to each element of a domain a unique element in the range. The corresponding element in the domain is called the *pre-image*. The second meaning refers to the image that is a pixelised representation of a picture. The desired meaning in different parts of the chapter should be clear from the context in which it is used.

Section 2.2 of this thesis describes how simplified decision functions are obtained, followed by a section describing how SVMs can be combined with the FFT to achieve

fast exhaustive classification. Section 2.4 describes the eye detection system, which also contains a sub-section on experiments using the XM2VTS database to test the performance of the system.

2.2 Simplified Decision Functions

The eye detection system uses FFT-assisted correlation, which requires simplified SVM decision functions. In this section we will show how this can be achieved using any trained SVM. Solutions for linear and non-linear SVMs are treated separately because an analytically exact simplified decision function can be easily derived for linear SVMs. Non-linear SVMs require special treatment that are dependent on the kernel type.

2.2.1 Linear Kernel

As described in the literature, the decision function for a linear SVM can be simplified and expressed in terms of the vector $\mathbf{w}$,

$$f_L(\mathbf{x}) = \sum_{i=1}^{l} y_i \alpha_i \mathbf{x} \cdot \mathbf{x}_i + b = \sum_{i}^{l} y_i \alpha_i \mathbf{x}_i \cdot \mathbf{x} + b = \mathbf{w} \cdot \mathbf{x} + b. \tag{2.4}$$

2.2.2 Non-linear Kernels

Usually one cannot find $\mathbf{w}$ as in the linear case because $\mathbf{\Phi}$ is not known explicitly, but defined implicitly by the kernel function. The non-linear kernel functions treated in our work are the Gaussian radial basis function kernels,

$$k(\mathbf{x}, \mathbf{z}) = e^{-\gamma \|\mathbf{x}-\mathbf{z}\|^2} \tag{2.5}$$

and the inhomogeneous polynomial kernels,

$$k(\mathbf{x}, \mathbf{z}) = (\mathbf{x} \cdot \mathbf{z} + c)^d. \tag{2.6}$$

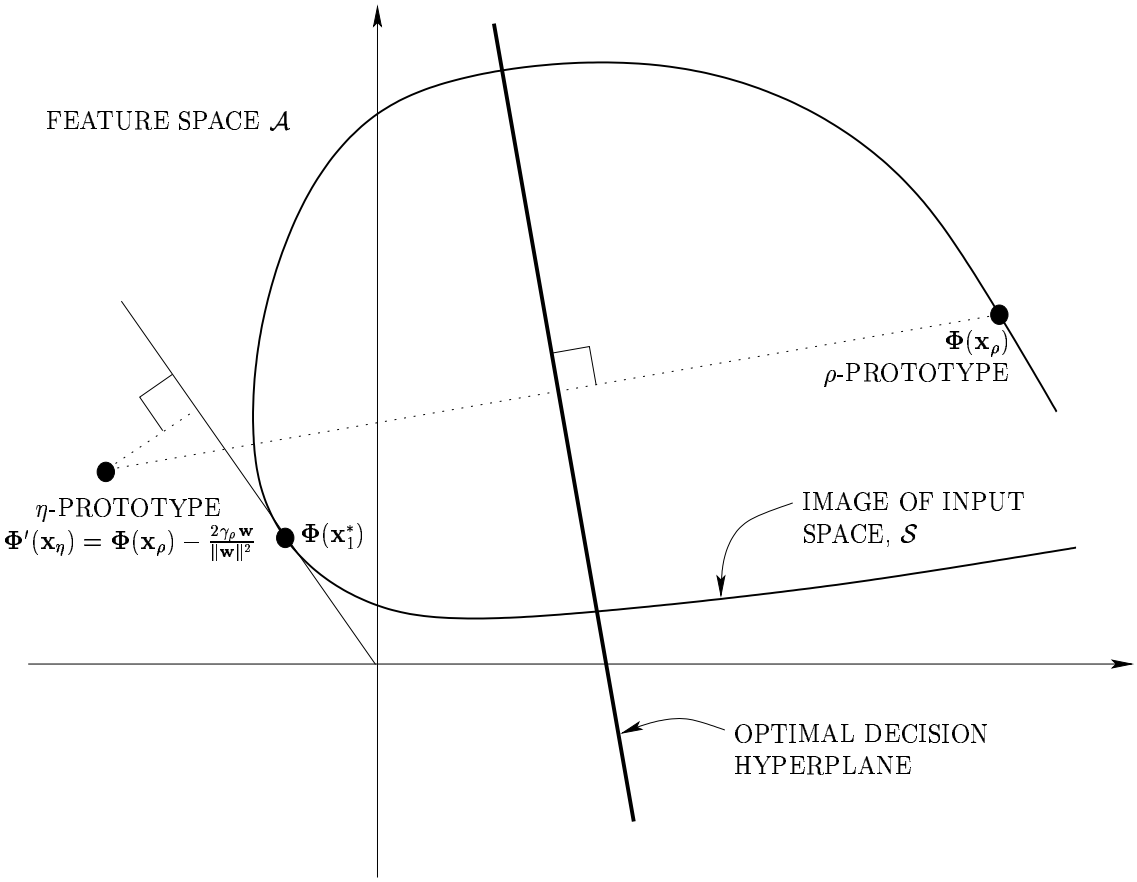


Figure 2.1: Illustration of the concept of η and ρ-prototype vectors used to represent the decision hyperplane of an SVM. The segment joining the two vectors is bisected by the hyperplane. The ρ-prototype is chosen so that it has a pre-image in the input space. In many cases the η-prototype does not have a pre-image so it is approximated by $c_1\mathbf{\Phi}(\mathbf{x}'_1)$.

If the input space is of dimension N, then the image of the input space will be at most an N-dimensional surface $\mathcal{S}$ in $\mathcal{A}$. Only those points on this surface have a corresponding pre-image in the input space [SMB+99].

The decision function of a trained non-linear SVM is simplified by approximating it with what we call ρ-prototype and η-prototype vectors. Figure 2.1 shows these prototypes in relation to the decision hyperplane. There are many ways to define these prototypes, but only one possibility is described here. The ρ-prototype vector is selected so that it lies on the surface $\mathcal{S}$. The simplest way to do this is to choose

one of the support vectors of the SVM. The η-prototype vector is then defined as,

$$\mathbf{\Phi}'(\mathbf{x}_\eta) = \mathbf{\Phi}(\mathbf{x}_\rho) - \frac{2\gamma_\rho \mathbf{w}}{\|\mathbf{w}\|^2} \tag{2.7}$$

where γ_ρ is the signed margin of the ρ-prototype to the decision hyperplane, defined as

$$\gamma_\rho = \mathbf{w} \cdot \mathbf{\Phi}(\mathbf{x}_\rho) + b. \tag{2.8}$$

With this definition of the η-prototype vector, the vector from the η to ρ prototype is normal to the decision hyperplane. The segment joining the prototypes is normal to, and bisected by, the decision hyperplane. The vector from the η to ρ prototype is a scaled version of $\mathbf{w}$, namely $\frac{2\gamma_\rho \mathbf{w}}{\|\mathbf{w}\|^2}$. The factor $\frac{2\gamma_\rho}{\|\mathbf{w}\|^2}$ is derivable by requiring the margin γ_η of the η-prototype to be equal to γ_ρ. We now state the equivalent SVM decision function:

Proposition 2.1 *Let η and ρ-prototypes represent vectors in feature space $\mathcal{A}$. Suppose the vector from the η-prototype to the ρ-prototype is a positively or negatively scaled version of $\mathbf{w}$ of a trained SVM. The decision function can then be expressed as*

$$\begin{aligned} f_N(\mathbf{x}) \;=\; & \frac{\|\mathbf{w}\|^2}{4\gamma_\rho} \{\mathbf{\Phi}'(\mathbf{x}_\eta) \cdot \mathbf{\Phi}'(\mathbf{x}_\eta) - \mathbf{\Phi}(\mathbf{x}_\rho) \cdot \mathbf{\Phi}(\mathbf{x}_\rho) - \\ & 2\left[\mathbf{\Phi}(\mathbf{x}) \cdot \mathbf{\Phi}'(\mathbf{x}_\eta) - \mathbf{\Phi}(\mathbf{x}) \cdot \mathbf{\Phi}(\mathbf{x}_\rho)\right]\} \end{aligned} \tag{2.9}$$

The parameter $\gamma_\rho = \mathbf{w} \cdot \mathbf{\Phi}(\mathbf{x}_\rho) + b$ is the signed margin of the ρ-prototype to the decision hyperplane.

Proof: Let the straight segment from the η-prototype to the ρ-prototype be normal to the decision hyperplane that also bisects it. And let the η-prototype be defined as shown in (2.7). Then,

$$\mathbf{\Phi}'(\mathbf{x}_\eta) \cdot \mathbf{\Phi}(\mathbf{x}) = \mathbf{\Phi}(\mathbf{x}_\rho) \cdot \mathbf{\Phi}(\mathbf{x}) - \frac{2\gamma_\rho}{\|\mathbf{w}\|^2} \mathbf{w} \cdot \mathbf{\Phi}(\mathbf{x}) \tag{2.10}$$

that is,

$$\mathbf{w} \cdot \mathbf{\Phi}(\mathbf{x}) = \frac{\|\mathbf{w}\|^2}{2\gamma_\rho} \left[\mathbf{\Phi}(\mathbf{x}_\rho) \cdot \mathbf{\Phi}(\mathbf{x}) - \mathbf{\Phi}'(\mathbf{x}_\eta) \cdot \mathbf{\Phi}(\mathbf{x})\right]. \tag{2.11}$$

The decision function can then be written as

$$\begin{aligned} f_N(\mathbf{x}) &= \mathbf{\Phi}(\mathbf{x}) \cdot \mathbf{w} + b \\ &= \mathbf{\Phi}(\mathbf{x}) \cdot \mathbf{w} - \mathbf{\Phi}(\mathbf{x}_\rho) \cdot \mathbf{w} + \gamma_\rho. \end{aligned} \tag{2.12}$$

It is desirable to remove references to $\mathbf{w}$ in the decision function as it is the term that requires the kernel evaluation of all support vectors, which is inefficient. References to $\|\mathbf{w}\|^2$ are acceptable as it can be easily calculated before-hand by evaluating $\sum_i^l \sum_j^l \alpha_i \alpha_j y_i y_j k(\mathbf{x}_i, \mathbf{x}_j)$ or approximated as $\sum_{i=1}^l \alpha_i$. Now, substituting (2.11) into (2.12) gives

$$\begin{aligned} f_N(\mathbf{x}) &= \frac{\|\mathbf{w}\|^2}{2\gamma_\rho} \left[\mathbf{\Phi}(\mathbf{x}_\rho) \cdot \mathbf{\Phi}(\mathbf{x}) - \mathbf{\Phi}'(\mathbf{x}_\eta) \cdot \mathbf{\Phi}(\mathbf{x})\right] - \mathbf{\Phi}(\mathbf{x}_\rho) \cdot \mathbf{w} + \gamma_\rho \\ &= \frac{\|\mathbf{w}\|^2}{2\gamma_\rho} \mathbf{\Phi}(\mathbf{x}_\rho) \cdot \mathbf{\Phi}(\mathbf{x}) - \frac{\|\mathbf{w}\|^2}{2\gamma_\rho} \mathbf{\Phi}'(\mathbf{x}_\eta) \cdot \mathbf{\Phi}(\mathbf{x}) - \frac{\|\mathbf{w}\|^2}{4\gamma_\rho} \mathbf{\Phi}(\mathbf{x}_\rho) \cdot \mathbf{\Phi}(\mathbf{x}_\rho) \\ &\quad + \frac{\|\mathbf{w}\|^2}{4\gamma_\rho} \mathbf{\Phi}(\mathbf{x}_\rho) \cdot \mathbf{\Phi}(\mathbf{x}_\rho) - \mathbf{\Phi}(\mathbf{x}_\rho) \cdot \mathbf{w} + \gamma_\rho \\ &= \frac{\|\mathbf{w}\|^2}{4\gamma_\rho} \{\mathbf{\Phi}'(\mathbf{x}_\eta) \cdot \mathbf{\Phi}'(\mathbf{x}_\eta) - \mathbf{\Phi}(\mathbf{x}_\rho) \cdot \mathbf{\Phi}(\mathbf{x}_\rho) - 2\left[\mathbf{\Phi}(\mathbf{x}) \cdot \mathbf{\Phi}'(\mathbf{x}_\eta)\right. \\ &\quad \left. - \mathbf{\Phi}(\mathbf{x}) \cdot \mathbf{\Phi}(\mathbf{x}_\rho)\right]\} \end{aligned} \tag{2.13}$$

as required. ■

2.2.2.1 Approximating the η-prototype

The map $\mathbf{\Phi}'$ is distinct from $\mathbf{\Phi}$, because in general $\mathbf{\Phi}'(\mathbf{x}_\eta) \in \mathcal{A}$ does not lie on $\mathcal{S}$. It is difficult to determine $\mathbf{\Phi}'$, so we aim to approximate $\mathbf{\Phi}'(\mathbf{x}_\eta)$ using $\mathbf{\Phi}$ induced by the kernel. This equates to solving the pre-image problem, an approximate solution to which is described in the literature. The solution used here has a similar vein to approximating some vector $\Psi \in \mathcal{A}$ with a reduced set expansion as described by Schölkopf et al. [SMB+99]. Specifically, $\mathbf{\Phi}'(\mathbf{x}_\eta)$ can be approximated with

$$\sum_{i=1}^{m} c_i \mathbf{\Phi}(\mathbf{x}'_i) \tag{2.14}$$

where m is the number of vectors, and $1 \leq m \leq m_a$. m is also the smallest integer such that

$$\left\| \mathbf{\Phi}'(\mathbf{x}_\eta) - \sum_{i=1}^{m} c_i \mathbf{\Phi}(\mathbf{x}'_i) \right\| < \tau_{h_1} \tag{2.15}$$

where τ_{h_1} is a preset threshold. If (2.15) is not satisfied, then $m = m_a$, where m_a is the upper bound on the number of terms. The vectors $\mathbf{x}'_i$ and their weights c_i can be determined by using reduced set selection or construction methods [SMB+99]. Our work focuses on the real-time application of SVMs to eye detection, so we concentrate on improving computational speed while not sacrificing too much on performance. This can be achieved by minimising the number of kernel evaluations in (2.9). Therefore, as well as for simplicity, we set m_a to 1 so that $\mathbf{\Phi}'(\mathbf{x}_\eta)$ is approximated by $c_1\mathbf{\Phi}(\mathbf{x}'_1)$. This requires only two kernel function evaluations to determine (2.9) for each $\mathbf{x}$ at run-time as evident in (2.22). The terms $\mathbf{\Phi}'(\mathbf{x}_\eta)\cdot\mathbf{\Phi}'(\mathbf{x}_\eta)$, $\mathbf{\Phi}(\mathbf{x}_\rho)\cdot\mathbf{\Phi}(\mathbf{x}_\rho)$ and $\frac{\|\mathbf{w}\|^2}{4\gamma_\rho}$ can be pre-calculated. Although only two vectors are used, namely $\mathbf{\Phi}(\mathbf{x}_\rho)$ and $\mathbf{\Phi}(\mathbf{x}'_1)$, a good approximation to the decision vector $\mathbf{w}$ is possible, as shown in the experiments described below.

The image vector $\mathbf{x}'_1$ and its corresponding weight c_1 can be determined by minimising the following objective function

$$h_1 = \|\mathbf{\Phi}'(\mathbf{x}_\eta) - c_1\mathbf{\Phi}(\mathbf{x}'_1)\|^2 = \left\|\mathbf{\Phi}(\mathbf{x}_\rho) - \frac{2\gamma_\rho\mathbf{w}}{\|\mathbf{w}\|^2} - c_1\mathbf{\Phi}(\mathbf{x}'_1)\right\|^2. \tag{2.16}$$

As pointed out by Schölkopf et al. [SMB+99], rather than minimising (2.16) it is preferable to minimise the distance from $\mathbf{\Phi}'(\mathbf{x}_\eta)$ to the orthogonal projection of $\mathbf{\Phi}'(\mathbf{x}_\eta)$ onto span $(\mathbf{\Phi}(\mathbf{x}'_1))$ because it is a lower-dimensional problem and is independent of c_1 so scaling problems are avoided. The problem in turn reduces to the minimisation of the following objective function

$$h_2 = -\frac{\mathbf{\Phi}'(\mathbf{x}_\eta)\cdot\mathbf{\Phi}(\mathbf{x}'_1)^2}{\mathbf{\Phi}(\mathbf{x}'_1)\cdot\mathbf{\Phi}(\mathbf{x}'_1)} = -\frac{\left[k(\mathbf{x}_\rho,\mathbf{x}'_1) - \frac{2\gamma_\rho}{\|\mathbf{w}\|^2}\sum_{i=1}^{l} y_i\alpha_i k(\mathbf{x}_i,\mathbf{x}'_1)\right]^2}{k(\mathbf{x}'_1,\mathbf{x}'_1)}. \tag{2.17}$$

We chose to minimise this objective function using unconstrained optimisation [PTVF92]. Gradient information with respect to $\mathbf{x}'_1$ is useful in achieving convergence. There are standard methods that do not require gradient information [PTVF92] but these have disadvantages such as slow convergence. This objective function can be evaluated in terms of kernel functions only. Explicit knowledge of $\mathbf{\Phi}$ is not required.

The gradients in terms of the inhomogeneous polynomial kernel are

$$\nabla_{\mathbf{x}'_1}(h_2^{\mathrm{P}}) = -\frac{2u}{t^d}\left[\mathbf{v} - \frac{du}{t}\mathbf{x}'_1\right] \tag{2.18}$$

where,

$$\begin{aligned} t &= \|\mathbf{x}'_1\|^2 + c \\ u &= [\mathbf{x}_\rho \cdot \mathbf{x}'_1 + c]^d - \frac{2\gamma_\rho}{\|\mathbf{w}\|^2}\sum_{i=1}^{l} y_i\alpha_i[\mathbf{x}_i \cdot \mathbf{x}'_1 + c]^d \\ \mathbf{v} &= d\left[[\mathbf{x}_\rho \cdot \mathbf{x}'_1 + c]^{d-1}\mathbf{x}_\rho - \frac{2\gamma_\rho}{\|\mathbf{w}\|^2}\sum_{i=1}^{l} y_i\alpha_i[\mathbf{x}_i \cdot \mathbf{x}'_1 + c]^{d-1}\mathbf{x}_i\right]. \end{aligned} \tag{2.19}$$

The gradients in terms of the radial basis function kernel are

$$\begin{aligned} \nabla_{\mathbf{x}'_1}(h_2^{\mathrm{R}}) &= -4\gamma\left[e^{-\gamma\|\mathbf{x}_\rho - \mathbf{x}'_1\|^2} - \frac{2\gamma_\rho}{\|\mathbf{w}\|^2}\sum_{i=1}^{l} y_i\alpha_i e^{-\gamma\|\mathbf{x}_i - \mathbf{x}'_1\|^2}\right] \times \\ &\quad \left[(\mathbf{x}_\rho - \mathbf{x}'_1)e^{-\gamma\|\mathbf{x}_\rho - \mathbf{x}'_1\|^2} - \frac{2\gamma_\rho}{\|\mathbf{w}\|^2}\sum_{i=1}^{l} y_i\alpha_i(\mathbf{x}_i - \mathbf{x}'_1)e^{-\gamma\|\mathbf{x}_i - \mathbf{x}'_1\|^2}\right] \end{aligned} \tag{2.20}$$

Once $\mathbf{x}_1$ has been determined the minimum of h_1 is found by setting

$$c_1 = \frac{\mathbf{\Phi}'(\mathbf{x}_\eta)\cdot\mathbf{\Phi}(\mathbf{x}'_1)}{\mathbf{\Phi}(\mathbf{x}'_1)\cdot\mathbf{\Phi}(\mathbf{x}'_1)} = \frac{k(\mathbf{x}_\rho, \mathbf{x}'_1) - \frac{2\gamma_\rho}{\|\mathbf{w}\|^2}\sum_{i=1}^{l} y_i\alpha_i k(\mathbf{x}_i, \mathbf{x}'_1)}{k(\mathbf{x}'_1, \mathbf{x}'_1)}. \tag{2.21}$$

Therefore, the decision function for a non-linear SVM can be approximated using the approximate η-prototype

$$f_N(\mathbf{x}) \approx f_1(\mathbf{x}) = \frac{\|\mathbf{w}\|^2}{4\gamma_\rho}\left\{c_1^2 k(\mathbf{x}'_1, \mathbf{x}'_1) - k(\mathbf{x}_\rho, \mathbf{x}_\rho) - 2\left[c_1 k(\mathbf{x}'_1, \mathbf{x}) - k(\mathbf{x}_\rho, \mathbf{x})\right]\right\}. \tag{2.22}$$

Because (2.22) is an approximation of the true decision function, the decision threshold can be adjusted to optimise performance on the *training* set. The optimal bias for each approximate decision function can be found by calculating the threshold that gives the maximum difference between the cumulative distributions of the scores obtained using the negative and positive *training* samples. In cases where the maximum difference is a plateau, meaning that the maximum occurs for a set of threshold values, the average threshold within this set is taken. However, it should

be noted that adjusting the threshold in this manner only optimises the performance for the training set. It is possible that better performance can be achieved on the testing set by not carrying out this modification.

In obtaining the expression in (2.22) we use the definition of the η-prototype given in (2.7). Now that the η-prototype is approximated with $c_1\mathbf{\Phi}(\mathbf{x}_1')$, we obtain the expression

$$c_1\mathbf{\Phi}(\mathbf{x}_1') \approx \mathbf{\Phi}(\mathbf{x}_\rho) - \frac{2\gamma_\rho \mathbf{w}}{\|\mathbf{w}\|^2}. \tag{2.23}$$

The fixed η-prototype vector $\mathbf{\Phi}'(\mathbf{x}_\eta)$ is approximated by $c_1\mathbf{\Phi}(\mathbf{x}_1')$. We can further improve this approximation by replacing γ_ρ with an optimal γ^* given by

$$\gamma^* = \frac{\gamma_\rho - \gamma_1}{2} \tag{2.24}$$

where $\gamma_1 = c_1\mathbf{w}\cdot\mathbf{\Phi}(\mathbf{x}_1') + b$. To explain why this is the case, we begin by saying that this optimisation is performed by keeping $c_1\mathbf{\Phi}(\mathbf{x}_1')$ fixed, while we vary the scaling factor in the second term of (2.23). In general,

$$c_1\mathbf{\Phi}(\mathbf{x}_1') \approx \mathbf{\Phi}'(\mathbf{x}_\eta) = \mathbf{\Phi}(\mathbf{x}_\rho) - \frac{c_s\mathbf{w}}{\|\mathbf{w}\|^2}. \tag{2.25}$$

As we vary c_s, the position of the η-prototype moves along the line joining it to the ρ-prototype, which is also normal to the decision hyperplane. Therefore, with $c_1\mathbf{\Phi}(\mathbf{x}_1')$ fixed, we aim to minimise

$$\begin{aligned} h_{c_s} &= \left\| c_1\mathbf{\Phi}(\mathbf{x}_1') - \mathbf{\Phi}(\mathbf{x}_\rho) + \frac{c_s\mathbf{w}}{\|\mathbf{w}\|^2} \right\|^2 \\ &= \|c_1\mathbf{\Phi}(\mathbf{x}_1') - \mathbf{\Phi}(\mathbf{x}_\rho)\|^2 + 2\left[c_1\mathbf{\Phi}(\mathbf{x}_1') - \mathbf{\Phi}(\mathbf{x}_\rho)\right]\cdot\frac{c_s\mathbf{w}}{\|\mathbf{w}\|^2} + \frac{c_s^2}{\|\mathbf{w}\|^2}. \end{aligned} \tag{2.26}$$

The minimum is then given by

$$\frac{\partial h_{c_s}}{\partial c_s} = 2\left[c_1\mathbf{\Phi}(\mathbf{x}_1') - \mathbf{\Phi}(\mathbf{x}_\rho)\right]\cdot\frac{\mathbf{w}}{\|\mathbf{w}\|^2} + 2\frac{c_s}{\|\mathbf{w}\|^2} = 0. \tag{2.27}$$

That is,

$$\begin{aligned} c_s &= \left[\mathbf{\Phi}(\mathbf{x}_\rho) - c_1\mathbf{\Phi}(\mathbf{x}_1')\right]\cdot\mathbf{w} \\ &= \left[\mathbf{\Phi}(\mathbf{x}_\rho)\cdot\mathbf{w} + b\right] - \left[c_1\mathbf{\Phi}(\mathbf{x}_1')\cdot\mathbf{w} + b\right] \\ &= \gamma_\rho - \gamma_1. \end{aligned} \tag{2.28}$$

The optimal γ^* is then given by

$$\gamma^* = \frac{c_s}{2}, \tag{2.29}$$

and therefore the improved approximation of $\mathbf{\Phi}'(\mathbf{x}_\eta)$ is

$$c_1\mathbf{\Phi}(\mathbf{x}'_1) \approx \mathbf{\Phi}'(\mathbf{x}_\eta) = \mathbf{\Phi}(\mathbf{x}_\rho) - \frac{2\gamma^*\mathbf{w}}{\|\mathbf{w}\|^2}. \tag{2.30}$$

We note that $\gamma^* = \gamma_\rho$ if $\gamma_1 = -\gamma_\rho$, that is the margin of $c_1\mathbf{\Phi}(\mathbf{x}'_1)$ is equal to the negative of the margin of $\mathbf{\Phi}(\mathbf{x}_\rho)$. In this case, no further improvements can be made to the approximation in (2.23).

Now, combining the use of γ^* and the readjustment of the decision threshold for optimal performance on the training set, we obtain the following approximate decision function

$$\begin{aligned} f_N(\mathbf{x}) &\approx f_1(\mathbf{x}) \\ &= \frac{\|\mathbf{w}\|^2}{4\gamma^*}\left\{c_1^2 k(\mathbf{x}'_1, \mathbf{x}'_1) - k(\mathbf{x}_\rho, \mathbf{x}_\rho) - 2\left[c_1 k(\mathbf{x}'_1, \mathbf{x}) - k(\mathbf{x}_\rho, \mathbf{x})\right]\right\} + \gamma_\rho - \gamma^* - b_T \end{aligned} \tag{2.31}$$

where b_T is the decision-threshold offset optimised for the training set as described above.

2.2.2.2 Choosing the ρ-prototype

One method of choosing the ρ-prototype is to simply perform exhaustive searching. That is, select each support vector of the SVM as the ρ-prototype and calculate h_1. However, this method is computationally expensive and can take a long time to complete when there are a large number of support vectors. Furthermore, simply choosing the ρ-prototype that minimises h_1 does not guarantee minimum classification errors. Errors in the approximate decision hyperplane will result in an increase in classification errors as the mean occurrence of the typical sample moves further away from the ρ and η-prototypes. This is illustrated in Figures 2.2 and 2.3.

We propose that the ρ-prototype be selected from one of three support vectors

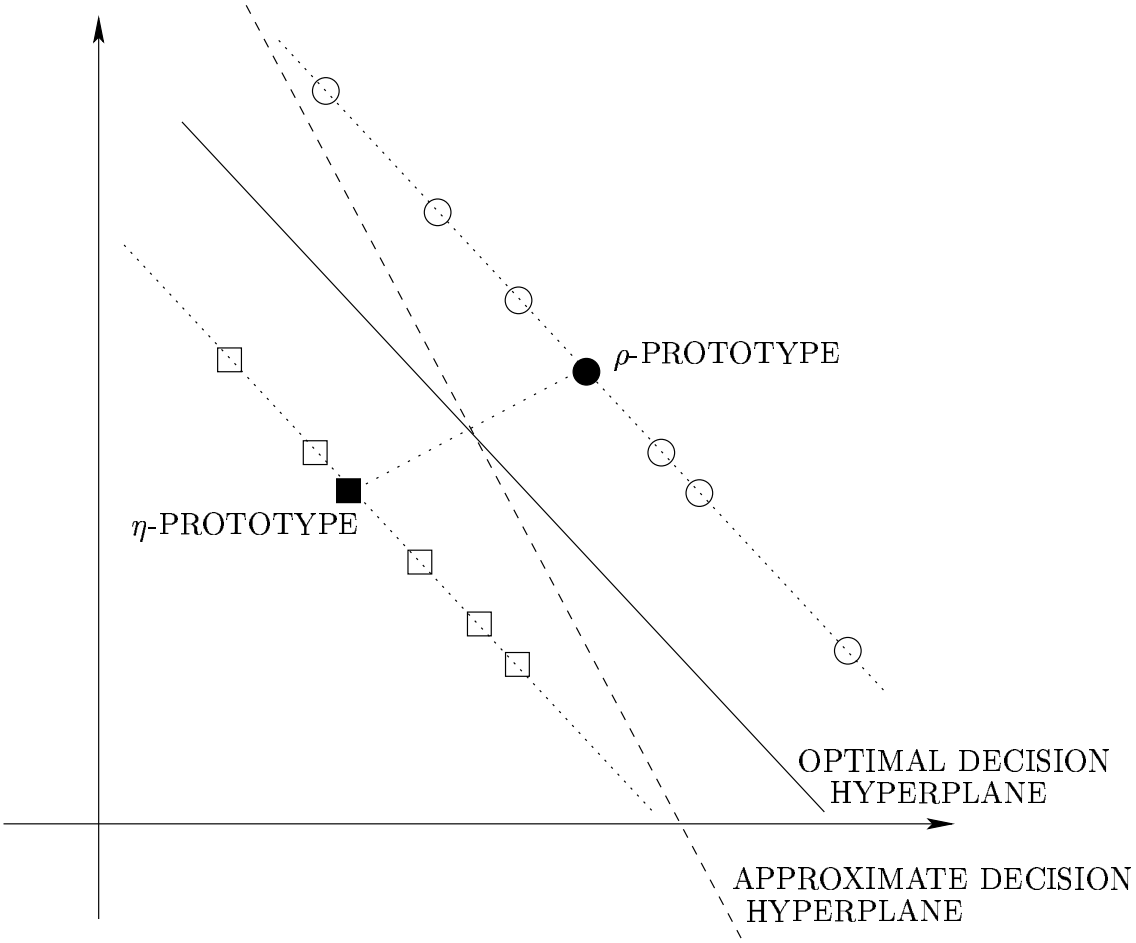


Figure 2.2: Illustration of the approximate decision hyperplane when the ρ-prototype is selected around the mean of one class. Although the approximate decision hyperplane does not represent the original decision hyperplane exactly, it still correctly separates the samples from the two classes. The circles and squares are support vectors of an SVM.

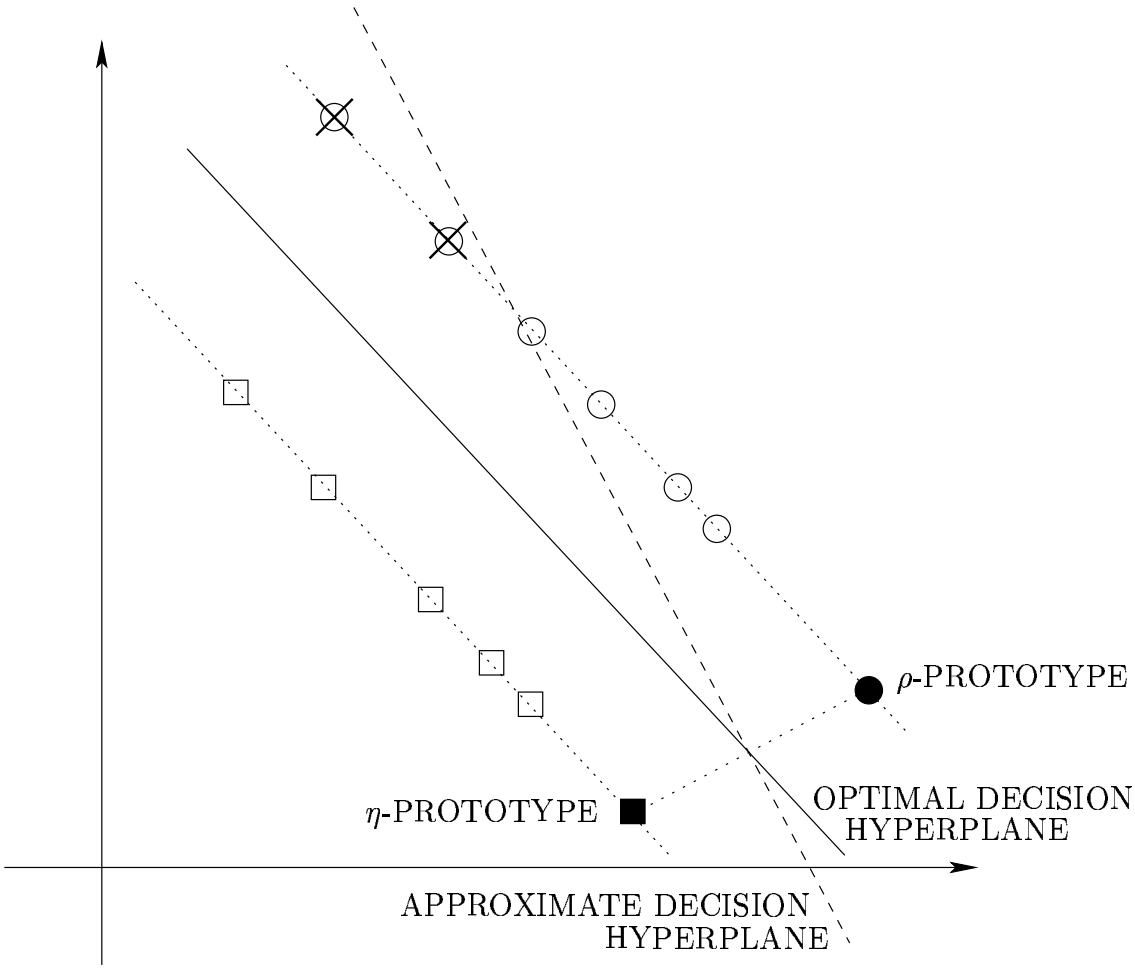


Figure 2.3: Illustration of the approximate decision hyperplane when the ρ-prototype is far from other vectors. A small error in the approximate decision hyperplane has resulted in two misclassified vectors, which are indicated with crosses. The number of classification errors increase as samples occur further away in a direction normal to the segment connecting the η and ρ-prototypes. The circles and squares are support vectors of an SVM.

$\{\mathbf{x}_{s_1}, \mathbf{x}_{s_2}, \mathbf{x}_{s_3}\}$, which minimises h_1. These support vectors are defined by

$$\begin{aligned}
\mathbf{x}_{s_1} &= \arg \min_{\mathbf{x}_i | i \in \mathrm{sv}^+} \left\| \mathbf{\Phi}(\mathbf{x}_i) - \frac{1}{n_{\mathrm{sv}}^+} \sum_{j \in \mathrm{sv}^+} \mathbf{\Phi}(\mathbf{x}_j) \right\|^2 \\
\mathbf{x}_{s_2} &= \arg \min_{\mathbf{x}_i | i \in \mathrm{sv}^-} \left\| \mathbf{\Phi}(\mathbf{x}_i) - \frac{1}{n_{\mathrm{sv}}^-} \sum_{j \in \mathrm{sv}^-} \mathbf{\Phi}(\mathbf{x}_j) \right\|^2 \\
\mathbf{x}_{s_3} &= \arg \min_{\mathbf{x}_i | i \in \mathrm{sv}} \left\| \mathbf{\Phi}(\mathbf{x}_i) - \frac{1}{n_{\mathrm{sv}}} \sum_{j \in \mathrm{sv}} \mathbf{\Phi}(\mathbf{x}_j) \right\|^2
\end{aligned} \tag{2.32}$$

where sv^+, sv^- and sv are the sets of indices of the positive, negative and all support vectors respectively. The number of vectors are represented by n_{sv}^+, n_{sv}^- and n_{sv} respectively. The vectors $\mathbf{x}_{s_1}$, $\mathbf{x}_{s_2}$ and $\mathbf{x}_{s_3}$ are the closest vectors to the center of mass of their respective classes. This method will perform well on data that have positive and negative classes clustered around their means with smaller variances than ones that are sparse and spread-out.

Another method of selecting the ρ-prototype is to select it randomly from the set of support vectors. This can be combined with the method just described above. If we select p number of random vectors, denoted by $\mathbf{x}_{r_1}$ to $\mathbf{x}_{r_p}$, we can then define the ρ-prototype vector as

$$\mathbf{x}_\rho = \arg \min_{\mathbf{y} \in \mathcal{P}} \left(\left\| \mathbf{\Phi}(\mathbf{y}) - \frac{2\gamma_\rho \mathbf{w}}{\|\mathbf{w}\|^2} - c_1 \mathbf{\Phi}(\mathbf{x}_1') \right\|^2 \right) \tag{2.33}$$

where $\mathcal{P} = \{\mathbf{x}_{s_1}, \mathbf{x}_{s_2}, \mathbf{x}_{s_3}, \mathbf{x}_{r_1}, \ldots, \mathbf{x}_{r_p}\}$. The term to be minimised in (2.33) is essentially h_1 as defined in (2.16). Alternatively, instead of using h_1 the recognition rate obtained on the training set can be used.

2.2.3 Experiments Using Simplified Decision Functions

We perform experiments to test the performance of SVMs with decision functions approximated using ρ and η prototypes. As mentioned above we use $m_a = 1$, that is, only one vector is used to approximate the η prototype. This means that two vectors are used, $\mathbf{\Phi}(\mathbf{x}_\rho)$ and $\mathbf{\Phi}(\mathbf{x}_1')$, to approximate the decision hyperplane. In particular, we use the approximate decision function given in (2.31), but we set $b_T = 0$ because

preliminary experiments show that slightly better results are obtained with this setting. The ρ-prototype is selected using the definition in (2.33), where p is set to 10. The approximate decision function generated by our prototype SVM method is compared with the reduced set approach proposed by Schölkopf et al. [SMB+99], where the number of reduced set vectors used is also set to two.

The ORL face database is used for performing face identification, without the rejection decision. That is, the identity of every unknown face presented to the system is assumed to have a corresponding match in the database. The ORL database consists of 40 people, each one with 10 face samples. The face samples for each person are labelled from 1 to 10. The following subsets of labels from each person are used for training:

- $S_1^R = \{1, 2, 3, 4, 5\}$;
- $S_2^R = \{1, 2, 4, 5, 9\}$;
- $S_3^R = \{1, 3, 7, 8, 10\}$; and
- $S_4^R = \{3, 5, 6, 8, 9\}$.

Those samples that are not used for training are used for testing, and they are:

- $S_1^T = \{6, 7, 8, 9, 10\}$;
- $S_2^T = \{3, 6, 7, 8, 10\}$;
- $S_3^T = \{2, 4, 5, 6, 9\}$; and
- $S_4^T = \{1, 2, 4, 7, 10\}$.

The training sets are used to create SVMs using the inhomogeneous polynomial kernel where $d = 3$ and $c = 1$. The one-against-all method is used for the multiclass classification encountered here. The resulting decision function is approximated using η and ρ prototypes, and also using the reduced set method with two vectors.

The performance of all classifiers are evaluated using the testing sets. The recognition results are summarised in Table 2.1. It shows that except for setting S_1 our prototype approximation method gives exactly the same results as the original SVMs, whereas the reduced set method gives inferior results for settings S_1 and S_3. The difference in performance is more evident when we examine the statistics of the difference between the original and approximated scores. Table 2.2 shows the statistics for the prototype and reduced set SVMs when compared to the original SVMs. The differences between the approximated and original scores are calculated, from which the following statistics are obtained:

- Average of the differences;
- Variance of the differences, given by $\frac{n\sum x^2-\left(\sum x\right)^2}{n^2}$, where x is the score difference and n is the total number of score differences;
- Minimum of the differences;
- Maximum of the differences;

Table 2.2 shows that the average score difference for the prototype SVM is about a factor of 10 times better than those from the reduced set SVM. Similarly, the prototype SVM's variance is roughly 5 times better. The minimum of the difference is better by a factor of about 5 to 175 times, and the maximum of the difference is better by a factor of about 3 times. Therefore, we have shown that the approximation of an original SVM decision function using prototype SVMs is decidedly more accurate than using reduced set SVMs.

In both prototype and reduced set methods, the decision threshold is adjusted to optimise performance on the *training* set. The reason for this is that both methods produce decision functions that are approximates of the true function. Better performance can be obtained by adjusting the bias for each approximate decision function, which in turn adjusts the decision threshold. The optimal bias for each approximate decision function can be found by calculating the threshold that

Partition	Orig SVM	Prototypes	Reduced Set
S_1	92.5 %	92.0 %	91.0 %
S_2	98.0 %	98.0 %	98.0 %
S_3	98.5 %	98.5 %	97.5 %
S_4	98.5 %	98.5 %	98.5 %

Table 2.1: Summary of results comparing the performance of prototype SVMs and reduced set SVMs. The results are obtained from face identification experiments using the ORL face database. The partition S_1 corresponds to using the subset S_1^R for training and S_1^T for testing. The same applies to S_2, S_3 and S_4.

gives the maximum difference between the unnormalised cumulative distributions of the scores obtained using the negative and positive *training* samples. In cases where the maximum difference is a plateau, meaning that it occurs for a series of threshold values, the average threshold within this set is taken.

2.3 Correlation Using Decision Functions

One method of human eye detection using SVMs is to perform classification at each possible location within a candidate image and indicate those that are likely to be human eyes based on the classification scores. This is essentially an exhaustive search. We chose to investigate this approach as opposed to other methods using heuristics because it can produce better results. Heuristics have been used to reduce search time complexity, but has the tendency to miss possible positive matches, or returns too many false positives if not defined properly. A robust method is to perform exhaustive searching, but carries with it the obvious drawback of long search times.

The previous section described how the decision function of SVMs can be simplified. Those simplifications re-express any decision function as a few simple dot products and terms that can be computed efficiently. The dot products operating on real data and computed at each possible location within a candidate image is equivalent to correlation in the spatial domain, which can be computed efficiently by performing

Partition	Average Diff.	Variance of Diff.	Min. Diff.	Max. Diff.
S_1				
Prototypes	0.004962	0.000279	0.000033	0.047182
Reduced Set	0.06384	0.001614	0.001169	0.158722
S_2				
Prototypes	0.010673	0.000219	0.000022	0.065178
Reduced Set	0.071415	0.001427	0.003878	0.203663
S_3				
Prototypes	0.006169	0.000342	0.000022	0.084046
Reduced Set	0.075357	0.001596	0.000834	0.283697
S_4				
Prototypes	0.008212	0.000305	0.000078	0.066704
Reduced Set	0.074212	0.001676	0.00035	0.191854

Table 2.2: Summary of results comparing the deviations of the approximate SVMs to the original SVMs. For each test image, the difference between the approximate and original decision function values is computed. This is then used to calculate the average, variance, minimum and maximum of the difference. The partition S_1 corresponds to using the subset S_1^R for training and S_1^T for testing. The same applies to S_2, S_3 and S_4.

the equivalent operation of multiplication in the frequency domain. Transformation of spatial data to the frequency domain is achieved by using the FFT. Increases in desktop computing power allows us to perform the required FFT operations plus the construction of the decision function in real-time without special hardware assistance.

2.3.1 Vectors and Images

An image is represented by a two-dimensional matrix. It also has an equivalent vector representation, normally created by row-wise lexicographic ordering of the matrix elements. Both representations are used in this section. We will denote an image as $\mathbf{X}$ and its corresponding vector as $\mathbf{x}$. The elements of an image are referenced by $x(i, j)$ and the elements of a vector are referenced by $x(i)$. Thus, in the analysis above the decision vector $\mathbf{w}$ has an equivalent matrix representation

$\mathbf{W}$. All image pixel values are assumed to have a normalised range, such as $[0, 1]$, for the purposes of training and testing SVMs.

2.3.2 Correlation Theorem

A well-known method for determining correlations in the time/spatial domain is to use the frequency domain, where multiplications are performed [Pou96]. Given images $f(i, j)$ and $g(i, j)$ with size $a \times b$ and $c \times d$ respectively, zero padded images $f_z(i, j)$ and $g_z(i, j)$ are created. Both zero padded images have size $m \times n$ with $m \geq a + c - 1$ and $n \geq b + d - 1$. Zero padding is required to avoid wraparound errors. Correlation is then defined (for the discrete case) as

$$f_z(i, j) \circ g_z(i, j) = \frac{1}{mn} \sum_{k=0}^{m-1} \sum_{l=0}^{n-1} f_z^*(k, l) g_z(i + k, j + l) \tag{2.34}$$

where * is the complex conjugate. Let $F(u, v) = \mathcal{F}[f(i, j)]$ and $G(u, v) = \mathcal{F}[g(i, j)]$ represent the two-dimensional Discrete Fourier Transforms of $f(i, j)$ and $g(i, j)$ respectively. The correlation theorem then states that

$$\mathcal{F}[f(i, j) \circ g(i, j)] = F^*(u, v) G(u, v) \tag{2.35}$$

2.3.3 Correlation Using Linear Kernels

Given an input image $\mathbf{X}$ from which we want to extract possible eye positions, we calculate $f_L(\mathbf{x}^{ij})$ as defined by (2.4) at each (i, j) position within $\mathbf{x}$. The matrix $\mathbf{X}^{ij}$ has the same dimensions as the template $\mathbf{W}$ and are sub-images of $\mathbf{X}$. The matrix $\mathbf{W}$ can be viewed as a weighted template for the correlation. The bias b is added to the correlation results to obtain the decision function response at each position. A map of possible eye locations $\mathbf{M}_L$ can be created,

$$\begin{aligned} m_L(i, j) = f_L(\mathbf{x}^{ij}) &= \frac{1}{mn} \sum_{k=0}^{m-1} \sum_{l=0}^{n-1} w_z^*(k, l) x_z(i + k, j + l) + b \\ &= \left(\mathcal{F}^{-1}[W_z^*(u, v) X_z(u, v)]\right)(i, j) + b \end{aligned} \tag{2.36}$$

where $\mathcal{F}^{-1}$ is the inverse two-dimensional Discrete Fourier Transform. Matching scores for possible eye locations are stored in the entries of the matrix $\mathbf{M}_L$.

2.3.4 Correlation Using Non-linear Kernels

To perform correlation with non-linear kernels we have to consider each kernel separately. We will demonstrate using inhomogeneous polynomial and radial basis function kernels. The approximate decision function f_1 as defined by (2.22) is used to create the correlations. Given an input image $\mathbf{X}$ from which we want to extract possible eye positions, we calculate the approximate decision function $f_1(\mathbf{x}^{ij})$ at each (i, j) position within $\mathbf{x}$. The matrix $\mathbf{X}^{ij}$ has the same dimensions as the training images and are sub-images of $\mathbf{X}$.

2.3.4.1 Correlation Using Inhomogeneous Polynomial Kernels

The major run-time computational component of the inhomogeneous polynomial kernel given by (2.6) is the dot product. Examining (2.22) reveals that the only terms that require evaluation at run-time are

$$k\left(\mathbf{x}'_1, \mathbf{x}^{ij}\right) = \left(\mathbf{x}'_1 \cdot \mathbf{x}^{ij} + c\right)^d \tag{2.37}$$

and

$$k\left(\mathbf{x}_\rho, \mathbf{x}^{ij}\right) = \left(\mathbf{x}_\rho \cdot \mathbf{x}^{ij} + c\right)^d. \tag{2.38}$$

The terms $k(\mathbf{x}'_1, \mathbf{x}'_1)$, $k(\mathbf{x}_\rho, \mathbf{x}_\rho)$ and $\frac{\|\mathbf{w}\|^2}{4\gamma_\rho}$ can be calculated once and stored for later use. In general, only terms involving the unknown $\mathbf{x}^{ij}$ need to be evaluated at run-time.

Let $c_a = c_1^2(\|\mathbf{x}'_1\|^2 + c)^d - (\|\mathbf{x}_\rho\|^2 + c)^d$, $\mathbf{y} = \mathbf{x}'_1$ and $\mathbf{z} = \mathbf{x}_\rho$. A map of possible eye locations $\mathbf{M}_{1P}$ can be created,

$$\begin{aligned} m_{1P}(i,j) &= f_1(\mathbf{x}^{ij}) \\ &= \frac{\|\mathbf{w}\|^2}{4\gamma_\rho}\left\{c_a - 2\left[c_1\left[\frac{1}{mn}\sum_{k=0}^{m-1}\sum_{l=0}^{n-1} y_z^*(k,l)x_z(i+k,j+l) + c\right]^d -\right.\right. \end{aligned}$$

$$\left[\frac{1}{mn}\sum_{k=0}^{m-1}\sum_{l=0}^{n-1} z_z^*(k,l)x_z(i+k,j+l)+c\right]^d\Bigg]\Bigg\} + \gamma_\rho - \gamma^*$$
$$= \frac{\|\mathbf{w}\|^2}{4\gamma_\rho}\left\{c_a - 2\left[c_1\left[y_x(i,j)+c\right]^d - \left[z_x(i,j)+c\right]^d\right]\right\} + \gamma_\rho - \gamma^* \tag{2.39}$$

where

$$y_x(i,j) = (\mathcal{F}^{-1}[Y_z^*(u,v)X_z(u,v)])(i,j) \tag{2.40}$$

and

$$z_x(i,j) = (\mathcal{F}^{-1}[Z_z^*(u,v)X_z(u,v)])(i,j). \tag{2.41}$$

The Fourier Transforms $Y_z^*(u,v)$ and $Z_z^*(u,v)$ can be calculated offline. Therefore, at run-time one forward Fourier Transform must be computed to create $X_z(u,v)$ plus two inverse Fourier Transforms must be applied for the correlations.

2.3.4.2 Correlation Using Radial Basis Function Kernels

Examining (2.22) reveals that the only terms that require evaluation at run-time are

$$k\left(\mathbf{x}_1', \mathbf{x}^{ij}\right) = e^{-\gamma\left\|\mathbf{x}_1' - \mathbf{x}^{ij}\right\|^2} = e^{-\gamma\left(\left\|\mathbf{x}_1'\right\|^2 - 2\mathbf{x}_1'\cdot\mathbf{x}^{ij} + \left\|\mathbf{x}^{ij}\right\|^2\right)} \tag{2.42}$$

and

$$k\left(\mathbf{x}_\rho, \mathbf{x}^{ij}\right) = e^{-\gamma\left\|\mathbf{x}_\rho - \mathbf{x}^{ij}\right\|^2} = e^{-\gamma\left(\|\mathbf{x}_\rho\|^2 - 2\mathbf{x}_\rho\cdot\mathbf{x}^{ij} + \left\|\mathbf{x}^{ij}\right\|^2\right)}. \tag{2.43}$$

As in the case of the inhomogeneous polynomial kernel, the terms $k(\mathbf{x}_1', \mathbf{x}_1')$, $k(\mathbf{x}_\rho, \mathbf{x}_\rho)$ and $\frac{\|\mathbf{w}\|^2}{4\gamma_\rho}$ can be calculated before run-time and only terms involving the unknown $\mathbf{x}^{ij}$ need to be evaluated at run-time. From (2.42) and (2.43) the major run-time computational components are $\|\mathbf{x}^{ij}\|^2$, $\mathbf{x}_1' \cdot \mathbf{x}^{ij}$ and $\mathbf{x}_\rho \cdot \mathbf{x}^{ij}$. The terms $\|\mathbf{x}_1'\|^2$ and $\|\mathbf{x}_\rho\|^2$ can be calculated before-hand. Correlation with respect to the two dot product components $\mathbf{x}_1' \cdot \mathbf{x}^{ij}$ and $\mathbf{x}_\rho \cdot \mathbf{x}^{ij}$ can be computed using the methods described above. The term

$$\left\|\mathbf{x}^{ij}\right\|^2 = \mathbf{x}^{ij}\cdot\mathbf{x}^{ij} = \sum_k \left(x^{ij}(k)\right)^2 \tag{2.44}$$

must be treated separately because the above methods cannot be applied and simply calculating this term for each (i, j) is too slow. Efficient computation of (2.44) for each (i, j) is achieved by exploiting its computational redundancies. The slowest operation in the computation of (2.44) is the multiplication. It suffices to compute the Hadamard product $\mathbf{X} \odot \mathbf{X}$ once and generate $\|\mathbf{x}^{ij}\|^2$ by adding the appropriate entries. Furthermore, it can be observed that $\|\mathbf{x}^{ij}\|^2$ and $\left\|\mathbf{x}^{(i+\Delta i)(j+\Delta j)}\right\|^2$ share many common terms when Δi and Δj are small enough. Therefore, we can compute $\left\|\mathbf{x}^{i(j+1)}\right\|^2$ based on the results from $\|\mathbf{x}^{ij}\|^2$ by removing and adding the required column. Similarly, $\left\|\mathbf{x}^{(i+1)j}\right\|^2$ can be computed by removing and adding the required row. The initial value $\|\mathbf{x}^{11}\|^2$ is computed by summing the required entries. Figure 2.4 illustrates this process.

Let $c_b = c_1^2 - 1$, $\mathbf{y} = \mathbf{x}'_1$ and $\mathbf{z} = \mathbf{x}_\rho$. A map of possible eye locations $\mathbf{M}_{1R}$ can be created,

$$\begin{aligned} m_{1R}(i,j) &= f_1(\mathbf{x}^{ij}) \\ &= \frac{\|\mathbf{w}\|^2}{4\gamma_\rho}\left\{c_b - 2\left[c_1 e^{-\gamma\left[\left\|\mathbf{x}'_1\right\|^2 - \frac{2}{mn}\sum_{k=0}^{m-1}\sum_{l=0}^{n-1} y_z^*(k,l)x_z(i+k,j+l) + \left\|\mathbf{x}^{ij}\right\|^2\right]} - \right. \\ &\quad \left. e^{-\gamma\left[\|\mathbf{x}_\rho\|^2 - \frac{2}{mn}\sum_{k=0}^{m-1}\sum_{l=0}^{n-1} z_z^*(k,l)x_z(i+k,j+l) + \left\|\mathbf{x}^{ij}\right\|^2\right]}\right]\right\} + \gamma_\rho - \gamma^* \\ = \frac{\|\mathbf{w}\|^2}{4\gamma_\rho}&\left\{c_b - 2\left[c_1 e^{-\gamma\left[\left\|\mathbf{x}'_1\right\|^2 - 2y_x(i,j) + \left\|\mathbf{x}^{ij}\right\|^2\right]} - e^{-\gamma\left[\|\mathbf{x}_\rho\|^2 - 2z_x(i,j) + \left\|\mathbf{x}^{ij}\right\|^2\right]}\right]\right\} + \gamma_\rho - \gamma^* \end{aligned} \tag{2.45}$$

where

$$y_x(i,j) = (\mathcal{F}^{-1}[Y_z^*(u,v)X_z(u,v)])(i,j) \tag{2.46}$$

and

$$z_x(i,j) = (\mathcal{F}^{-1}[Z_z^*(u,v)X_z(u,v)])(i,j). \tag{2.47}$$

The Fourier Transforms $Y_z^*(u,v)$ and $Z_z^*(u,v)$ can be calculated offline. As in the case with inhomogeneous polynomial kernels, at run-time one forward Fourier Transform must be computed to create $X_z(u,v)$ plus two inverse Fourier Transforms must be applied for the correlations. In addition, $\|\mathbf{x}^{ij}\|^2$ must also be evaluated at run-time using the method described above.

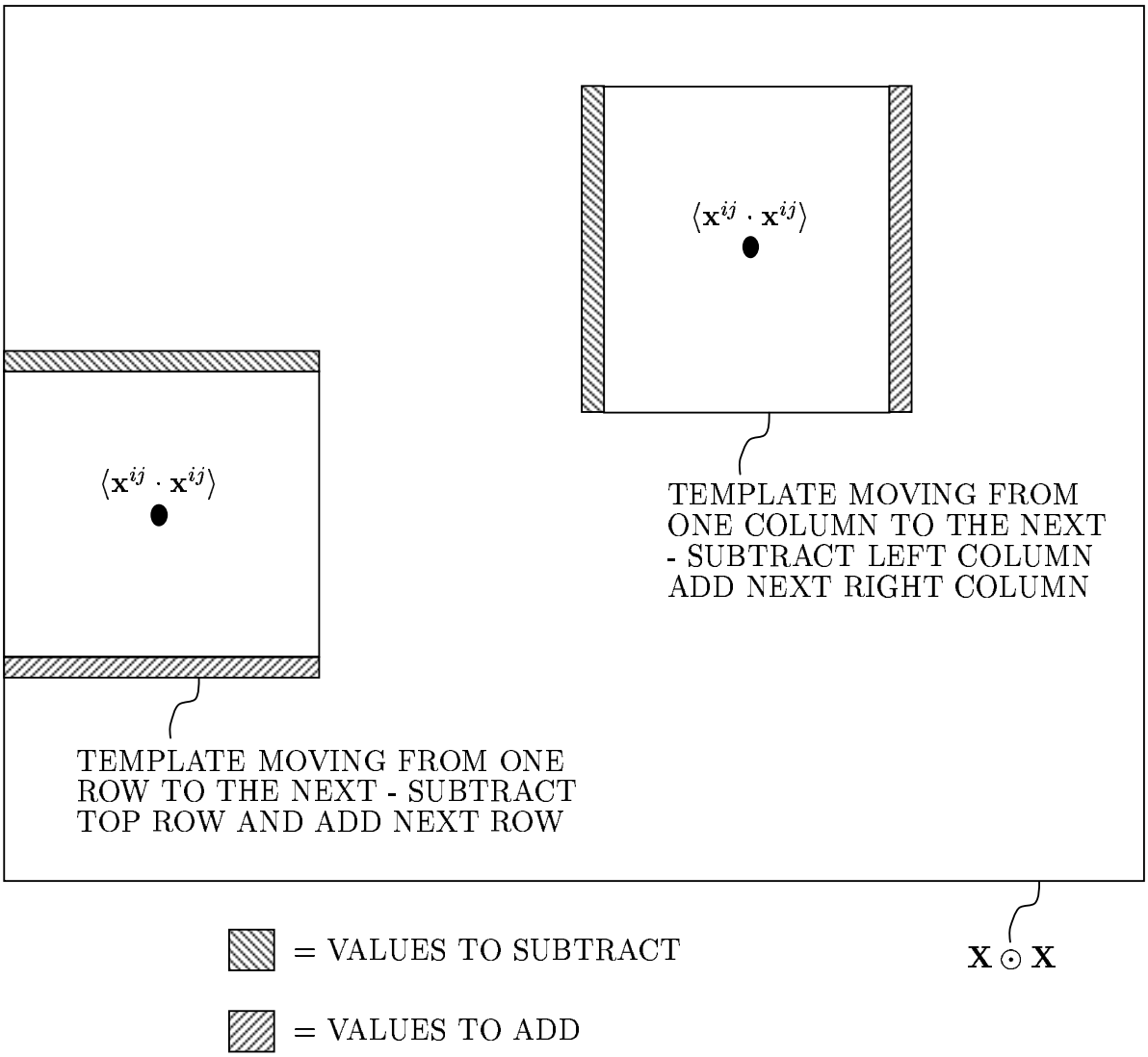


Figure 2.4: Efficient computation of the terms $\|\mathbf{x}^{ij}\|^2$ for all (i, j) by exploitation of redundancies. Multiplication operations only have to be performed once to compute the Hadamard product $\mathbf{X} \odot \mathbf{X}$. The dot products of sub-images with themselves can then be calculated by adding and subtracting adjacent rows and columns. The figure illustrates the calculation of these dot products by using results from the previous row, as shown by the block on the left where $\mathbf{x}^{(i+1)j} \cdot \mathbf{x}^{(i+1)j}$ is computed based on results from $\mathbf{x}^{ij} \cdot \mathbf{x}^{ij}$ by removing and adding the required row, and using results from the previous column, as shown by the block on the right where $\mathbf{x}^{i(j+1)} \cdot \mathbf{x}^{i(j+1)}$ is computed based on results from $\mathbf{x}^{ij} \cdot \mathbf{x}^{ij}$ by removing and adding the required column.

2.3.5 Implementation Notes

The definition of the eye location maps $\mathbf{M}_L$, $\mathbf{M}_{1P}$ and $\mathbf{M}_{1R}$ have entries that refer to the top left corner of the templates. Appropriate offsets must be added to obtain references to other parts of the template. Alternatively, the templates can be shifted.

2.4 Eye Detection System

2.4.1 Motivation

The first major processing task in most face recognition and tracking applications is to locate faces in the scene. A recent study discovered that an area of the human brain responds to stimuli as faces, with categorisation occurring later [LHK02]. This may suggest that the human brain too performs face detection/extraction before recognition in a two step process.

This chapter focuses on face detection by eye location. The reason for this research focus is that knowing the location of the eyes allows us to perform normalisations such as head size and in-plane rotation correction. Beyond the scope of this chapter is the normalisation of out-of-plane head rotation, which cannot be accounted for using just the two eye locations alone.

2.4.2 System Details

The eye detection system consists of left eye and right eye detection sub-systems. Each eye detection sub-system consists of two detectors. These two detectors operate on differently scaled versions of the input image. This is to further account for different head sizes even though it is accounted for to a smaller extent by the training data described below. Each detector generates one of the following eye location

maps $\mathbf{M}_L^s$, $\mathbf{M}_{1P}^s$ or $\mathbf{M}_{1R}^s$, where $\mathbf{M}_L$ is the map generated using a linear SVM, $\mathbf{M}_{1P}$ is the map generated using a polynomial SVM approximated using f_1 described in (2.31), $\mathbf{M}_{1R}$ is the map generated using a radial basis function SVM approximated using f_1 described in (2.31), $s \in \{L_1, L_2, R_1, R_2\}$ indicates which detector generated the map, L_1 is left-eye detector number 1, L_2 is left-eye detector number 2, R_1 is right-eye detector number 1, and R_2 is right-eye detector number 2. We will use $\mathbf{M}^s$ to refer to a map of any kernel type generated by detector s.

The eye detection system is illustrated in Figure 2.5. The first step involves feeding each frame of captured image data, from devices such as video or still cameras, to the system. The input image frames are assumed to be 576 pixels high and 720 pixels wide. The image is re-scaled to a size of 104 pixels high by 130 pixels wide ("RESIZE IMAGE 1"), and also to 86 pixels high by 108 pixels wide ("RESIZE IMAGE 2"). The dimensions were chosen based on the sizes of the training images used, so as to maintain an allowable tolerance on variations in head size.

The Fourier Transforms of the images are then computed and used to generate an eye location map $\mathbf{M}^s$ for each of the detectors. Using a pre-selected threshold τ_t obtained experimentally to achieve an acceptable level of performance, the two detectors for each eye are then combined. The maps are examined and if $m^s(i,j) > \tau_t$ then the location (i,j) is recorded. The recorded locations from the two maps for each eye can then be combined.

At this point, it is probable that many locations have passed the thresholding stage. To further reduce the number of possibilities we eliminate those locations (i,j) that have a score less than that of at least one point in its neighbourhood of points $\{(i+\Delta i, j+\Delta j) | -n_i \leq \Delta i < n_i, -n_j \leq \Delta j < n_j\}$. This can be interpreted as local thresholding where every location other than the local maximum is deleted. An inefficient method of solving this problem would be to compare every point to every other point to see if it is a neighbourhood point with a better score. This problem is also known as two-dimensional range searching, and efficient algorithms have been documented [Sed88]. One solution, called the grid method, involves dividing up

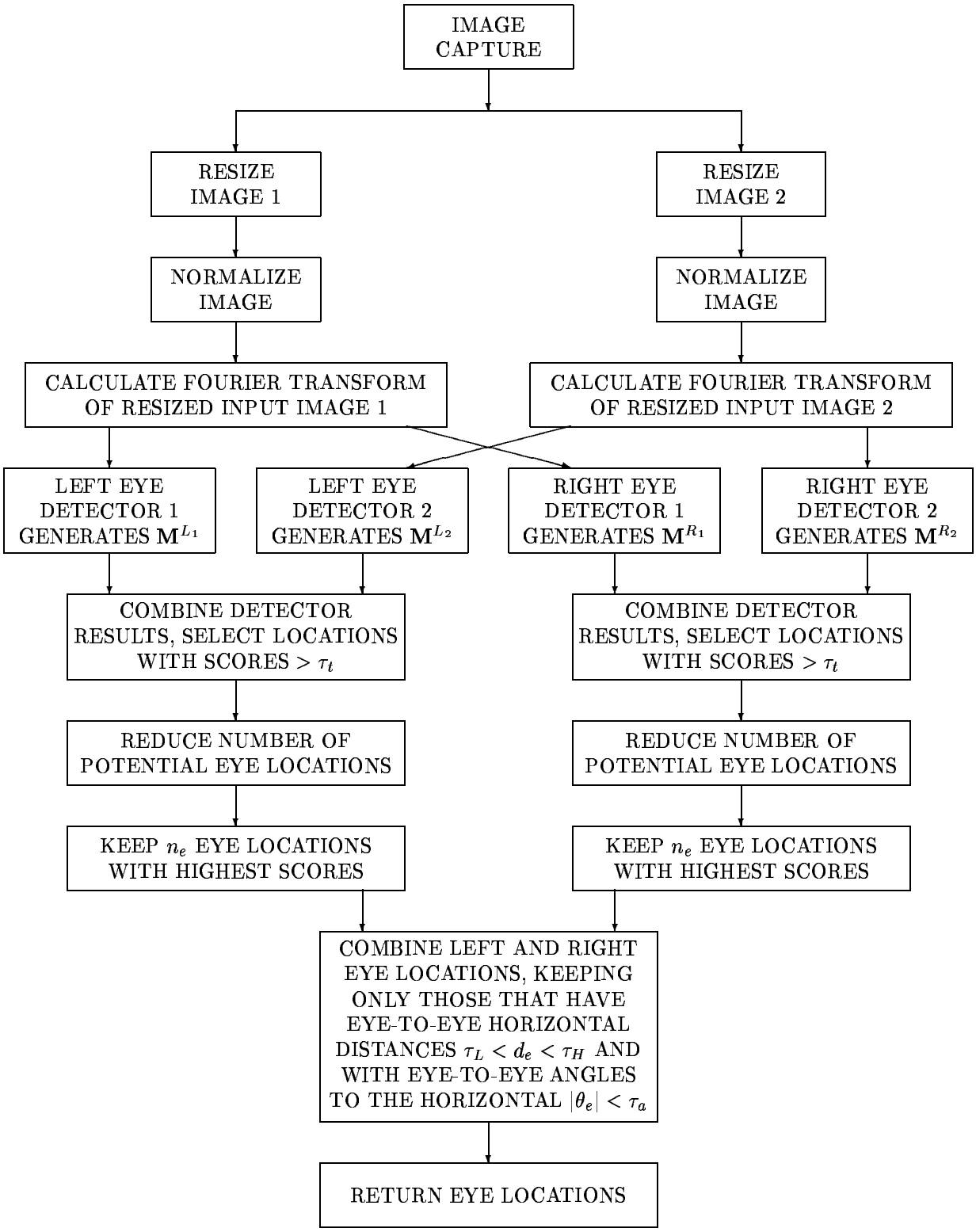


Figure 2.5: Block diagram of the eye detection system.

the search region into uniform rectangular blocks. Points are then assigned to their corresponding blocks. With a local neighbourhood search region centered on each point being examined, comparisons only need to be performed with those points in the blocks that overlap with the neighbourhood region. In our implementation we used a neighbourhood region defined by $n_i = 1$ and $n_j = 1$.

The scores at the remaining locations are then sorted for each eye. The locations with the n_e highest scores are kept, the rest are deleted. Now n_e possible locations are left for each eye. In our implementation we used $n_e = 5$.

The final step involves matching each left eye position with a right eye position and verifying that they satisfy some predetermined properties. Two properties are used: the horizontal distance d_e between the left and right eyes must be greater than a lower threshold τ_L and less than an upper threshold τ_H; and the absolute value of the angle of the segment joining the left and right eyes to the horizontal axis $|\theta_e|$ must be less than the maximum permitted angle τ_a. In our implementation we used $\tau_L = 10$ pixels, $\tau_H = 70$ pixels and $\tau_a = 20$ degrees.

Up to n_e^2 pairs of possible eye locations can be returned by the system. Optionally, further elimination can be performed because a single eye cannot have more than one matching pair, but carries with it the risk of deleting valid eye pairs. This system is aimed at access control so the assumption is that we only need to detect a single face in the image. Therefore, only the best left and right eye locations are used.

2.4.3 Training and Testing Data

Images of human faces are used as training data. Each training image consists of a single human face with manually located eye positions. A template is extracted around each eye, which we refer to as a training template. Two training templates are extracted for each training image, one for each eye. Positive training templates are those that corresponded to correct eye positions. Negative training templates

are those obtained by randomly selecting a sub-image in the training image that is "far enough" from a correct eye position. In our implementation "far enough" is anything further than 7 pixels in the horizontal and vertical directions from a valid eye position.

Two sets of training templates are used to train two SVMs to detect the left and right eyes. The left eye SVM is used to create the maps $\mathbf{M}^{L_1}$ and $\mathbf{M}^{L_2}$, which differed only in image size. The same is true for the right eye SVM and $\mathbf{M}^{R_1}$ and $\mathbf{M}^{R_2}$.

The training templates are 40 pixels high and 36 pixels wide. The training images are not centered on the eye, but rather at an offset so that it also encompassed the other parts of the face including the nose and the mouth. It is usually difficult to recognise an eye using low quality images if the template consists of nothing but the eye. Here we utilise information from the whole face to locate the eye. This increased robustness to eye-glasses, occlusions over the eyes, direction of stare, pose and eyelids that are partially or fully closed. Training images for the left eye are extracted with the coordinate $(9, 9)$ centered on the eye. The coordinate $(9, 27)$ is centered on the right eye.

Experiments are carried out using the XM2VTS face database [MMK+99, MHJ+00]. It consisted of a total of 2360 face images obtained from 295 people. Eight face samples from each person are taken over four different sessions with two samples recorded per session. The Lausanne Protocol [MHJ+00] describes a standard partitioning of the database to evaluate access control systems. The database is partitioned into client training data, client evaluation data, client test data, impostor evaluation data and impostor test data. Two configurations are specified. For training we use the client training data from protocol Configuration I, consisting of 600 images from 200 clients from three different sessions. Eye locations are manually labelled in all 2360 faces, which are then used in training to obtain the positive class samples and used during testing to evaluate system performance.

The face images are 576 pixels high and 720 pixels wide in 16 million colors, which

are converted to grayscales prior to further processing. Although the images used are of very high quality, resolution reduction is performed because the system itself operates on low resolution images as described above. Therefore, the system is also operable on lower quality input images.

To increase invariance to planar rotations and scaling, additional positive training templates are created by rotating and scaling each training image around each eye. The scale factors used are $\{0.15, 0.1625, 0.175, 0.1875\}$ and the rotation angles used are $\{-12, 6, 0, 6, 12\}$ in degrees. Some training images had faces with inherent planar rotations due to different poses. These are corrected before performing our own rotations. The number of negative training samples are similarly increased by scaling and rotating the original negative samples. This gives an initial set of 12000 positive and 12000 negative training samples. Extra negative samples are added by the bootstrapping method where incorrect classifications using the training data are extracted, appended to the training database and the SVMs re-trained. This gives a total of 24806 training samples for the left eye SVM and 24895 for the right eye SVM.

Testing is carried out using the XM2VTS database samples that are not part of the client training data. There are 1760 test images and prior to processing they are down-sized by the system to "RESIZE IMAGE 1" and "RESIZE IMAGE 2" and converted to grayscales. Down-sizing is done by averaging, thus interlacing effects such as blurring due fast motion are preserved. This tests the system on blurred images.

2.4.4 Experiments

This sub-section describes the performance of the eye detection system based on the following configurations, each one stating the kernel and their training parameter(s):

1. (L_a) Linear, $C = 1$;

2. (L_b) Linear, $C = 5$;

3. (L_c) Linear, $C = 10$;

4. (P_a) Inhomogeneous Polynomial, $d = 2, c = 1, C = 0.1$;

5. (P_b) Inhomogeneous Polynomial, $d = 4, c = 1, C = 0.25$;

6. (P_c) Inhomogeneous Polynomial, $d = 3, c = 3, C = 1$;

7. (R_a) Radial Basis Function, $\gamma = 0.1, C = 1$;

8. (R_b) Radial Basis Function, $\gamma = 1, C = 1$; and

9. (R_c) Radial Basis Function, $\gamma = 1, C = 5$.

The regularisation constant C, obtained experimentally, determines the trade-off between classifier capacity and the number of training errors. Eye detection results using the XM2VTS face database are shown in Tables 2.3 and 2.4. The top half of Table 2.3 shows the left eye detection results and the right eye detection results are in the bottom half. The distance in pixels from the automatically detected eye position to the correct (manually located) position is shown in the second left-most column. We refer to this as the Eye Pixel Distance (EPD), and the corresponding percentage of the average left-to-right eye pixel distance is also shown in the table in brackets. The average left-to-right eye distance is approximately 37 pixels. Each table entry represents the percentage of test samples having automatically detected eye locations that are at or less than the corresponding EPD. The best results from each classifier class have their columns highlighted, namely columns L_b, P_b and R_b. The rows corresponding to an EPD of 7 (19.0 %) are also highlighted. For the purposes of evaluating the system's performance we subjectively choose this as the maximum distance from the correct eye position that an automatically determined eye location can be deemed correct. Some experimental results satisfying this criteria are shown in Figure 2.6, which also demonstrate the system's robustness to eyeglasses, reflections off eyeglasses, facial hair, partial occlusions due to mainly to hair, pose and lower quality images due to fast motion in an interlaced video

format. Examples considered as incorrect detections are shown in Figure 2.7. The images in both figures are results from using the R_b configuration.

	Eye Pixel Distance	L_a	$\boldsymbol{L_b}$	L_c	P_a	$\boldsymbol{P_b}$	P_c	R_a	$\boldsymbol{R_b}$	R_c
Left Eye	1 (2.7 %)	30.0	**31.1**	29.4	26.4	**33.5**	28.9	25.3	**29.2**	28.9
	2 (5.4 %)	56.9	**58.1**	56.1	52.1	**63.2**	55.2	49.4	**56.1**	55.2
	3 (8.1 %)	80.6	**80.2**	78.6	75.9	**84.7**	76.8	73.4	**79.3**	78.2
	4 (10.8 %)	90.7	**89.6**	87.8	86.8	**92.0**	85.2	84.9	**90.3**	87.8
	5 (13.6 %)	95.3	**95.5**	94.6	92.6	**95.8**	91.7	91.6	**96.9**	94.6
	6 (16.3 %)	96.6	**96.7**	96.2	93.8	**96.9**	93.5	93.2	**97.8**	96.3
	7 (19.0 %)	**97.0**	**97.4**	**97.2**	**94.9**	**97.5**	**95.2**	**94.4**	**98.1**	**97.0**
	8 (21.7 %)	97.4	**97.7**	97.6	95.6	**98.0**	95.7	95.3	**98.4**	97.3
	9 (24.4 %)	97.6	**97.7**	97.6	96.1	**98.0**	96.0	96.0	**98.4**	97.3
Right Eye	1 (2.7 %)	26.1	**26.1**	24.6	21.6	**28.5**	23.3	20.3	**26.9**	24.3
	2 (5.4 %)	56.4	**56.6**	54.8	48.1	**58.8**	52.7	46.0	**56.6**	52.3
	3 (8.1 %)	81.8	**81.8**	80.3	74.7	**83.0**	78.8	73.1	**80.9**	77.4
	4 (10.8 %)	92.5	**92.9**	91.2	86.6	**93.1**	89.5	85.2	**92.0**	89.0
	5 (13.6 %)	95.5	**95.9**	94.8	92.4	**96.0**	92.9	91.4	**95.8**	93.6
	6 (16.3 %)	96.5	**96.6**	95.9	94.0	**96.8**	94.2	93.2	**96.8**	95.2
	7 (19.0 %)	**97.0**	**97.1**	**96.7**	**94.9**	**97.2**	**94.9**	**94.4**	**97.4**	**96.3**
	8 (21.7 %)	97.6	**97.7**	97.3	95.7	**97.8**	95.4	95.3	**98.1**	96.6
	9 (24.4 %)	98.0	**97.7**	97.3	96.6	**97.9**	95.6	96.4	**98.2**	96.8

Table 2.3: Individual eye detection results, where the top-half shows the left eye detection results and the bottom-half shows the right eye detection results. Each table entry represents the percentage of test samples having automatically detected eye locations that are at, or less than, the corresponding Eye Pixel Distance (EPD). The EPD is the distance of an automatically detected eye location to the known correct eye position. The row with an EPD of 7 is the subjective upper limit in which a detected location is considered as correct. The columns of the best performing configuration in each class are highlighted. L_a to L_c are linear kernels, P_a to P_c are inhomogeneous polynomial kernels and R_a to R_c are radial basis function kernels.

The systems that failed to detect either or both eyes in some test images are those with configurations $L_b(1)$, $L_c(1)$, $P_c(11)$, $R_b(1)$ and $R_c(3)$, where the number in the brackets is the number of undetected samples out of the total of 1760 test samples. The other configurations L_a, P_a, P_b and R_a had detection results for all test samples.

The results in Table 2.3 reveal that the system running with configuration P_b using the inhomogeneous polynomial kernel consistently outperformed those based on the linear kernel. The radial basis function kernel configuration R_b only performed better than those based on the linear and inhomogeneous polynomial kernels for larger EPDs. In general, final configuration selection will depend on the required system specifications. If the requirement is for a system that performs well at 7 EPD or less then it would be advisable to use configuration R_b, which has an accuracy of 98.1 % in detecting the left eye and 97.4 % in detecting the right eye. However, if the requirement is for a system that performs well in locating eye positions up to 2 EPD accuracy then configuration P_b gives the best performance at 63.2 % for the left eye and 58.8 % for the right eye. Although the left and right eye detectors could be based on different configurations, the best results coincide with the use of the same configurations. The results indicate that the right eye detector does not perform as well as the left eye detector. This is likely to be due to the asymmetry in the training data and the use of different negative class samples for each detector.

Where Table 2.3 shows the results for each individual eye, Table 2.4 shows the ability of the system in locating both eyes within a specified range. The table comprises three square matrices. The top matrix belongs to L_b, followed by P_b and R_b. The left eye EPD increases down each column and the right eye EPD increases across each row. Each entry in the table represents the percentage of test samples that had the left and right eyes detected at, or less than, the corresponding EPDs. At the subjectively determined maximum acceptable distance of 7 EPD for both the left and right eyes, configuration R_b performed best at a 96.9 % detection rate.

Our results are comparable to those in the literature [SB02], where they report a 99.0 % detection rate for each eye and a 97.4 % rate for the successful detection

	Eye Pixel Distance	1	2	3	4	5	6	**7**	8	9
	1	10.9	20.5	27.9	30.5	30.7	30.7	30.7	30.9	30.9
	2	17.9	37.3	51.9	56.3	57.1	57.2	57.3	57.6	57.6
	3	22.8	48.6	69.3	76.7	78.6	78.9	79.1	79.5	79.5
	4	24.8	52.7	75.9	85.2	87.7	88.2	88.5	88.9	88.9
L_b	5	25.9	55.7	79.7	90.5	93.2	93.9	94.2	94.7	94.7
	6	26.0	56.1	80.7	91.5	94.4	95.1	95.4	96.0	96.0
	7	26.1	56.2	81.1	92.2	95.1	95.7	**96.1**	96.6	96.6
	8	26.1	56.4	81.4	92.4	95.3	96.0	96.4	96.9	96.9
	9	26.1	56.4	81.4	92.4	95.3	96.0	96.4	97.0	97.0
	1	12.8	23.6	30.1	32.5	32.9	33.0	33.0	33.1	33.1
	2	21.4	41.8	56.0	61.0	62.0	62.2	62.3	62.5	62.5
	3	26.4	53.2	73.4	81.1	82.8	83.3	83.5	83.9	83.9
	4	27.5	56.3	78.4	87.6	89.8	90.5	90.7	91.2	91.3
P_b	5	28.2	58.0	81.0	90.8	93.4	94.1	94.4	94.9	95.0
	6	28.4	58.3	81.7	91.8	94.5	95.3	95.6	96.1	96.1
	7	28.4	58.3	82.2	92.3	95.0	95.8	**96.1**	96.6	96.7
	8	28.4	58.5	82.5	92.6	95.4	96.2	96.5	97.1	97.2
	9	28.4	58.5	82.6	92.7	95.5	96.3	96.6	97.2	97.2
	1	10.7	20.2	25.9	28.1	28.6	28.8	28.9	28.9	28.9
	2	18.1	36.3	48.8	53.7	54.9	55.3	55.5	55.6	55.6
	3	23.7	48.3	67.4	75.1	77.4	78.0	78.3	78.7	78.8
	4	25.3	53.0	74.9	84.9	88.0	88.6	89.1	89.5	89.7
R_b	5	26.8	56.2	79.5	90.6	94.1	95.1	95.7	96.1	96.3
	6	26.9	56.5	80.3	91.4	95.1	96.0	96.6	97.0	97.2
	7	26.9	56.5	80.5	91.6	95.2	96.3	**96.9**	97.3	97.4
	8	26.9	56.5	80.6	91.8	95.5	96.5	97.1	97.6	97.7
	9	26.9	56.5	80.7	91.8	95.5	96.5	97.2	97.7	97.8

Table 2.4: Three square matrices of detection percentages, left eye EPD increases down the columns, and right eye EPD increases across the rows. Each table entry represents the percentage of samples that have their left and right eye EPDs at, or lower than, the EPD value indicated by the corresponding row and column. The EPD is the distance of an automatically detected eye location to the known correct eye position. L_b is a linear, P_b is an inhomogeneous polynomial and R_b is a radial basis function kernel.

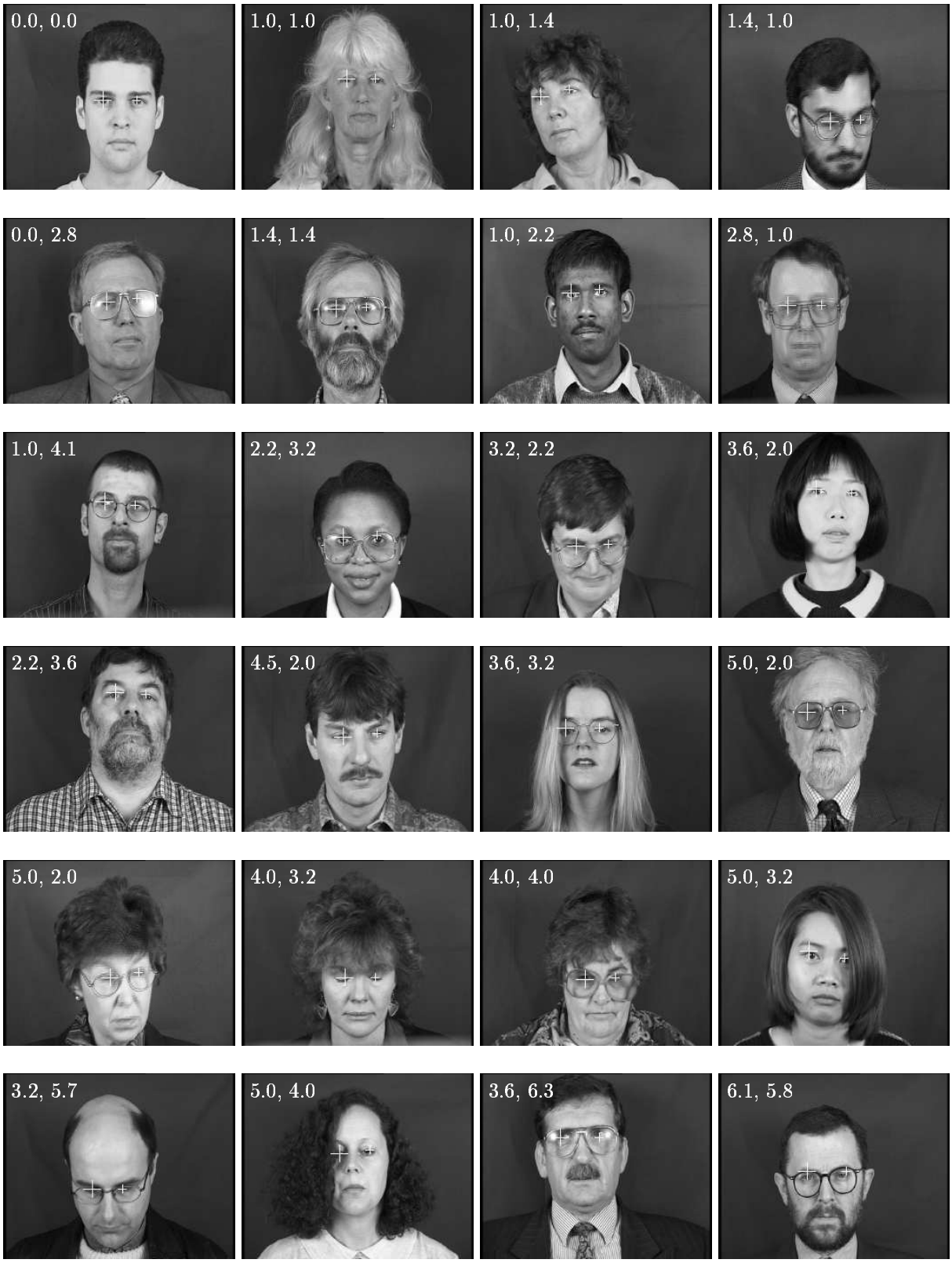


Figure 2.6: Results considered as good detections where both eyes are less than or equal to seven pixels away from the correct position. The top-left corner of each image displays the distances of the left and right eyes from the correct position, where the first number corresponds to the left eye and the second to the right eye.

Figure 2.7: Results considered as bad detections where either or both eyes are greater than seven pixels away from the correct position. The top-left corner of each image displays the distances of the left and right eyes from the correct position, where the first number corresponds to the left eye and the second to the right eye.

of both eyes and mouth using the XM2VTS database. Some differences limit the extent in which the systems can be directly compared. In particular, they use images from the M2VTS database for training, unspecified extra test images are used and detection performance is evaluated by visual inspection without a specific definition of what constitutes a correct or incorrect detection.

Our results can be improved by using multiple vectors to represent the η-prototype, instead of using just one vector. The appropriate simplified decision function representation is just a simple extension of the one vector case. It is likely that the optimisation method for finding the multiple vectors will be in an incremental fashion. And the optimal coefficients for each vector will most likely be calculated at each incremental step. In addition, the generation of the eye map will need to be modified to incorporate the use of multiple vectors for the η-prototype. Further investigation and implementation of this technique is left for future work.

The average processing times to detect both eyes using the best configurations for each kernel type are L_b (39 ms, 25 fps), P_b (68 ms, 15 fps) and R_b (110ms, 9 fps), where the rough equivalent in frames per second is indicated with the unit fps. The processing hardware consists of an AMD Athlon XP 2000+ based PC with 1 Gigabyte of DDR-RAM. Software code runs on the Windows XP operating system and is written in C++ and compiled with the Intel C++ 6.0 Compiler with optimisation for speed. Training of the SVMs are done using LIBSVM (available at http://www.csie.ntu.edu.tw/ cjlin/libsvm/), which has the training times of L_b (7010 s), P_b (9361 s) and R_b (13079 s) for the left eye, and L_b (7140 s), P_b (9398 s) and R_b (12753 s) for the right eye. For non-linear kernel configurations, the times required for prototype generation are P_b (751 s) and R_b (3011 s) for the left eye, and P_b (807 s) and R_b (2785 s) for the right eye. The FFTs are performed using the FFTW package (available at http://www.fftw.org/).

2.5 Conclusions

We demonstrate how Support Vector Machines can be combined in a novel way with the Fast Fourier Transform to achieve efficient exhaustive searching of an input image for human eyes. This requires the use of simplified decision functions, which are easily obtainable for the linear kernel. However, for non-linear kernels we introduce the concept of η and ρ-prototypes to obtain approximate simplified decision functions, which may be kernel dependent. We show how they can be calculated for the inhomogeneous polynomial and radial basis function kernels. Experiment results demonstrate that our method produces better approximations of the original decision function compared to those obtained using reduced set methods where an equivalent number of vectors are used.

Support Vector Machines are trained in the usual way to detect the left and right eyes in a template-based approach. The training images encompass the relevant eye and also other parts of the face to obtain more robustness to factors that may change the appearance of the eye. The trained Support Vector Machines are then modified according to the kernel type and used with the Fast Fourier Transform to generate maps of possible eye locations. The maps are analysed and the most likely left and right eye locations extracted. In experiments using the XM2VTS face database our system successfully detects both left and right eyes in 96.9 % of all test samples. The same system successfully detects 98.1 % of all left eyes and 97.4 % of all right eyes. These results are achieved using the radial basis function kernel. The linear kernel based system has the fastest processing time, followed by the inhomogeneous polynomial kernel and the radial basis function kernel. All of them operate in real-time, ranging from 39 ms to 110 ms to process one image frame, which is roughly equivalent to 25 and 9 frames per second respectively.

The reduction of the number of support vectors in Support Vector Machines is very important in fast pattern classification. For example, Burges [Bur96] uses reduced set vectors for handwritten digit classification. A list of the number of reduced set vectors used for each digit is shown in Table 2.5. Our proposed method can

Digit	No. Support Vectors	No. Reduced Set Vectors
0	292	10
1	95	6
2	415	22
3	403	14
4	375	14
5	421	18
6	261	12
7	228	10
8	446	24
9	330	20

Table 2.5: The number of support vectors and the corresponding number of reduced set vectors used for NIST handwritten digit classification. Using the indicated number of reduced set vectors achieves a classification performance that is approximately equivalent to that obtained using the original support vectors. Data obtained from Burges [Bur96].

potentially outperform the reduced set method for the handwritten digit test. This investigation is left for future work.

Chapter 3

Fractal Neighbour Distance

3.1 Introduction

It is well known that humans have the ability to recognise objects with relative ease and speed. However, it is also true that producing an automatic system capable of performing this task with accuracy comparable to that of humans is extremely difficult. The exact processes in the brain required for recognition of visual sensory signals still remain largely unknown due to the brain's complexity and the difficulty in obtaining accurate measurements. Therefore, biologically inspired artificial recognition systems are usually limited in mimicking the visual tasks that we humans perform effortlessly. On the other hand, we can approach the problem from the point of view of the object to be recognised, detached from the biological model of the brain.

One of the most widely accepted algorithms for human face recognition is the eigenface method [GA97, KS90, Pen84, TP91a, TP91b]. This method is based on principal components analysis. One of its main advantages is the dimensionality reduction that enables recognition to be performed rapidly. However, one of its serious drawbacks is its dependence on translation and illumination, which is predominant in the first three principal components. Discarding those components can yield better

recognition rates [BHK97], but they may contain critical feature information on some databases, thus resulting in a degraded performance. Another method known as linear discriminant analysis (LDA) [EC97] performs better than eigenfaces, which also uses eigenvector analysis, but on scatter matrices.

Rigid templates have also been used for object recognition [BP92, BP93]. In general, this method involves the correlation of a rigid template with an unknown input image. A metric is defined and the rigid model template that minimises this metric is taken as the best match. One of the major drawbacks of this method is the inability of the rigid template to account for non-linear distortions of the image of the object. These may be due to the object's orientation in three-dimensional space or changes within the object itself, such as different human face expressions. Deformable templates/graphs have been proposed to overcome these limitations [BLM90, LVB$^+$93, SW91, Yui91]. Classification is performed by minimising a heuristic cost/energy function associated with the amount of model distortions between the unknown input image and the ones in the database. Deformable templates/graphs can be computationally expensive and is limited in performance by the accuracy of the model.

Artificial neural networks have also been proposed [LGTB97, ZYL97], but they suffer from several drawbacks, including the need to retrain the whole network whenever samples are added or removed from the database. There are also convergence problems associated with having large networks, which is often required when databases become large.

The recognition ability of systems based on normalised anthropometric feature ratios [BP92, BP93, Kan77] and analytic and holistic approaches [LY98] depend on the ability to detect features and to model them reliably, and the relevancy of those features in defining that object uniquely. The combination of these factors limits the performance of these systems.

On analysing natural objects, such as coastlines, rock formations and clouds, many exhibit self-similar properties [Man82]. Self-similar properties refer to features that

are similar under different scales of magnification. Such objects are called fractals [Man82]. Images of fractals can be described by a set of non-linear transformations. This notion can be extended to objects that are not truly fractal in nature. In fact, all objects contain some degree of self-similarity within their constituent parts. This idea is utilised in the Collage Theorem [Bar93], which enables us to find non-linear transformations that approximate any given image. This process is referred to as fractal image coding [Bar93, BH93, BJ88, BS88, Jac93, Jac90, Jac94, Jac92]. The set of transformations approximating a given image is referred to as the fractal code of that image. By minimising the number of bits required to represent the fractal code, fractal image compression is achieved [BH93, BS88, Fis95a, Fis92, Jac90]. We propose that due to the self-similarity in the transformations, fractal image coding can be naturally adapted for recognition. This process centers on the measurement of the input-output characteristics of a fractal transformation. This method is implemented to perform human face recognition.

The work here was inspired by the material published by a number of researchers [KHS97, NC96, NCC96], but the ideas published in [KHS00] was developed independently. The general approach is based on the Euclidean metric and the uniqueness of the attractor of a fractal code. Here we extend the work by Neil et al. [NC96, NCC96] to include grayscale images and we investigate the effects of varying the contractivity factor and using different encoding schemes. In our method, an unknown image is modified with the fractal code of a known object. The difference between the unknown image before and after the modification is then determined using the Euclidean distance measure. This results in a new distance measure that we will refer to as the Fractal Neighbour Distance (FND). Recognition of an unknown image is achieved by selecting the entry from a database of fractal codes that minimises this distance. The performance of this new classifier is compared to algorithms based on the nearest neighbour classifier and the eigenface method under the same experimental conditions. Comparisons with other algorithms is also shown using results quoted from the literature. The use of fractal image coding for recognition is similar to that used by Kouzani et al. [KHS00], which was developed

independently. The difference between our work and that in the literature is that we will demonstrate that this new method is inherently invariant, to a given extent, to translations, rotations, scaling and illumination differences. Without extra effort, the method is invariant through the properties of the transformations that constitute the fractal code.

Section 3.4 describes the method of using fractal image coding for object recognition. Throughout this chapter, when we refer to "recognition" we are in fact referring to the more specific problem of identification. Section 3.5 presents the experiments we have conducted along with the results. This is followed by some discussion.

3.2 Fractals

Fractals are mathematical sets that exhibit self-similarity under all scales of magnification [Bar93, Man82]. Figure 3.1 shows an example of the construction of a fractal object, formally known as the Sierpinski Triangle. This fractal code is represented by three contractive affine transformations of itself. Each of the three transformations places reduced-by-half copies of the whole at three different positions. The results of applying this code to a black input image is shown in Figure 3.1(a). On continued application of the transformations to each successive resultant image an approximation to the attractor is reached (Figure 3.1(d)), which remains approximately the same with further iterations. The attractor of a fractal code is invariant to further applications of the code. The true attractor (the fractal object) is reached only after an infinite number of iterations, but usually a good approximation is available after several iterations. It must be noted that the attractor is not dependent in any way on the initial image used. Figure 3.1(e)-(i) illustrates this property. The attractor is unique for a given set of transformations that constitute the fractal code. Moreover, these transformations must be contractive for the iteration process to converge to the attractor. This condition is fulfilled if those transformations, represented by f, constitute a contraction mapping in a metric

space (X, d) satisfying:

$$d(f(\mathbf{x}), f(\mathbf{y})) \leq sd(\mathbf{x}, \mathbf{y}), \tag{3.1}$$

where $s < 1$ and $\mathbf{x}, \mathbf{y} \in X$. X is the set of images, s is called the contractivity factor of f, and d is a distance measure such as the Euclidean distance. Repeated application of f to any element $\mathbf{x}$ results in the unique attractor $A = f(A)$. The rates at which $\mathbf{x}$ approaches A increases as the contractivity factor s approaches zero and decreases as s approaches one.

3.3 Fractal Image Coding

3.3.1 Background

In general, it is almost impossible to find an optimal fractal code for any arbitrary image, except for a subclass of images that are genuine fractal objects in the mathematical sense [Man82]. In fact, this problem is found to belong to a class of problems that are NP-hard [RH97], which are not yet known to be solvable in polynomial time. However, Barnsley [Bar93, BH93, BJ88, BS88] proposes a technique of approximating the fractal code of an arbitrary image that gives satisfactory results, though it is still not an approximating algorithm for this fractal coding optimisation problem [RH97]. Fractal image coding based on the iterated function system was first proposed by Barnsley and Jacquin [Bar93, BH93, BJ88, BS88, Jac93, Jac90, Jac94, Jac92]. This coding scheme has since been successfully used to compress images [Bar93, BH93, BJ88, BS88, Fis95a, Fis92, Jac93, Jac90, Jac94, Jac92, PY95, PY97, Vrs95]. It has also been used for image segmentation [PMST94] and target recognition. The process involves the encoding of images in terms of block affine transformations in a manner that resembles vector quantisation [RG86]. One of the main differences between vector quantisation and fractal image coding is that the latter derives the codebook from the image being encoded, rather than from separate training images. Fractal image coding

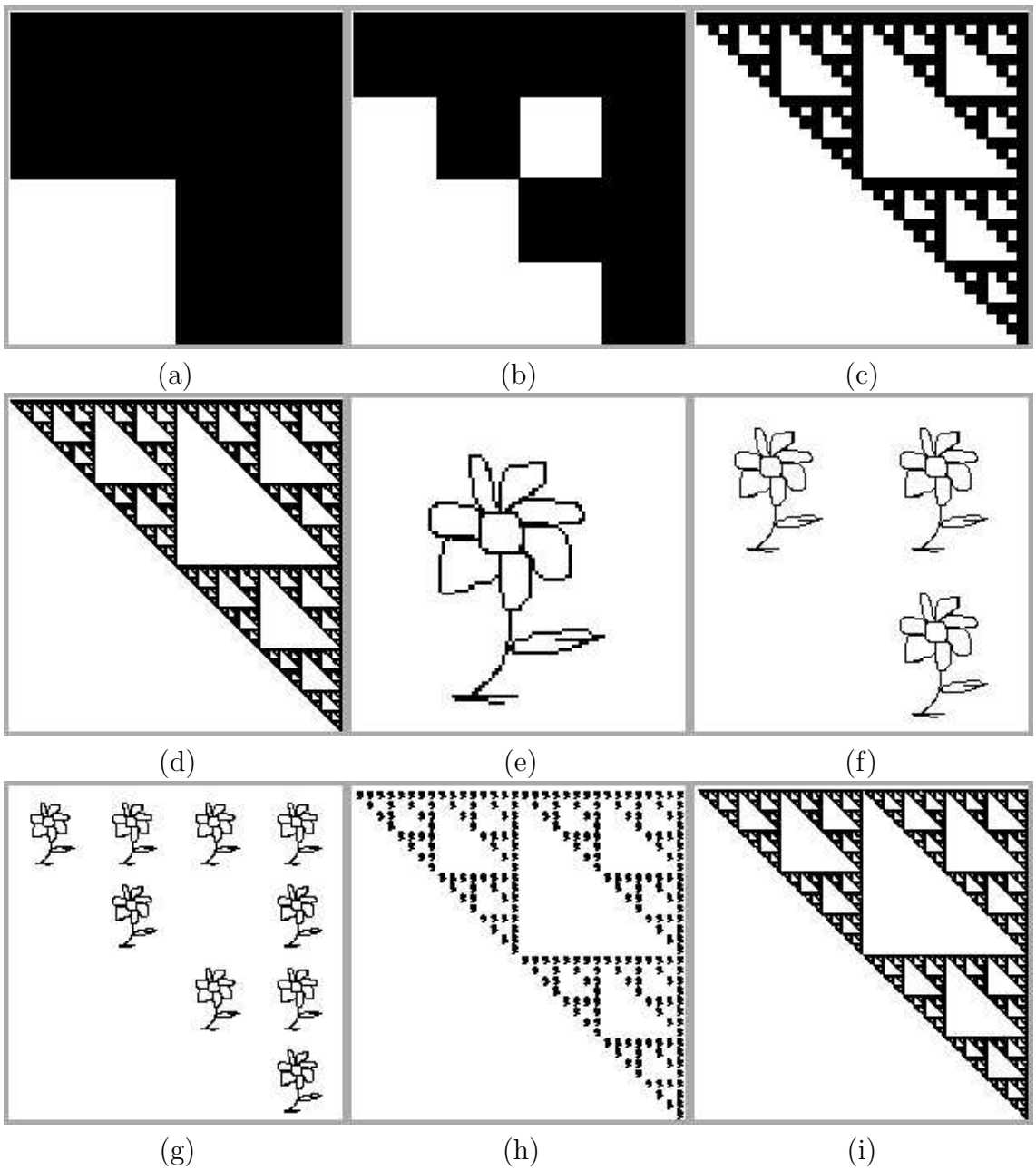


Figure 3.1: The iterative process that generates the Sierpinski Triangle fractal. It also shows the uniqueness of the attractor and its independence from the initial image: (a) the first iteration using the Sierpinski Triangle transformations with a black input image; (b) second iteration; (c) fifth iteration; (d) tenth iteration (this is an approximation of the attractor); (e) using a flower as an initial image; (f) first iteration of the flower; (g) second iteration of the flower; (h) fifth iteration of the flower; (i) tenth iteration of the flower (this is an approximation of the attractor).

exploits the self-similarity in subsets of an image to obtain sets of contractive transformations, the union of which reproduces an approximation of the original image that improves with further iterations of those transformations. This technique is grounded on the Collage Theorem [Bar93], which involves a search for self-similar sub-regions within the original image.

3.3.2 Basic Concepts

In fractal image coding an arbitrary image is encoded into a set of equations. These equations are usually affine transformations that transform a sub-image, called a domain block, into another sub-image, called a range block. An image is divided into non-overlapping range blocks, and a search for a best matching domain block is performed for each range block. Domain blocks are usually larger than range blocks, and are similar to one another under an affine transformation. Examples of these transformations, collectively referred to as a fractal code, are shown in Figure 3.2. The mapping from a domain to range block involves decimating the domain block to the size of the range block, multiplying each element in the resulting matrix by a constant contrast scaling factor α, and an addition of another factor, γ. The location of the domain block and the choices of α and γ are made such that the resulting block approximates the range block as closely as possible under practical constraints, such as the imposition of a maximum allowable search time. The factors α and γ can be calculated by equalising the dynamic range and mean value between the domain and range blocks. Applying these transformations collectively and repeatedly results in closer approximations to the original image.

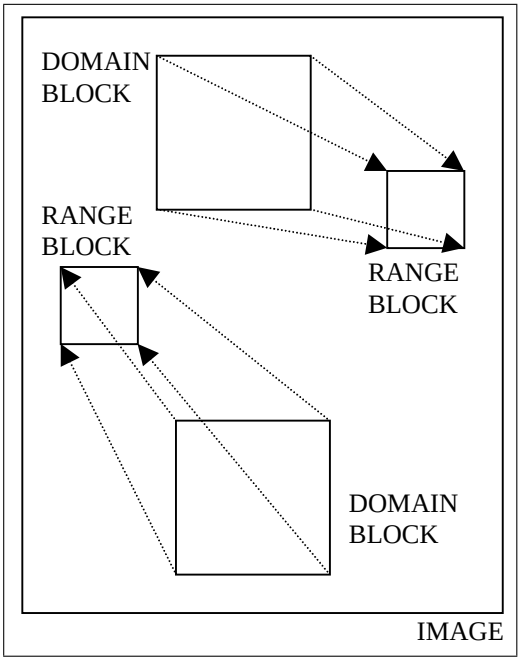


Figure 3.2: Example of the matching process of the domain block to the range block when determining the transformations that constitute a fractal code.

3.3.3 Notations

We let $\tau = \bigcup_{n=1}^{N} \tau_n$ where N is the total number of contractive sub-transformations τ_n, that map a domain block to an approximated-range block. Non-approximated-range blocks are those that are obtained from the original image itself. As a convention, we refer to non-approximated-range blocks as just range blocks. Both types of range blocks have the same shape and location on the original image, but the approximated type is derived from the mapping from domain blocks and the other comes directly from the original image.

Iteration of the overall transformation τ, regenerates an approximation to the original image. The transformations τ_n considered in this chapter are based on decimation, contrast scaling and illuminance shifting of the domain blocks as described by

$$\tau_n(\mathbf{p}_i) = \alpha_n(\mathbf{D}_{n,r}\mathbf{p}_i^{(n_D)}) + \gamma_n. \tag{3.2}$$

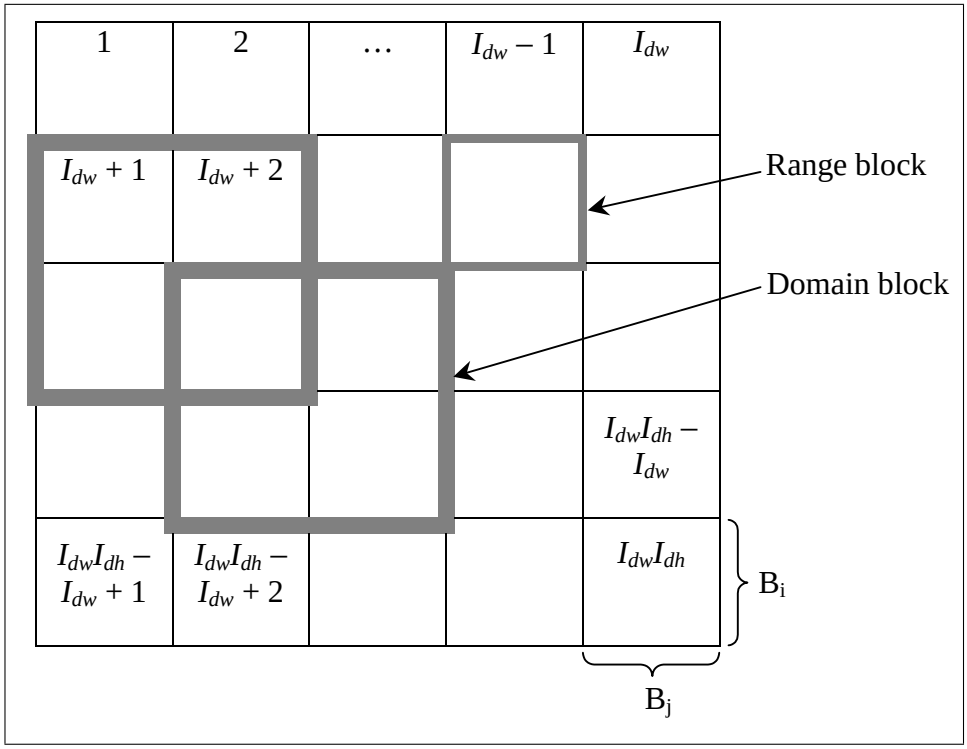


Figure 3.3: Uniform image partitioning. Domain block dimensions are twice as large as range block dimensions. The range blocks are also ordered row-wise. Each range block has size B_i by B_j. There are I_{dw} number of horizontal range blocks, and I_{dh} number of vertical range blocks.

The nth transformation τ_n operates on a domain block of the input image $\mathbf{p}_i$. The domain block used by the nth transformation is denoted as $\mathbf{p}_i^{(n_D)}$. A decimation by factor r filter, denoted as $\mathbf{D}_{n,r}$, is followed by contrast scaling α_n, and an illuminance shift γ_n. In our implementation, the domain and range blocks are square sub-images of $\mathbf{p}_i$. Furthermore, the range sub-images are non-overlapping, so that $\tau_n(\mathbf{p}_i) \bigcap \tau_m(\mathbf{p}_i) = \emptyset$, for $m \neq n$. During encoding the best approximated-range block is found for each range block by searching and transforming from a pool of domain blocks. Figure 3.3 illustrates the partitioning of an image into non-overlapping range blocks with some possible domain blocks shown. The figure also shows that domain blocks consist of sub-blocks that have the same size and position as range blocks.

3.3.4 Implementation

The following assumptions apply in the implementation of our fractal encoder:

i. The range blocks are square non-overlapping uniform blocks of size 4 by 4 pixels. For generality, let B_i be the height and B_j be the width. Other block sizes were also experimented with, and for the image sizes used the 4 by 4 block gives the best recognition results.

ii. The height and width of the domain blocks are twice as large as the height and width of the range blocks.

iii. The domain blocks overlap by half in the vertical and horizontal directions. Having overlapping domain blocks increases encoding accuracy as the probability of locating the optimal domain block that matches a given range block increases [Jac93, Jac94, Jac92].

iv. The mapping of domain blocks to approximated-range blocks are affine transformations described by Equation (3.2). Isometric transformations such as reflections and rotations of the domain block are not used, reducing the amount of searching required.

v. The Euclidean norm is used for distance measures. In this case the distance between any two given images, say $\mathbf{p}$ and $\mathbf{q}$ with height I_h and width I_w, is defined as the root mean square (RMS) difference between those images:

$$d(\mathbf{p},\mathbf{q}) = \|\mathbf{p}-\mathbf{q}\|_2 = \sqrt{\sum_{k=0}^{I_h}\sum_{l=0}^{I_w}(\mathbf{p}_{(k,l)}-\mathbf{q}_{(k,l)})^2}. \tag{3.3}$$

vi. There are no search restrictions on the domain pool. All possible domain blocks are searched for the one that minimises $d(\mathbf{p}_i^{(n_R)}, \tau_n(\mathbf{p}_i))$, where $\mathbf{p}_i^{(n_R)}$ denotes the nth range block in the image $\mathbf{p}_i$.

vii. The value of α_n for the nth domain to approximate-range block transformation is fixed to a constant value α. This is imposed to aid the calculation of s.

viii. The value of γ_n is determined by the following equation:

$$\gamma_n = \frac{1}{B_i \times B_j} \left(\sum_{i=1}^{B_i} \sum_{j=1}^{B_j} \mathbf{p}^{(n_R)}(i,j) - \alpha \sum_{i=1}^{B_i} \sum_{j=1}^{B_j} (\mathbf{D}_{n,r} \mathbf{p}^{(n_D)})(i,j) \right). \tag{3.4}$$

The value α is the contrast scaling factor that modifies the dynamic range of the transformed domain block, and γ_n adjusts the mean of the transformed domain block so that it is the same as the range block. This scheme is similar to the one used by Jacquin [Jac93, Jac90, Jac94, Jac92] with the exception that isometries are not considered as noted in assumption (iv) above.

3.3.5 Contractivity Factor

The contractivity factor s is an important element in the FND as it ensures fractal code convergence and influences recognition rates. In this chapter we establish a relationship between the contractivity factor and the recognition rate. This relationship, along with the effects of other factors, is further investigated in the next chapter. The contractivity factor has a lower bound of zero as described above. The rate of convergence of a code is determined only by the upper bound of s. If the absolute value of s is less than one, convergence is guaranteed. Otherwise, the decoding process has the possibility of diverging, where the magnitudes of some values operated on by the fractal code increases without bound as the number of iterations increases. In this case, the ensuing resultant image of decoding does not eventually converge to a stable attractor. Therefore, by controlling this upper bound we can manage the properties of the fractal code. Here we use the simplified version of Theorem 4.1, which is described in Chapter 4 for the calculation of s. In this chapter we only use the result. The proof and explanations for the theorem are given in Chapter 4. The simplified calculation of the contractivity factor is

$$s \leq \sqrt{\frac{t\alpha^2}{r}}, \tag{3.5}$$

where t is the maximum number of times any sub-domain block is used, α is the fixed contrast scaling factor for all transformations, and r is the decimation factor. Domain sub-blocks have the same size and position as the range blocks, as shown in Figure 3.3. The value of t is set to six, because it was discovered experimentally that this value gave the best tradeoff between encoding accuracy and the approximation accuracy of s. The value of r is set to four for decimation by two in the horizontal and vertical directions. The simplicity of Equation (3.5) is made possible by assumption (vii) above, where all domain to range block transformations assume a constant contrast scaling factor of α.

3.4 Recognition Using the Fractal Neighbour Distance

3.4.1 The Fractal Neighbour Distance

The problem of object recognition can be stated as: for a database of known objects, assign an unknown object to one of the objects in the database. In this section, the fractal neighbour distance implemented as a classifier for the problem of human face recognition is described. In particular, we will focus on identification, as opposed to verification, which is a different problem [RPM98]. In identification, the identity of an unknown input image is gauged based on a database of known individuals. An error rate can then be defined as F_{ERR}/F_N, where F_{ERR} is the number of incorrect matches, and F_N is the total number of unknown input images used, assuming those images have their identity in the database. This is in contrast to verification where the input is an unknown image with a claimed identity. The input image is then accepted or rejected by the algorithm based on that identity and a pre-set threshold. Some identification systems also incorporate an accept/reject feature, but this is not considered here for simplicity.

The FND gives a quantitative measure of the input-output characteristics of the

fractal code of an encoded image. The classification system based on the FND is illustrated in Figure 3.4. The FND is defined as

$$d_{\mathrm{FN}}(\mathbf{x}_j, \mathbf{p}_i) \equiv d(f_j(\mathbf{p}_i), \mathbf{p}_i), \tag{3.6}$$

where f_j is the jth code in a database of fractal codes with an attractor that is an approximation of the training image $\mathbf{x}_j$, $f_j(\mathbf{p}_i)$ is the first iteration of the jth code with $\mathbf{p}_i$ as the input image. The FND is in fact not a true metric because it does not satisfy the symmetry condition. It is possible to redefine the FND such that it is symmetric by, for example, defining it as $d(f_j(\mathbf{p}_i), \mathbf{p}_i) + d(f_{\mathbf{p}_i}(\mathbf{x}_j), \mathbf{x}_j)$. However, in preliminary experiments this was found to give inferior results. Therefore, we settle for the definition of the FND shown in Equation (3.6). Next, we establish a relationship between the FND and the contractivity factor. The metric d in Equation (3.1) satisfies the following properies:

(i). $d(\mathbf{x}, \mathbf{y}) = 0$ iff $\mathbf{x} = \mathbf{y}$,

(ii). $d(\mathbf{x}, \mathbf{y}) = d(\mathbf{y}, \mathbf{x})$, and

(iii). $d(\mathbf{x}, \mathbf{y}) + d(\mathbf{y}, \mathbf{z}) \geq d(\mathbf{x}, \mathbf{z})$ for any $\mathbf{x}, \mathbf{y}, \mathbf{z}$ in the metric space.

Let f be a mapping in the metric space (X, d). Then f is called a contraction mapping if there exists a constant $s < 1$ such that for all $\mathbf{x}, \mathbf{y} \in X$ Equation (3.1) is satisfied. From the triangle inequality (iii) we can write

$$d(\mathbf{p}_i, f_j^n(\mathbf{p}_i)) \leq d(\mathbf{p}_i, f_j(\mathbf{p}_i)) + d(f_j(\mathbf{p}_i), f_j^2(\mathbf{p}_i)) + \ldots + d(f_j^{(n-1)}(\mathbf{p}_i), f_j^n(\mathbf{p}_i)) \tag{3.7}$$

where $\mathbf{p}_i$ is the ith input face image, and f_j^n is the jth fractal code in the database applied n times. It can then be shown that

$$d(\mathbf{p}_i, f_j^n(\mathbf{p}_i)) \leq \frac{1 - s^n}{1 - s} d(\mathbf{p}_i, f_j(\mathbf{p}_i)). \tag{3.8}$$

Using the triangle inequality we can also show that

$$d(f_j(\mathbf{p}_i), \mathbf{p}_i) \leq (s + 1) d(\mathbf{p}_i, f_j^n(\mathbf{p}_i)) + s d(f_j^n(\mathbf{p}_i), f_j^{n-1}(\mathbf{p}_i)). \tag{3.9}$$

Combining Equations (3.8) and (3.9) we obtain the following expression,

$$\frac{1-s}{1-s^n}d(\mathbf{p}_i, f_j^n(\mathbf{p}_i)) \leq d(\mathbf{p}_i, f_j(\mathbf{p}_i)) \leq (s+1)d(\mathbf{p}_i, f_j^n(\mathbf{p}_i)) + sd(f_j^{n-1}(\mathbf{p}_i), f_j^n(\mathbf{p}_i)) \tag{3.10}$$

Now, in the limit as the number of iterations n approach infinity we obtain the following expression,

$$\lim_{n\to\infty}\left(\frac{1-s}{1-s^n}\right) \leq \lim_{n\to\infty}\frac{d(\mathbf{p}_i, f_j(\mathbf{p}_i))}{d(\mathbf{p}_i, f_j^n(\mathbf{p}_i))} \leq \lim_{n\to\infty}(1+s) + \lim_{n\to\infty}\left(\frac{sd(f_j^{n-1}(\mathbf{p}_i), f_j^n(\mathbf{p}_i))}{d(\mathbf{p}_i, f_j^n(\mathbf{p}_i))}\right). \tag{3.11}$$

Note the following property,

$$d(f_j^m(\mathbf{p}_i), f_j^n(\mathbf{p}_i)) \leq s^{m\wedge n} d\left(f_j^{m-(m\wedge n)}(\mathbf{p}_i), f_j^{n-(m\wedge n)}(\mathbf{p}_i)\right) \tag{3.12}$$

where $|s| < 1$ and $\wedge$ represent the minimum operator. Combining this property with Equation (3.11) we derive

$$1 - s \leq \frac{d(f_j(\mathbf{p}_i), \mathbf{p}_i)}{d(\mathbf{p}_i, \tilde{\mathbf{p}}_j)} \leq 1 + s, \tag{3.13}$$

where $\tilde{\mathbf{p}}_j$ is the attractor of the fractal code f_j. We observe from Equation (3.13) that the FND $d(f_j(\mathbf{p}_i), \mathbf{p}_i)$ is bounded by a function of the contractivity factor s. Combining Equations (3.5) and (3.13) gives,

$$1 - \sqrt{\frac{t\alpha^2}{r}} \leq \frac{d(f_j(\mathbf{p}_i), \mathbf{p}_i)}{d(\mathbf{p}_i, \tilde{\mathbf{p}}_j)} \leq 1 + \sqrt{\frac{t\alpha^2}{r}}. \tag{3.14}$$

Interestingly, in the limit as s, or α, approaches zero the FND classifier approximately becomes the nearest neighbour classifier depending on the approximation accuracy of $\tilde{\mathbf{p}}_j$ to $\mathbf{p}_j$. This can be observed in Equation (3.15) by applying the Squeeze Law in calculus to Equation (3.13)

$$\lim_{s\to 0} d(f_j(\mathbf{p}_i), \mathbf{p}_i) = d(\tilde{\mathbf{p}}_j, \mathbf{p}_i) = d(\mathbf{p}_i, \tilde{\mathbf{p}}_j) \approx d(\mathbf{p}_i, \mathbf{p}_j). \tag{3.15}$$

The choice of encoding scheme also affects the recognition rate, because it affects the way features are represented in the transformations.

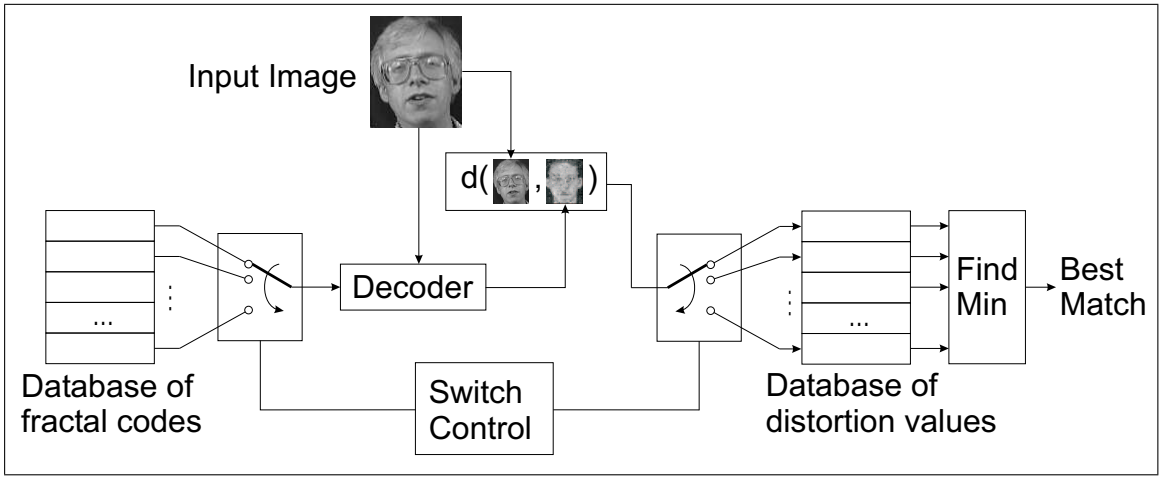


Figure 3.4: Block diagram of the Fractal Neighbour Distance based object recognition system, implemented for face recognition. The input image is decoded with each fractal code in the database. The fractal code that minimises the difference between the input image and the resultant image after one iteration of decoding is taken as the best match.

3.4.2 Invariance

The fundamental mechanism behind this recognition scheme lies in the uniqueness of the attractor of a fractal code. The FND calculates the distance between the input image and the result after applying one iteration of the fractal code approximating the reference image. If the input to the fractal code is close to the attractor of the code, then the FND will be small compared to an input that is further away from the attractor. It is obvious that if the input image is close to the attractor of the fractal code, then the result of one iteration will have little effect on the input image, and thus the FND value will be small. Of interest is when the input image is far from the attractor in the Euclidean sense, but still belongs to the same object class represented by the reference image. This occurs when the input is a distorted version of the reference. The FND has built-in compensation for some of these distortions, and this is due to the way self-similar regions are modelled by the fractal code. There is some invariance to rotations, scaling, translations, and illumination. Invariance to illumination occurs as a result of the FND measuring the distance between the input image and the resultant image after one decoding process. A fractal code with a large contractivity factor implies that the decoding

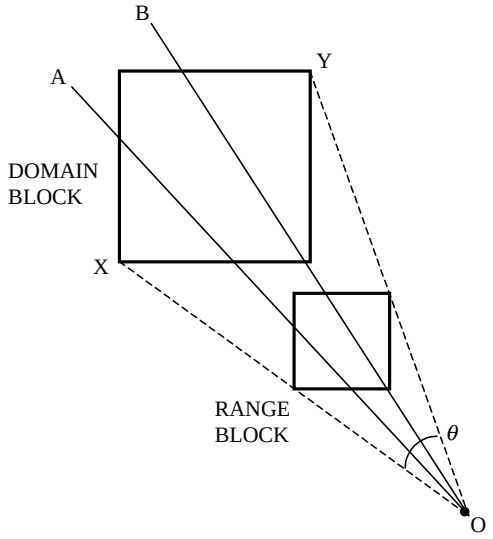


Figure 3.5: The invariance of a self-similar transformation for a line segment. All line segments between OX and OY passing through O have parts of it represented correctly by the self-similar domain to range transformation.

process will take longer to converge. This in turn implies that the resultant image after one decoding process will exhibit more similarities to the input image in terms of illumination levels. This renders the resultant image less sensitive to the actual illumination level in the attractor of the fractal code used to generate it, and more towards the illumination level of the input image. This effect can be described as a sort of automatic illumination adaptation to the input image.

We shall consider fractal codes consisting of affine transformations where the domain and range regions are square blocks with a decimation factor of two. The illustration in Figure 3.5 shows a domain to range block transformation. From the fractal encoding process we are ensured that the range block is a geometrically scaled, contrast and illumination adjusted version of the domain block - this transformation is self-similar. For this transformation to be invariant when the image underneath is distorted we require that transformation still be self-similar. Consider any line segment passing through O and lying between line segments OX and OY, the range block is a scaled version of the domain block. For this reason that particular domain to range transformation represents a self-similarity in all line segments through O within a range of angles given by θ. This argument can be extended to other shapes apart from line segments. That domain to range transformation has represented a

feature in an object that is invariant to rotations, translations and scaling as long as the combination of those distortions result in a new shape that is within the extent of the line segments OX and OY in the manner just described. This type of invariance can also apply to an object of arbitrary shape, as described below. It can be shown that the extent of line segment invariance as a function of θ can be expressed by the following equation:

$$\cos\theta = 1 - \frac{1}{2(T-1)^2}\frac{R_x^2 + R_y^2}{d_{RD}^2} \tag{3.16}$$

where R_x and R_y are the range block width and height respectively, d_{RD} is the Euclidean distance from the range block to the domain block, and T is the size ratio of the domain block to the range block. We see that the maximum extent is achieved when the range block sits at the center of the domain block, but this does not always produce the best domain to range block mapping with the maximum similarity between the two blocks.

The invariance just described occurs when the limit of the distortions are within the extent of a range block. However, it is also applicable to larger distortions as long as those distortions have pushed a self-similar region into the extent of the domain and range blocks.

For an object of arbitrary shape, Figure 3.6 illustrates the invariance to image distortions due to translation, rotation and scaling, or feature changes because of plastic deformations of the original object. Figure 3.6(a) shows the image of a curve before distortions with one domain to approximated-range block transformation. Note that the domain block is similar to the approximated-range block under an affine transformation. Figure 3.6(b) shows that same image after a slight rotation, scaling and translation, but retaining the same block transformation. The second figure shows that even after the distortions the domain block is still similar to the approximated-range block under the same affine transformation. Therefore, that transformation is invariant to the distortions we have applied. This block transformation has captured an invariant feature of the original object. This invariance is

strongest if the distortions are within the dimensions of one range block, becomes less as the distortions increase, and ceases to be invariant in this sense when the distortions extend outside the bounds of the range block.

However, invariance still exists when the distortions displace the original image outside the original range bounding box in such a way that the transformation still captures a self-similarity, but for a different part of the image. Therefore, there are three contributors to distortion invariance:

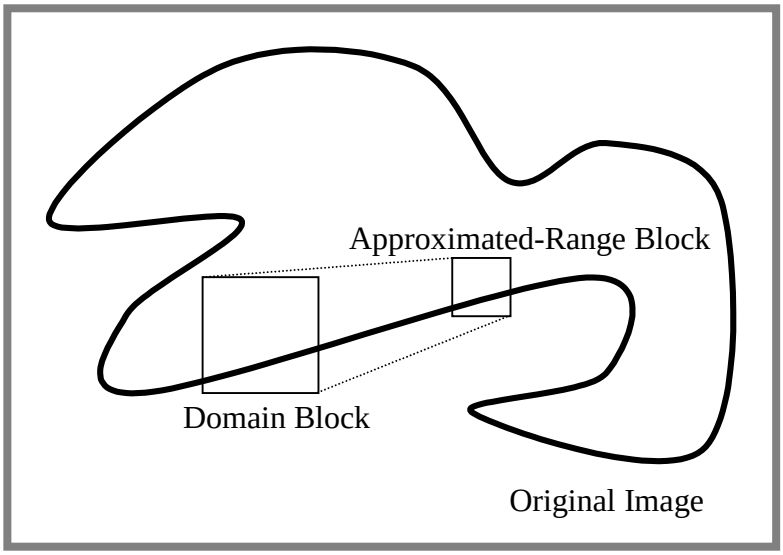
A. Automatic illumination adaptation from the iterative nature of the fractal decoding process. The contractivity factor determines the rate of convergence of the decoding process, thus directly affecting this type of invariance.

B. Invariance due to distortions that displace the original image no more than the bounding box of a range block. This invariance is range block size and shape dependent.

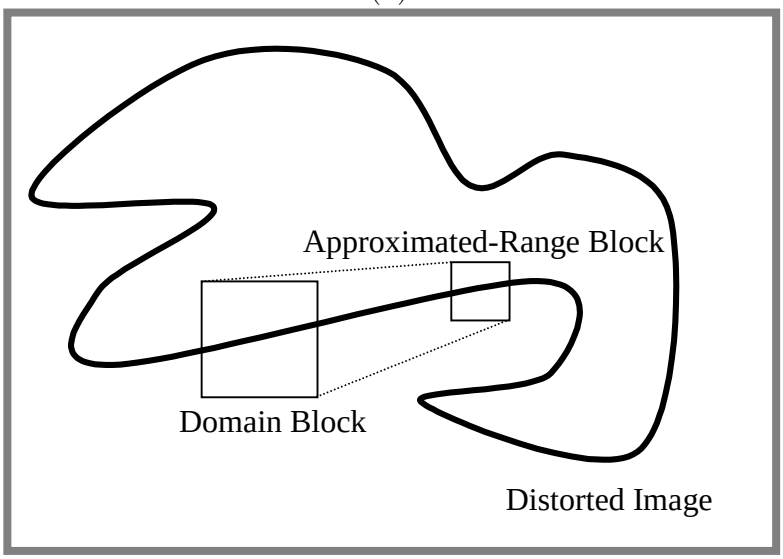
C. Invariance due to distortions that displace the original image more than the bounding box of a range block. This invariance is image dependent.

For invariance types (B) and (C) we do not consider the domain block because it is assumed that the domain block is larger than the range block, so the determining bounding box is dependent only on the range block.

The invariances just described are most effective against illumination and affine-type distortions. Therefore, these invariances also extend to out-of-plane rotations because such distortions can be described by a perspective transformation, which in turn can be modeled by a set of affine transformations [Hil90]. On the other hand, invariance does not extend to partial occlusions that may occur during out-of-plane rotations.



(a)



(b)

Figure 3.6: Invariance of the domain to approximated-range block transformation to rotation, translation, and scaling: (a) an original curve with domain to approximated-range block transformation shown; (b) the same block transformation on the distorted curve that is rotated, translated and scaled. Notice that the same transformation has still captured a similarity in the distorted curve. That is, the domain block is still similar, under an affine transformation, to the approximated-range block. Therefore, this transformation has extracted an invariant feature in the original curve.

3.4.3 Algorithm

In summary, the algorithm for recognition using the FND is as follows:

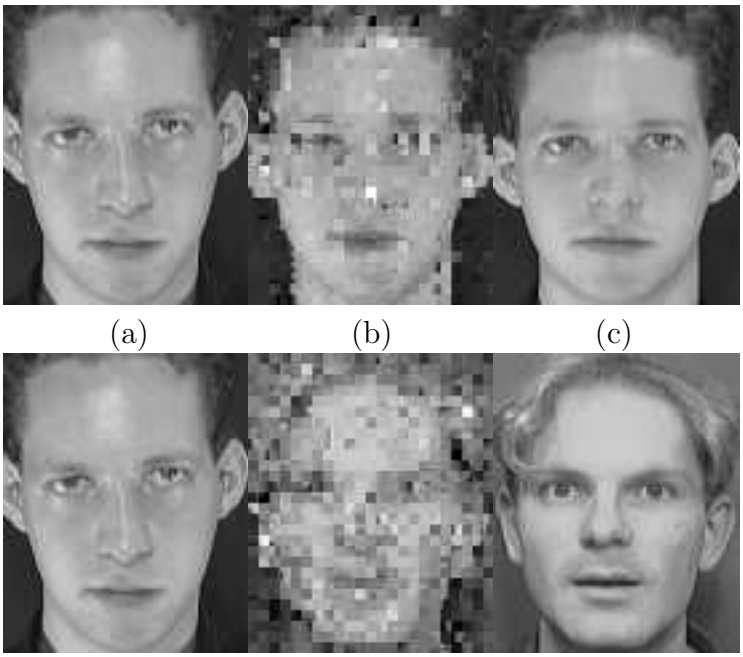
- Create a database of fractal codes representing the set of training faces.
- Consider unknown input images $\mathbf{p}_i$, where $1 \leq i \leq V$, and V is the number of input images. For each fractal code f_j in the database, where $1 \leq j \leq W$, and W is the number of training images, find $d(f_j(\mathbf{p}_i), \mathbf{p}_i)$.
- The best matching database entry to the input image is the one where $d(f_j(\mathbf{p}_i), \mathbf{p}_i)$ is minimised i.e.

$$j_{\text{best}} = \arg \min_{1 \leq j \leq W} d(f_j(\mathbf{p}_i), \mathbf{p}_i). \tag{3.17}$$

Figure 3.7 shows the results of decoding one input image using two different fractal codes. One of these fractal codes belongs to the same class as the input image. The figure shows that decoding with the fractal code of the same class results in less blocky images that are also visually more similar to the input image. For a more meaningful comparison, the two fractal codes are generated from two face images of different people in the same pose.

From the reasoning described in Figure 3.6, recognition results can be expected to improve if the input image is shifted in the horizontal and vertical directions before recognition. The amount of shift chosen is the dimensions of the range blocks. This increases the amount of invariance type (B). Thus, we can replace Equation (3.17) with,

$$\begin{aligned} j_{\text{best}} \quad = \quad & \arg \min_{1 \leq j \leq W} (d(f_j(\mathbf{p}_i^{\text{left}}), \mathbf{p}_i^{\text{left}}), d(f_j(\mathbf{p}_i^{\text{up}}), \mathbf{p}_i^{\text{up}}), d(f_j(\mathbf{p}_i^{\text{right}}), \mathbf{p}_i^{\text{right}}), \\ & d(f_j(\mathbf{p}_i^{\text{down}}), \mathbf{p}_i^{\text{down}}), d(f_j(\mathbf{p}_i), \mathbf{p}_i)). \end{aligned} \tag{3.18}$$



(a) (b) (c)

(d) (e) (f)

Figure 3.7: Results of one decoding iteration on one input face using two different fractal codes: (a) an unregistered face image of person A; (b) result of one decoding iteration using the fractal code of person A in a frontal pose; (c) person A frontal pose used to generate the fractal code that decodes the input image; (d) same unregistered face image of person A; (e) result of one decoding iteration using the fractal code of person B in a frontal pose; (f) person B frontal pose used to generate the fractal code that decodes the input image.

The direction of the shifts of the input image are indicated by superscripts in Equation (3.18), so $\mathbf{p}_i^{\text{right}}$ and $\mathbf{p}_i^{\text{up}}$ indicate input images that have been shifted right and up by an amount equal to the horizontal and vertical range block dimensions respectively. Experimental results, described later, indicate that recognition performance improves as a result of this modification.

Due to this extra shifting, domain and range blocks at the edge of the shifted images will be filled in by zero pixels. Therefore, the Euclidean distance given by Equation (3.3) is no longer accurate. We propose using a modified Euclidean norm for this particular purpose:

$$d_z(\mathbf{p},\mathbf{q}) = \sqrt{\sum_{k=0}^{I_h}\sum_{l=0}^{I_w}{}_{\mathbf{p}_{(k,l)}\neq 0,\,\mathbf{q}_{(k,l)}\neq 0}(\mathbf{p}_{(k,l)}-\mathbf{q}_{(k,l)})^2}. \tag{3.19}$$

We refer to the distance measure described by Equation (3.19) as the non-zero Euclidean norm. The components of the Euclidean distance are calculated only if the two pixels $\mathbf{p}_{(k,l)}$ and $\mathbf{q}_{(k,l)}$ are both non-zero. This effectively disregards those pixels that are at the borders of the shifted image, based on the assumption that most images rarely have zero pixel values. This distance is used for both encoding and recognition, so Equation (3.17) becomes,

$$j_{\text{best}} = \arg\min_{1\leq j\leq W} d_z(f_j(\mathbf{p}_i),\mathbf{p}_i). \tag{3.20}$$

and Equation (3.18) becomes,

$$\begin{aligned} j_{\text{best}} \;=\; & \arg\min_{1\leq j\leq W}(d_z(f_j(\mathbf{p}_i^{\text{left}}),\mathbf{p}_i^{\text{left}}), d_z(f_j(\mathbf{p}_i^{\text{up}}),\mathbf{p}_i^{\text{up}}), d_z(f_j(\mathbf{p}_i^{\text{right}}),\mathbf{p}_i^{\text{right}}),\\ & d_z(f_j(\mathbf{p}_i^{\text{down}}),\mathbf{p}_i^{\text{down}}), d_z(f_j(\mathbf{p}_i),\mathbf{p}_i)). \end{aligned} \tag{3.21}$$

For consistency the modification to the distance is also applied to the case where shifting is not performed, as shown in Equation (3.20).

3.4.4 Limitations

As mentioned above, our method has limited invariance to rotations, translations, scaling and illumination. This invariance is inherent in each domain to approximated-range block transformation and is the result of its representation of a self-similar feature in the image. The natural limits of its invariance depends on the contractivity factor, as given by invariance type (A), the bounds of the range block, as given by invariance type (B), and on the image's self-similarities under more extreme distortions, as given by invariance type (C).

Currently, there is no way to theoretically estimate the optimum value of the contractivity factor s, that will give the best recognition rate. One of the main difficulties is the fact that the optimum value is dependent on both the image in the database and the image to be recognised. Therefore, it is likely that any scheme that optimises s needs to re-encode the image in the database with the new s after analysing both images. We will further investigate this in future work. The experimental estimates of the best s are shown in Section 3.5.

Another drawback of this method is the extra computation required in decoding each fractal code in the database prior to using the Euclidean distance. However, optimisation of the decoding program code and the use of smaller image sizes gives good results and reduces computation time. The asymptotic time complexity of the recognition program is linear to the database size, that is $O(n)$.

3.5 Experiments

Various experiments are conducted to test the performance of the FND based classifier against other known methods. Experiments carried out for this chapter also include those based on the nearest neighbour classifier and the eigenface method, so a meaningful direct comparison with our method is possible. Other results such as those using self-organising maps (SOM) [Koh90], convolutional networks [LGTB97],

hidden markov models (HMM) [Sam94, SH94], probabilistic decision-based neural networks (PDBNN) [LKL97], and the combination of discrete cosine transforms (DCT) with neural networks [SBO+97], which use the same ORL database, are quoted from the literature for comparison. In our experiments, four different sets of training and test images from the same database are used. The results are averaged and used for comparison with the other recognition systems. As we have no exact knowledge of the selections of training and test images in the literature, an average of four different database selections are used to give a rough indication. In our experiments, recognition rates for individual database selections reach as high as 99.5%. The average recognition rate over the four selections is 98.25%. Experiments are also carried out on the Yale face database, and the results are shown below. The experiment in Section 3.5.5 quantifies the degree of invariance the FND measure has over the Euclidean distance.

3.5.1 ORL Face Database

The images in the Olivetti Research Laboratory database consist of faces of different people taken from different angles under varying lighting conditions. The images are centered on the face, though only approximately. Recognition is performed on the faces without further feature extraction or face/head detection. No effort is made to detect features or center the faces, as there are algorithms available that can perform that task quite satisfactorily [CWY95, CTB92, JM96, LBP95, RBK95, Vin95, YC96]. The database consists of 40 people, with 10 samples from each person, giving a total of 400 face images. The images are of size 92×112 pixels in 8-bit gray levels.

3.5.2 Experiment Conditions

Normalisations are not performed on the input faces. The faces in the ORL database are treated as the result of some face detection algorithm, noting that it still contain some varying poses. This is desirable since practical face detection and head extraction algorithms are not 100% accurate in determining the positions of the features. Therefore, the following experiments are designed to evaluate the discrimination ability and the inherent invariances of each method for a given data set, and not the face identification system as a whole. Hence no special regard is given to illumination variations or exact face locations as they are pre-processing and pre-recognition problems. During testing, each input test image is compared using a given algorithm to all training data of all persons in the database. No generic descriptions are used for each class of training images.

For the experiments in Sections 3.5.3 and 3.5.4, one to five face samples per person are used for training. The ones that are not used for training are used for testing. Four different sets of training and test images are used, and the average calculated. Three of those sample sets are selected at random, and the remaining set is selected by consecutively taking the first five faces of each person for training.

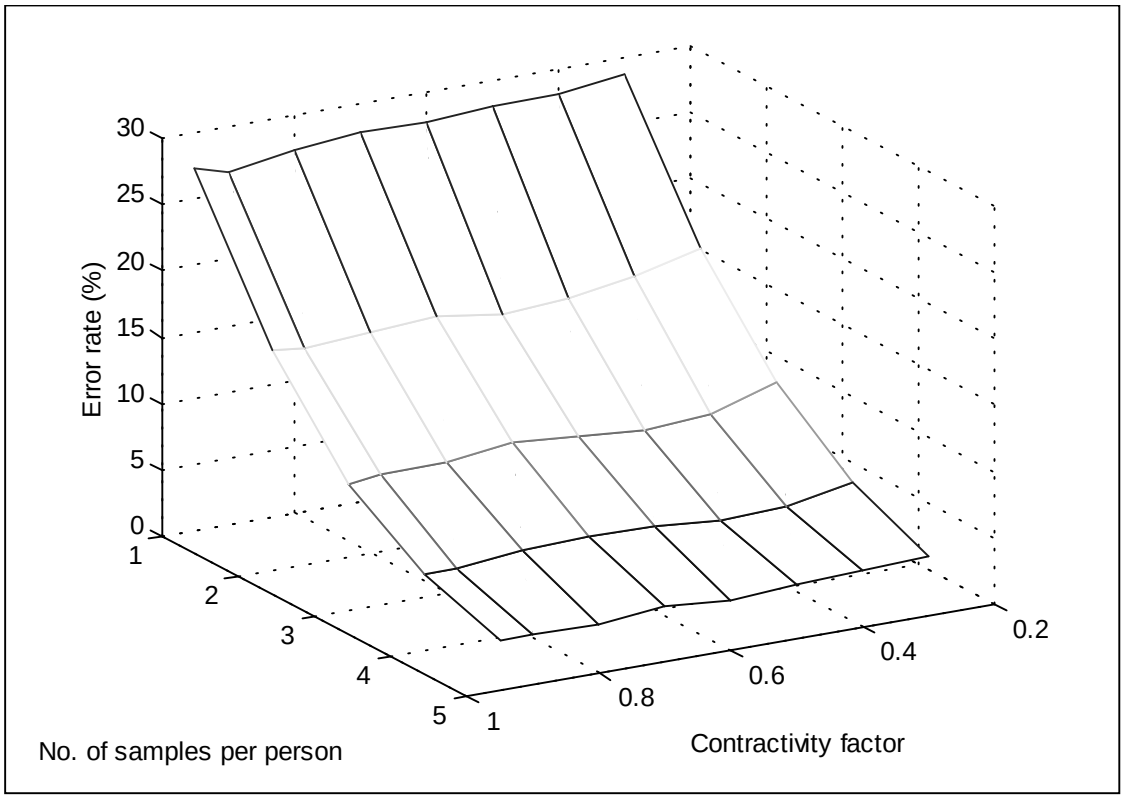
3.5.3 Fractal Neighbour Distance Based Classifiers

The choice of encoding scheme affects the recognition rate for the reasons discussed above. We experiment with three types of encoding schemes: uniform block, quadtree [Fis95a], and HV-partition [Fis95a] encoding. Shifting as described by Equation (3.21) is only performed for the uniform block encoding case. In all encoding schemes domain block dimensions are twice the size of range block dimensions, and the domain pool consists of all possible domain blocks.

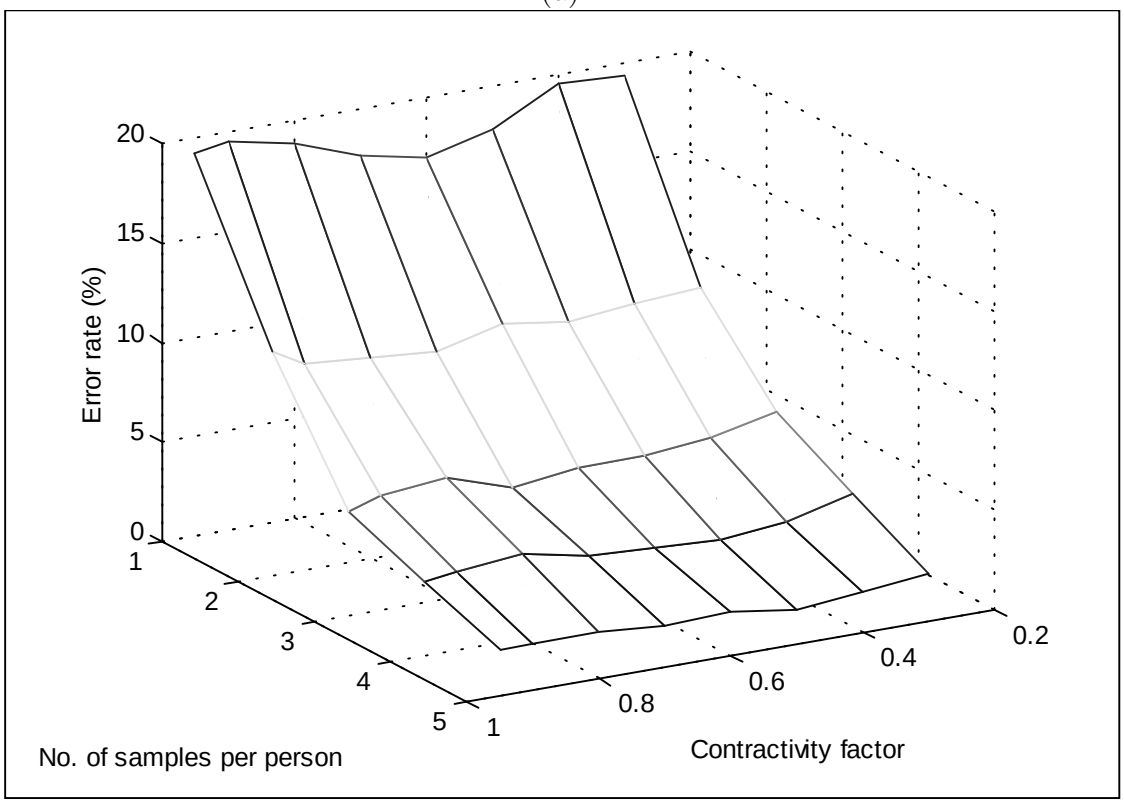
3.5.3.1 Uniform Block Encoding

In uniform block encoding the domain and range blocks are of constant uniform size, and the contractivity factors are controlled using Equation (3.5). The range block distribution is shown in Figure 3.9(a). For this encoding type, two different sets of experiments are performed. One of these has images in their original sizes, and the other has images that are decimated by a factor of two. The latter is used for experiments that involved shifting of the input image, as described by Equation (3.21). The reduction in image resolution is to counteract the effect of a four-fold increase in computational complexity due to the shifting of the input in four extra directions. To further reduce the computation time required, the decoding origin is shifted instead of the input image. This avoids the overhead of copying an image to each of the four shift positions, whilst maintaining a similar effect to an actual shift. Decoding occurs at the various shift offsets from the upper left-hand corner of the input image, and domain blocks crossing the image border are padded with zeros. Consequently, the computational times for the two methods remain approximately the same.

The maximum usage of a sub-domain block in a transformation is limited to $t = 6$. So, Equation (3.5) becomes $s \leq \sqrt{6\alpha^2/r} = \sqrt{6\alpha^2/4} = \sqrt{3/2}\alpha$. Other values of t could have been used, but it is found that $t = 6$ gives a reasonably good image approximation, and at the same time still gives control over s. If t is too large, the actual usage of a sub-domain block may not reach t. Thus, the estimate of s may not agree with the actual value. If t is too small image quality suffers because of a reduced domain pool. Figure 3.8 shows the error surfaces obtained from the two different sets of experiments. An average error rate of 3.75% is obtained using uniform encoding without shifting and with a contractivity factor of 0.9 when five samples per person is used. With shifting, an average error rate of 1.75% is obtained when the contractivity factor is 0.5.



(a)



(b)

Figure 3.8: Average error surface using uniform block fractal encoding: (a) original image sizes used; (b) images reduced by half and extra image shifting used as in Equation (3.21).

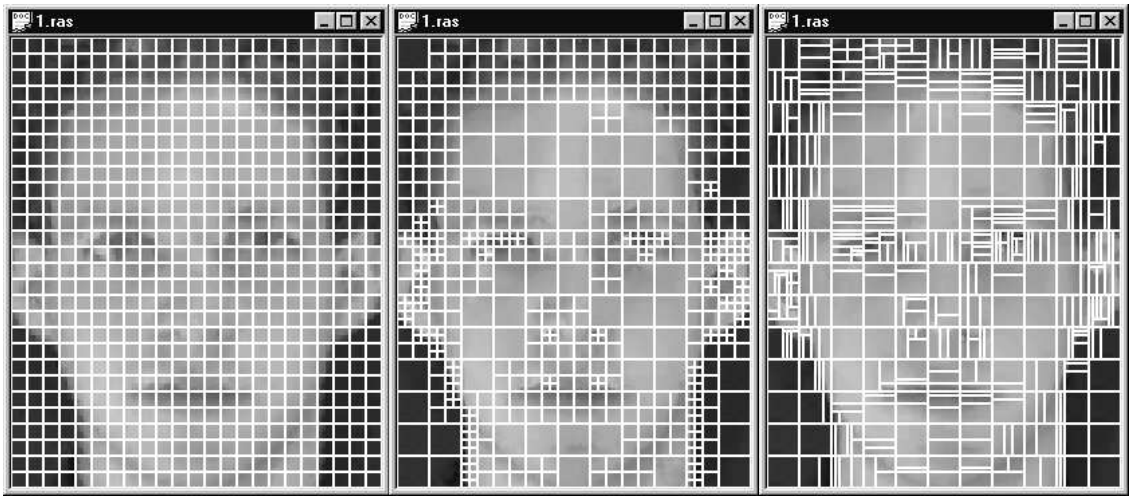
3.5.3.2 Quad-tree Partition Encoding

This type of encoding uses domain and range blocks of varying square sizes. Moreover, range/domain blocks of a particular size are twice as small as the next larger size and twice as large as the next lower size. Each range block without a best matching approximated-range block that is lower than some pre-set root-mean-square (RMS) threshold is divided into smaller range blocks. The new range blocks are half their original size in the horizontal and vertical directions, and the corresponding domain pool is searched again for their best match. Hence, the encoding process involves recursive partitioning.

The contractivity factor for this encoding type is not controlled. In particular, α is allowed to take any value in the range $[0.1, 0.7]$. An upper limit of 0.7 is placed on the value of α to ensure convergence of the fractal code. Figure 3.9(b) shows an example of the distribution of quad-tree range blocks. It is discovered that limiting the value of α to $[0.1, 1.0)$ result in some code having partial convergence. The reason for this can be seen in Equation (3.5), where an α value less than 1.0 does not always imply a contractivity factor of less than 1.0. So, for this experiment, α is limited to $[0.1, 0.7]$. This range is enough to guarantee total convergence for all the images in the ORL database. A graph of the results is shown in Figure 3.10(a). An average recognition error rate of 4.75% is obtained when five training samples per person are used.

3.5.3.3 HV-partition Encoding

This encoding scheme gives higher encoding accuracy because it utilise rectangular range/domain blocks. This encoding scheme also use non-overlapping range blocks and encoding involve recursive partitioning. In our implementation, the image is divided into large uniform square blocks. These blocks are then partitioned using the HV-partition scheme. Range block partitioning occur recursively



(a) (b) (c)

Figure 3.9: Range block distributions for different encoding schemes: (a) uniform block partitions; (b) quad-tree partitions; (c) HV-partitions.

in the horizontal or vertical directions yielding two new rectangles. They are partitioned at positions with strong vertical or horizontal edges, and continued until the range to approximated-range block comparison is lower than a pre-set root-mean-square (RMS) threshold. Figure 3.9(c) shows an example of the distribution of HV-partitioned range blocks.

The contractivity factor for this encoding type is not controlled. Again, to ensure convergence α is limited to the range $[0.1, 0.7]$ with truncation to the values outside that range. An expression for the contractivity factor for this encoding scheme is not derived, but α set to this range guaranteed complete convergence for the ORL images. A graph of the results is shown in Figure 3.10(b). An average recognition error rate of 4.63% is obtained when five training samples per person are used.

3.5.4 Other Types of Classifiers

3.5.4.1 Eigenface

The eigenface approach is a popular method for performing human face recognition. This method reduces the dimensions of the space spanned by a set of training faces using principal component analysis (PCA). Given a set of vectors formed by concatenating the rows of two-dimensional digital face images, a set of principal component vectors is selected. These components are orthogonal vectors that give the largest variance in the set of training vectors, which are taken as the basis vectors that span the space of human face images in the set of all possible images. Comparing the weights from an unknown face vector projected onto the basis then enables classification to be performed. One of the drawbacks of this method is that the principal components must be recalculated if more samples are deemed significant in generating the eigenface space. It has also been proven empirically and experimentally [BHK97] that as the number of principal components used approaches the number of training vectors, its recognition performance approaches that of the nearest neighbour classifier.

Forty eigenfaces are used to form the basis of eigenpictures. Such a high number of eigenfaces are retained to improve its performance. In general, its discrimination ability decreases with the reduction in the number of eigenfaces used, and is upper bounded by the nearest neighbour classifier [BHK97]. The weights for an input image are determined in terms of those 40 vectors. Recognition is conducted by finding the face in the database that has the closest set of weights to the input image in terms of the Euclidean distance. The input image is then assigned to the class of the best matching face in the database. A graph of the results is shown in Figure 3.10(c). An average recognition error rate of 6.75% is obtained when five training samples per person are used.

3.5.4.2 Nearest Neighbour

In this approach, a distance measure indicates the degree of separation between two samples in the pattern space. Here the Euclidean distance shown in Equation (3.3) is used. A set of samples from each person constitutes a class representing that person. An unknown input sample $\mathbf{p}_i$ can then be assigned to the class with a sample closest to it in terms of that distance measure. The results are shown in Figure 3.10(d). An average recognition error rate of 6.13% is obtained when five training samples per person are used.

3.5.4.3 Nearest neighbour with Shifting

Here the nearest neighbour method in the previous sub-section is implemented with extra shifting similar to that described by Equation (3.18). The following equation describes this extra shifting:

$$\begin{aligned} j'_{\text{best}} &= \arg \min_{1 \le j \le W} (d(\mathbf{p}_j, \mathbf{p}_i^{\text{left}}), d(\mathbf{p}_j, \mathbf{p}_i^{\text{up}}), d(\mathbf{p}_j, \mathbf{p}_i^{\text{right}}), \\ &\quad d(\mathbf{p}_j, \mathbf{p}_i^{\text{down}}), d(\mathbf{p}_j, \mathbf{p}_i)). \end{aligned} \tag{3.22}$$

In order to obtain a fair comparison with the method described in Section 3.5.3.1 that use image shifting, image sizes are reduced by half for this experiment. The shift amount in the horizontal and vertical directions is equal to the horizontal and vertical dimensions of the range blocks used in the fractal encoding experiments. To enable a fair comparison, the shift directions are chosen to be same as those in the fractal encoding experiments.

Results are shown in Figure 3.10(e). An average recognition error rate of 4.13% is obtained when five training samples per person are used. Figure 3.11 shows a graph comparing all the ORL face database experiment results.

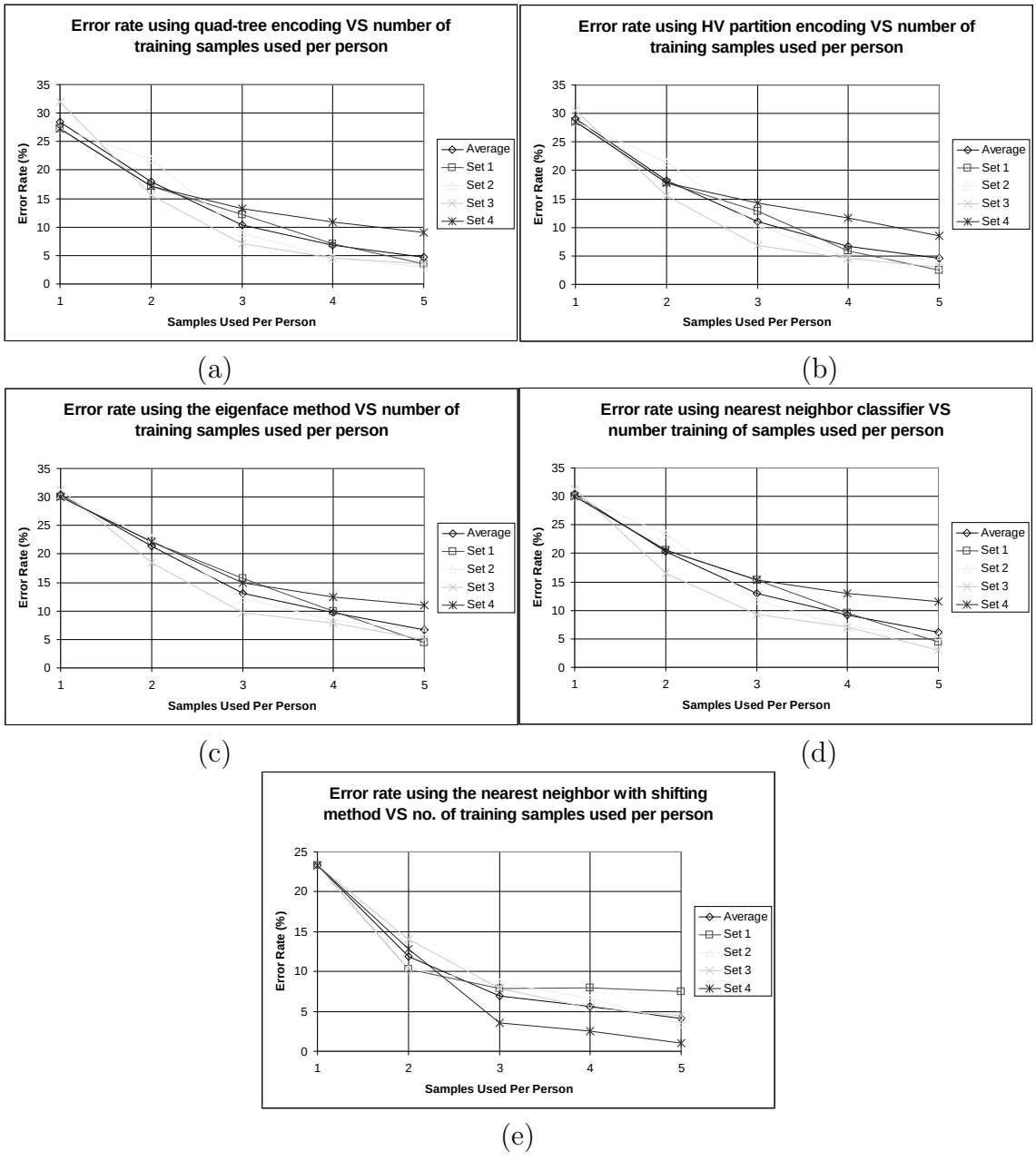


Figure 3.10: Recognition error rates of other methods: (a) quad-tree fractal encoding; (b) HV-partition fractal encoding; (c) eigenfaces; (d) nearest neighbour classifier using Euclidean distance measure; (e) nearest neighbour using the Euclidean distance with extra shifting in four directions as described by Equation (3.22).

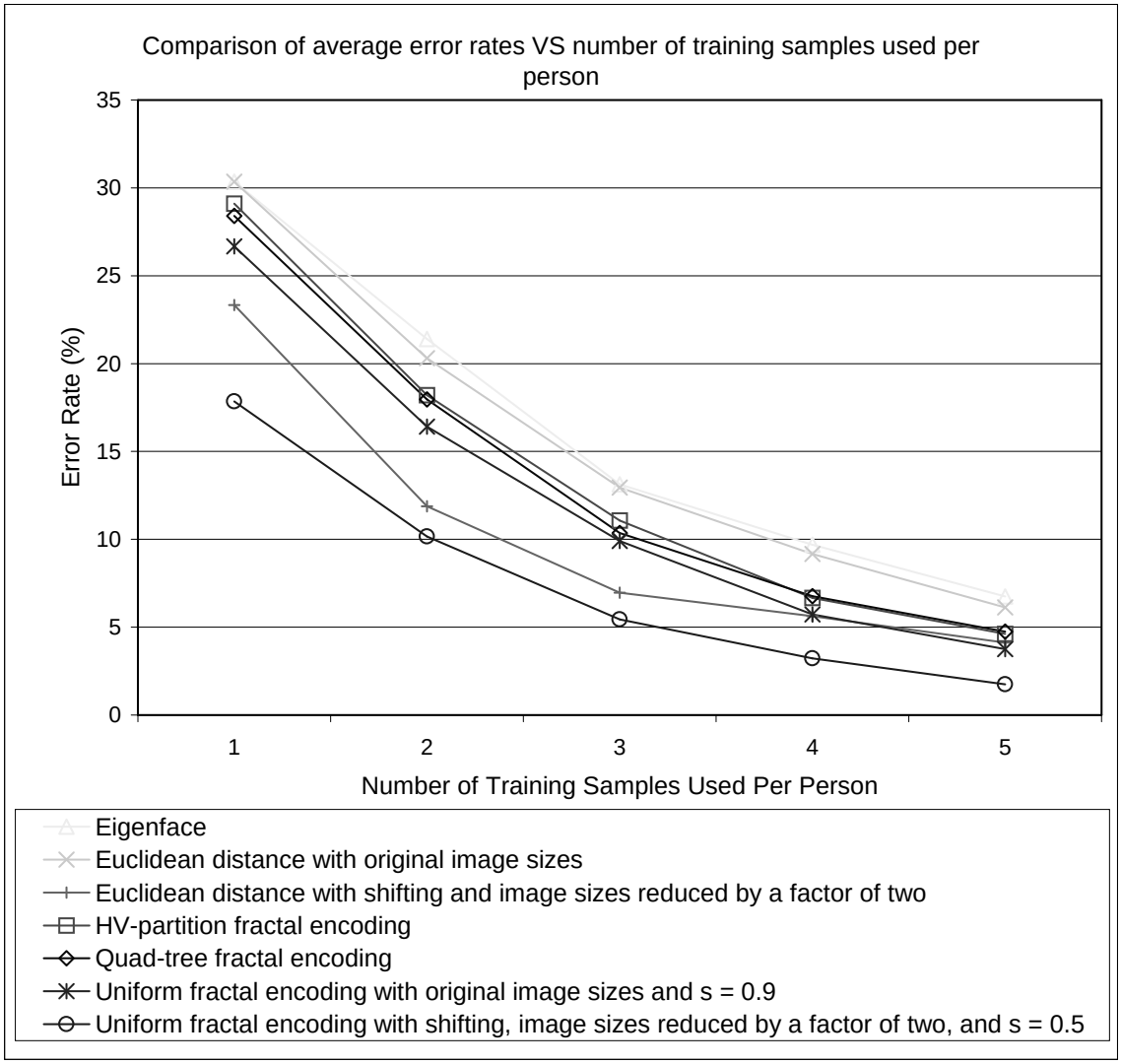


Figure 3.11: Average error rate results obtained from ORL experiments. The average is taken over four different sample sets.

3.5.5 Invariance Tests

In order to evaluate the relative performance of the FND and the Euclidean distance measures, comparisons between the two methods must be put within a relevant context. The aim of this experiment is to investigate using real data and under controlled test conditions if the FND outperforms the Euclidean distance in terms of recognition rate. On this basis, performance can be evaluated in terms of the number of recognition errors obtained over a set of test images. The error rate defined in Section 3.4.1 can be used.

Selected face images from the Olivetti Research Laboratory database are used. One face image from each person is used for training. The test faces used are distortions of various types and degrees of each of the training faces. In particular, 40 training faces are used, one for each person. Accordingly, there are 40 faces from which all test faces are derived through the following controlled distortions: rotation, scaling, horizontal translation, vertical translation, illumination, and horizontal and vertical perspective transformation. The latter two correspond to out-of-plane rotations of the image plane if the face image is treated as a flat two-dimensional surface. To remove any lighting effects or skin tone variations that might bias the recognition process or influence our own illumination tests, all training and test images are edge extracted versions obtained using Sobel operators. For image consistency over all distortion types the faces are rescaled to a square image of 64×64 pixels. A circular mask is also applied prior to any recognition or fractal encoding to reduce the background area and eliminate any artifacts generated from the rotation experiments. Figure 3.12 shows some example images and their distorted versions. Fractal encoding types with maximum contrast scaling factor limits of 0.3, 0.5 and 0.95 are investigated. For these experiments, no limits are imposed on the number of times sub-domain blocks can be used in the fractal codes, and uniform block encoding is used. Assumption (vii) Section 3.3.4 is relaxed to allow all α values up to and including the chosen limits.

Test faces for the rotation experiment are generated by rotating each face in an

anti-clockwise manner from 0° to 360° in steps of 2°. For the scaling experiment, faces are re-scaled using factors of 0.8 to 1.3 in steps of 0.005. Horizontal and vertical translations range from -20 to 20 pixels from the center in steps of 1 pixel. Illumination variation is synthesised by adding grayscale offsets for each pixel starting from -255 to 255 in steps of 5. Truncation is performed for resulting pixel values less than zero or larger than 255. Horizontal and vertical perspective transformations are applied with perspective factors of -100% to 100%. The amount of scaling and translations are chosen with the above limits because recognition performance differences are not significant between the two methods for values outside those ranges.

The results are summarised in Table 3.1. Recognition is performed for each distorted test image using the FND algorithm described in Section 3.4.3 utilising Equation (3.17), and the direct Euclidean distance based method also known as the nearest neighbour classifier. The latter algorithm is similar to the one described in Section 3.4.3, but employs databases of unmodified training images and utilises the Euclidean distance for image comparison. The index of the best matching face in the database is then found using

$$j'_{\text{best}} = \arg \min_{1 \leq j \leq W} d(\mathbf{p}_j, \mathbf{p}_i). \tag{3.23}$$

Both recognition methods have similar algorithms, differing only in the distance used. This permits a fair comparison of their performance in terms of recognition rate and their invariance to various distortions. The entries in Table 1 show the error rates for each deformation averaged over all test faces and all degrees of distortions of that type.

All FND results are better than the Euclidean distance results except for one case. This occurs when α is 0.3 during the illumination tests. Rotation tests show that the best case FND has an error rate that is 1.7% better than the Euclidean distance method. Scaling tests reveal a 8.3% improvement, horizontal translation 6.4%, vertical translation 6.2%, illumination shift 9.3%, horizontal perspective 10.5% and

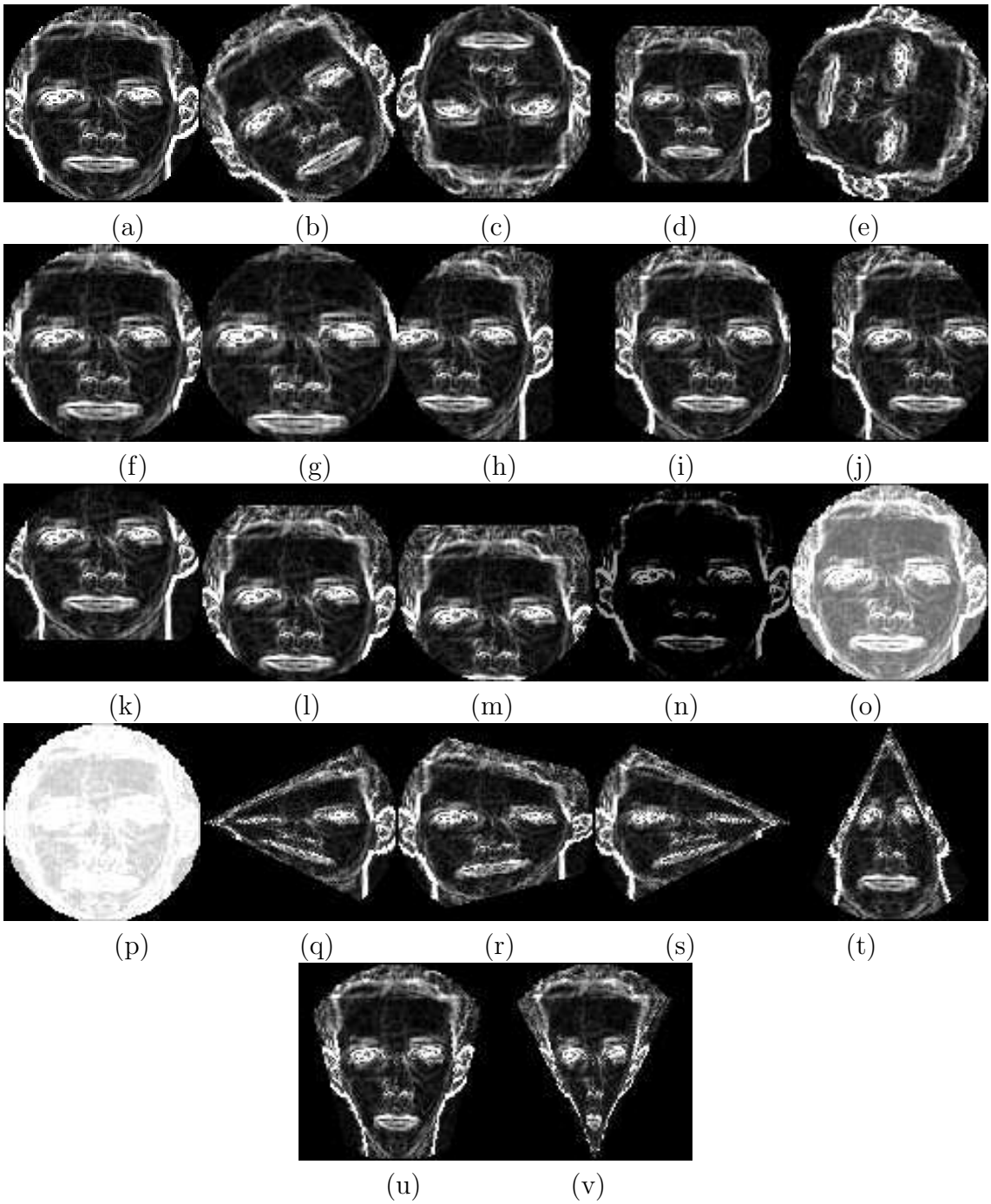


Figure 3.12: Examples of the types of distorted test images used for evaluating recognition rates in controlled deformation experiments: (a) a training face image, also used as a starting point for all other distorted versions that are used as test faces based on this training sample; (b)-(d) rotations at 30°, 180° and 260° respectively; (e)-(g) scaling at factors 0.8, 1.1 and 1.3 respectively; (h)-(j) horizontal translation at pixel offsets of −20, 10 and 20 respectively; (k)-(m) vertical translation at pixel offsets of −20, 10 and 20 respectively; (n)-(p) illumination shift with grayscale values of −100, 100 and 200 respectively; (q)-(s) horizontal perspective transformation at −100%, 50% and 100% respectively; (t)-(v) vertical perspective transformation at −100%, 50% and 100% respectively.

	Euclidean (%)	FND $\alpha = 0.3$(%)	FND $\alpha = 0.5$(%)	FND $\alpha = 0.95$(%)
Rotation	84.8	83.1	83.3	84.0
Scaling	39.1	32.3	32.1	30.8
Horizontal translation	66.8	62.5	61.6	60.4
Vertical translation	82.8	78.4	77.3	76.6
Illumination transformation	39.0	43.0	37.4	29.7
Horizontal perspective	39.9	37.6	32.4	29.4
Vertical perspective	31.8	26.3	22.0	17.7

Table 3.1: Results of recognition experiments comparing the invariances of the FND based classifier and the Euclidean distance based classifier to various deformations. The entries in the table indicate the recognition error rate for each deformation type averaged over all 40 test faces and all degrees of deformation for that type.

vertical perspective 14.1%. The implications of these results are further discussed in Section 3.6.

3.5.6 Yale Database

For this face database, only uniform fractal encoding with extra shifting using the FND is used in the experiments. This database consists of 15 individuals, with 11 different views for each person. The full faces including the head boundaries are located manually, and extracted into a passport style photograph. Expressions vary, but the poses are relatively consistent. Illumination varies widely, from center lighting to left lighting and right lighting. In one experiment, a simple pre-processing algorithm involving separate histogram normalisations of the left half and right half of the images is used to partially account for the large illumination differences. The experiments are performed using the "leave-one-out" method [BHK97]. The results are summarised in Table 3.2, and are discussed in the next section. The error rate without histogram corrections is 18.8%, with errors from extreme lighting effects

Method	Error rate	Errors due to lighting	Errors due to other factors
Full face	31/165 (18.8%)	30/31 (96.8%)	1/31 (3.2%)
Full face with independent left half and right half histogram equalisation	15/165 (9.1%)	15/15 (100%)	0/15 (0%)

Table 3.2: Results of "leaving-one-out" experiments using the Yale face database.

accounting for 96.8% of the errors. Non-lighting effects such as pose and expression account for 3.2% of the errors. With histogram corrections, the error rate is 9.1%, all of them due to lighting differences.

3.6 Discussion

Figure 3.8(a) and Figure 3.8(b) demonstrates that the number of views used in the training set has a more pronounced effect on the error rate than s. This can be observed by the relative flatness of the surfaces of s in relation to the error rate compared to the number of training samples used. The results in the two figures are obtained from actual face recognition experiments, which are subject to various poses, expressions and slight variations in illumination. However, under controlled distortion conditions as investigated in Section 3.5.5, the value of s, which is determined by α, has a pronounced effect on the error rate for some types of distortions and not others. The results in Table 1 demonstrate that α has the largest impact on the error rate for illumination, and for horizontal and vertical perspective changes. For controlled illumination tests, the error rate improve on average by 13.3% when α is increased from 0.3 to 0.95. In fact, when α is set to 0.3 the error rate is worse than the one obtained by the Euclidean distance method. For horizontal and vertical perspective tests, the average error rate improve by 8.2% and 8.6% respectively as α is increased from 0.3 to 0.95. This shows that the inherent invariances in the FND classifier also extend to out-of-plane distortions. With the exception of the rotation tests, the average error rates are 6% to 14% better than

the ones obtained by the Euclidean distance method when α was set to 0.95. The rotation tests actually show that the error rate increased by 0.9% as α increased from 0.3 to 0.95. This demonstrates that rotational distortion behaves inversely to the other distortion types investigated with respect to the relationship of α to the recognition rate. However, as shown in Figure 3.6 invariance to rotation still exists when the different types of distortions are combined in the right amounts. The significant effects of α in illumination and perspective tests naturally become less pronounced as the different types of distortions are combined. However, we can still analyse the results from such an experiment to gain insight into the mechanics of s, with the hope that it may be used to further improve the recognition rate in the future.

Figure 3.8(a) shows the recognition error surface as a function of the number of training samples per person and the contractivity factor. When a small number of samples are used the contractivity factor has a more significant effect on improving the error rate. With one sample per person, the error rate decreases with an increasing contractivity factor. Increasing the contractivity factor has a less pronounced effect with five samples per person. The reason for this is that changes in lighting, illumination and pixel-wise differences due to image distortions in the input image have a more significant impact on recognition ability if a small number of samples are used. This effect reduces as the number of samples increases. Increasing the contractivity factor increases the systems tolerance to those differences between the input image and the corresponding image used for training. The best results are obtained when the contractivity factor is around 0.9.

Figure 3.8(b) shows the error surface for the case when uniform block partitions are used with simulated shifting using decoding offsets. In this case, the best results are obtained when the contractivity factor is 0.5. It is purely coincidental that the images are also reduced by 0.5, as there is no direct relationship between the size of the images to the contractivity factor. Because of the extra shifting performed, the pixel-wise differences due to translations and other distortions are reduced. Therefore, only structural and feature differences dominate for input faces that

match with the corresponding face in the database. This illustrates the case when smaller contractivity factors lead to better results, in contrast to the previous case where larger contractivity factors are favored. This is more evident for the case where one sample is used for each person. The minimum value for this part of the surface in Figure 3.8(a) occurs when s is 0.9, and it occurs when s is 0.6 in Figure 3.8(b).

Figure 3.11 shows a comparison of the ORL face database results using the methods investigated in this chapter. When uniform blocks are used with shifting as described by Equation (3.21), the results are better than all the other methods. Uniform block partitions perform better than non-uniform ones. In particular, both the quad-tree and HV-partition fractal encoding methods give inferior results. The non-uniform encoding schemes impose a spatial structure on the fractal code due to the variability of the range block sizes. This spatial structure reduces the invariance of the FND to input distortions because that structure requires the input image to have features that are in close proximity to those of the encoded image to produce a small FND value. Both quad-tree and HV-partition based fractal codes give better approximations to the original image, but this is at the expense of using smaller range blocks at selected locations. From Figure 3.6 we can observe that smaller range blocks will reduce the amount of invariance to distortions. Ultimately it is a tradeoff between fractal code approximation accuracy and the amount of invariance to distortions. Therefore, object recognition using FND and finding fractal codes with increased accuracy demands opposing attributes. For recognition, reduced approximation accuracy is desirable if distortion invariance is increased, but only to the extent where the original image is still uniquely represented. With non-uniform encoding, unless the input image has features that conform to the structure of the fractal code, the FND does not give as good a measure of structural similarity in the images as the uniform case.

In addition, from Figure 3.11, we see that as predicted the performance of the eigenface system with 40 principal components is worse than the nearest neighbour classifier. The same figure also shows the result of using the nearest neighbour

classifier that is modified to include extra image shifting. Its performance is, for most parts, better than the non-shifted uniform encoding method, but levels out much faster than the other approaches. At five samples per person, the uniform-encoding-without-shifting algorithm outperforms this classifier. The best results are obtained when uniform encoding is used with shifting.

The experiments using the Yale database demonstrate the limits of the invariance of the FND classifier. Although the faces in the database has little pose or expression variations, the range of illuminations is beyond the capabilities of the FND. It is compelling to note that 96.8% of the errors from the algorithm without histogram corrections is due to images with left/right/center lighting. The other 3.2% are due to other factors such as pose and expression. This shows that under more extreme lighting conditions errors from non-lighting effects also tend to increase. With the algorithm that implements simple histogram corrections, all errors are due to left/right/center lighting. This illustrates a case where the FND handles geometrical deformations well, but not illumination variations.

The computational complexity of the FND algorithm is linear to the number of fractal codes in the database, i.e. $O(n)$, where n is the number of fractal codes in the database. With five training samples per person the recognition time for one input face is 3.2 seconds, and the time to train (encode) one sample is 0.31 seconds (performed on an Intel Pentium II 400MHz system running Windows 95. The times quoted are without any code optimisations). The average error rate obtained using our proposed system on the ORL database is compared with other systems from the literature, and the results are shown in Table 3.3. Shaded entries in that table indicate results that are obtained using a different machine configuration. Therefore, their classification and training times cannot be compared to those we obtained using our machine specifications, as indicated by non-shaded entries in the table.

Although our method is slightly more than an order of magnitude slower than some systems, it has a smaller training time. Our method also supports incremental

Method	Error rate	Classification time (s)	Training time
Eigenface[a]	6.75%	1.8	3 min 46 s
Euclidean[a]	6.13%	1.5	5.3 s
Euclidean with shifting[b]	4.13%	0.8	5.3 s
HV-partition FND[a]	4.63%	6.5	24 min 50 s
Quad-tree FND[a]	4.75%	9.1	35 min 10 s
Uniform partition FND[a]	3.75%	4.0	18 min
Uniform partition FND with shifting[b,c]	1.75%	3.2	61 s
2DCT+image synthesis+neural network	2.4%	0.1	2.95 min
SOM+CN	3.8%	0.5	4 h
PDBNN	4%	< 0.1	20 min
2D-HMM	5%	240	n/a
HMM	13%	n/a	n/a

Table 3.3: Comparison of our results with others quoted from the literature. The last five entries of this table are taken from Lawrence et al. [LGTB97], Lin et al. [LKL97], and Satonaka et al. [SBO+97]. The classification and training times quoted there are obtained on a different machine configuration, so they cannot be compared directly to our results. The first seven rows of the table report results that are obtained on the exact same machine configuration. [a]Original image sizes are used. [b]Image sizes are reduced by half before any processing. [c]Best result from our work.

training where faces can be added and removed from the database without re-training the system, which is required for both neural network and eigenface based systems. This is not entirely true for eigenfaces, where it is possible to add faces to the database without re-calculating the principal components. However, re-calculation is still required if the eigenface space must be modified by the addition of new training samples. Our system has the further advantage of being simple to implement; thus, further speed increases are possible through optimisations in software and hardware.

3.7 Conclusions

In this chapter, we present an object recognition system based on the fractal image coding technique. Fractal image coding reduces redundancies in images by exploiting the self-similarities within the image. Because of this, it also works well in extracting features and shape information in images and representing them in the form of block affine transformations. It is found that the input-output characteristics of a fractal decoder could be used for recognition. Utilizing the invariant properties of the fractal code to image distortions and illuminations, a further improvement to the recognition rate is achieved. We discover that the recognition rate is dependent on both the fractal encoding scheme and the contractivity factor of the fractal code. We also demonstrate the existence of a relationship between the contractivity factor and the recognition rate. To test our method, we successfully implement our system to perform human face recognition.

Our method achieves an average error rate of 1.75% on the ORL database when half the images in the database is used for training and the other half is used for testing. This method also supports incremental modification to the object database, where the addition or removal of a sample from the database does not require the re-training of the classifier.

The Fractal Neighbour Distance classifier perform better than other methods when uniform block partitioning is used. Experiments show that non-uniform block partitioning methods such as quad-tree and HV schemes introduce implicit structures that reduce distortion invariance. Regardless of the block partitioning type all FND methods still perform better than the Euclidean distance based nearest neighbour classifier. The eigenface method achieves the worst overall result. Further improvements to the uniform block partitioning FND classifier are attained with further image shifting prior to the decoding process.

Invariances inherent in the Fractal Neighbour Distance classifier are quantified through experiments using controlled distortion tests. The limits of the invariances

are explored by the use of the Yale face database. For future work, the invariances of this classifier will be analysed further with more complex image deformations.

Chapter 4

Controlling Convergence and Recognition Rates

4.1 Introduction

As described in Chapter 3, naturally occurring objects often exhibit a unique property where some of its parts are similar to other parts under some transformation. Mandelbrot introduces the term *fractal* to describe irregular objects having detail at all levels of magnification [Man82]. These objects also tend to have self-similar parts. Of interest are two-dimensional objects or projections of objects onto the two-dimensional plane.

As mentioned above, fractals can be described by a set of contractive transformations. These transformations collectively possess the property that repeated applications on consecutive results, commencing with any initial image, produces a unique image. This image is called the attractor, which is also a fractal. A reasonable approximation to this attractor is usually reached after several iterations. It is an NP-hard problem to find a set of transformations that can generate exactly any given image [RH97]. Fractal image coding overcomes this problem to an extent by using a collage of contractive transformations whose union approximates

any given arbitrary image. The transformations generated by this coding scheme is also referred to as a *fractal code*. The convergence factors in fractal image coding are the contractivity and eventual contractivity factors, which indicate convergence of the decoding process. This process must converge for it to regenerate a stable approximate of the original image.

This chapter focuses on the effects of changing fractal code properties such as the contrast scaling and illuminance shift factors for the Fractal Neighbour Distance (FND). This distance measure is discovered to have several invariant properties, which includes invariance to rotation, scaling, translation and illumination. The convergence characteristics and illuminance shift factors of a fractal code has a strong influence on the illumination invariance of the FND. Controlling the illuminance shift factors is straightforward. Controlling convergence has not been possible before, so we will present a method that can determine the convergence factors, and control of it is achieved through understanding the mechanisms underlying the fractal decoding process. The previous chapter demonstrated that the rate of convergence can be an important factor in fractal image coding based object recognition. Furthermore, with the proposed methods it is now possible to generate fractal codes with greater control over the properties of the resulting code.

The results given in this chapter also serve to eliminate the need for a proof of convergence for newly derived fractal encoding schemes by providing a common theorem for calculating convergence factors. It is true that it suffices to restrict the range of scaling factors to ensure convergence, but in this way there is no absolute control over convergence, especially eventual convergence, as required by our recognition experiments. It has been shown that it is possible to ensure eventual convergence by orthogonalising decimated domain blocks [OL95], but there is still no control over which decoding iteration convergence can be guaranteed.

Jacquin [Jac92, Jac93, Jac89] proposes that the contractivity factor of a fractal code is equal to the highest contrast scaling factor. Other methods of estimation have also been proposed [Kom95, HH94]. The matrix norm is used in the approach

suggested by Lundheim [Lun95]. These methods provide an estimate of the contractivity factor, but they have the drawback of being dependent on the encoding scheme. In particular, there are assumptions that domain and range blocks are square or rectangular and domain blocks must not overlap. Furthermore, there are no practical methods of determining eventual convergence. Eventual convergence means a code has guaranteed convergence because it has a contractivity factor of less than one after a finite number of decoding iterations.

Similar to Chapter 3, we only consider the task of identification in this chapter. Where we refer to "recognition" in this chapter we are in fact referring to identification. Generally it is not difficult to convert an algorithm that performs identification into one that does verification. For simplicity and evaluation purposes we only consider identification in this chapter. The same reasoning holds true for only considering identification in the previous chapter. Verification is treated in later chapters.

4.2 Background

An alternative treatment of fractal image coding to that in Chapter 3 is covered in this section, with more attention paid to the contractivity and eventual contractivity factors. A more complete treatment can be found in the works of various researchers [Jac92, Jac93, Jac89, Jac94, PY97, PY95, Dav98, Fis95a, Fis92, WdJ99]. In addition, the notations to be used in later sections are presented.

4.2.1 Fractal Image Coding

Fractal image coding involves the coding of an image into a set of sub-transformations, each of which obeys a set of properties. One of these is the condition that the transformations must be contractive. For a metric space (X, d) a transformation $f : X \to X$ is referred to as contractive if the following condition is

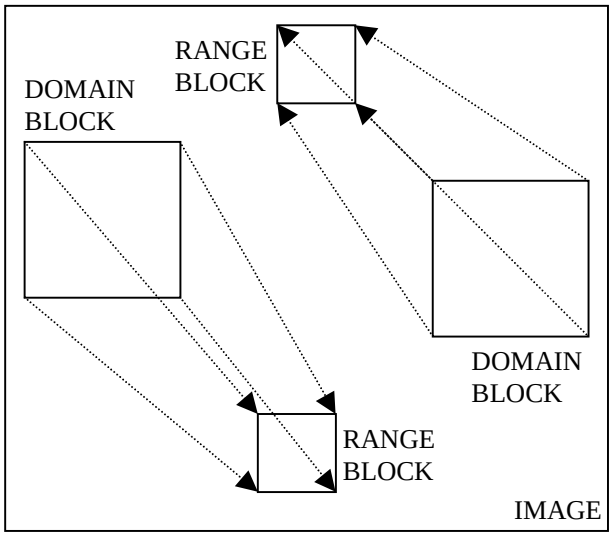


Figure 4.1: Affine domain to range block transformations in a fractal code. Executing all transformations collectively and repeatedly brings the image closer to the original image used to calculate the fractal codes.

satisfied:

$$d(f(\mathbf{x}), f(\mathbf{y})) \leq sd(\mathbf{x}, \mathbf{y}) \tag{4.1}$$

where s is called the contractivity factor and $s < 1$, X is the set of all two-dimensional digital images $\mathbf{x}, \mathbf{y} \in X$, and d is a distance measure. The factor s can be found by calculating the Lipschitz factor of f, which is a bound on how much f increases the norm of a vector in X. Throughout this chapter, d represents the Euclidean distance. The mapping characteristics of f determine the value of s. A finite set of mappings f_i constitute a fractal code if $d(\mathbf{x_{orig}}, f_i(\mathbf{x_{orig}}))$ is minimised, where $\mathbf{x_{orig}}$ is the image to be approximated with the fractal code. Sub-images in $\mathbf{x_{orig}}$, referred to as domain regions, are located, extracted and operated on by f_i, and the fractal code is defined as $f = \cup_{i=1}^{N} f_i(\mathbf{x_{orig}})$, where $f_i(\mathbf{x_{orig}}) \cap f_j(\mathbf{x_{orig}}) = \emptyset$ for $i \neq j$ and N mappings. Each $f_i(\mathbf{x_{orig}})$ is known as a range region. Domain regions are usually larger than range regions. Furthermore, f has a unique attractor A with the property that $A = f^{(\infty)}(\mathbf{x}) = f(A)$, for any $\mathbf{x} \in X$. In this chapter, we shall consider mappings f_i that are affine transformations.

Figure 4.1 illustrates some domain to range transformations in a fractal encoded image. Each range block is transformed from one domain block. We shall only

consider range blocks that do not overlap, and the union of which covers the entire image. However, overlapping of domain blocks is allowed. Encoding involves finding for each range block the best domain block obtained from a pool of domain blocks and subjected to an affine transformation, such that some cost function between them is minimised. This cost function can simply be the root-mean-square (RMS) difference. It is then obvious that using a larger pool of domain blocks usually results in better encodings, at the expense of a longer encoding time. In practice the domain pool is deliberately restricted by using domain blocks at incremental steps larger than one pixel value.

If the contractivity factor satisfies the condition $s < 1$ convergence is guaranteed, but this is not a necessary condition [Fis95a]. In particular, convergence is assured as long as the contractivity factor satisfies that condition for some iterate of f. That is,

$$d(f^{(n)}(\mathbf{x}), f^{(n)}(\mathbf{y})) \leq s_n d(\mathbf{x}, \mathbf{y}) \tag{4.2}$$

where the nth iterate of f is denoted by $f^{(n)}$, and s_n is the contractivity factor for $f^{(n)}$. We will refer to s_1 as the contractivity factor, and it is equivalent to just s. And the eventual contractivity factor refers to s_n for all $n > 1$.

4.2.2 Notations

We will refer to a generalised set of domain pixels of any shape as a domain region, and similarly for range regions. Then domain blocks and range blocks are just special cases of domain and range regions that are square in shape. Furthermore, we will refer to a range region that corresponds to a single pixel as a fundamental range region. A fundamental domain region is a set of sub-domain regions transforming to a fundamental range region. Consequently, a fundamental range region has the same dimensions as a fundamental sub-domain region. These regions are considered fundamental because they consitute the smallest units on which these transformations can take place. In general, a region in an image can be both a range region and a sub-domain region for two different transformations at the same

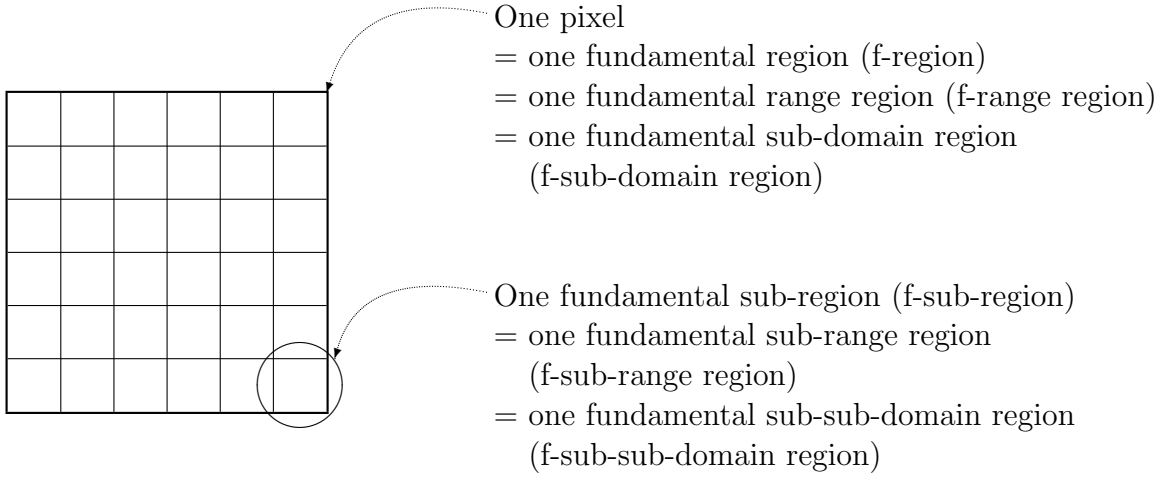


Figure 4.2: Illustration of fundamental regions. A single pixel is a fundamental region. Each pixel is further divided into sub-regions. In this example there are 36 sub-regions. Note that a fundamental domain region is made up of many fundamental regions. The abbreviations for each region type are shown.

time. In addition, we will refer to all fundamental range and sub-domain regions as fundamental regions. Figure 4.2 shows an illustration of the fundamental regions and also the abbreviations that will be used throughout this chapter.

Domain and range regions are vectorised by forming a column vector from consecutively labeled elements within the regions. The notations for representing a fractal code are the same as the one used by Lundheim [Lun95]. We shall employ the simplest non-linear transformations that comprise the fractal codes that are affine transformations. Let T represent a fractal code, then:

$$T\mathbf{x} = \mathbf{L}\mathbf{x} + \mathbf{t} \tag{4.3}$$

where $\mathbf{L}$, the matrix, and $\mathbf{t}$, the constant term, form the non-linear transformation $\mathbf{T}$. The input is an image $\mathbf{x} \in X$ that has been vectorised into a column formed by consecutive elements of each consecutive range-region in the original image. Other permutations of this vector are also valid and does not affect the derivations that follow.

The matrix $\mathbf{L}$ is composed of sub-transformations that fetches domain regions, decimate, multiply by a constant value and places the resulting range region into the appropriate position in the vectorised output image. Formally this is expressed

as:

$$\mathbf{L} = \sum_{n=1}^{N} \alpha^{(n)} \mathbf{P}_n \mathbf{D}_{n,u,t} \mathbf{F}_n \tag{4.4}$$

This term is composed of the summation of N non-overlapping range regions. Each range region is generated by a sequence of operators. The operators are defined in a similar way to the one proposed by Lundheim and are described in more detail in Appendix A. The operator $\mathbf{F}_n$ fetches the appropriate domain region, $\mathbf{D}_{n,u,t}$ decimates that region by factors u and t in the vertical and horizontal directions respectively, and $\mathbf{P}_n$ places the transformed region back to the appropriate location.

There are several ways to determine the contrast scaling factor $\alpha^{(n)}$, for the nth range region. The encoding schemes implemented in this chapter use Jacquin's [Jac92] method of calculating $\alpha^{(n)}$ by taking the ratio of the dynamic ranges of range and decimated domain regions. That is,

$$\alpha^{(n)} = \frac{|\max(\mathbf{p}^{(n)}) - \min(\mathbf{p}^{(n)})|}{|\max(\mathbf{D}_{n,u,t}\mathbf{p}^{(n_D)}) - \min(\mathbf{D}_{n,u,t}\mathbf{p}^{(n_D)})|} \tag{4.5}$$

where the nth domain region is given by $\mathbf{p}^{(n_D)}$, and the nth range region is given by $\mathbf{p}^{(n)}$. Another method of determining $\alpha^{(n)}$ is described by Fisher [Fis95a], which minimises the RMS distance between the transformed domain region and the range region. However, it was discovered that this version produces lower recognition rates in our face recognition experiments. The constant in Equation (4.3) is given by:

$$\mathbf{t} = \left[\gamma^{(1)} \dots \gamma^{(1)} \gamma^{(2)} \dots \gamma^{(2)} \dots \gamma^{(N)} \dots \gamma^{(N)}\right]^T \tag{4.6}$$

Other permutations of $\mathbf{t}$ in Equation (4.6) are also valid. The individual illuminance shift factor for each range region adjusts the mean of the transformed region to match the one in the original range region. This factor can be determined by:

$$\gamma^{(n)} = \frac{1}{B_i B_j} \left[\sum_{k=1}^{B_i} \sum_{l=1}^{B_j} \mathbf{p}^{(n)}(k,l) - \alpha^{(n)} \sum_{k=1}^{B_i} \sum_{l=1}^{B_j} (\mathbf{D}_{n,u,t}\mathbf{p}^{(n_D)})(k,l) \right] \tag{4.7}$$

where B_i and B_j are the vertical and horizontal sizes of the range region under examination, and n_D is the domain region that transforms to the nth range region.

4.3 The Contractivity Factor

The work in this chapter on the calculation of the convergence factors is an extension of the method proposed by Lundheim for the calculation of the contractivity factor. That method has the restriction that domain regions must not overlap. Specifically, for decimation by sub-sampling:

$$s = \sqrt{\max_{j \in Q_D} \sum_{n=R(j)} [\alpha^{(n)}]^2} \tag{4.8}$$

and for decimation by averaging:

$$s = \frac{1}{\sqrt{r}} \sqrt{\max_{j \in Q_D} \sum_{n=R(j)} [\alpha^{(n)}]^2} \tag{4.9}$$

where Q_D is the set of non-overlapping domain blocks, $R(j)$ is the set of indices of range blocks that transform from the jth domain block, and r is the number of sub-domain blocks that are decimated to one sub-range block. The derivations of these equations are performed in a discrete domain framework.

We will now present theorems for the calculation of convergence factors that apply to all fractal encoding schemes based on affine transformations with the following properties:

A. The decimation of fundamental domain to fundamental range regions must be held constant to the same integer value larger than, or equal to, one for all fundamental transformations.

B. Each range region is the result of a transformation from one domain region. Nevertheless, the theorems in this chapter are easily extended to transformations consisting of multiple fixed regions and/or multiple domain regions. Multiple fixed and domain regions are described by various researchers [WdJ99, GAH94b, GAH94a, GAH96].

C. The transformations are those that satisfy Equation (4.3).

D. No restrictions apply to the shapes of the domain or range regions, how the domain regions overlap, the amount of isometries applied, or the domain pool size.

Theorem 4.1 *Consider a fractal code approximating an arbitrary two-dimensional image satisfying properties (A) to (D). If the decimation ratio is $u:1$ in the horizontal and vertical directions, then the contractivity factor for the fractal code can be calculated in terms of fundamental regions and the l_2 norm by:*

$$s = \sqrt{\frac{1}{u^2} \max_{1 \le j \le N} \sum_{i=1}^{N} \rho_{i,j} [\alpha^{(i)}]^2} \tag{4.10}$$

where $\alpha^{(i)}$ is the contrast scaling factor for the ith f-range region, N is the total number of pixels in the image, and $\rho_{i,j} \ge 0$ is an integer representing the number of times the transformation for the ith f-range region utilises the jth f-sub-domain region.

Proof: See Appendix B. ■

Corollary 4.2 *Consider a fractal code approximating an arbitrary two-dimensional image satisfying properties (A) to (D). If the decimation ratio is $u:1$ and $t:1$ in the vertical and horizontal directions respectively, then the contractivity factor for the fractal code can be calculated in terms of fundamental regions and the l_2 norm by:*

$$s = \sqrt{\frac{1}{ut} \max_{1 \le j \le N} \sum_{i=1}^{N} \rho_{i,j} [\alpha^{(i)}]^2} \tag{4.11}$$

where $\alpha^{(i)}$ is the contrast scaling factor for the ith f-range region, N is the total number of pixels in the image, and $\rho_{i,j} \ge 0$ is an integer value representing the number of times the transformation for the ith f-range region utilises the jth f-sub-domain region.

Proof: See Appendix C. ■

4.4 Eventual Contractivity

Theorem 4.3 *Consider a fractal code approximating an arbitrary two-dimensional image satisfying properties (A) to (D). If the decimation ratio is $u:1$ and $t:1$ in the vertical and horizontal directions respectively, then the contractivity factor for the fractal code after the lth iteration can be calculated in terms of fundamental regions and the l_2 norm by:*

$$s_l = \sqrt{\frac{1}{(ut)^l} \max_{1 \le j \le N} \sum_{i=1}^{N(ut)^l} \rho'_{i,j} \left[\prod_{k=1}^{l} \alpha^{(i,k)} \right]^2} \tag{4.12}$$

where $l \ge 1$, $\alpha^{(i,k)}$ is the contrast scaling factor of the kth transformation in the path of transformations from the jth f-sub-domain region to the ith f-sub-range region, N is the total number of pixels in the image and,

$$\rho'_{i,j} = \begin{cases} 1 & \textit{if the } i\textit{th} \text{ f-sub-range } \textit{region utilises the } j\textit{th f-sub-domain region} \\ 0 & \textit{otherwise} \end{cases} \tag{4.13}$$

Proof: See Appendix D. ■

Equation (4.12) can also be used where decimation is by the sub-sampling method. Here, all sub-domain regions coincide with one single sub-domain region. Then the averaging performed in Equation (4.12) results in the same effect as sub-sampling. In the limiting case as t approaches u, $l = 1$, and with the use of non-overlapping domain regions, Equation (4.12) simplifies to Equation (4.8) and Equation (4.9) when decimation is by averaging and sub-sampling respectively. Jacquin [Jac93, Jac89] propose that the contractivity factor is given directly by the contrast scaling factor, α. Equation (4.12) confirms this, but more specifically the eventual contractivity factor can be determined directly by α. That is, if α is less than one, then there is some $l \ge 1$ such that $s_l < 1$. It is interesting to observe that for $\alpha < 1$ it is possible for s_1 to be larger than one. The result also confirms that convergence can still be guaranteed when there is no geometrical contraction, that is when $u = 1$ and $t = 1$.

Fractal codes without spatial contraction was investigated by Gharavi-Alkhansari and Huang [GAH94b, GAH94a, GAH96].

The theorems and equations presented above use fundamental regions. Traditional composite domain and range blocks, like the ones used by Jacquin, can be constructed by a set of fundamental regions. Isometries and other massic transformations [Jac92, Jac93, Jac89] between composite domain and range blocks are irrelevant in our analysis of the convergence factor, as long as the transformations can be reduced to simple affine transformations on fundamental regions.

4.4.1 Implementation

The calculation of Equation (4.12) can be implemented using a tree structure, which we call the influence-tree. Recall that each pixel is an f-range region, so a tree structure is created for each pixel in turn. The nodes of the trees store transform information such as the coordinates of the f-range region, the coordinates of the f-sub-domain regions, and the transform parameters. The root of that tree is the pixel being examined. The next level is created by inserting all f-sub-domain regions that are transformed to the f-sub-range regions in that f-range region. Each level represents one iteration of the fractal code, therefore the height of the tree is equal to the number of iterations at which we want to calculate the contractivity factor. Each leaf in the tree is an f-sub-domain region, and its existence indicates a usage by an f-sub-range region in the root f-range region. The usage of each f-sub-domain region is recorded as the sum of all multiplications of all the contrast scaling factors along the path to each root node. The maximum of this record is used to calculate s_l.

We observe that each level of the influence-tree increases by the power of ut, thus this algorithm has exponential complexity. In practice we found that the eventual contractivity factor can be calculated up to the sixth or seventh iteration for a tree that increases by a power of four with each iteration before the time required for

the calculation increases significantly. Redundancies in the tree can be exploited to reduce computation times in future.

It is known that fractal codes with spatial contraction converges faster than those without. As a further insight into the fractal image decoding process we can say that domain to range transformations with larger spatial contraction offers faster convergence because each pixel experiences an influence from other pixels that increases exponentially by the power of ut with each iteration. This corresponds to an increase in the generation of higher frequency components with each iteration.

4.5 FND for Object Recognition

4.5.1 The Fractal Neighbour Distance

We introduce a name for the classifier utilising the FND described in Equation (3.6) as the Fractal Neighbour Classifier (FNC). Classification of an unknown sample is performed by assigning to it the class of the nearest neighbour training sample with respect to the FND. Our experiments show that the FNC performs better than the nearest neighbour classifier (based on the Euclidean distance) in terms of reducing the relative distance between two images of the same class (see Experiment 4.6.2). The FND, which was defined in Equation (3.6) can also be written as:

$$d_{\mathrm{FN}}(\mathbf{u}, \mathbf{x}) \equiv d(f_{\mathbf{u}}^{(1)}(\mathbf{x}), \mathbf{x}) \tag{4.14}$$

where $\mathbf{x}$ is an input image, $f_{\mathbf{u}}$ is the fractal code whose attractor is an approximation of the reference image $\mathbf{u}$, $f_{\mathbf{u}}^{(1)}(\mathbf{x})$ is the result of the first iteration of $f_{\mathbf{u}}$ using $\mathbf{x}$ as the input, and d is the Euclidean distance.

The FND requires only the fractal code of the reference image. As such, compression issues normally associated with fractal image coding are not relevant here. Our interest lies in the fractal code itself and not how well it lends itself to compression, and thus it is not considered in our research.

4.5.2 Invariance in FND

Section 3.4.2 in Chapter 3 describes the invariances inherent in the FND. It is noted that the FND is invariant to illumination differences between the input and the reference image. Here we further discuss this invariance in terms of eventual convergence. By controlling how fast the input image converges to the attractor we can control how much pixel values change after the application of $f(\mathbf{x})$. Having slow convergence puts less emphasis on illumination differences between the input and reference image, and thus textural differences will dominate in the comparison between $f(\mathbf{x})$ and $\mathbf{x}$. There are practical limits on how slow we can make the convergence using the theorems presented above. In particular, it takes considerably more computation time to ensure eventual convergence at higher iteration levels. To summarise, illumination invariance is determined by the rate of convergence, which in turn is governed by the contractivity and eventual contractivity factors. Slower convergence results in more invariance to illumination differences between the input and reference image. The illumination shift factor γ, also influences this invariance type.

From the description above we see that the FND can be a better distance measure for tasks that require comparisons between class samples such as recognition because of its inherent invariance properties. This hinges on the assumption that the fractal code approximation for the reference sample can be obtained.

4.5.3 The Gamma Limit

The γ factor as defined in Equation (4.7) adds or subtracts uniformly all values in the destination range region. Therefore, it also has an effect on illumination invariance. Reducing the allowable range for this factor to $[-\gamma_l, \gamma_l]$ restricts the number of pixel value changes between iterations, and thus increases its invariance to illumination differences between the input and reference image. However, restricting this factor also reduces the approximation accuracy of the attractor image to the original

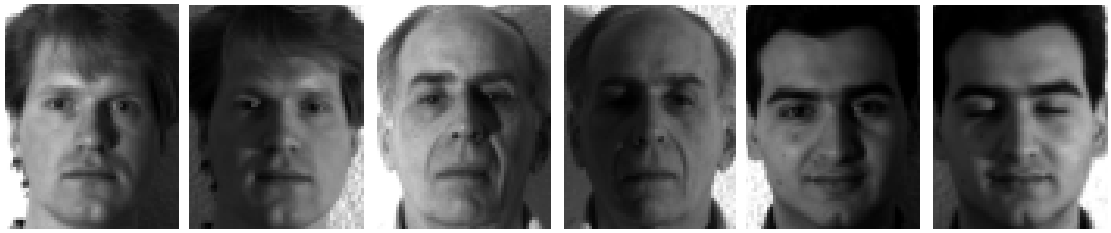


Figure 4.3: Examples of faces in the Yale database under more extreme left/right lighting conditions.

image. Consequently, finding the optimum value of this limit to obtain the best recognition rate requires the consideration of the trade-off between illumination invariance and approximation accuracy. In practice this is difficult to determine theoretically, therefore it is approximated experimentally.

4.5.4 Human Face Recognition

The aim of this chapter is to illustrate the advantage of FND as a distance measure, and not as a recognition system. Therefore, face detection and image normalisations are kept to a minimum. The Olivetti Research Labs (ORL) and the Yale Face Databases are used in our experiments. Although the ORL database is considered an easy database, it is used because the faces are centered on the image so that face detection is not required, and it contain faces of varying pose and expressions. The Yale database has face samples under more extreme lighting conditions, so it is used to test the FND's illumination invariance. Examples of faces from the Yale database are shown in Figure 4.3.

The FNC face recognition system is view-based and involves the creation of a fractal code database for the set of test face samples. One fractal code is created for each sample. Samples from the same person are kept separate and not combined into a generic class description. For each unknown input face the FND is calculated using each fractal code in the database. The unknown image is then matched to the person with the fractal code that generates the smallest FND value. The recognition error

rate can then be calculated as:

$$e_r = \frac{n_e}{n_i} \tag{4.15}$$

where n_e is the number of incorrect matches for n_i unknown input faces. It is assumed that the input face is of one of the persons in the database, so a rejection process is not required.

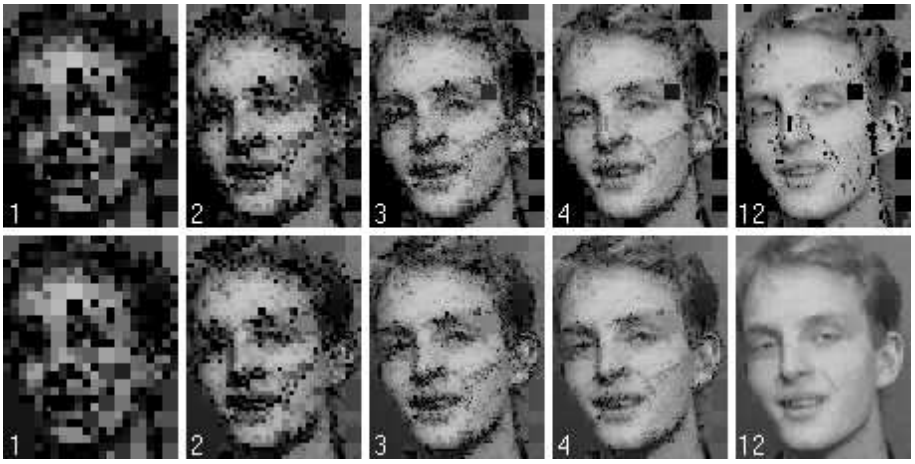
To further improve the recognition rate by increasing its invariance, the FND can be calculated at extra image shift positions. Five positions are used: left, right, up, down and center. The shift amounts are made equal to the physical dimensions of the smallest range block to obtain more type-C invariance as mentioned in Section 3.4.2 in the Chapter 3.

4.6 Experiments

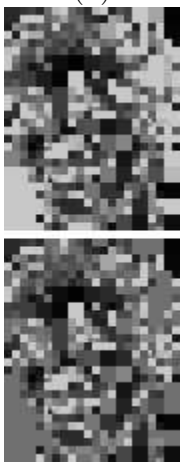
Experiments are performed to illustrate how Equation (4.12) can be used to control convergence, the reduction of the relative distance of inter-class to intra-class samples, how the FND can be used for face recognition, and the effects of convergence properties on the performance of the FND.

4.6.1 Controlling Convergence

This experiment demonstrates how a non-converging code can be made to converge by altering the contrast scaling factors of a few transformations. The approach taken here uses the influence-tree structure described above. All sub-domain blocks that transform to a root pixel constitute a branch in the tree. The leaves are examined to determine the f-sub-domain region that contributes to the contractivity factor, and all contrast scaling factors on the path from that f-sub-domain region to the root pixel is truncated to a fixed threshold. This process is repeated until a convergent code is obtained at the required iteration number.



(a)



(b)

Figure 4.4: Comparison of results before and after α modification to ensure convergence: (a) results of decoding with an initial black image using a fractal code with quad-tree partitions having a maximum contrast scaling factor of 1.2. The top row shows the results before α modification. The bottom row shows the results after α modification with the threshold set at 0.7 examined up to the first three iterations. The numbers on the bottom left corner of each image is the iteration number; (b) the intensities in these two images show the α values for each range block. The top image shows the α value distribution before the modification, and the bottom image is the result after the modification. The higher intensity blocks have α values of 1.2. After α modification some of these have changed to 0.7.

This algorithm is implemented and applied to the fractal code of a face image from the ORL database with the maximum allowable α set at 1.2. Initially this code is non-convergent as shown in the top row of Figure 4.4(a). We arbitrarily choose to have convergence at the third iteration with the most influential α factors set to $\alpha_M = 0.7$. The contrast factors at the nodes in the influence-tree is continually modified to α_M until the eventual contractivity factor is less than one after the third iteration. The eventual contractivity factor is re-calculated at the third level everytime a path from a leaf to the root is modified to check for convergence. The resulting eventual contractivity factors for the first few iterations are shown in Figure 4.5, and the outcomes from decoding with the modified fractal code is shown in Figure 4.4(a). Figure 4.4(b) shows the distribution of α values before and after the convergence modification. This figure reveals that a large uniform area initially with large α factors has been modified to α_M while retaining the larger α factors at higher frequency regions. This is expected because range blocks in regions with similar shade characteristics tend to be transformed from the same domain block while using the same encoding algorithm. That domain block will have a major contribution to the overall contractivity factor. Therefore, using this algorithm, the α factors for uniform regions are modified before edge regions. This graceful reduction in block approximation from uniform regions to edge regions to obtain convergence is desirable in our recognition task as edges contain more image structure information.

4.6.2 Relative Distance Reduction

The aim of this experiment is to demonstrate how FNC clusters samples from the same class closer together in relation to other class samples than the nearest neighbour classifier (NNC). Standard between-class and within-class scatter matrices cannot be used as a fair indicator of separability in the case of the FNC.

Let f_i denote the fractal code of the ith training sample, where $1 \leq i \leq n_T$, and n_T is the total number of training samples. Let x be a sample with an unknown

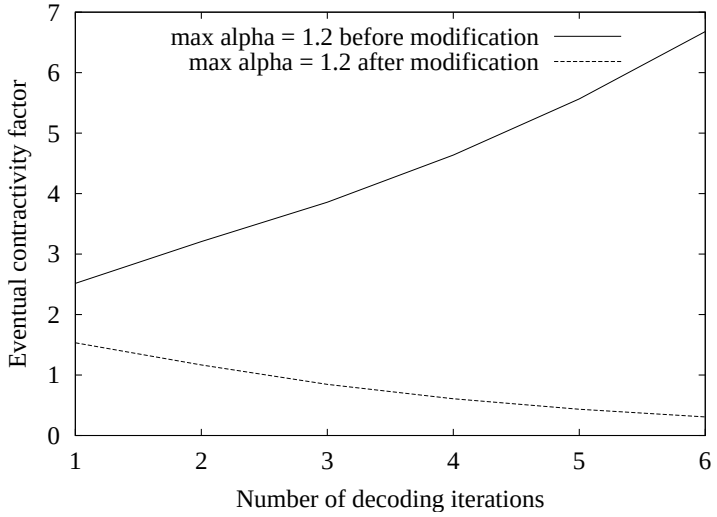


Figure 4.5: The eventual contractivity factor s_l, versus the number of decoding iterations using the fractal codes of a face image. The graph shows a comparison of the eventual contractivity factors before and after α modification. The ORL face image used is encoded with a maximum α factor of 1.2. The fractal code is then modified up to the third level of decoding, truncating α values to a value of 0.7. Not all α values are affected, thus the fractal code converged whilst still possessing a maximum α value of 1.2.

category. The set of n_T pairs $(f_1(\mathbf{x}), \theta_1), \ldots, (f_{n_T}(\mathbf{x}), \theta_{n_T})$, where $f_i(\mathbf{x})$ take values in the metric space X upon which is defined a metric d, then identifies the category of each $f_i(\mathbf{x})$ as θ_i. Each θ_i takes values in the set $\{1, 2, \ldots, n_C\}$, where n_C is the number of categories. θ_i is considered to be the index of the category to which $f_i(\mathbf{x})$ belongs. Given the pair (x, θ), where x takes values in the metric space X, we wish to find θ where x is the only known measurement. The definition of the FND in Equation (4.14) suggests that $f_i(\mathbf{x})$'s are distributed around $\mathbf{x}$. Ideally, $f_j(\mathbf{x})$ is closer to x under the metric d for all j where $1 \leq j \leq n_T$ and $\theta_j = \theta$ than $f_i(\mathbf{x})$ is to x for all i where $i \neq j$, $1 \leq i \leq n_T$ and $\theta_i \neq \theta$. A similar representation of the category assignments can be used for the NNC, given by $(x_1, \theta_1), \ldots, (x_{n_T}, \theta_{n_T})$. We now define the following relative distance difference to measure the performance of the FNC with respect to the NNC for an input sample x, where its category θ is assumed known,

$$R_1 = \frac{d(f_a(\mathbf{x}), \mathbf{x})}{d(f_b(\mathbf{x}), \mathbf{x})} - \frac{d(\mathbf{x}_k, \mathbf{x})}{d(\mathbf{x}_l, \mathbf{x})} \tag{4.16}$$

where $a = \arg\min_i \{d(f_i(\mathbf{x}), \mathbf{x}) | \theta_i = \theta\}$, $b = \arg\min_i \{d(f_i(\mathbf{x}), \mathbf{x}) | \theta_i \neq \theta\}$, $k =$

FNC Recognition Error Rate		11/200 (5.5%)
NNC Recognition Error Rate		18/200 (9%)
R_1	number of negative R_1s	117 (out of 200)
	mean R_1	-0.01545
	sum R_1	-3.090
R_2	number of R_2s < 1	117 (out of 200)
	mean R_2	0.9926

Table 4.1: Results from experiments on the Fractal Neighbour Classifier versus the Nearest Neighbour Classifier, based on the relative distance performance measures R_1 and R_2. The ORL face database is used. Five samples per person is used for training and the rest used for testing. R_1 negative or $R_2 < 1$ indicates that FNC performs better than NNC.

$\arg\min_i\{d(\mathbf{x}_i, \mathbf{x})|\theta_i = \theta\}$, and $l = \arg\min_i\{d(\mathbf{x}_i, \mathbf{x})|\theta_i \neq \theta\}$. The two ratio terms in Equation (4.16) give the relative distance of the nearest within-class neighbour to the nearest between-class neighbour. We can also define the ratio of the relative distances as a performance measure,

$$R_2 = \frac{d(f_a(\mathbf{x}), \mathbf{x})d(\mathbf{x}_l, \mathbf{x})}{d(f_b(\mathbf{x}), \mathbf{x})d(\mathbf{x}_k, \mathbf{x})} \tag{4.17}$$

The FNC outperforms NNC for an input x if R_1 is negative or if R_2 is less than one. Table 4.1 summarises the results of the calculations of R_1 and R_2 where five samples per person from the ORL face database are used for training. The remaining samples are used for testing as input samples. Extra image shifting for the FNC as described in Section 4.5.4 is not used so that a fair performance comparison with the NNC method is possible. The results from the table indicate that the FNC performed better than the NNC in reducing the relative distances of its nearest neighbours, which was correspondingly reflected in the recognition error rate.

It is interesting to note that R_1 and R_2 are indications of class separation, in the same way as the MIN/MAX ratio described by Brunelli and Poggio [BP92]. The results indicate that the FNC does indeed result in an increased class separation.

4.6.3 Application to Face Recognition

Human face recognition is performed using the FNC in the manner described above. With the aid of Equation (4.12) experiments are performed to investigate the effects of using α values that result in contractivity factors that are larger than one, whilst maintaining eventual convergence. The encoding scheme allows α values up to a certain threshold, instead of a fixed α as was used in Chapter 3. This results in better domain to range region matches. The method of modifying α is described by the following algorithm, which ensures convergence after the lth decoding iteration.

A. Create a tree structure for each pixel, where the pixel is at the root of the tree. Each branch from that root represents a transformation from an f-sub-domain region to that pixel. In turn, each f-sub-domain region has branches representing transformations to that region. Each level of the tree represents one iteration of the fractal code.

B. Calculate s_l using the tree structures.

C. If s_l is larger than one, find the f-sub-domain regions at the leaves of the trees that contribute to the value of s_l. For those regions, traverse up the tree and impose an upper limit by truncating each value of α to some threshold less than one.

D. Repeat step B until s_l is less than one.

Note that with this algorithm the threshold for α must be less than one for the modification process to halt.

Two different encoding schemes are used for the experiments: uniform partitioning and quad-tree partitioning. In uniform partitioning, the range regions are created by uniformly partitioning an image into square blocks. For our experiments range block sizes of 4×4 pixels are used and domain blocks are twice as large. Quad-tree [Fis95b] encoding initially involves uniform partitioning. Each block is then

examined in turn, searching for a best matching domain block. If the resulting range block from a transformation using the best matching domain block compared to the range block in the original image is above a pre-set threshold, then that range block is partitioned uniformly into four smaller uniform blocks. This process is repeated until all range block approximations are below that threshold.

Figure 4.6 shows the results of the face recognition experiments using the ORL Face Database where various encoding parameters are used. In the case where uniform partitioning is used (Fractal (i)-(iii)) the parameters are chosen to illustrate the effects of varying s. In the quad-tree case (Fractal (iv)-(vi)) the parameters are chosen to illustrate the effects of s and also the effects of using different minimum range block sizes. Comparison between the uniform and quad-tree methods is facilitated by using the same encoding parameters where possible, as shown in Fractal (i) and Fractal (iv). In all the FND experiments the average s is calculated by averaging the contractivity factors of all database entries for each encoding type. This is required because the values of s for each fractal code in the database can be slightly different although the same α range of values are used. Fractal (i) and (iv) uses maximum α factors larger than one, so eventual convergence is chosen to be guaranteed at the third or fourth iteration using a modified α of 0.7 with the influence-tree modification technique described above. A maximum of 20 modification iterations is imposed. The other fractal schemes use maximum α factors less than one so convergence is guaranteed even though their s values are larger than one. This can be confirmed by calculating their eventual contractivity factors. All classifiers described here that use the FND utilise the extra image shifts as described in Section 4.5.4. The figure also shows results from using the Eigenface method [TP91a] and the nearest neighbour classifier using the direct Euclidean distance.

An average recognition error rate of 1.1% is achieved at five training samples per person using the Fractal (i) method, which has uniform partitioning, average s of 1.8 and γ_l at 40. At this number of samples per person the recognition error

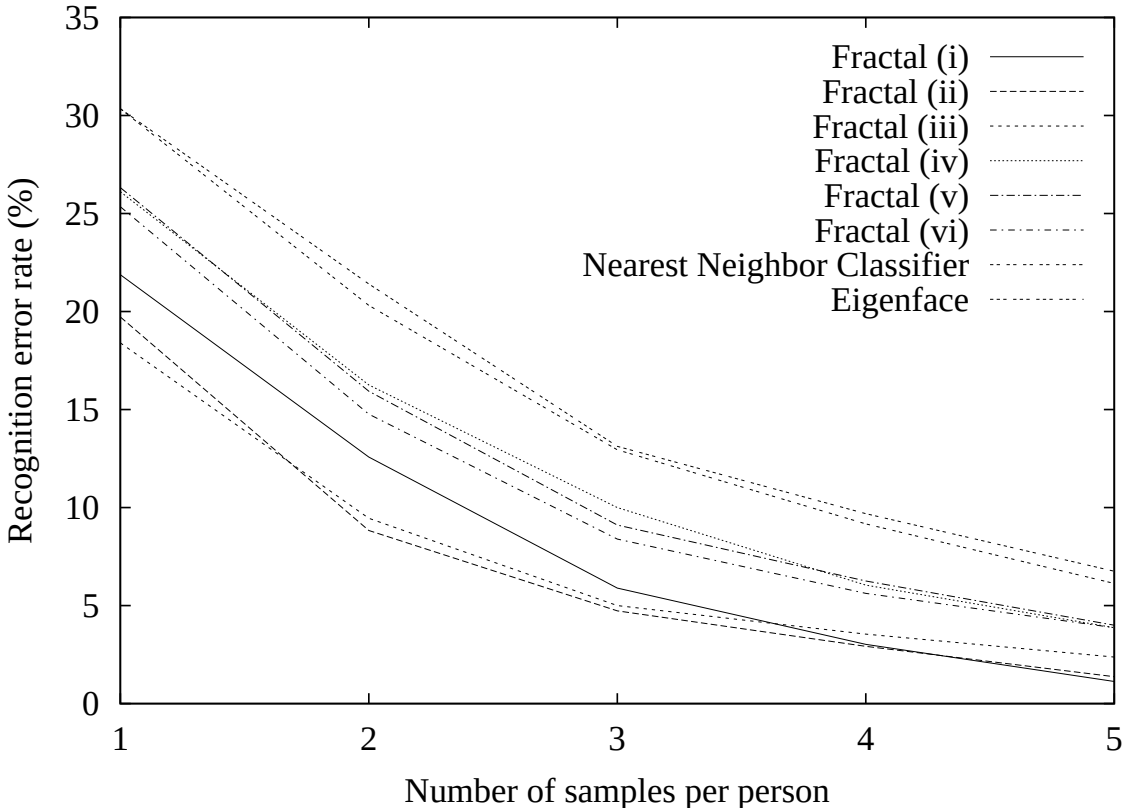


Figure 4.6: Recognition results using FND with uniform partitioning and quad-tree partitioning, nearest neighbour classifier (using Euclidean distance) and Eigenface. Fractal (i) - uniform partitioning, maximum $\alpha = 2.0$, minimum $\alpha = 0.95$, modified $\alpha = 0.7$, $\gamma_l = 40$, average $s = 1.8$; Fractal (ii) - uniform partitioning, maximum $\alpha = 0.9$, minimum $\alpha = 0.0$, no gamma limit, average $s = 1.3$; Fractal (iii) - uniform partitioning, maximum $\alpha = 0.5$, minimum $\alpha = 0.0$, no gamma limit, average $s = 0.85$; Fractal (iv) - quad-tree partitioning, maximum $\alpha = 2.0$, minimum $\alpha = 0.95$, modified $\alpha = 0.7$, $\gamma_l = 40$, minimum range size$= 2 \times 2$, average $s = 1.9$; Fractal (v) - quad-tree partitioning, maximum $\alpha = 0.7$, minimum $\alpha = 0.0$, no gamma limit, minimum range size$= 2 \times 2$, average $s = 1.1$; Fractal (vi) - quad-tree partitioning, maximum $\alpha = 0.7$, minimum $\alpha = 0.0$, no gamma limit, minimum range size$= 4 \times 4$, average $s = 1.1$.

rate decreases as average s increases with appropriate gamma limits. The recognition performance of fractal codes with higher contractivity factors improves as the number of views in the database increases, but degrades faster as the number of views decreases. This effect is more noticable as s increases. This can be explained by an increase in dependence on geometrical features as convergence rate slows and dependence on actual pixel value differences decrease. As the number of views increases the amount of geometrical information available for each person increases, therefore with slower convergence we can expect the recognition ability to be a stronger function of the number of views. This property can be observed as a greater change in the recognition error rate with the number of samples per person for the curve generated with uniform partitioning and average s at 1.8 as compared to that with average s at 1.3 or 0.85. This effect can also be observed with quad-tree partitioning with average s at 1.9 and 1.1 at the same minimum range block sizes. The effects of using larger minimum range block sizes (of 4×4 pixels) for this partitioning type reduces the error rate, as predicted by the discussion on invariance in Section 4.5.2. Obviously this improvement is at the cost of image approximation accuracy. The amount of error rate reduction is more than that obtained from increasing s. Comparing the curves of Fractal (i) to (vi) reveals that the quad-tree partitioning scheme yielded inferior results to all those obtained using the uniform partitioning scheme, regardless of the value of s or the minimum range block size. This is due to the use of non-uniform range blocks in the quad-tree scheme, which imposes an extra structural constraint on the image, which reduces its invariance to distortions. Although better approximations are possible they have less generalisation ability. In addition, usually smaller range blocks tend to be used at selected areas, which again leads to less invariance.

Table 4.2 shows the results of experiments using the Yale Face Database. Results of six experiments are shown. The first experiment, Method (A), uses FND with fractal codes that have contrast scaling factors truncated at 0.5, and histogram equalisation is performed independently for the left and right half of the test faces. The fractal codes in the second experiment, Method (B), have no gamma limits

"LEAVING-ONE-OUT" OF YALE DATABASE			
Method	Error rate	Errors due to lighting	Errors due to other factors
(A)	14/165 (8.5%)	11/14 (78.6%)	3/14 (21.4%)
(B)	32/165 (19.4%)	31/32 (96.9%)	1/32 (3.1%)
(C)	8/165 (2.4%)	7/8 (87.5%)	1/8 (12.5%)
(D)	2/165 (1.2%)	2/2 (100.0%)	0/2 (0.0%)
(E)	34/165 (20.6%)	33/34 (97.1%)	1/34 (2.9%)
(F)	37/165 (22.4%)	19/165 (11.5%)	18/165 (10.9%)

Table 4.2: Results from experiments with the Yale Face Database. The leave-one-out testing methodology is used. The face images are manually extracted to include the full face with hair and some background. The first method of implementation, Method (A), used FND with fractal codes generated using uniform partitioning, maximum $\alpha = 0.5$, minimum $\alpha = 0.0$, with the left and right half of the test face independently histogram equalised. The next three methods of implementation used FND with fractal codes generated using uniform partitioning, maximum $\alpha = 2.0$, minimum $\alpha = 0.95$, modified $\alpha = 0.7$, convergence guaranteed at the third or fourth iteration, and the maximum number of influence-tree modification iterations set at 20. The methods differed in the following way: (B) no gamma limits, no pre-processing; (C) no gamma limits, left and right half of input test face independently histogram equalised; (D) $\gamma_l = 6$, left and right half of input test face independently histogram equalised. The last two experiments were based on the nearest neighbour classifier using the Euclidean distance: (E) no pre-processing; (F) left and right half of input test face independently histogram equalised.

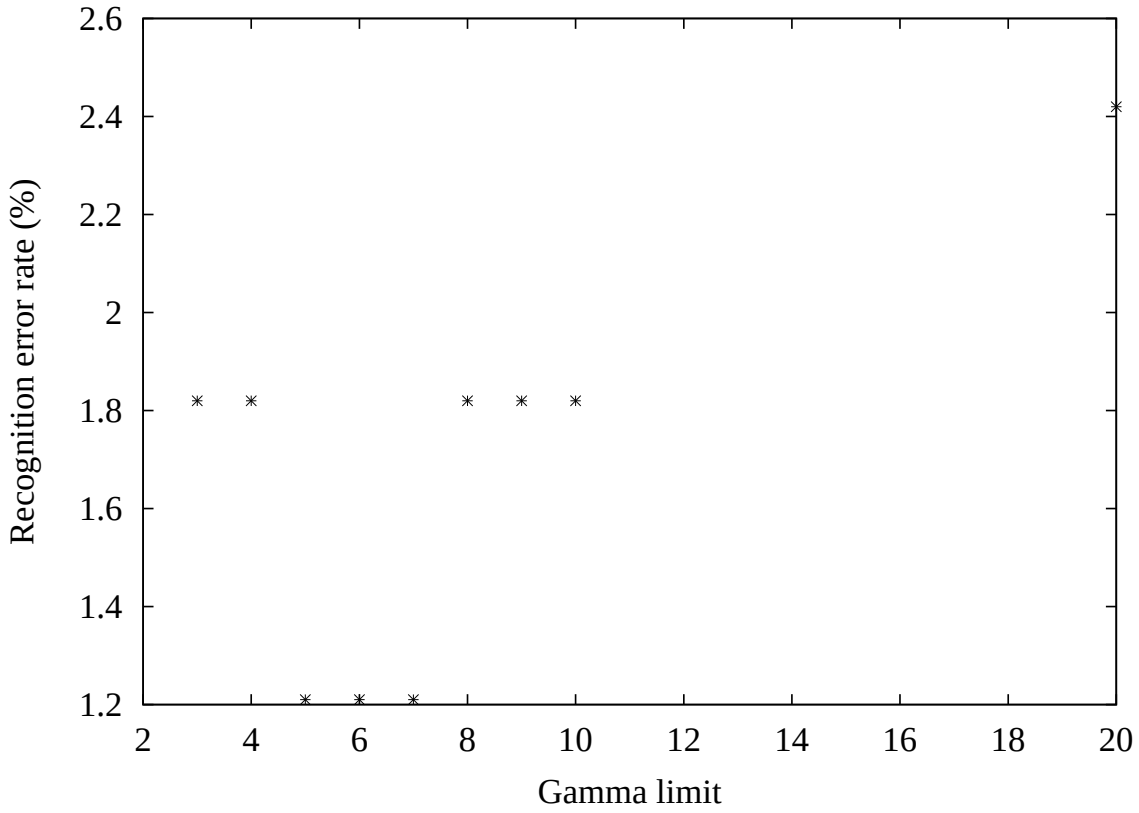


Figure 4.7: The relationship between recognition error rate and the gamma limit. The Yale Face Database is used with recognition performed using the FND with fractal codes generated using uniform partitioning, maximum $\alpha = 2.0$, minimum $\alpha = 0.95$, modified $\alpha = 0.7$, convergence guaranteed at the third or fourth iteration, and the maximum number of influence-tree modification iterations set at 20.

and no pre-processing is performed. The next experiment has no gamma limits, but utilises the same histogram equalisation operation. The fourth experiment uses $\gamma_l = 6$ and histogram equalisation of the input faces. Independent left and right half histogram equalisation is used to reduce the effects of severe left/right lighting conditions in some of the faces in the Yale collection. Figure 4.7 is a graph of experiment results showing the recognition error rate as a function of the gamma limit. The best gamma limit (γ_l=6) is used in Method (D) in Table 4.2. The fifth and sixth experiments are based on the nearest neighbour classifier method using the Euclidean distance, which include extra image shifting as described in Section 4.5.4 by the same amounts as those in the first four experiments. This is done so that a fair comparison is possible.

The results show that the FND does not perform well under extreme lighting conditions and is not much better than the Euclidean distance. However, just using simple histogram equalisations to reduce the extreme lighting conditions coupled with gamma limits improve the results dramatically to just 1.2%. This is competitive with the method by Belhumeur et al. [BHK97] where they quote a result of 0.6% using the Fisherface method under similar experimental conditions. Other results from that paper are Linear Subspace (15.6%), Correlation (20.0%), Eigenface without the first three components (10.8%) and Eigenface (19.4%). In comparison, Table 4.2 also reveals that the Euclidean distance method performed worse with the extra histogram equalisation than without, which show a significant rise in errors due to lighting. This emphasises the invariances in the FND that does not exist with the Euclidean distance.

4.7 Conclusions

A method of applying fractal image coding to object recognition is presented in this chapter. The Fractal Neighbour Classifier (FNC) based on the Fractal Neighbour Distance (FND) is introduced for recognition. The effects of the eventual contractivity factor on this distance measure is investigated. As such we require absolute control over the eventual convergence properties of a fractal code. To do this we derive theorems and present practical techniques that enable us to control the eventual contractivity factor. The effects of the illumination shift factor on the error rate is also investigated. It is found that imposing limits on this factor can increase the FND's invariance to illumination differences.

The FNC is implemented to perform human face recognition. The ORL and Yale Face Databases are used in the experiments. The ORL database is used as it does not require additional face detection or pre-recognition processing. The aim is to test the FND as a distance measure and its inherent invariances to distortions, and not as a complete face recognition system. The Yale database is used to test the

FND under more extreme lighting conditions. A recognition error rate of 1.1% is obtained on the ORL database, and 1.2% on the Yale database. The latter result is competitive with Fisherface results obtained using the same database under similar experimental conditions.

In addition, experiments are performed that illustrate how the eventual contractivity factor can be calculated at any given iteration, and how it can be used to control the iteration number at which we want convergence to be guaranteed. The techniques for doing this apply to a general class of fractal codes consisting of affine transformations with integral decimation in the horizontal and vertical directions.

The performance of the FND as a distance measure is also compared to that of the Euclidean distance using two proposed relative distance measures. With respect to these performance measures the FND is better than the Euclidean distance, which is correspondingly reflected in the improved recognition error rate. They results also reveal that the FNC in fact increases class separation.

For future work, the possibility of having an optimal method for modifying the contrast scaling factor α, to ensure convergence can be investigated. At the moment, a simple method of modifying the most significant α factors is used. Ideally, there should be a minimal effect on the image quality as a result of the modification, typically by reducing the number of changes to α. The effects of the contractivity and eventual contractivity factors on image quality is left for future investigations.

Furthermore, a faster algorithm can be derived to calculate the eventual contractivity factors by eliminating redundancies in the tree structure used to track the transformations. Currently, there is a practical limit of six to seven iterations for most encoding types before the amount of computation time required becomes excessive.

Chapter 5

Face Verification and Identification System

5.1 Introduction

Face verification is mainly used in access control applications. The usual mode of operation involves an individual attempting to gain access to a secured area or resource by providing a claimed identity together with a live image of their face. The system must extract the face of the individual and then compare it with a stored database of faces of the person with the identity that is claimed by that individual. In a practical application security issues must be taken into account, for example verifying that the extracted face is obtained from a live image and not from a photograph held up to the camera. We do not consider liveness detection in our work. The performance of verification systems are measured in terms of false accept rates (FAR) and false reject rates (FRR).

Face identification is used to identify an unknown face given a database of faces of known identities. Some identification systems also have a reject decision, where the unknown face is deemed not in the database. Similar to many experiments carried out by numerous researchers, our system does not implement this feature.

We assume that the identity of the unknown input face is the same as one of those in the database. The performance of identification systems is measured in terms of recognition error rates or positive recognition rates, similar to that described in Chapter 3 Section 3.4.

In this chapter we describe a face verification and identification system based on FNDs and the FFT-assist SVM eye detection system described in Chapter 2. Furthermore, we use FNDs that have been modified to operate with weights on subregions within the face. The motivation for these modifications comes from the findings from the work of several authors that the eye region has more discriminating information than other parts of the face [NS98, BP93]. In some cases using information from the whole face results in reduced recognition rates. For the identification system, we also propose the use of a voting scheme based on the weighted FND. For the verification system, we present the use of normalised scores and client specific thresholds. Using normalised scores give better results, and make our system competitive with current state-of-the-art methods in the literature. We also demonstrate that using client specific thresholds can potentially further improve verification results significantly, and in some cases outperform all the methods in the literature to which we compare our results.

The previous chapters describe the concept of using FNDs for face recognition and FFT-assisted SVM for eye detection as separate entities. In this chapter we present a complete face verification and identification system that combines the modified FND and FFT-assisted SVM eye detector. The modified FNDs are described in the next section. Additionally, for the sake of comparison, we also present a novel verification system that uses FNDs for both face localisation and verification. This method is specific only to the XM2VTS database.

5.2 Weighted FNDs

In this chapter, we aim to improve the performance of the FND by modifying it to operate over a set of locally weighted regions, from which a global score is derived. This modification is motivated by the fact that different parts of the face have different amounts of information for discrimination. It was pointed out by various researchers [NS98][BP93] that the region around the eyes have more information for discrimination compared to other parts of the face.

We use rectangular weighting functions. Let the height of an image in pixels be denoted by I_h and the width by I_w, and define b_t, b_b, b_l, b_r as the normalized ratios in the range $[0, 1]$, representing the top, bottom, left and right boundaries, respectively, of where the weighting function begins and ends. We then define the weighting function as

$$w(k, l, b_t, b_b, b_l, b_r, c_1, c_2) = \begin{cases} c_1, & \text{if } \; b_t < \frac{k}{I_h} \leq b_b \; \text{ and } \; b_l < \frac{l}{I_w} \leq b_r \\ c_2, & \text{otherwise.} \end{cases} . \qquad (5.1)$$

This weighting function is characterized by the boundaries b_t, b_b, b_l, b_r and the costs c_1 and c_2. The cost value c_1 is applied inside the boundaries and c_2 is the cost value outside those boundaries. We require the weighted FND to operate with different weighting functions. Let $\mathcal{W} = \left\{ \left(b_t^{(1)}, b_b^{(1)}, b_l^{(1)}, b_r^{(1)}, c_1^{(1)}, c_2^{(1)}\right), \left(b_t^{(2)}, b_b^{(2)}, b_l^{(2)}, b_r^{(2)}, c_1^{(2)}, c_2^{(2)}\right), \ldots, \left(b_t^{(n)}, b_b^{(n)}, b_l^{(n)}, b_r^{(n)}, c_1^{(n)}, c_2^{(n)}\right) \right\}$ be the set containing n weighting functions and $(b_t, b_b, b_l, b_r, c_1, c_2) \in \mathcal{W}$ be an element in that set.

Vertical and horizontal shifts, similar to those described in Equation (3.21) are also used. Let $\mathcal{H}$ be the finite set of desired shifts, where $(m, n) \in \mathcal{H}$ is an element in that set. In each element, m represents the number of pixels to be shifted in the vertical direction, and n is the number of pixel shifts in the horizontal direction. In this case, an image $\mathbf{q}$ shifted by (m, n) is then denoted by $\mathbf{q}_{mn}$.

The weighted FND (WFND) is then defined as

$$d_{wFN}(\mathbf{p}, \mathbf{q}) = \sum_{(b_t, b_b, b_l, b_r, c_1, c_2) \in \mathcal{W}} \min_{(m,n) \in \mathcal{H}} (d_{wnz}(f_{\mathbf{p}}(\mathbf{q}_{mn}), \mathbf{q}_{mn}, b_t, b_b, b_l, b_r, c_1, c_2)) \quad (5.2)$$

where

$$d_{wnz}(\mathbf{x}, \mathbf{y}, b_t, b_b, b_l, b_r, c_1, c_2) = \sqrt{\frac{\sum_{k=1}^{I_h} \sum_{l=1}^{I_w} \underset{x(k,l) \neq 0, y(k,l) \neq 0}{w(k, l, b_t, b_b, b_l, b_r, c_1, c_2)} \left(x(k,l) - y(k,l)\right)^2}{N_z}} \tag{5.3}$$

and

$$\begin{aligned} N_z \;=\; & |\{(k,l) : x(k,l) \neq 0, y(k,l) \neq 0, w(k, l, b_t, b_b, b_l, b_r, c_1, c_2) \neq 0, \\ & k = 1, \ldots, I_h \text{ and } l = 1, \ldots, I_w\}| \end{aligned} \tag{5.4}$$

The pixel value at coordinates (k, l) in the image $\mathbf{x}$ is denoted by $x(k, l)$. The same applies to the image $\mathbf{y}$. The WFND uses a normalized version of the non-zero Euclidean norm. The unnormalized version is shown in Equation (3.19). The factor N_z is the number of (k, l) positions where $x(k, l)$, $y(k, l)$ and $w(k, l, b_t, b_b, b_l, b_r, c_1, c_2)$ are all non-zero.

The weighted FND matches features at the local level and derives a local score. The local scores are summed to derive a global score. Effectively, the weighted FND uses each weighting function to search for a best matching region over a local area specified by the set of desired shifts $\mathcal{H}$. The value from the position that minimizes the locally weighted FND over this search region is selected and summed over all weighting functions to obtain the global score.

As described in previous chapters, the FND has some invariance to translation, scale, intensity and rotation. Therefore, each locally weighted FND has some of these invariances. The minimization of the locally weighted FND over a search area is computed independently for each weighting function, so their combined representation has the potential to be more invariant to non-linear distortions. This is because the relationship between local regions, in terms of their position within the image, is not geometrically restricted. The positions of the local regions depend on $\mathcal{H}$ and the local features that minimize the locally weighted FND. Therefore, the geometrical relationship between local regions can be complex and non-linear, increasing the chances of the WFND being invariant to complex deformations. This

is advantageous when considering images of human faces, which can deform non-linearly. However, it should be pointed out that the amount of invariance available from using the WFND rely on the local features being located accurately. If incorrect features are located using the locally weighted FND then we can expect degraded performance.

5.3 Verification System

5.3.1 Verification System Using FNDs

This verification system (VS-FND) uses a template-based face localisation algorithm using the FND. This algorithm is specific only to the type of face samples used. The FND is used for face verification. This was the system used to create the results as published by Matas et al. [MHJ+00], which took part in the face authentication competition held as part of the 15th International Conference on Pattern Recognition 2000. Our system managed to win first prize, despite the fact that our results were not the best. The reason being that the results submitted by other authors either did not conform to the Laussane Protocol or were submitted after the due date.

The block diagram for system VS-FND is shown in Figure 5.1. Face localisation and extraction is carried out on the captured image using an FND template based method, which utilises a significant number of assumptions about the input image. Face verification is then carried out using the FND by comparing the input face with those in a database.

5.3.1.1 Face Localisation Subsystem for VS-FND

As mentioned above, the face localisation and extraction subsystem implements an FND template based method utilising a significant number of assumptions about

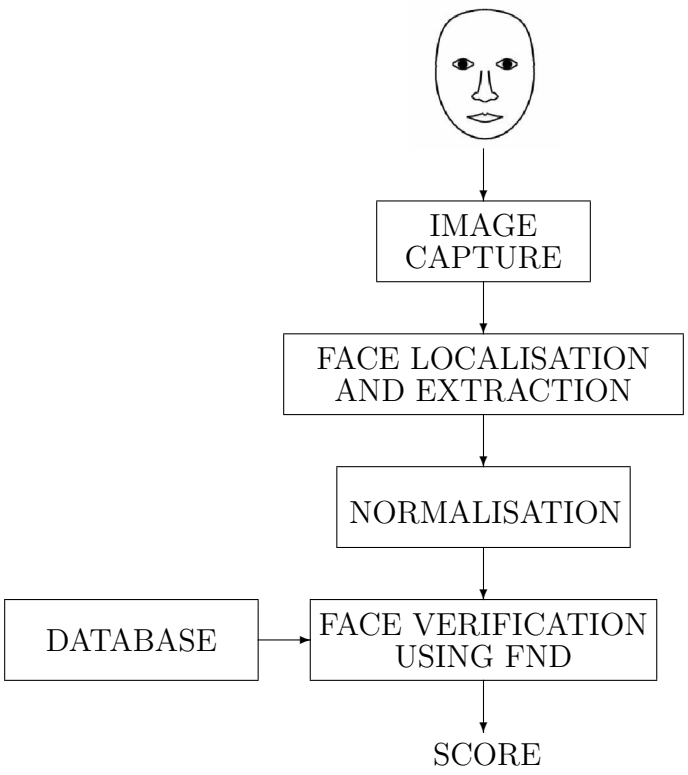


Figure 5.1: Face verification system VS-FND. Face localisation and extraction is template based and uses assumptions about the input image. The extracted face is 76 pixels high and 56 pixels wide.

Figure 5.2: Example of faces from the XM2VTS face database.

the input image. In particular, we assume that the captured images are those from the XM2VTS database [MMK$^+$99]. The database consists of 2360 faces from 295 people, taken in front of a blue background. The images are 576 pixels high and 720 pixels wide. Some of these faces are shown in Figure 5.2

The blue background assists the process of locating the head boundary. This can be done by simple colour thresholding. Input images to the face localisation algorithm are of those from the XM2VTS database with red, green and blue colour components available. A template of size 32×32 is used (shown in Figure 5.3). This template, used for comparison with the image underneath using the FND, was found experimentally to perform well in locating faces. The dark regions in the template correlate with the positions of the eyes, nose and mouth. In a grayscale image, the eyes, nose and mouth generally have darker pixels. The nose and mouth regions are combined accounting for moustaches and beards. Symmetry is implicitly imposed by the symmetry of the template.

A. For $i = 4.5$ to 6.5 in steps of 0.5.

B. Reduce the size of the input image by the factor of i to obtain image height I_h and image width I_w.

C. Restrict the search region to an area within the resized image that starts at $0.035 \times I_h$ from the top, $0.07 \times I_w$ from the left, $0.28 \times I_h$ from the bottom

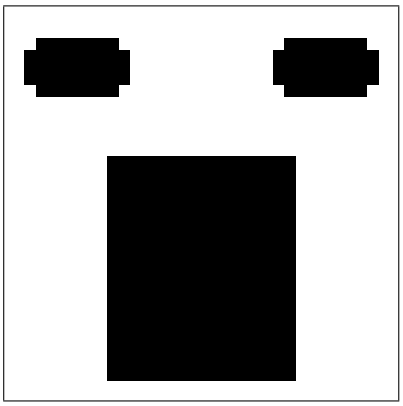


Figure 5.3: Template used for FND-based face detection. The dark areas correspond to the position of the eyes, nose and mouth. The nose and mouth are encompassed in one region. This template can be used to locate faces with slight in-plane rotations, as long as they fit within the template structure.

and $0.3 \times I_w$ from the right. The non-symmetrical search region is due to the assumption that the template used for correlation has a size of 32×32 pixels with the origin at the top left hand corner.

D. Further reduce the search region by removing from it the positions that begin with a blue pixel. We consider a pixel as blue if the blue minus the red component pixel value and the blue minus the green component pixel value are both greater than 40. A bounding box of the search region can be obtained.

E. The top edge of the search bounding box is set forward by a factor of 0.15 of the total height to move past the hair and forehead of the face. The final search region is shown in Figure 5.4. The following steps are performed for each position in the search region.

F. Extract the 32×32 pixel sub-image at the current search position and calculate the percentage of blue pixels. If this ratio is greater than 18% then skip to the next search position.

G. This step enforces the condition that there are no significant edges between the eyes and the nose. In the extracted 32×32 pixel sub-image, the gradient is calculated using the variance operator for the image strip from the 7th row

to the 10th row inclusively. The image strip is four pixels high by 32 pixels wide. If the sum of the gradients is greater than 10000 then skip to the next search position.

H. Symmetry is enforced in this step. An imaginary vertical line of reflection is created that divides the image into two equal halves. If a pixel $p_o(x, y)$ and its corresponding mirrored pixel $p_m(x, y)$ are non-zero we calculate

$$c_d(x, y) = |p_o(x, y) - p_m(x, y)| - \min(p_o(x, y), p_m(x, y)). \tag{5.5}$$

The extracted sub-image is divided into two halves: the eye region, and the nose plus mouth region. The nose and mouth region, beginning from the eleventh row onwards, follows the eye region. Two sums are calculated as a measure of the symmetry of the face,

$$r_e = \sum_{y=1}^{10} \sum_{x} c_d(x, y) \tag{5.6}$$

and

$$r_m = \sum_{y=11}^{32} \sum_{x} c_d(x, y). \tag{5.7}$$

If $r_e > -300$ or $r_m > -300$ then skip to the next search position, otherwise proceed to the next step.

I. The FND is used here to compare the extracted sub-image with the template. The template image is fractal encoded with a maximum sub-domain block usage of six, similar to the one described in Section 3.5.3.1. The upper bound for the contrast scaling factor α is set to 0.5 and the lower bound is set at zero. All search positions are attempted for the one that minimises $d_{FN}(\mathbf{x}_T, \mathbf{p}_s(x, y))$ using the Euclidean norm, where $\mathbf{x}_T$ is the template image, and $\mathbf{p}_s(x, y)$ is the extracted 32×32 sub-image at position (x, y).

This method performs reasonably well with the XM2VTS face database. However, it can only locate upright faces with accomodation for some slight in-plane head rotations by the design of the template. Making the eye, nose and mouth regions in the template larger may provide more invariance to rotations, but detection

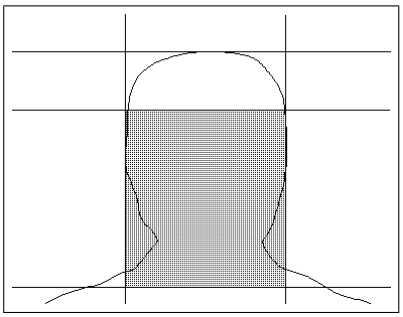


Figure 5.4: Approximate template search region for face finding. The bounding box for the head is found by colour thresholding in a restricted search space. The bounding box is further restricted by moving the top edge below the hair and some of the forehead region.

accuracy decreases. Furthermore, we cannot use it to normalise for in-plane head rotations. Depending on the size of the search region, which is image dependent, each face extraction requires 200 ms to 500 ms of processing time using a PC configured with an Athlon XP 2000+ processor, 1 Gigabyte of DDR-RAM and the software written in C++ compiled with Intel C++ 6.0 running on the Windows XP platform. This is the same PC configuration as the one described in Section 2.4.4. Examples of located faces are shown in Figure 5.5.

All the faces from the XM2VTS database are detected satisfactorily if we consider a detection as correct if most parts of the face are captured within the extracted rectangle as shown in Figure 5.5(b). Weaknesses in the method are:

- Only approximate face detection is possible. This is observable by comparing the eye positions of the first and second image on the second row of Figure 5.5(b). The first image had the eyes positioned closer to the top border than the second image;
- Extracted face sizes can vary quite significantly, which is observable by comparing the fourth image on the third row of Figure 5.5(b) to most of the other extracted faces.
- In-plane head rotations are not accounted for. For example, the third image

(a)

(b)

Figure 5.5: Face localisation using system VS-1: (a) original input images; (b) detected faces. Most of the faces from the XM2VTS database are detected satisfactorily, but there are variations in position, size and in-plane head rotation.

on the first row, second image on the second row and the first image on the fourth row of Figure 5.5(b) are not in an upright position.

5.3.1.2 Verification Subsystem for VS-FND

The extracted faces are 76 pixels high and 56 pixels wide. Each client face is extracted and fractal encoded using a uniform range block distribution with a maximum sub-domain block usage of six, similar to the one described in Section 3.5.3.1. The upper bound for the contrast scaling factor α is set to 0.5 and the lower bound is set at zero. Verification is performed by calculating the FND between the input image and those in the database with the claimed identity. A verification score between 0.0 and 1.0 is obtained by calculating

$$S_v = \max\left\{0.0, \min\left\{1.0, \max_{i,m,n}\left(1 - \frac{d_{FN}(\mathbf{x}_{ci}, \mathbf{x}^{(m,n)})}{100}\right)\right\}\right\} \tag{5.8}$$

where d_{FN} is the FND, $\mathbf{x}_{ci}$ is the ith face image for the claimed client identity c and (m, n) are the vertical and horizontal shifts similar to the ones described in Section 3.4.3. The vertical offset m ranges from -8 to 8 in steps of two pixels and the horizontal offset n ranges from -4 to 4 in steps of two pixels. The FND in Equation (5.8) is calculated in terms of the normalised version of the non-zero Euclidean norm as shown in Equation (5.9).

$$d_{FN}(\mathbf{p}, \mathbf{q}) = d_{nz}(f_{\mathbf{p}}(\mathbf{q}), \mathbf{q}) = \frac{d_z(f_{\mathbf{p}}(\mathbf{q}), \mathbf{q})}{\sqrt{N_z}} \tag{5.9}$$

where d_z is the non-zero Euclidean norm defined in Equation (3.19) and N_z is the number of pixel positions in $f_{\mathbf{p}}(\mathbf{q})$ and $\mathbf{q}$ where both are non-zero.

The higher the score S_v the better the match. The claimed identity c is accepted as true if $S_v > \tau_c$, where the acceptance threshold τ_c was found experimentally.

5.3.2 Verification System Using SVMs, FFTs and FNDs

This verification system (VS-WFND) combines a face localisation sub-system based on the eyes detection system described in Chapter 2 with the weighted FND. The

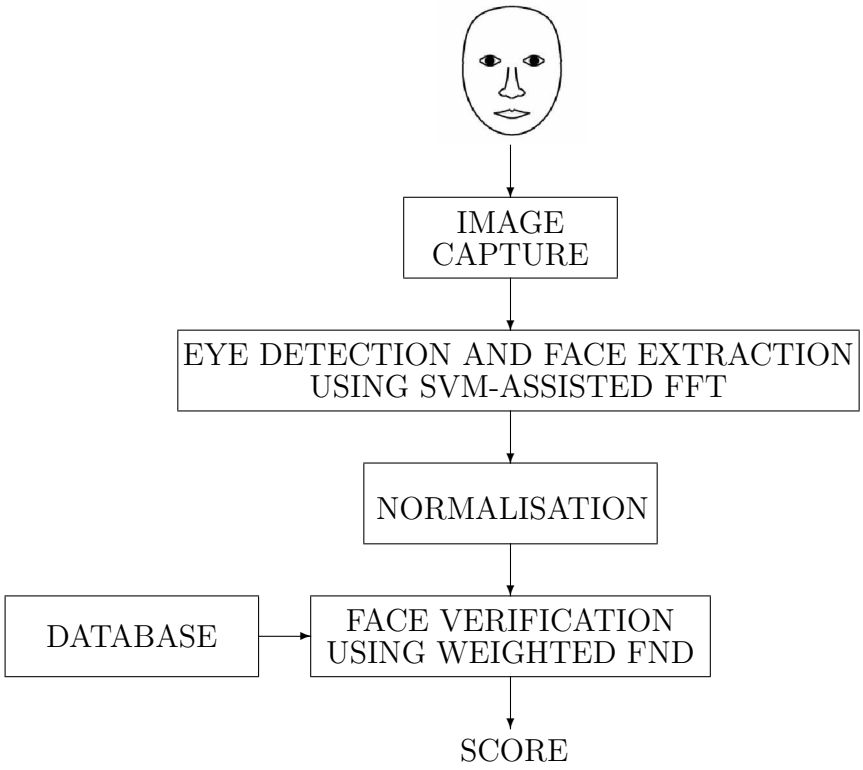


Figure 5.6: Face verification system VS-WFND. Eye detection uses the system described in Chapter 2. The face is extracted using the eye positions, where normalisations for size and in-plane rotations are also performed. The extracted face is 76 pixels high and 56 pixels wide.

weighted FND is used for image matching. The face extraction sub-system firstly detects the eyes using the FFT-assisted SVM eye detection system as described in Chapter 2. The eye locations are then used to extract a face that is normalised with respect to size and in-plane rotations. This alleviates some of the problems encountered by the face localisation sub-system in VS-FND. The block diagram for VS-WFND is shown in Figure 5.6.

5.3.2.1 Face Localisation Subsystem for VS-WFND

The algorithm for the VS-WFND face extraction sub-system is as follows:

A. Apply the low pass filter to the input image and find eye locations using the system described in Chapter 2.

B. Rotate the image around the left (or right) eye so that the left and right eyes lie on the horizontal. Simple pixel mapping can be used for finding the rotated image. However, more sophisticated algorithms employing pixel weighting are recommended because they provide higher quality rotations in terms of image smoothness and perceived quality. One example used in this system is the so-called Rotate-by-Shear. The C++ source code is available at http://www.codeproject.com/bitmap/rotatebyshear.asp. It performs rotation by skewing in the horizontal and vertical directions by the required amounts.

C. The rotated image is resized so that the horizontal eye to eye distance is 22 pixels.

D. Extract the face with a height of 76 and width of 56 pixels. With the image origin at the top-left corner, the left eye is at row 34 and column 17, and the right eye is at row 34 and column 39.

5.3.2.2 Verification Subsystem for VS-WFND

The extracted faces have the same size as those from VS-FND, that is 76 pixels high and 56 pixels wide. Verification is performed by calculating the weighted FND between the input image and those in the database with the claimed identity.

$$S_v = \max\left\{0.0, \min\left\{1.0, \max_i\left(1 - \frac{d_{wFN}(\mathbf{x}_{ci}, \mathbf{x})}{100}\right)\right\}\right\} \tag{5.10}$$

where d_{wFN} is the weighted FND, $\mathbf{x}_{ci}$ is the ith face image for the claimed client identity c. The higher the score S_v the better the match. The claimed identity c is accepted as true if $S_v > \tau_c$, where the acceptance threshold τ_c was found experimentally.

The knowledge gained from the experiment results described in Chapters 3 and 4 are used in selecting the parameters for the following experiments. The faces used here are more accurately located and normalised compared to the ones used in the

ORL database. As such we expect that better results are obtained by using smaller contractivity factors for the fractal codes. The explanation for this is given in Section 3.6, which basically says that when more normalisations are performed the difference between faces are dominated by shape and feature differences. Therefore, using smaller contractivity factors in this case should give better results. Fractal codes with smaller contractivity factors have faster convergence during the decoding process. All experiments use a domain to range block size ratio of two. The settings used in our experiments are divided into six categories:

Category 1 - Variation of α:

A. Uniform range block distribution, unrestricted sub-domain block usage, minimum $\alpha = 0$, maximum $\alpha = 0.4$, $\mathcal{H} = \{(-4, 0), (4, 0), (0, 0), (0, -4), (0, 4)\}$, $\mathcal{W} = \{(0.0, 0.5, 0.0, 0.0, 1.0, 0.0025)\}$, extracted face dimensions (76×56), left-eye center coordinates (34, 17), right-eye center coordinates (34, 39), eye detector R_b.

B. Uniform range block distribution, unrestricted sub-domain block usage, minimum $\alpha = 0$, maximum $\alpha = 0.5$, $\mathcal{H} = \{(-4, 0), (4, 0), (0, 0), (0, -4), (0, 4)\}$, $\mathcal{W} = \{(0.0, 0.5, 0.0, 0.0, 1.0, 0.0025)\}$, extracted face dimensions (76×56), left-eye center coordinates (34, 17), right-eye center coordinates (34, 39), eye detector R_b.

C. Uniform range block distribution, unrestricted sub-domain block usage, minimum $\alpha = 0$, maximum $\alpha = 0.6$, $\mathcal{H} = \{(-4, 0), (4, 0), (0, 0), (0, -4), (0, 4)\}$, $\mathcal{W} = \{(0.0, 0.5, 0.0, 0.0, 1.0, 0.0025)\}$, extracted face dimensions (76×56), left-eye center coordinates (34, 17), right-eye center coordinates (34, 39), eye detector R_b.

Category 2 - Maximum sub-domain block usage of six and variation of w_{h2}:

A. Uniform range block distribution, maximum sub-domain block usage of six (see Section 3.5.3.1), minimum $\alpha = 0$, maximum $\alpha = 0.5$, $\mathcal{H} =$

$\{(-4,0),(4,0),(0,0),(0,-4),(0,4)\}$, $\mathcal{W} = \{(0.0, 0.5, 0.0, 0.0, 1.0, 0.0025)\}$, extracted face dimensions (76×56), left-eye center coordinates (34, 17), right-eye center coordinates (34, 39), eye detector R_b.

B. Uniform range block distribution, maximum sub-domain block usage of six (see Section 3.5.3.1), minimum $\alpha = 0$, maximum $\alpha = 0.5$, $\mathcal{H} = \{(-4,0),(4,0),(0,0),(0,-4),(0,4)\}$, $\mathcal{W} = \{(0.0, 0.6, 0.0, 0.0, 1.0, 0.0025)\}$, extracted face dimensions (76×56), left-eye center coordinates (34, 17), right-eye center coordinates (34, 39), eye detector R_b.

Category 3 - Variation of $\mathcal{W}$

A. Uniform range block distribution, unrestricted sub-domain block usage, minimum $\alpha = 0$, maximum $\alpha = 0.5$, $\mathcal{H} = \{(-4,0),(4,0),(0,0),(0,-4),(0,4)\}$, $\mathcal{W} = \{(0.0, 0.7, 0.0, 0.0, 1.0, 0.0025)\}$, extracted face dimensions (76 × 56), left-eye center coordinates (34, 17), right-eye center coordinates (34, 39), eye detector R_b.

B. Uniform range block distribution, unrestricted sub-domain block usage, minimum $\alpha = 0$, maximum $\alpha = 0.5$, $\mathcal{H} = \{(-4,0),(4,0),(0,0),(0,-4),(0,4)\}$, $\mathcal{W} = \{(0.0, 0.6, 0.0, 0.0, 1.0, 0.0025)\}$, extracted face dimensions (76 × 56), left-eye center coordinates (34, 17), right-eye center coordinates (34, 39), eye detector R_b.

C. Uniform range block distribution, unrestricted sub-domain block usage, minimum $\alpha = 0$, maximum $\alpha = 0.5$, $\mathcal{H} = \{(-4,0),(4,0),(0,0),(0,-4),(0,4)\}$, $\mathcal{W} = \{(0.0, 0.4, 0.0, 0.0, 1.0, 0.0025)\}$, extracted face dimensions (76 × 56), left-eye center coordinates (34, 17), right-eye center coordinates (34, 39), eye detector R_b.

D. Uniform range block distribution, unrestricted sub-domain block usage, minimum $\alpha = 0$, maximum $\alpha = 0.5$, $\mathcal{H} = \{(-4,0),(4,0),(0,0),(0,-4),(0,4)\}$, $\mathcal{W} = \{(0.0, 0.3, 0.0, 0.0, 1.0, 0.0025)\}$, extracted face dimensions (76 × 56),

left-eye center coordinates (34, 17), right-eye center coordinates (34, 39), eye detector R_b.

E. Uniform range block distribution, unrestricted sub-domain block usage, minimum $\alpha = 0$, maximum $\alpha = 0.5$, $\mathcal{H} = \{(-4, 0), (4, 0), (0, 0), (0, -4), (0, 4)\}$, $\mathcal{W} = \{(0.1, 0.6, 0.0, 0.0, 1.0, 0.0025)\}$, extracted face dimensions (76×56), left-eye center coordinates (34, 17), right-eye center coordinates (34, 39), eye detector R_b.

F. Uniform range block distribution, unrestricted sub-domain block usage, minimum $\alpha = 0$, maximum $\alpha = 0.5$, $\mathcal{H} = \{(-4, 0), (4, 0), (0, 0), (0, -4), (0, 4)\}$, $\mathcal{W} = \{(0.05, 0.55, 0.0, 0.0, 1.0, 0.0025)\}$, extracted face dimensions (76×56), left-eye center coordinates (34, 17), right-eye center coordinates (34, 39), eye detector R_b.

G. Uniform range block distribution, unrestricted sub-domain block usage, minimum $\alpha = 0$, maximum $\alpha = 0.5$, $\mathcal{H} = \{(-4, 0), (4, 0), (0, 0), (0, -4), (0, 4)\}$, $\mathcal{W} = \{(0.0, 0.5, 0.1, 0.9, 1.0, 0.0025)\}$, extracted face dimensions (76×56), left-eye center coordinates (34, 17), right-eye center coordinates (34, 39), eye detector R_b.

H. Uniform range block distribution, unrestricted sub-domain block usage, minimum $\alpha = 0$, maximum $\alpha = 0.5$, $\mathcal{H} = \{(-4, 0), (4, 0), (0, 0), (0, -4), (0, 4)\}$, $\mathcal{W} = \{(0.0, 0.5, 0.0, 0.0, 1.0, 0.0064)\}$, extracted face dimensions (76×56), left-eye center coordinates (34, 17), right-eye center coordinates (34, 39), eye detector R_b.

I. Uniform range block distribution, unrestricted sub-domain block usage, minimum $\alpha = 0$, maximum $\alpha = 0.5$, $\mathcal{H} = \{(-4, 0), (4, 0), (0, 0), (0, -4), (0, 4)\}$, $\mathcal{W} = \{(0.0, 0.5, 0.0, 0.0, 1.0, 0.04)\}$, extracted face dimensions (76×56), left-eye center coordinates (34, 17), right-eye center coordinates (34, 39), eye detector R_b.

J. Uniform range block distribution, unrestricted sub-domain block usage, minimum $\alpha = 0$, maximum $\alpha = 0.5$, $\mathcal{H} = \{(-4, 0), (4, 0), (0, 0), (0, -4), (0, 4)\}$, $\mathcal{W} = \{(0.0, 0.5, 0.0, 0.0, 1.0, 0.25)\}$, extracted face dimensions (76×56), left-eye center coordinates (34, 17), right-eye center coordinates (34, 39), eye detector R_b.

K. Uniform range block distribution, unrestricted sub-domain block usage, minimum $\alpha = 0$, maximum $\alpha = 0.5$, $\mathcal{H} = \{(-4, 0), (4, 0), (0, 0), (0, -4), (0, 4)\}$, $\mathcal{W} = \{(0.0, 0.5, 0.0, 0.0, 1.0, 0.0001)\}$, extracted face dimensions (76×56), left-eye center coordinates (34, 17), right-eye center coordinates (34, 39), eye detector R_b.

L. Uniform range block distribution, unrestricted sub-domain block usage, minimum $\alpha = 0$, maximum $\alpha = 0.5$, $\mathcal{H} = \{(-4, 0), (4, 0), (0, 0), (0, -4), (0, 4)\}$, $\mathcal{W} = \{(0.0, 0.5, 0.0, 0.0, 1.0, 0.0)\}$, extracted face dimensions (76×56), left-eye center coordinates (34, 17), right-eye center coordinates (34, 39), eye detector R_b.

M. Uniform range block distribution, unrestricted sub-domain block usage, minimum $\alpha = 0$, maximum $\alpha = 0.5$, $\mathcal{H} = \{(-4, 0), (4, 0), (0, 0), (0, -4), (0, 4)\}$, $\mathcal{W} = \{(0.0, 0.5, 0.0, 0.0, 1.0, 1.0)\}$, extracted face dimensions (76×56), left-eye center coordinates (34, 17), right-eye center coordinates (34, 39), eye detector R_b.

Category 4 - Variation of normalised face size and position:

A. Uniform range block distribution, unrestricted sub-domain block usage, minimum $\alpha = 0$, maximum $\alpha = 0.5$, $\mathcal{H} = \{(-4, 0), (4, 0), (0, 0), (0, -4), (0, 4)\}$, $\mathcal{W} = \{(0.0, 0.5, 0.0, 0.0, 1.0, 0.0025)\}$, extracted face dimensions (56×44), left-eye center coordinates (14, 12), right-eye center coordinates (14, 32), eye detector R_b.

B. Uniform range block distribution, unrestricted sub-domain block usage, minimum $\alpha = 0$, maximum $\alpha = 0.5$, $\mathcal{H} = \{(-4, 0), (4, 0), (0, 0), (0, -4), (0, 4)\}$,

$\mathcal{W} = \{(0.0, 0.5, 0.0, 0.0, 1.0, 0.0025)\}$, extracted face dimensions (100×76), left-eye center coordinates (36, 24), right-eye center coordinates (36, 52), eye detector R_b.

Category 5 - Variation of eye detector used:

A. Uniform range block distribution, unrestricted sub-domain block usage, minimum $\alpha = 0$, maximum $\alpha = 0.5$, $\mathcal{H} = \{(-4, 0), (4, 0), (0, 0), (0, -4), (0, 4)\}$, $\mathcal{W} = \{(0.0, 0.5, 0.0, 0.0, 1.0, 0.0025)\}$, extracted face dimensions (76×56), left-eye center coordinates (34, 17), right-eye center coordinates (34, 39), eye detector L_b.

B. Uniform range block distribution, unrestricted sub-domain block usage, minimum $\alpha = 0$, maximum $\alpha = 0.5$, $\mathcal{H} = \{(-4, 0), (4, 0), (0, 0), (0, -4), (0, 4)\}$, $\mathcal{W} = \{(0.0, 0.5, 0.0, 0.0, 1.0, 0.0025)\}$, extracted face dimensions (76×56), left-eye center coordinates (34, 17), right-eye center coordinates (34, 39), eye detector P_b.

Category 6 - Non-uniform range block distributions:

A. Quad-tree range block distribution, unrestricted sub-domain block usage, maximum range block size (8×8), minimum range block size (2×2), minimum $\alpha = 0$, maximum $\alpha = 0.4$, $\mathcal{H} = \{(-4, 0), (4, 0), (0, 0), (0, -4), (0, 4)\}$, $\mathcal{W} = \{(0.0, 0.5, 0.0, 0.0, 1.0, 0.0025)\}$, extracted face dimensions (76×56), left-eye center coordinates (34, 17), right-eye center coordinates (34, 39), eye detector R_b.

B. Quad-tree range block distribution, unrestricted sub-domain block usage, maximum range block size (8×8), minimum range block size (2×2), minimum $\alpha = 0$, maximum $\alpha = 0.5$, $\mathcal{H} = \{(-4, 0), (4, 0), (0, 0), (0, -4), (0, 4)\}$, $\mathcal{W} = \{(0.0, 0.5, 0.0, 0.0, 1.0, 0.0025)\}$, extracted face dimensions (76×56), left-eye center coordinates (34, 17), right-eye center coordinates (34, 39), eye detector R_b.

C. Quad-tree range block distribution, unrestricted sub-domain block usage, maximum range block size (8×8), minimum range block size (2×2), minimum $\alpha = 0$, maximum $\alpha = 0.6$, $\mathcal{H} = \{(-4, 0), (4, 0), (0, 0), (0, -4), (0, 4)\}$, $\mathcal{W} = \{(0.0, 0.5, 0.0, 0.0, 1.0, 0.0025)\}$, extracted face dimensions (76×56), left-eye center coordinates (34, 17), right-eye center coordinates (34, 39), eye detector R_b.

D. Quad-tree range block distribution, unrestricted sub-domain block usage, maximum range block size (8×8), minimum range block size (2×2), minimum $\alpha = 0$, maximum $\alpha = 0.95$, $\mathcal{H} = \{(-4, 0), (4, 0), (0, 0), (0, -4), (0, 4)\}$, $\mathcal{W} = \{(0.0, 0.5, 0.0, 0.0, 1.0, 0.0025)\}$, extracted face dimensions (76×56), left-eye center coordinates (34, 17), right-eye center coordinates (34, 39), eye detector R_b.

E. HV-partitioned range block distribution, unrestricted sub-domain block usage, maximum range block size (8×8), minimum range block size (2×2), minimum $\alpha = 0$, maximum $\alpha = 0.5$, $\mathcal{H} = \{(-4, 0), (4, 0), (0, 0), (0, -4), (0, 4)\}$, $\mathcal{W} = \{(0.0, 0.5, 0.0, 0.0, 1.0, 0.0025)\}$, extracted face dimensions (76×56), left-eye center coordinates (34, 17), right-eye center coordinates (34, 39), eye detector R_b.

In addition, experiments are also performed using manually located eye positions. This is used in place of the FFT-assisted SVM eye detector. The extracted faces are then verified using the weighted FND. This tests the performance of the weighted FND independently of the face localisation system.

5.3.3 Performance Comparison

The XM2VTS face database [MMK+99] is used to evaluate the performances of VS-FND and VS-WFND. The database consists of 2360 images of a single frontal pose shot of faces from 295 people. There are eight face samples of each person, obtained from four sessions spread out in monthly intervals. Two shots are taken

in each session. The Lausanne Protocol [LM98] describes a testing procedure using this database for evaluating face verification systems. The subjects are randomly divided into three categories, where the number of subjects in each category is shown inside parentheses:

- Clients (200).
- Impostors for use during system evaluation (25).
- Impostors for use during system testing (70).

A weakness with many testing protocols is that the results are obtained using decision thresholds that have been adjusted to the test samples so that the best results are obtained. In a real-world situation the results are obtained using decision thresholds that have been set a priori. The Lausanne Protocol attempts to simulate real-world behaviour by dividing the sample set into training, evaluation and testing sets. The training set is used to train the verification algorithms. The evaluation set is used for setting the decision threshold to satisfy certain performance levels expressed in terms of error rates. This decision threshold is fixed and used to obtain results using the testing set.

The subjects in the Clients category are further subdivided into the training, evaluation and testing categories. The Lausanne protocol specifies two different subdivisions called Configuration I and Configuration II, shown in Tables 5.1 and 5.2 respectively. Altogether, for Configuration I, there are $200 \times 3 = 600$ client training images, $200 \times 3 = 600$ client evaluation images, $200 \times 2 = 400$ client test images, $25 \times 8 = 200$ evaluation images of impostors and $70 \times 8 = 560$ test images of impostors. For Configuration II, there are $200 \times 4 = 800$ client training images, $200 \times 2 = 400$ client evaluation images, $200 \times 2 = 400$ client test images, $25 \times 8 = 200$ evaluation images of impostors and $70 \times 8 = 560$ test images of impostors.

The results for Configuration I obtained using VS-FND are shown in Tables 5.3 and 5.4. Configuration II results using VS-FND are shown in Tables 5.5 and 5.6.

Training Data	Session 1, Shot 1
	Session 2, Shot 1
	Session 3, Shot 1
Evaluation Data	Session 1, Shot 2
	Session 2, Shot 2
	Session 3, Shot 2
Test Data	Session 4, Shot 1
	Session 4, Shot 2

Table 5.1: Division of clients into training, evaluation and testing data for Configuration I.

Training Data	Session 1, Shot 1
	Session 1, Shot 2
	Session 2, Shot 1
	Session 2, Shot 2
Evaluation Data	Session 3, Shot 1
	Session 3, Shot 2
Test Data	Session 4, Shot 1
	Session 4, Shot 2

Table 5.2: Division of clients into training, evaluation and testing data for Configuration II.

In both configurations, decision thresholds are set using results from the evaluation sets, which are then used for testing on the test sets. Three threshold settings are examined, corresponding to the conditions $FAR = FRR$, $FRR(FAR = 0)$ and $FAR(FRR = 0)$ on the evaluation set. The false accept rate (FAR) is the rate at which an impostor is accepted as a valid client, and the false reject rate (FRR) is the rate at which a valid client is rejected as an impostor. The equal error rate is denoted by $FAR = FRR$, $FRR(FAR = 0)$ is the false reject error rate when the threshold is set such that the false accept error rate is zero, and $FAR(FRR = 0)$ is the false accept error rate when the threshold is set such that the false reject error rate is zero. The results obtained using the test sets show both FAR and FRR error rates for each threshold setting, because when the FAR/FRR is equal to zero in the evaluation case it is not necessarily zero in the test case. That is, for the test sets we report $FAR(FAR = FRR)$, $FRR(FAR = FRR)$, $FAR(FAR = 0)$, $FRR(FAR = 0)$, $FAR(FRR = 0)$ and $FRR(FRR = 0)$. Note that the condition in parentheses, such as $(FAR = FRR)$ and $(FAR = 0)$, *always* refer to *evaluation* set results. Therefore, for example, the test set result $FAR(FAR = FRR)$ refers to the test FAR result when the threshold is set to the one that gives $FAR = FRR$ on the evaluation set. One commonly used measure of performance is the Half Total Error Rate (HTER) [BMM01], which is defined as

$$HTER = \frac{FAR + FRR}{2}. \tag{5.11}$$

In the case when the evaluation set is used, and the threshold can be set for the condition $FAR = FRR$, then $HTER = FAR = FRR$.

In evaluating the VS-WFND, the results from using only one weighting function is shown. It is found from preliminary experiments that using one weighting function gives better results for the verification system. We conduct experiments using more than one weighting function, each located at different positions, but the results are inferior to the ones quoted here. We also try using the weighting functions specified in the next section describing the identification system, and again with similar inferior results.

Due to the amount of computation required for each experiment we could only afford to perform all of them for the evaluation set of Configuration I. The evaluation set is used to select the best settings, which are then used on the testing set. These settings are then used for Configuration II experiments. In this case, the decision threshold is set based on Configuration II's evaluation set results.

The results reveal that applying a weighting factor that emphasises the top part of the face and de-emphasises the bottom part improves the recognition rate. The top part of the face encompassing the eyes and some of the nose region tend to be more invariant than the bottom part of the face, which is more sensitive to facial expressions and facial hair. Tables 5.7 to 5.12 show the evaluation results for VS-WFND using settings specified in categories 1 to 6. From the evaluation results in Tables 5.7 to 5.12 we choose the setting with the best performance and use it in further experiments. Using the error rate when $FAR = FRR$ as the performance indicator, category 3 setting C gives the best performance with $FAR = FRR = 9.4\%$. However, we choose to use category 1 setting B, which has the next best $FAR = FRR$ of 9.9%, for its lower $FRR(FAR = 0)$. Our selected system setting, category 1 setting B, gives results of $FAR = FRR = 9.9\%$, $FAR(FRR = 0) = 88.6\%$ and $FRR(FAR = 0) = 62.6\%$ on the evaluation set of Configuration I. This setting is used on the testing set of Configuration I and the results are shown in Table 5.13, which shows an $HTER$ of 11.8%. The same setting, except the decision thresholds, is used on the evaluation set of Configuration II. The results are shown in Table 5.14, which reveals an equal error rate of $FAR = FRR = 12.1\%$, $FAR(FRR = 0) = 90.8\%$ and $FRR(FAR = 0) = 83.5\%$. The thresholds are determined on this evaluation set for the three conditions described, which are then used on the test set. The results obtained using this test set are shown in Table 5.15. In this case, the FAR at the threshold where $FAR = FRR$ for the evaluation set is 11.4%, and the FRR error rate is 16.5%, giving an $HTER$ of 13.95%.

In Figures 5.7 to 5.10, the FAR and FRR error rates for category 1 setting B are plotted against the decision threshold using the evaluation and test results from Configuration I and II. It shows the relationship of the FAR and FRR error rates to

the decision threshold. The equal error rate is the point at which the FAR and FRR curves meet. The threshold where the equal error rate occurs for the evaluation set is indicated by an arrow labelled '1'. The arrow labelled '2' is the threshold where the FRR error rate is zero for the evaluation set. Similarly, the arrow labelled '3' is the threshold where the FAR error rate is zero for the evaluation set. For comparison, the same arrows are drawn at the same locations for the test sets. Ideally, the points where the equal error rate, zero FAR and zero FRR occur in the set correspond exactly to those of the evaluation set. Except for point '2', points '1' and '3' in Figure 5.7 correspond closely to those in Figure 5.8. There is greater variance between the evaluation and test sets of Configuration II, as evident in the mismatch of the three points shown in Figures 5.9 and 5.10.

We also perform experiments using manually located eye positions, bypassing the automatic eye detection system. The eye positions are then used to normalise the face. The aim is to test the FND based recognition engine with accurately normalised faces. The following setting is used:

- Uniform range block distribution, unrestricted sub-domain block usage, minimum $\alpha = 0$, maximum $\alpha = 0.5$, $\mathcal{H} = \{(0,0)\}$, $\mathcal{W} = \{(0.0, 0.5, 0.0, 0.0, 1.0, 0.0025)\}$, extracted face dimensions (76×56), left-eye center coordinates (34, 17), right-eye center coordinates (34, 39).

This setting is equivalent to category 1 setting B, except that the automatic eye detector R_b is replaced with manually located eye positions and no shifting is carried out, so that $\mathcal{H}$ consists only of the element $(0,0)$. Shifting is not required because the faces are already correctly aligned using the manually located eye positions. In fact, preliminary experiments show that in some cases using shifting with faces aligned using manually located eye positions produces inferior results. The error rates at $FAR = FRR$, $FAR(FRR = 0)$ and $FRR(FAR = 0)$ for configuration I are shown in Tables 5.16 and 5.17. Table 5.16 shows the evaluation results, revealing an equal error rate of $FAR = FRR = 5.0\%$, $FAR(FRR = 0) = 62.0\%$ and $FRR(FAR = 0) = 35.0\%$. The same decision thresholds are used for the test

experiment and the results are summarised in Table 5.17. In this case, the FAR error rate at the threshold where $FAR = FRR$ for the evaluation set is 5.0%, and the FRR error rate 12.8%, giving an $HTER$ of 8.9% for this test set. When we compare the FRR and FAR versus the threshold curves in Figures 5.11 and 5.12, we see that the FRR curve in the test result has a larger spread towards the lower thresholds compared to those from the evaluation experiment. The reason for this is that the client test data is obtained from session four, which is a completely different session to those used for training and evaluation. The training and evaluation data are obtained from the same sessions, differing only in the shot number. Therefore, we can expect that the thresholds set using the evaluation set will not be optimal for the client test data. This problem is alleviated by normalising the output scores, which we describe in Section 5.3.3.1.

The error rates at $FAR = FRR$, $FAR(FRR = 0)$ and $FRR(FAR = 0)$ for configuration II are shown in Tables 5.18 and 5.19. Table 5.18 shows the evaluation results, revealing an equal error rate of $FAR = FRR = 9.1\%$, $FAR(FRR = 0) = 88.0\%$, and $FRR(FAR = 0) = 68.8\%$. The same decision thresholds are used for the test experiment and the results are summarised in Table 5.19. In this case, the FAR error rate at the threshold where $FAR = FRR$ for the evaluation set is 8.6%, and the FRR error rate is 10.0%, giving an $HTER$ of 9.3% for this test set. When we compare the FRR and FAR versus the threshold curves in Figures 5.13 and 5.14, we see that, with the exception of point '2', points '1' and '3' correspond roughly to the expected locations.

As evident in the results, the thresholds set using the evaluation set may not be optimal for the test set. Ideally, the system should be set up with thresholds that are also optimal on the test set. However, it is natural for test sets to be different, in one form or another, to evaluation sets. Therefore, less than optimal performance can be expected on the test sets. This reflects the real-life situation where non-adaptive thresholds have to be preset for any operation. Again, this problem can be alleviated by normalising the output scores, which is described in Section 5.3.3.1.

Graphs showing the FRR versus FAR error rates are shown in Figures 5.15 and 5.16. Figure 5.15 relate to Configuration I results, and Figure 5.16 relate to Configuration II results. Both graphs show the test and evaluation results for systems using category 1 setting B and manually located eye positions. The graphs illustrate the performance improvement gained from using manually located eye positions. The equal error rates can be obtained from the graphs by locating the points of intersection with the straight line having a slope of 1 through the origin.

We now compare our results to the ones in the literature that also use the XM2VTS database with the Lausanne Protocol testing procedures. The results from the literature are summarised in Tables 5.20 to 5.23. Table 5.20 shows the Configuration I evaluation results, table 5.22 shows the Configuration I test results. Configuration II evaluation results are shown in Table 5.21, and the test results are shown in Table 5.23. The results are extracted from Matas et al. [MHJ+00]. The labels in the table are:

- AUT - Aristotle University of Thessaloniki [KTP00]. Elastic graph matching is used. Features are extracted based on multiscale dilation and erosion operations computed at each node of the grid. The amount of grid deformation and feature differences are computed and combined into a verification score.
- IDIAP - Dalle Molle Institute for Perceptual Artificial Intelligence [BYAM99]. Elastic graph matching is used. Gabor responses are calculated at each node using six orientations and three resolutions.
- UniS-A-G-NC - University of Surrey, full automatic registration, global threshold, normalised correlation.
- UniS-A-S-NC - University of Surrey, full automatic registration, client-specific threshold, normalised correlation.
- UniS-S-G-NC - University of Surrey, semi-automatic registration, global threshold, normalised correlation.

- UniS-S-S-NC - University of Surrey, semi-automatic registration, client-specific threshold, normalised correlation.
- UniS-A-G-SM - University of Surrey, full automatic registration, global threshold, special metric.
- UniS-A-S-SM - University of Surrey, full automatic registration, client-specific threshold, special metric.
- UniS-S-G-SM - University of Surrey, semi-automatic registration, global threshold, special metric.
- UniS-S-S-SM - University of Surrey, semi-automatic registration, client-specific threshold, special metric.
- UniS-SVM - University of Surrey, full automatic registration, support vector machine.

The methods proposed by the University of Surrey involve the use of robust correlation and also support vector machines [JKLM99, MHJ+00]. Havran et al. [HHC+02] also reports results using the XM2VTS database. However, it is not clear which configuration is used or if they followed the Lausanne Protocol. In any case, they report an equal error rate of 5.62 % on the training set, and an $HTER$ of 5.21% on the evaluation set.

In terms of $HTER$, our Configuration I results are comparable to the ones in the literature, where we achieve an $HTER$ of 8.9% and the results obtained by others range from 2.4% to 9.3% as shown in Table 5.22. However, Configuration II results show that our method generally has larger error rates. If we examine the training, evaluation and test images in Configuration II, we notice that the evaluation and test images are taken from different sessions. The sessions are separated by monthly intervals. At each session there are variations in hair style, facial hair, eye glasses and pose. We attribute most of the errors to pose variations, which are mainly in terms of out-of-plane rotations. There are also some slight illumination differences,

but the majority of variation is from pose. In-plane rotations are accounted for by normalising the face using the eye locations. Out-of-plane rotations are not accounted for. We discover that the standard and weighted FND can be sensitive to such distortions. This problem is not revealed in earlier experiments using the ORL database because of the limited size of the database. This problem is revealed and investigated further in the following sections.

5.3.3.1 Normalised Scores

The previous section reveals an asymmetry in the evaluation and test set FAR and FRR results. This is reflected in the inferior threshold setting for the test set, and can be observed in Figure 5.12, where the arrow labelled "1" is not located at the ideal threshold where FAR is equal to FRR. Here we describe a method of normalising the output scores that alleviates this problem, and improves the $HTER$ in the process. Our aim is to normalise the output score S_v defined in Equation (5.10).

Firstly, in the database of client training images, we calculate the average image for each client. That is,

$$\mathbf{x}_{c,avg} = \frac{1}{n_c} \sum_{j}^{n_c} \mathbf{x}_{cj} \tag{5.12}$$

where $\mathbf{x}_{cj}$ is the j-th training image for client c and n_c is the number of training images for client c. Let l denote the claimed identity during a verification run, and S_v^l denote the output score for the unknown input face $\mathbf{x}$ claiming to be l. The normalised form of the output score is then given by

$$S_n^l = a_n \frac{S_v^l}{\|\mathbf{x} - \mathbf{x}_{l,avg}\|} \tag{5.13}$$

where a_n is a scaling factor to adjust the normalised score to the range $[0, 1]$. In our experiments we use $a_n = 1000$.

Configuration I evaluation results are shown in Table 5.24, and the corresponding test results are shown in Table 5.25. Configuration II evaluation results are shown in Table 5.26, and the corresponding test results are shown in Table 5.27.

The Configuration I evaluation error rate when $FAR = FRR$ is 7.6%. This is slightly inferior to the unnormalised scores method, which achieves an error rate of 5.0% as shown in Table 5.16. However, the superiority of the normalised scores method lies in the consistency of results. This is evident when we examine the Configuration I test results, where an $HTER$ of 7.7% is achieved. This result is competitive with the ones from the literature where the test set $HTER$ range from 2.4% to 9.3% as shown in Table 5.22. At 7.7% the result here is better than the 8.3% reported by IDIAP, which use elastic graph matching.

The Configuration II evaluation error rate when $FAR = FRR$ is 8.3%, and the corresponding $HTER$ obtained on the test set is 8.2%. Again, consistency between the evaluation and test sets is demonstrated. The results here are better than the ones that use unnormalised scores, where the evaluation error rate at $FAR = FRR$ is 9.1% and the test set $HTER$ is 9.3% as shown in Tables 5.18 and 5.19. However, for reasons described in the previous section, the results here are still slightly inferior to those in the literature, which report $HTER$s in the range from 1.1% to 7.5% as shown in Table 5.23. Our test set $HTER$ result of 8.2% is only competitive with the result from IDIAP, which report an $HTER$ of 7.5%.

The advantage of using normalised scores over the unnormalised version can be further demonstrated by observing the plots of FRR versus FAR. Configuration I results are shown in Figure 5.21 and Configuration II results are shown in Figure 5.22. The graphs also include a line with slope one emanating from the origin. The points of intersection between the curves and this line are the points where $FAR = FRR$. The graphs show that the evaluation and test curves obtained using normalised scores are in closer correlation than those that use unnormalised scores. This property is more distinct in Configuration I. In both configurations the test results obtained using normalised scores are better than those obtained using the unnormalised version.

We also experiment with other definitions of S_n^l. In particular, we also use a normalisation scheme where instead of using the average client training image, the scores

are normalised with each corresponding client training image. However, inferior results are obtained with this method.

5.3.3.2 Client Specific Thresholding

So far, the methods in the previous sections have used a global decision threshold only. That is, the same threshold is applied to all clients. Now we examine the performance of the system when we allow thresholds to be set independently for each client. We only consider the case where the thresholds are set to achieve the condition $FAR = FRR$.

We use a threshold selection scheme similar to that described in Section 2.2.3. Here the thresholds are optimised on the *evaluation* set for the condition $FAR = FRR$ in an incremental fashion. The basic idea behind the method is that finding the threshold where $FAR = FRR$ is equivalent to minimising $|FAR - FRR|$ with respect to the threshold. The algorithm for determining the client specific thresholds in pseudo-code is:

```
%%%%% For client specific threshold determination on evaluation set %%%%%
TOTAL_NUM_CLIENTS = 600
TOTAL_NUM_IMPOSTORS = 40000

%%%%% For client specific threshold determination on testing set %%%%%
TOTAL_NUM_CLIENTS = 400
TOTAL_NUM_IMPOSTORS = 112000

%%%%% CONSTANTS %%%%%
MAX_DISCRETE = 2000000

%%%%% VARIABLES %%%%%
score = 0
numPositives = 0
numNegatives = 0
clientScores[1..MAX_DISCRETE] = 0
impostorScores[1..MAX_DISCRETE] = 0
positiveCumulative[1..MAX_DISCRETE] = 0
negativeCumulative[1..MAX_DISCRETE] = 0
normalisedPositiveCumulative[1..MAX_DISCRETE] = 0
```

```
21  normalisedNegativeCumulative[1..MAX_DISCRETE] = 0
22  distributionDiff[1..MAX_DISCRETE] = 0
23  bestCount = 0
24  bestIndex = 0
25  numBest = 0
26  bestThreshold = 0
27
28  %%%%% MAIN LOOP %%%%%
29  for each claimed client identity C
30    for each face E in client evaluation/test set
31      score = verify face E
32      discretise score to range [1..MAX_DISCRETE]
33      clientScores[score] = clientScores[score] + 1
34    for each face I in impostor evaluation/test set
35      score = verify face I
36      discretise score to range [1..MAX_DISCRETE]
37      impostorScores[score] = impostorScores[score] + 1
38    positiveCumulative[1] = clientScores[1]
39    for i = 2 to MAX_DISCRETE step 1
40      positiveCumulative[i] = clientScores[i] + clientScores[i-1]
41    negativeCumulative[MAX_DISCRETE] = impostorScores[MAX_DISCRETE]
42    for i = MAX_DISCRETE-1 to 1 step -1
43      negativeCumulative[i] = impostorScores[i] + impostorScores[i+1]
44    for i = 1 to MAX_DISCRETE step 1
45      normalisedPositiveCumulative[i] = (positiveCumulative[i] + numPositives)/TOTAL_NUM_CLIENTS
46      normalisedNegativeCumulative[i] = (negativeCumulative[i] + numNegatives)/TOTAL_NUM_IMPOSTORS
47      distributionDiff[i] = |normalisedNegativeCumulative[i] - normalisedPositiveCumulative[i]|
48
49    bestCount = distributionDiff[0]
50    bestIndex = 0
51    numBest = 1
52    for i = 2 to MAX_DISCRETE step 1
53      if distributionDiff[i] < bestCount
54       bestCount = distributionDiff[i]
55       bestIndex = i
56       numBest = 1
57      else if distributionDiff[i] == bestCount
58       bestIndex = bestIndex + i
59       numBest = numBest + 1
60    bestIndex = bestIndex/numBest
61    numPositives = numPositives + positiveCumulative[bestIndex]
62    numNegatives = numNegatives + negativeCumulative[bestIndex]
63    bestThreshold = map bestIndex from discrete to actual threshold value
```

This pseudo-code is operable on evaluation and test set data. When evaluation data

is used, lines 6 and 7 must be commented out, and when test data is used, lines 2 and 3 must be commented out. The use of test data is only for evaluation purposes. In practice, it cannot be used for setting thresholds because test data should only be used for simulating run-time data, as described in Section 5.3.3.

Variables are initialised in the top part of the pseudo-code. The main loop starts from line 28. Lines 30 to 37 of the pseudo-code performs verification on all client and impostor evaluation/test data. Verification scores are discretised and normalised to the range $[0, \text{MAX\_DISCRETE}]$, which are then used to create a histogram of scores for clients and impostors. The histogram of client scores are used to create a cumulative distribution that is an accumulation of histogram values from lower scores to higher scores. The histogram of impostor scores are used to create a cumulative distribution that is an accumulation of histogram values from higher scores to lower scores. The cumulative distributions are then normalised in lines 45 and 46, where the normalised values are stored in `normalisedPositiveCumulative[i]` and `normalisedNegativeCumulative[i]`. The value at `normalisedPositiveCumulative[i]` represents the current FRR at the threshold represented by index `i`. Similarly, `normalisedNegativeCumulative[i]` represents the current FAR at the threshold represented by index `i`. The threshold where $FAR = FRR$ can then be determined by finding the index that minimises $|$`normalisedPositiveCumulative[i]` $-$ `normalisedNegativeCumulative[i]`$|$. This index is then converted to the corresponding threshold value. Therefore, in each loop starting at line 29, we find the threshold that minimises $|FAR - FRR|$. Also, in cases where the maximum difference is a plateau, meaning that it occurs for a series of threshold values, the average threshold within this set is taken. This is done in lines 49 to 60 of the pseudo-code. The variables `numPositives` and `numNegatives` are updated in lines 61 and 62. `numPositives` keeps track of the current number of samples that have been falsely rejected, and `numNegatives` keeps track of the current number of samples that have been falsely accepted.

The experiment results using this method of client specific thresholding are summarised in Tables 5.28 and 5.29. Results using normalised scores are also shown. In both configurations, the evaluation $HTER$s obtained from using unnormalised scores are better than those obtained using normalised scores. However, the test results using a thresholds set using evaluation data show that using normalised scores give better $HTER$s compared to using unnormalised scores. The evaluation results show that the method is very competitive with those in the literature, where our unnormalised scores variant achieved an $HTER$ of 2.2% and 3.2% on Configuration I and II respectively. For Configuration I, this is in fact better than all the methods quoted from the literature listed in Table 5.20. For Configuration II, our result of 3.2% is better than half of the methods in the literature listed in Table 5.21. In all cases, our method performs significantly better than the elastic graph matching based methods of AUT and IDIAP listed in those tables. AUT reports rates of 8.1% and 6.5%, and IDIAP reports rates of 8.0% and 7.0%, for the evaluation set of Configuration I and II respectively. For the evaluation set of Configuration I, our method performs better than those from the University of Surrey, which also include methods that use client specific thresholding. Their best result is 2.5%, compared to our result of 2.2%. However, our method is slightly inferior on the evaluation set of Configuration II, where they achieve an error rate of 1.3% compared to our result of 3.2%.

With client specific thresholding our method is very competitive with methods in the literature when using the evaluation sets of either configurations. However, our results obtained using the test sets are inferior. Using Configuration I and thresholds set using evaluation data, our method achieves an $HTER$ of 14.7% using unnormalised scores, and 9.8% using normalised scores. Using Configuration II, our method achieves an $HTER$ of 13.4% using unnormalised scores, and 11.5% using normalised scores. Results in the literature listed in Table 5.22 for Configuration I show results that range from 2.4% to 9.3%. Configuration II results in the literature, listed in Table 5.23, range from 1.1% to 7.5%. Nevertheless, our results show that

when using the test sets, normalised scores give better results compared to un-normalised scores.

For both configurations and using normalised and un-normalised scores, the majority of the errors in our test runs are FRR errors. Our FAR results are actually quite competitive. Again, the problem of asymmetry in the results is evident. To investigate the optimal result achievable using our method on the test sets, we apply our client specific threshold calculation algorithm to the test data. The results are also shown in Tables 5.28 and 5.29. In this case, for Configuration I, our method based on un-normalised scores achieves an $HTER$ of 2.8%, and 3.8% when normalised scores are used. For Configuration II, the un-normalised scores variant achieves an $HTER$ of 4.0%, and the normalised scores variant achieves an $HTER$ of 3.5%. These results are again competitive with the results in the literature. What this demonstrates is that the Weighted FND method does in fact achieve good class separation. This is reflected in the fact that it is actually possible to achieve small $HTER$s on the test set. However, in practice, the problem of asymmetry between the evaluation and test results remains to be solved. This problem is likely to be alleviated by using other methods of score normalisation, or normalisations in the steps prior to score generation. The discovery of an appropriate solution is left for future work.

We note that better performance can be expected if we have more client evaluation samples. In comparison, there are plenty of impostor evaluation samples. The Lausanne Protocol only specifies three client evaluation samples per client for Configuration I, and two per client for Configuration II. This is not enough to model the distribution of client evaluation samples effectively.

FAR=FRR	FAR (FRR = 0)	FRR (FAR = 0)
12.9 %	94.4 %	70.5 %

Table 5.3: Error rates obtained using the evaluation set of Configuration I and VS-FND.

FAR (FAR=FRR)	FRR (FAR=FRR)	HTER (FAR=FRR)	FAR (FRR = 0)	FRR (FRR = 0)	FAR (FAR = 0)	FRR (FAR = 0)
13.6 %	12.3 %	12.95 %	94.0 %	0.0 %	0.0 %	81.3 %

Table 5.4: Error rates obtained using the test set of Configuration I and VS-FND.

FAR=FRR	FAR (FRR = 0)	FRR (FAR = 0)
14.1 %	98.4 %	80.8 %

Table 5.5: Error rates obtained using the evaluation set of Configuration II and VS-FND.

FAR (FAR=FRR)	FRR (FAR=FRR)	HTER (FAR=FRR)	FAR (FRR = 0)	FRR (FRR = 0)	FAR (FAR = 0)	FRR (FAR = 0)
13.0 %	12.3 %	12.65 %	98.1 %	0.0 %	0.0 %	84.8 %

Table 5.6: Error rates obtained using the test set of Configuration II and VS-FND.

Category 1 Experiment	FAR=FRR	FAR (FRR = 0)	FRR (FAR = 0)
A	10.1 %	94.4 %	71.0 %
B	9.9 %	88.6 %	62.6 %
C	9.9 %	94.3 %	62.3 %

Table 5.7: Error rates for evaluation experiments using Configuration I, VS-WFND and category 1.

Category 2 Experiment	FAR=FRR	FAR (FRR = 0)	FRR (FAR = 0)
A	10.2 %	90.1 %	60.0 %
B	10.0 %	95.8 %	62.0 %

Table 5.8: Error rates for evaluation experiments using Configuration I, VS-WFND and category 2.

Category 3 Experiment	FAR=FRR	FAR (FRR = 0)	FRR (FAR = 0)
A	10.2 %	95.0 %	66.2 %
B	9.7 %	92.4 %	61.8 %
C	9.4 %	87.8 %	65.5 %
D	9.8 %	91.1 %	92.8 %
E	11.6 %	94.5 %	67.7 %
F	10.1 %	91.5 %	65.0 %
G	10.1 %	91.4 %	71.7 %
H	9.9 %	88.9 %	63.0 %
I	9.7 %	89.3 %	63.5 %
J	10.2 %	92.4 %	65.5 %
K	10.1 %	88.9 %	63.0 %
L	10.0 %	88.9 %	63.0 %
M	12.2 %	98.9 %	74.4 %

Table 5.9: Error rates for evaluation experiments using Configuration I, VS-WFND and category 3.

Category 4 Experiment	FAR=FRR	FAR (FRR = 0)	FRR (FAR = 0)
A	17.9 %	99.6 %	84.9 %
B	10.7 %	94.9 %	65.3 %

Table 5.10: Error rates for evaluation experiments using Configuration I, VS-WFND and category 4.

Category 5 Experiment	FAR=FRR	FAR (FRR = 0)	FRR (FAR = 0)
A	10.0 %	89.9 %	62.2 %
B	9.5 %	69.7 %	92.8 %

Table 5.11: Error rates for evaluation experiments using Configuration I, VS-WFND and category 5.

Category 6 Experiment	FAR=FRR	FAR (FRR = 0)	FRR (FAR = 0)
A	9.8 %	93.8 %	66.7 %
B	9.7 %	94.1 %	63.9 %
C	9.7 %	93.4 %	66.0 %
D	9.5 %	91.1 %	65.3 %
E	10.2 %	93.9 %	70.9 %

Table 5.12: Error rates for evaluation experiments using Configuration I, VS-WFND and category 6.

FAR (FAR=FRR)	FRR (FAR=FRR)	HTER (FAR=FRR)	FAR (FRR = 0)	FRR (FRR = 0)	FAR (FAR = 0)	FRR (FAR = 0)
12.1 %	11.5 %	11.8 %	87.0 %	0.0 %	0.0 %	71.4 %

Table 5.13: Error rates obtained using the test set of Configuration I, VS-WFND and category 1 setting B.

FAR=FRR	FAR (FRR = 0)	FRR (FAR = 0)
12.1 %	90.8 %	83.5 %

Table 5.14: Error rates obtained using the evaluation set of Configuration II, VS-WFND and category 1 setting B.

FAR (FAR=FRR)	FRR (FAR=FRR)	HTER (FAR=FRR)	FAR (FRR = 0)	FRR (FRR = 0)	FAR (FAR = 0)	FRR (FAR = 0)
11.4 %	16.5 %	13.95 %	88.7 %	0.0 %	0.0 %	82.0 %

Table 5.15: Error rates obtained using the test set of Configuration II, VS-WFND and category 1 setting B.

FAR=FRR	FAR (FRR = 0)	FRR (FAR = 0)
5.0 %	66.2 %	35.0 %

Table 5.16: Error rates obtained using the evaluation set of Configuration I, VS-WFND and manually located eye positions.

FAR (FAR=FRR)	FRR (FAR=FRR)	HTER (FAR=FRR)	FAR (FRR = 0)	FRR (FRR = 0)	FAR (FAR = 0)	FRR (FAR = 0)
5.0 %	12.8 %	8.9 %	59.3 %	0.3 %	0.1 %	62.0 %

Table 5.17: Error rates obtained using the test set of Configuration I, VS-WFND and manually located eye positions.

FAR=FRR	FAR (FRR = 0)	FRR (FAR = 0)
9.1 %	88.0 %	68.8 %

Table 5.18: Error rates obtained using the evaluation set of Configuration II, VS-WFND and manually located eye positions.

FAR (FAR=FRR)	FRR (FAR=FRR)	HTER (FAR=FRR)	FAR (FRR = 0)	FRR (FRR = 0)	FAR (FAR = 0)	FRR (FAR = 0)
8.6 %	10.0 %	9.3 %	84.3 %	0.0 %	0.0 %	68.0 %

Table 5.19: Error rates obtained using the test set of Configuration II, VS-WFND and manually located eye positions.

	FAR=FRR	FAR (FRR = 0)	FRR (FAR = 0)
AUT	8.1 %	48.4 %	19.0 %
IDIAP	8.0 %	54.9 %	16.0 %
UniS-A-G-NC	5.7 %	96.4 %	26.7 %
UniS-A-S-NC	5.3 %	99.3 %	25.3 %
UniS-S-G-NC	3.5 %	81.1 %	16.2 %
UniS-S-S-NC	3.3 %	92.9 %	15.7 %
UniS-A-G-SM	10.0 %	99.8 %	93.8 %
UniS-A-S-SM	7.0 %	97.3 %	63.7 %
UniS-S-G-SM	6.5 %	93.9 %	40.0 %
UniS-S-S-SM	2.5 %	84.2 %	25.7 %
UniS-SVM	6.9 %	-	-

Table 5.20: XM2VTS database results quoted from literature. Error rates obtained using the evaluation set of Configuration I.

	FAR=FRR	FAR (FRR = 0)	FRR (FAR = 0)
AUT	6.5 %	36.9 %	18.8 %
IDIAP	7.0 %	59.4 %	19.8 %
UniS-S-G-NC	1.3 %	43.5 %	9.3 %
UniS-S-S-NC	1.3 %	55.4 %	8.5 %
UniS-S-G-SM	3.5 %	42.0 %	18.8 %
UniS-S-S-SM	1.3 %	23.1 %	18.8 %

Table 5.21: XM2VTS database results quoted from literature. Error rates obtained using the evaluation set of Configuration II.

	FAR (FAR=FRR)	FRR (FAR=FRR)	HTER (FAR=FRR)	FAR (FRR = 0)	FRR (FRR = 0)	FAR (FAR = 0)	FRR (FAR = 0)
AUT	8.2 %	6.0 %	7.1 %	46.6 %	0.8 %	0.5 %	20.0 %
IDIAP	8.1 %	8.5 %	8.3 %	54.5 %	0.5 %	0.5 %	20.5 %
UniS-A-G-NC	7.6 %	6.8 %	7.2 %	96.5 %	0.3 %	0.0 %	27.5 %
UniS-A-S-NC	7.4 %	6.8 %	7.1 %	99.4 %	0.0 %	0.0 %	24.3 %
UniS-S-G-NC	3.5 %	2.8 %	3.15 %	81.2 %	0.0 %	0.0 %	14.5 %
UniS-S-S-NC	3.3 %	3.0 %	3.15 %	93.5 %	0.0 %	0.0 %	14.0 %
UniS-A-G-SM	9.8 %	8.8 %	9.3 %	100.0 %	0.0 %	0.0 %	97.3 %
UniS-A-S-SM	5.8 %	7.3 %	6.55 %	99.6 %	0.0 %	0.0 %	58.5 %
UniS-S-G-SM	6.5 %	5.3 %	5.9 %	94.1 %	0.0 %	0.0 %	37.5 %
UniS-S-S-SM	2.3 %	2.5 %	2.4 %	85.0 %	0.3 %	0.0 %	24.3 %
UniS-SVM	7.7 %	6.3 %	7.0 %	-	-	-	-

Table 5.22: XM2VTS database results quoted from literature. Error rates obtained using the test set of Configuration I.

	FAR (FAR=FRR)	FRR (FAR=FRR)	HTER (FAR=FRR)	FAR (FRR = 0)	FRR (FRR = 0)	FAR (FAR = 0)	FRR (FAR = 0)
AUT	6.2 %	3.5 %	4.85 %	34.7 %	0.8 %	0.5 %	16.3 %
IDIAP	7.7 %	7.3 %	7.5 %	0.3 %	54.2 %	1.0 %	18.0 %
UniS-S-G-NC	1.3 %	1.8 %	1.55 %	44.2 %	0.3 %	0.0 %	9.0 %
UniS-S-S-NC	1.2 %	1.5 %	1.35 %	55.6 %	0.3 %	0.0 %	8.5 %
UniS-S-G-SM	3.5 %	3.8 %	3.65 %	42.1 %	0.5 %	0.0 %	19.5 %
UniS-S-S-SM	1.2 %	1.0 %	1.1 %	22.6 %	0.3 %	0.0 %	20.5 %

Table 5.23: XM2VTS database results quoted from literature. Error rates obtained using the test set of Configuration II.

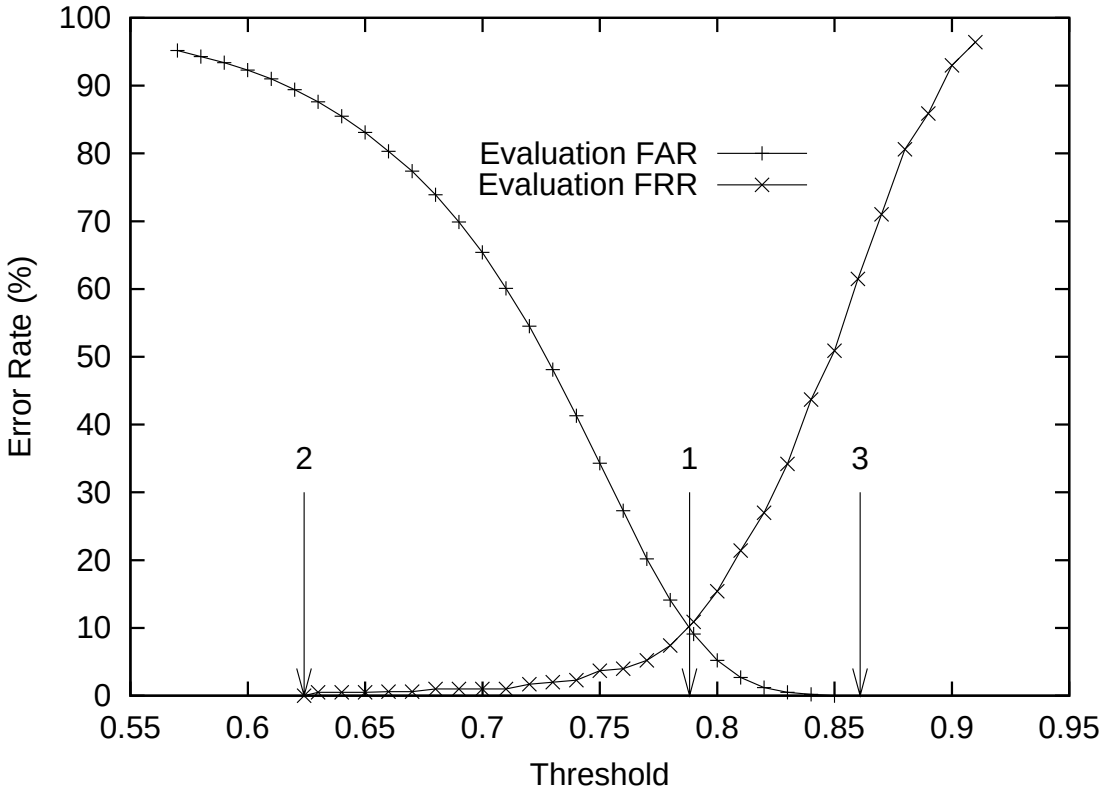


Figure 5.7: Configuration I evaluation set results for VS-WFND using category 1 setting B: FAR and FRR error rates versus the decision threshold. Arrow 1 is the position of the threshold for FAR=FRR. Arrow 2 is the position of the threshold for FRR=0. Arrow 3 is the position of the threshold for FAR=0.

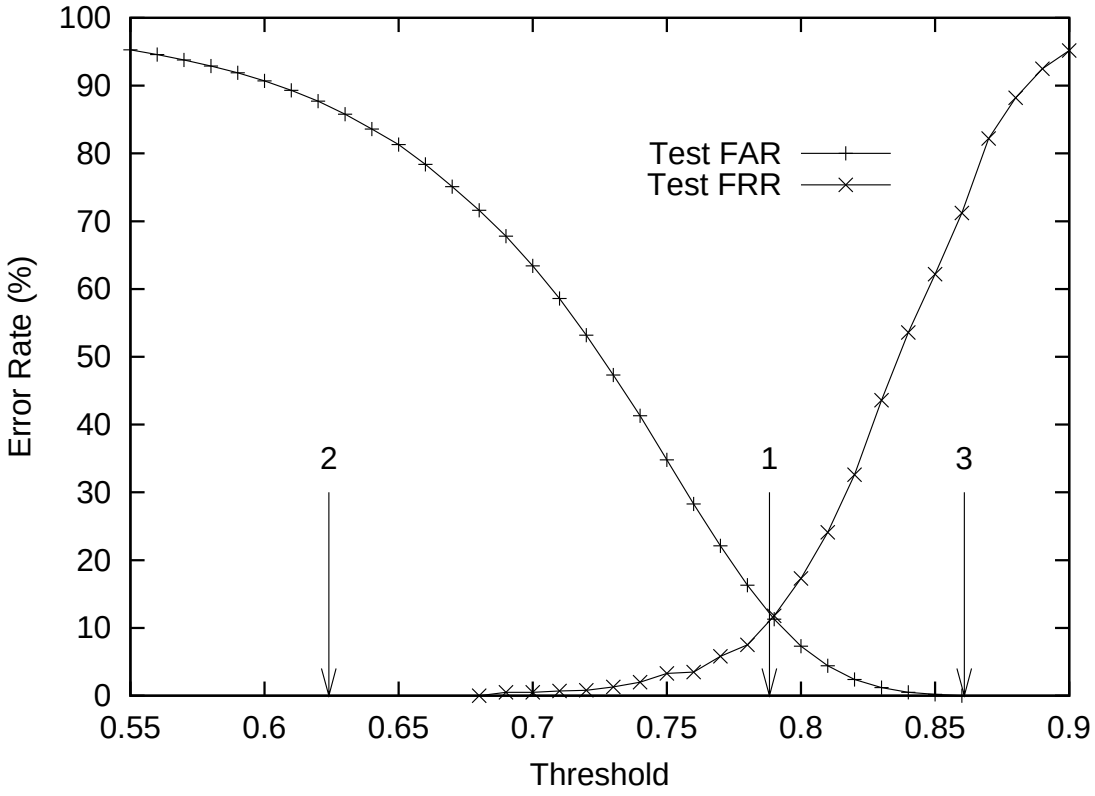


Figure 5.8: Configuration I test set results for VS-WFND using category 1 setting B: FAR and FRR error rates versus the decision threshold. The three arrows correspond to the thresholds selected using the evaluation results. Arrow 1 corresponds to FAR=FRR, arrow 2 to FRR=0 and arrow 3 to FAR=0 in the evaluation set.

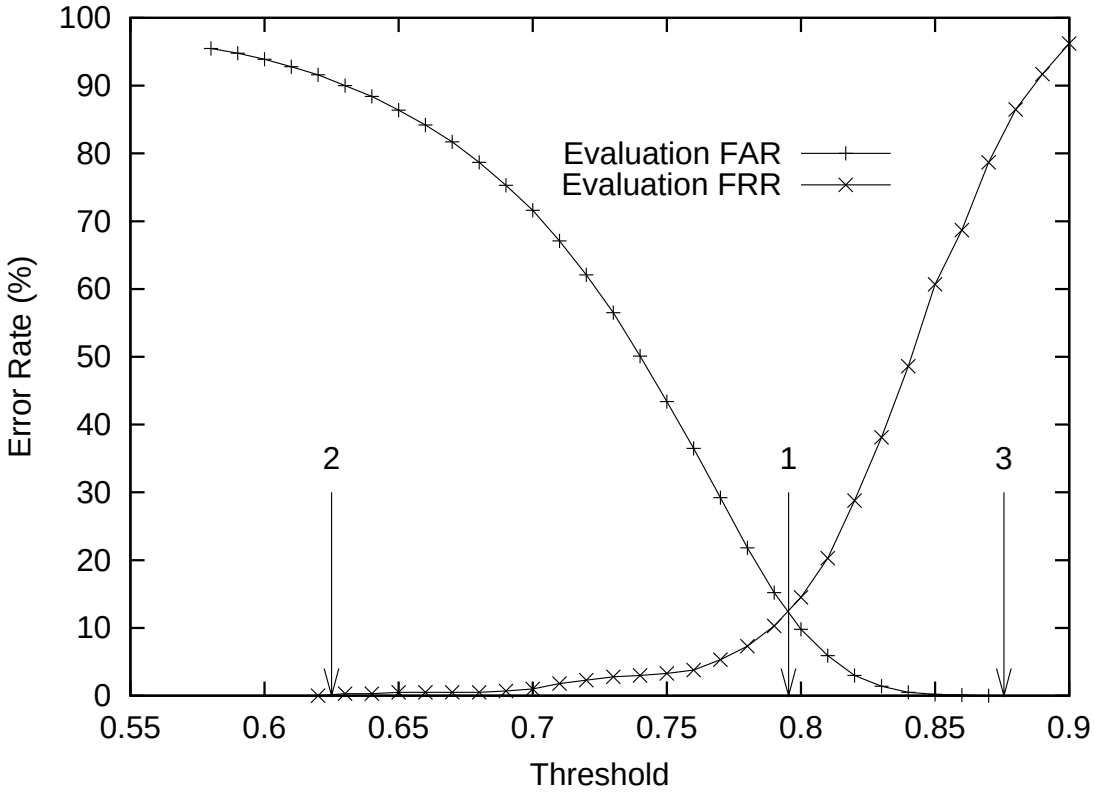


Figure 5.9: Configuration II evaluation set results for VS-WFND using category 1 setting B: FAR and FRR error rates versus the decision threshold. Arrow 1 is the position of the threshold for FAR=FRR. Arrow 2 is the position of the threshold for FRR=0. Arrow 3 is the position of the threshold for FAR=0.

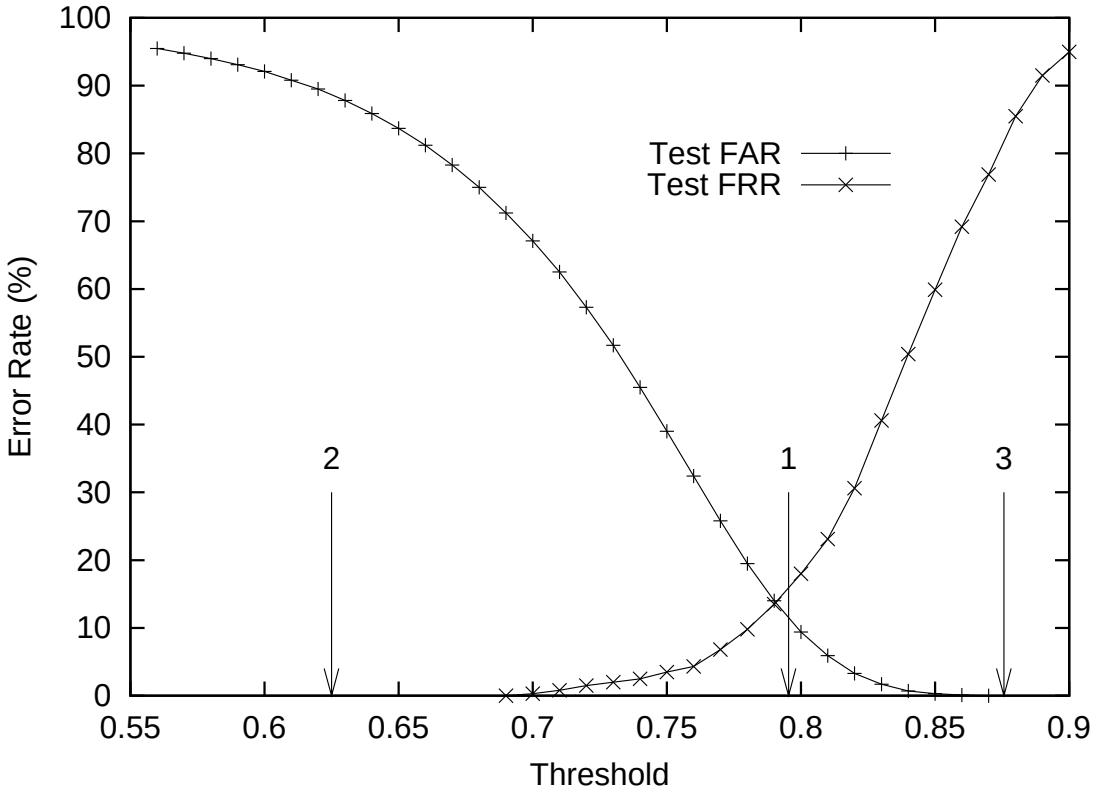


Figure 5.10: Configuration II test set results for VS-WFND using category 1 setting B: FAR and FRR error rates versus the decision threshold. The three arrows correspond to the thresholds selected using the evaluation results. Arrow 1 corresponds to FAR=FRR, arrow 2 to FRR=0 and arrow 3 to FAR=0 in the evaluation set.

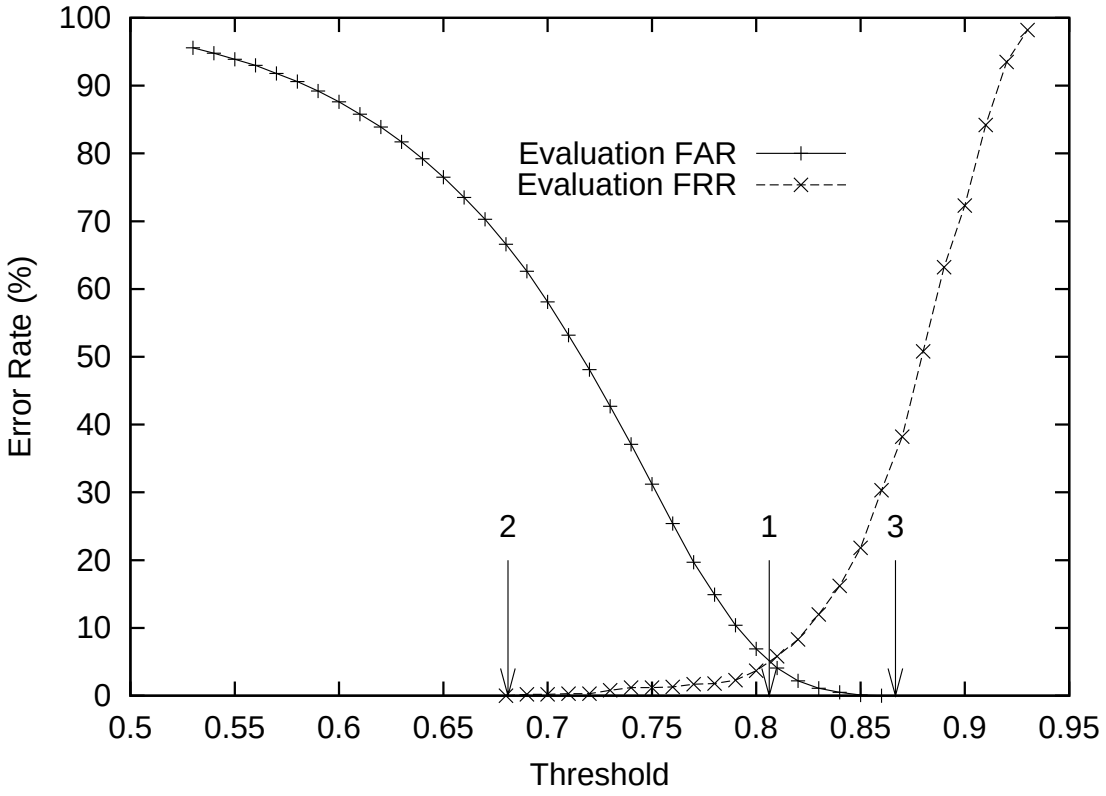


Figure 5.11: Configuration I evaluation set results for VS-WFND using manually located eye positions: FAR and FRR error rates versus the decision threshold. Arrow 1 is the position of the threshold for FAR=FRR. Arrow 2 is the position of the threshold for FRR=0. Arrow 3 is the position of the threshold for FAR=0.

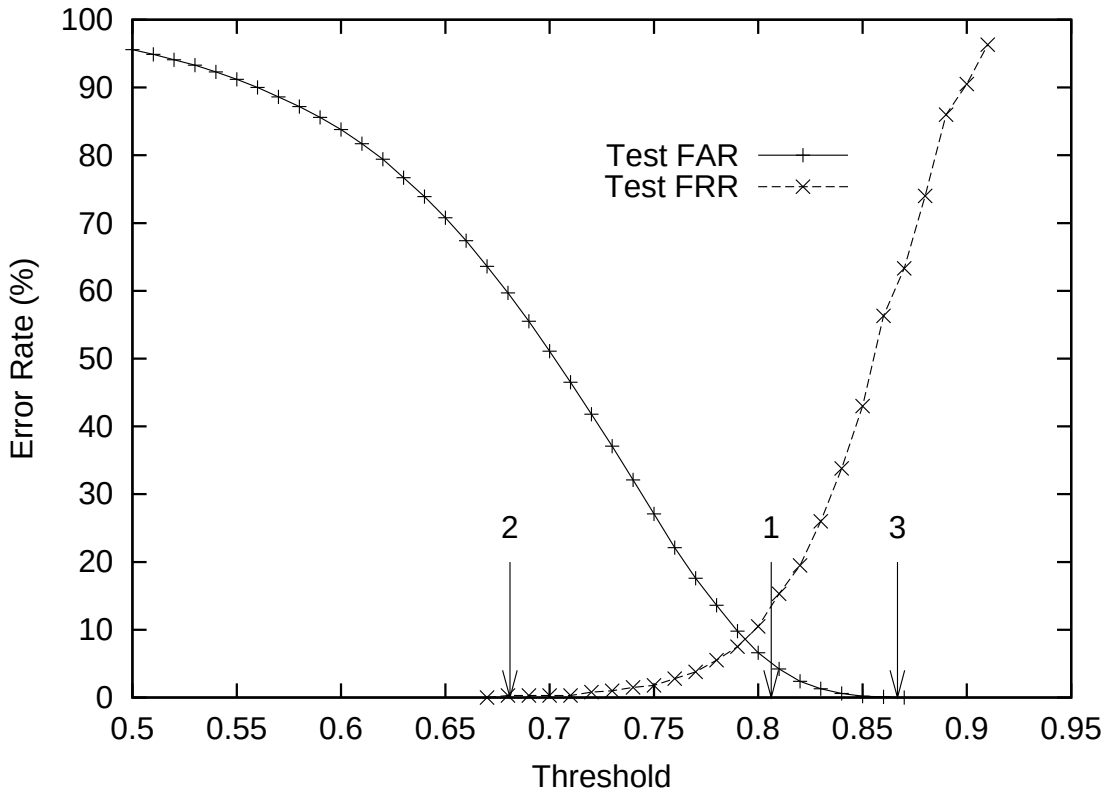


Figure 5.12: Configuration I test set results for VS-WFND using manually located eye positions: FAR and FRR error rates versus the decision threshold. The three arrows correspond to the thresholds selected using the evaluation results. Arrow 1 corresponds to FAR=FRR, arrow 2 to FRR=0 and arrow 3 to FAR=0 in the evaluation set.

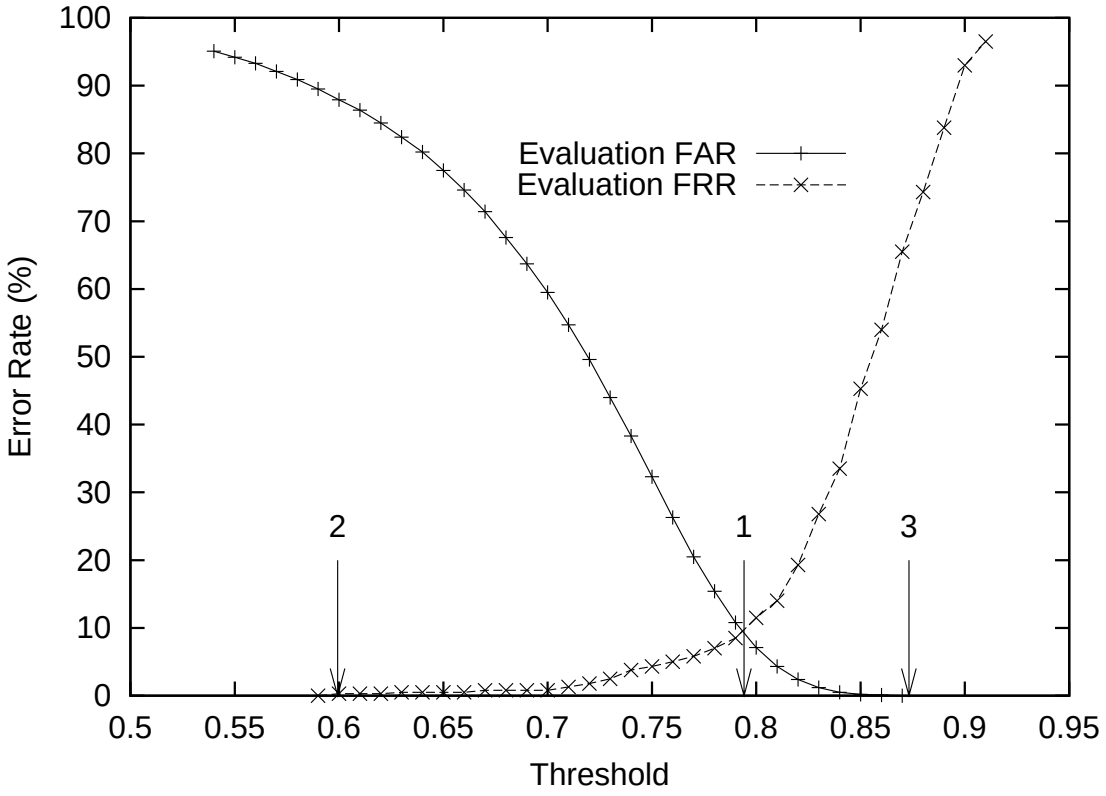


Figure 5.13: Configuration II evaluation set results for VS-WFND using manually located eye positions: FAR and FRR error rates versus the decision threshold. Arrow 1 is the position of the threshold for FAR=FRR. Arrow 2 is the position of the threshold for FRR=0. Arrow 3 is the position of the threshold for FAR=0.

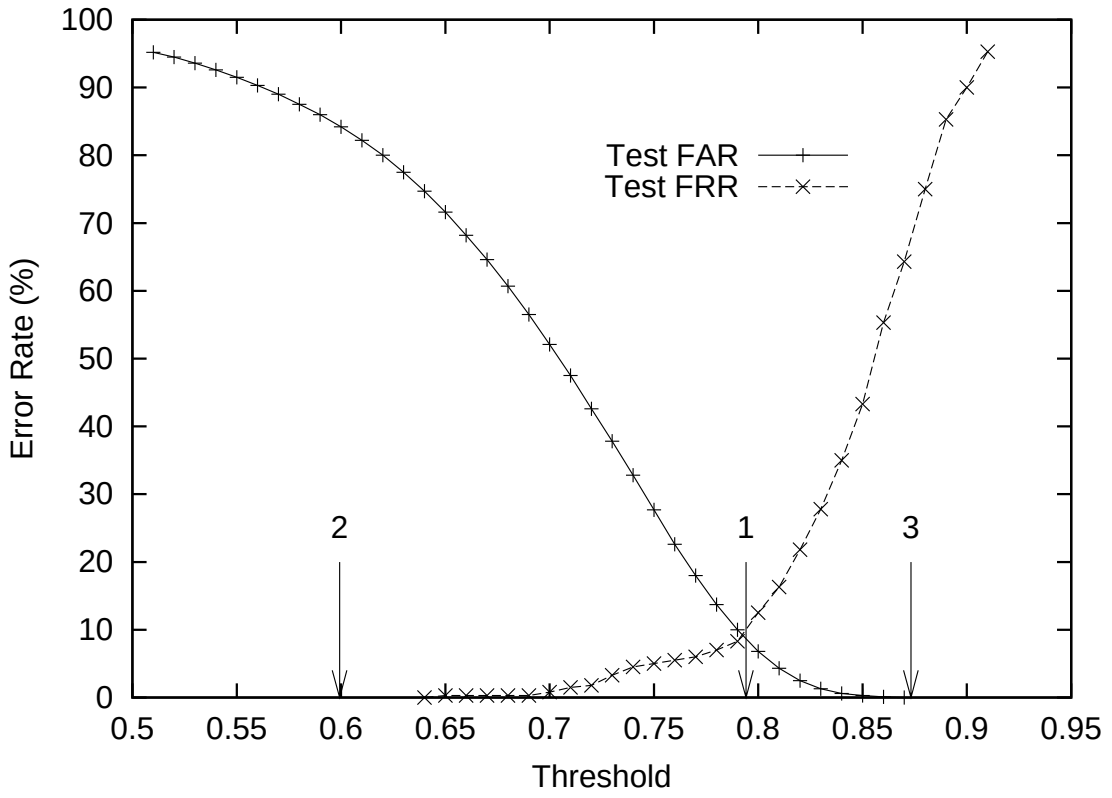


Figure 5.14: Configuration II test set results for VS-WFND using manually located eye positions: FAR and FRR error rates versus the decision threshold. The three arrows correspond to the thresholds selected using the evaluation results. Arrow 1 corresponds to FAR=FRR, arrow 2 to FRR=0 and arrow 3 to FAR=0 in the evaluation set.

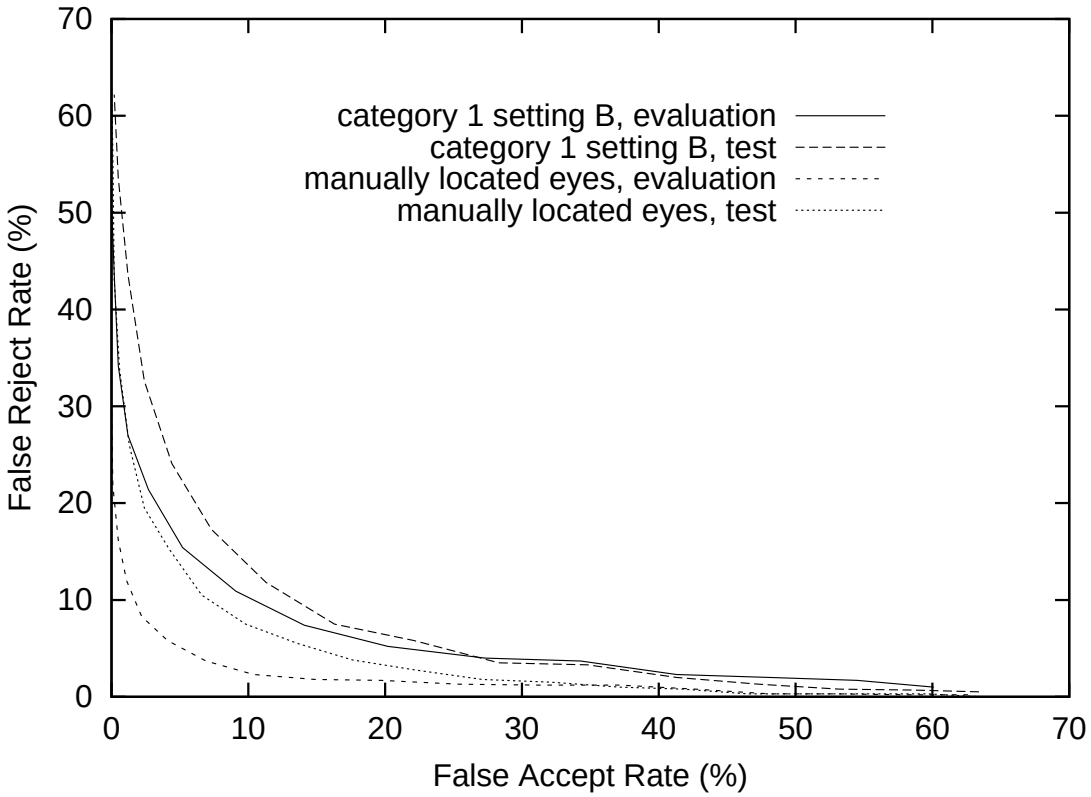


Figure 5.15: Configuration I results for VS-WFND showing the relationship between the false reject rate and the false accept rate. Results for the evaluation and test of the system using category 1 setting B and manually located eye locations are shown.

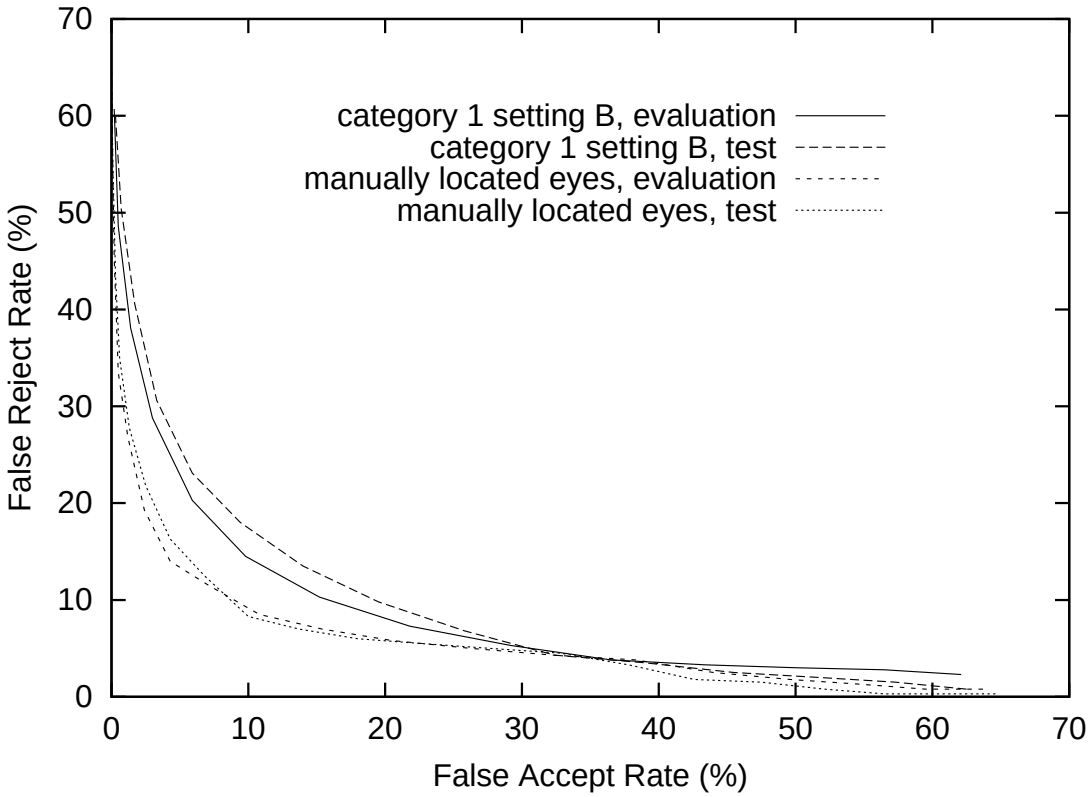


Figure 5.16: Configuration II results for VS-WFND showing the relationship between the false reject rate and the false accept rate. Results for the evaluation and test of the system using category 1 setting B and manually located eye locations are shown.

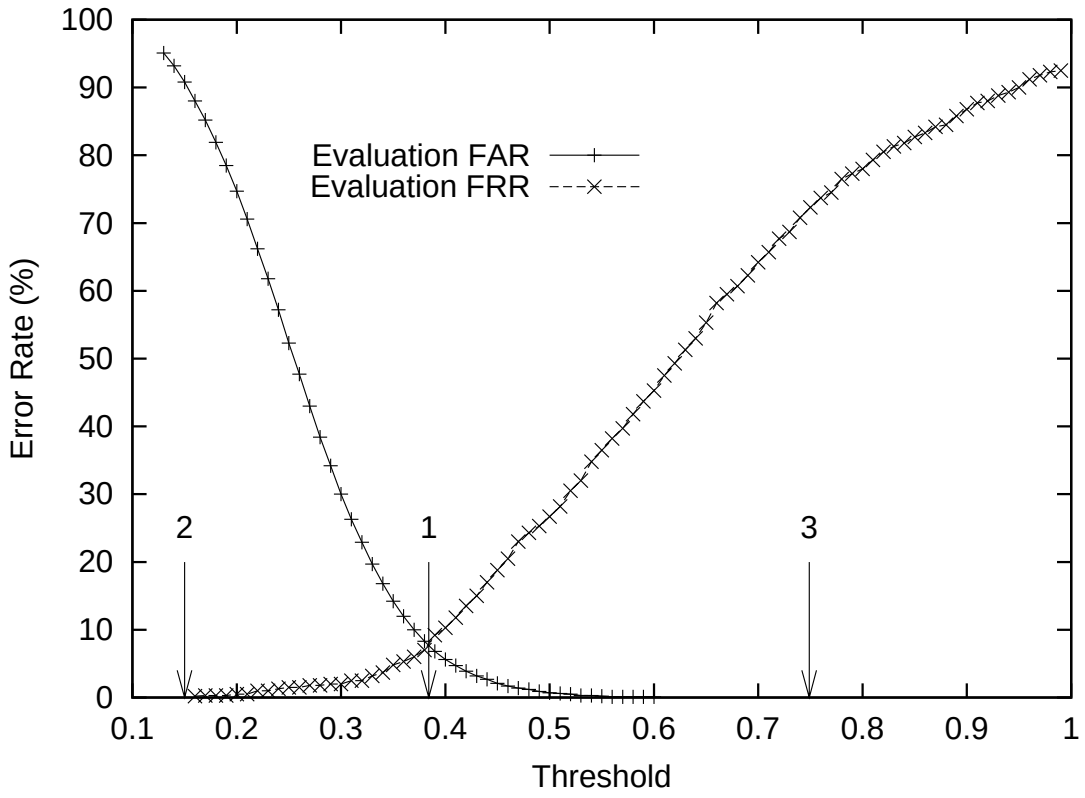


Figure 5.17: Configuration I evaluation set results for VS-WFND using manually located eye positions and normalised scores: FAR and FRR error rates versus the decision threshold. Arrow 1 is the position of the threshold for FAR=FRR. Arrow 2 is the position of the threshold for FRR=0. Arrow 3 is the position of the threshold for FAR=0.

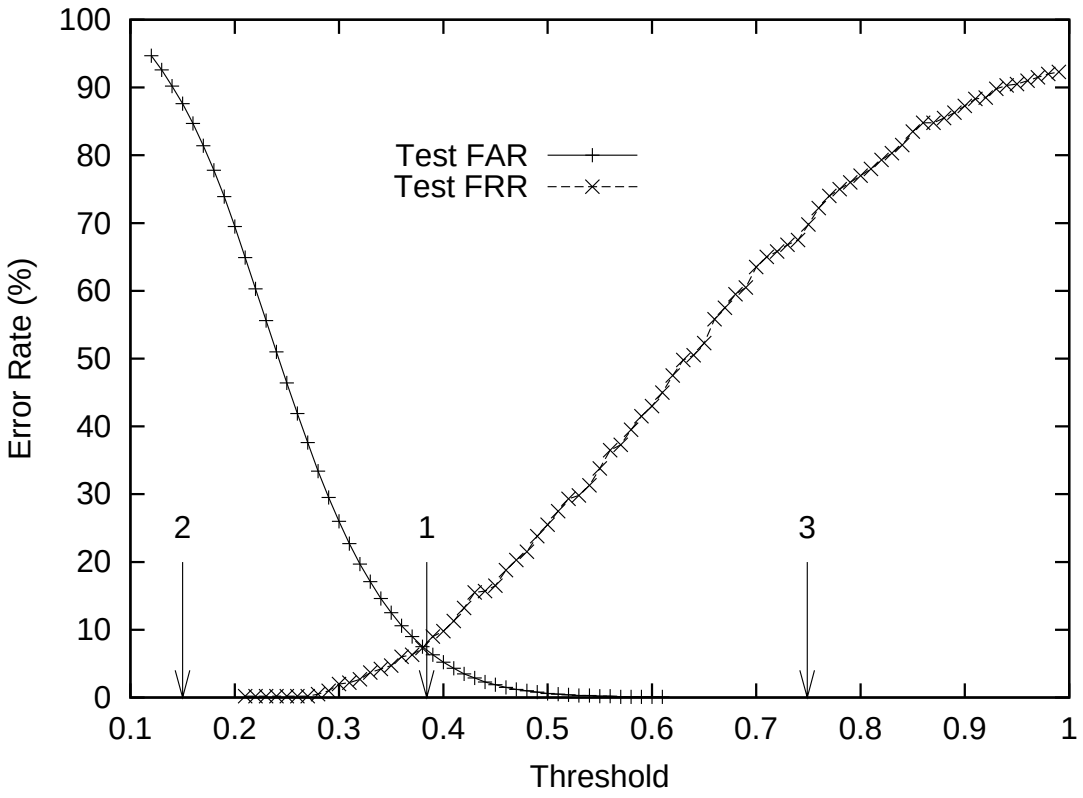


Figure 5.18: Configuration I test set results for VS-WFND using manually located eye positions and normalised scores: FAR and FRR error rates versus the decision threshold. The three arrows correspond to the thresholds selected using the evaluation results. Arrow 1 corresponds to FAR=FRR, arrow 2 to FRR=0 and arrow 3 to FAR=0 in the evaluation set.

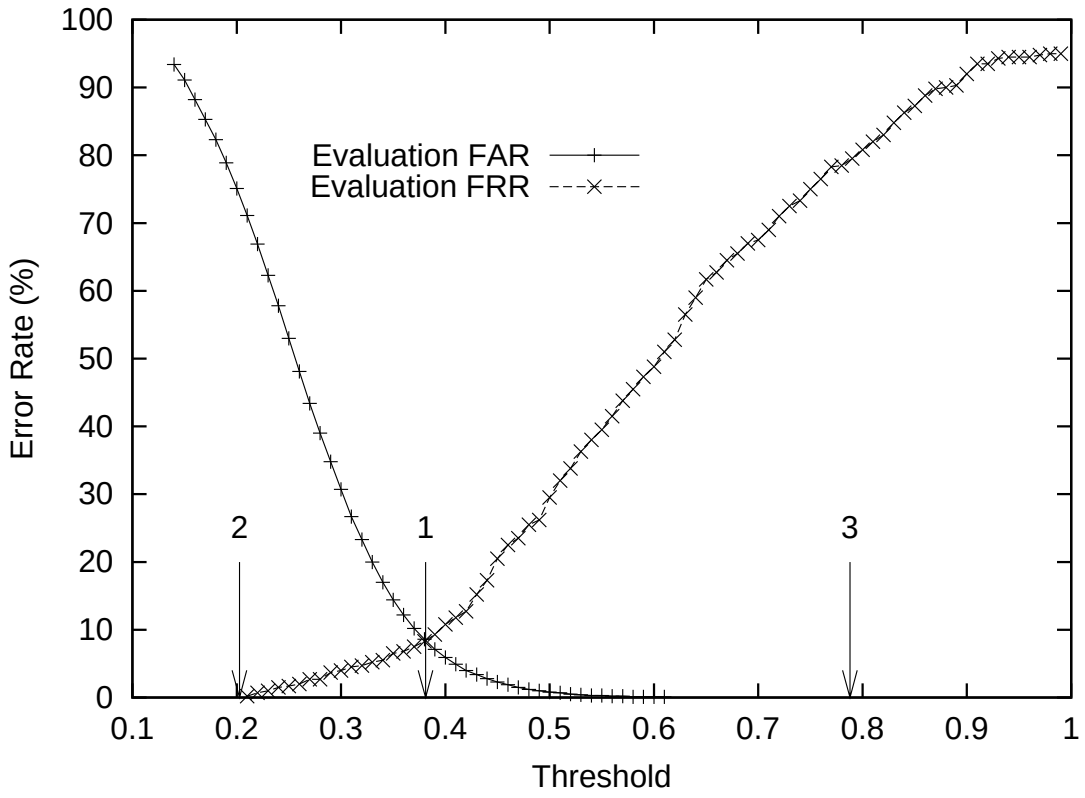


Figure 5.19: Configuration II evaluation set results for VS-WFND using manually located eye positions and normalised scores: FAR and FRR error rates versus the decision threshold. Arrow 1 is the position of the threshold for FAR=FRR. Arrow 2 is the position of the threshold for FRR=0. Arrow 3 is the position of the threshold for FAR=0.

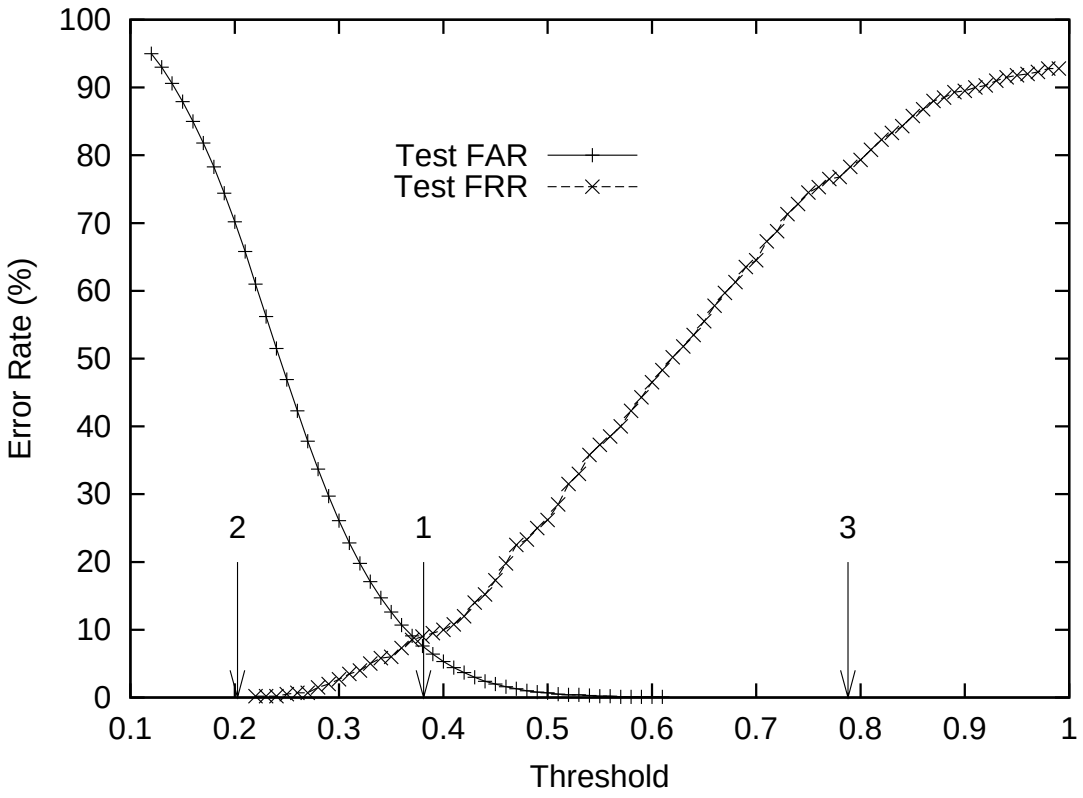


Figure 5.20: Configuration II test set results for VS-WFND using manually located eye positions and normalised scores: FAR and FRR error rates versus the decision threshold. The three arrows correspond to the thresholds selected using the evaluation results. Arrow 1 corresponds to FAR=FRR, arrow 2 to FRR=0 and arrow 3 to FAR=0 in the evaluation set.

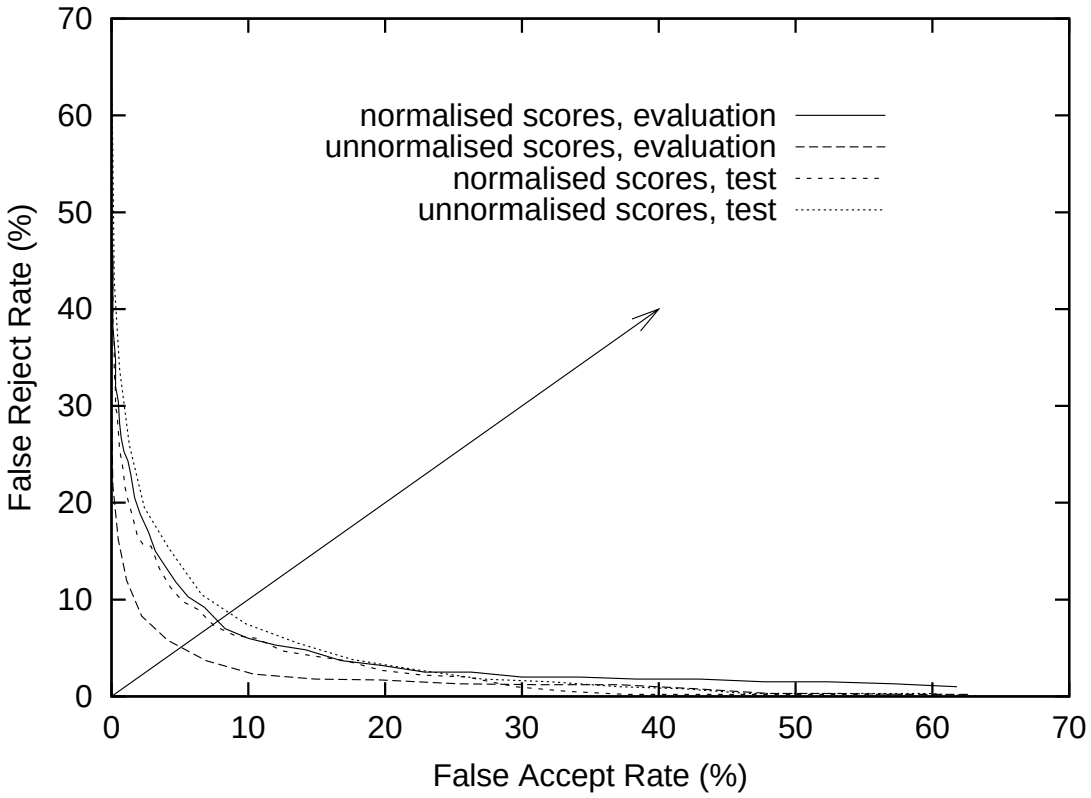


Figure 5.21: Configuration I results for VS-WFND showing the relationship between the false reject rate and the false accept rate. Results for the evaluation and test of the system using manually located eye locations with and without score normalisations are shown. Also shown is a line with slope 1, and the points of intersection between the curves and this line are the points where $FAR = FRR$.

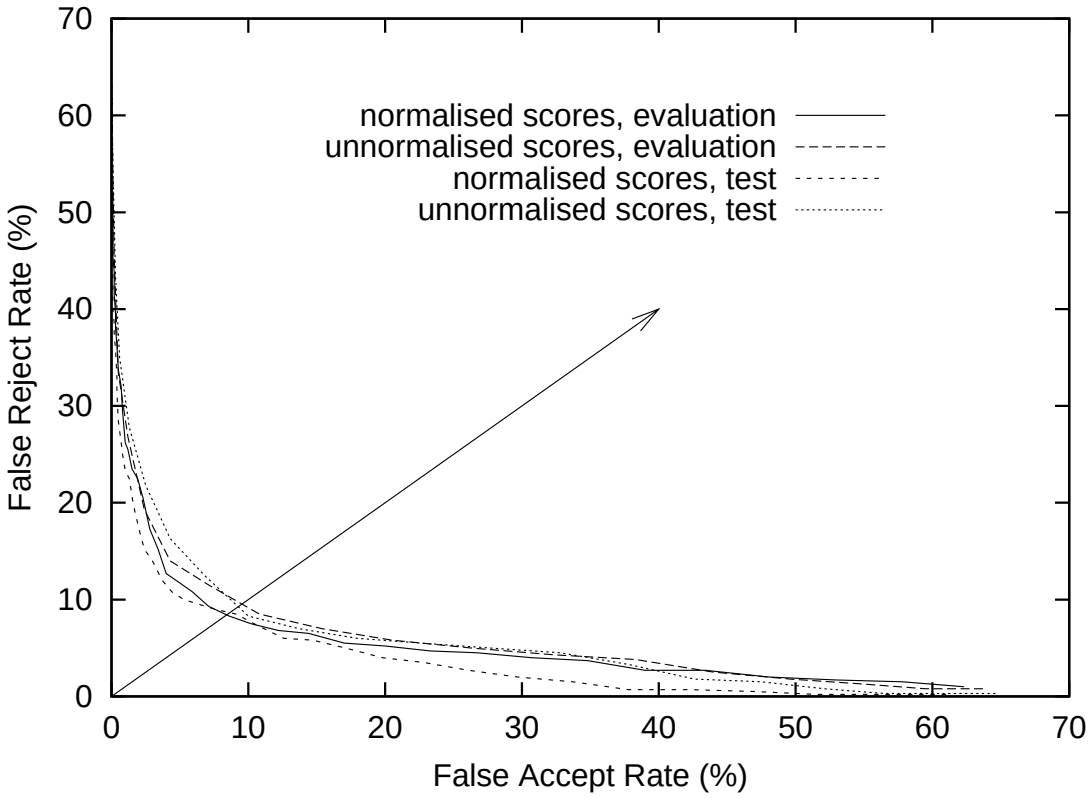


Figure 5.22: Configuration II results for VS-WFND showing the relationship between the false reject rate and the false accept rate. Results for the evaluation and test of the system using manually located eye locations with and without score normalisations are shown. Also shown is a line with slope 1, and the points of intersection between the curves and this line are the points where $FAR = FRR$.

FAR=FRR	FAR (FRR = 0)	FRR (FAR = 0)
7.6 %	90.8 %	72.3 %

Table 5.24: Error rates obtained using the evaluation set of Configuration I, VS-WFND, manually located eye positions and normalised scores.

FAR (FAR=FRR)	FRR (FAR=FRR)	HTER (FAR=FRR)	FAR (FRR = 0)	FRR (FRR = 0)	FAR (FAR = 0)	FRR (FAR = 0)
7.0 %	8.3 %	7.7 %	87.6 %	0.0 %	0.0 %	69.3 %

Table 5.25: Error rates obtained using the test set of Configuration I, VS-WFND, manually located eye positions and normalised scores.

FAR=FRR	FAR (FRR = 0)	FRR (FAR = 0)
8.3 %	74.0 %	79.3 %

Table 5.26: Error rates obtained using the evaluation set of Configuration II, VS-WFND, manually located eye positions and normalised scores.

FAR (FAR=FRR)	FRR (FAR=FRR)	HTER (FAR=FRR)	FAR (FRR = 0)	FRR (FRR = 0)	FAR (FAR = 0)	FRR (FAR = 0)
7.4 %	9.0 %	8.2 %	69.0 %	0.0 %	0.0 %	78.3 %

Table 5.27: Error rates obtained using the test set of Configuration II, VS-WFND, manually located eye positions and normalised scores.

	FAR (FAR=FRR)	FRR (FAR=FRR)	HTER
Evaluation, un-normalised scores	2.2 %	2.2 %	2.2 %
Test, thresholds determined on evaluation set, un-normalised scores	2.3 %	27.0 %	14.7 %
Test, thresholds determined on test set, un-normalised scores	2.8 %	2.8 %	2.8 %
Evaluation, normalised scores	4.3 %	4.3 %	4.3 %
Test, thresholds determined on evaluation set, normalised scores	4.6 %	15.0 %	9.8 %
Test, thresholds determined on test set, normalised scores	3.8 %	3.8 %	3.8 %

Table 5.28: Error rates obtained using Configuration I, VS-WFND, manually located eye positions and client specific thresholding.

	FAR (FAR=FRR)	FRR (FAR=FRR)	HTER
Evaluation, un-normalised scores	2.5 %	3.8 %	3.2 %
Test, thresholds determined on evaluation set, un-normalised scores	3.2 %	23.5 %	13.4 %
Test, thresholds determined on test set, un-normalised scores	4.0 %	4.0 %	4.0 %
Evaluation, normalised scores	2.8 %	4.3 %	3.6 %
Test, thresholds determined on evaluation set, normalised scores	3.1 %	19.8 %	11.5 %
Test, thresholds determined on test set, normalised scores	3.5 %	3.5 %	3.5 %

Table 5.29: Error rates obtained using Configuration II, VS-WFND, manually located eye positions and client specific thresholding.

5.4 Identification System

An identification system based on the FFT-assisted SVM eye detection and the weighted FND is presented here. The face localisation and extraction component is exactly the same as the one for VS-WFND described in Section 5.3.2.1, where the extracted height is 76 pixels and extracted width is 56 pixels. The weighted FND is used for identifying an unknown individual. The set of weighting functions used in most of the experiments reported here is

$$\begin{aligned}\mathcal{W}_1 \;=\; & \left\{\left(\frac{i}{76}, \frac{i+s_h}{76}, \frac{j}{56}, \frac{j+s_w}{56}, 1.0, 0.0\right) : i \in \{7,9,\ldots,29\}, j \in \{0,2,\ldots,12\}\right\} \cup \\ & \left\{\left(\frac{i}{76}, \frac{i+s_h}{76}, \frac{j}{56}, \frac{j+s_w}{56}, 1.0, 0.0\right) : i \in \{7,9,\ldots,29\}, j \in \{18,20,\ldots,30\}\right\}\end{aligned} \tag{5.14}$$

where s_h is the height and s_w is the width of the window function. The cost c_1 is set to one and c_2 is set to zero for all members of $\mathcal{W}_1$. This set of weighting functions emphasizes the region around the top part of the face including the eyes, and de-emphasizes the bottom part. The top part of the face encompassing the eyes and some of the nose region tend to be more invariant than the bottom part of the face, which is more affected by facial expressions and facial hair.

The set containing a single weighting function used in category 1 setting B is also used in an experiment here. That set is $\mathcal{W}_2 = \{(0.0, 0.5, 0.0, 0.0, 1.0, 0.0025)\}$.

5.4.1 Experiment Results

The client training and evaluation sets from Configurations I and II of the XM2VTS database are used. The client training set is used to create the database of fractal codes. The client evaluation data is used for the identification test. For Configuration I, there are training and testing data from each session, where each session is taken at different times several months apart. In Configuration II, the training and evaluation data are taken from different sessions. As a result of the images being obtained at different times, there tends to be greater variance between the training

and evaluation sets in terms of pose and lighting. This allows us to evaluate the system's sensitivity to faces taken at different times.

In all the experiments described here, the extracted face dimensions are 76×56, with the left-eye center coordinates at (34, 17) and the right-eye center coordinates at (34, 39). Other extracted face dimensions with different left-eye and right-eye coordinates are also tested but not reported here because they give similar or slightly inferior results. The normalised scores method described in Section 5.3.3.1 is also adapted for facial identification, but preliminary results are significantly inferior to the ones reported here.

The following settings are used in our experiments, where the eye detector R_b from Chapter 2 is used in all settings except 8 to 11:

1. Fractal codes in the training database are created using uniform range block distribution, maximum sub-domain block usage of six (see Section 3.5.3.1), minimum $\alpha = 0$ and maximum $\alpha = 0.5$ and the set of weighting functions $\mathcal{W}_1$ is used.

   (a) Weighting functions have a height of $s_h = 26$ and width of $s_w = 26$. The set of shifts is $\mathcal{H} = \{(2i, 2j) : i = -1, \ldots, 1 \text{ and } j = -1, \ldots, 1\}$.

   (b) Weighting functions have a height of $s_h = 26$ and width of $s_w = 26$. The set of shifts is $\mathcal{H} = \{(2i, 2j) : i = -2, \ldots, 2 \text{ and } j = -2, \ldots, 2\}$.

   (c) Weighting functions have a height of $s_h = 26$ and width of $s_w = 26$. The set of shifts is $\mathcal{H} = \{(3i, 3j) : i = -2, \ldots, 2 \text{ and } j = -2, \ldots, 2\}$.

   (d) Weighting functions have a height of $s_h = 26$ and width of $s_w = 26$. The set of shifts is $\mathcal{H} = \{(4i, 4j) : i = -2, \ldots, 2 \text{ and } j = -2, \ldots, 2\}$.

   (e) Weighting functions have a height of $s_h = 22$ and width of $s_w = 22$. The set of shifts is $\mathcal{H} = \{(2i, 2j) : i = -2, \ldots, 2 \text{ and } j = -2, \ldots, 2\}$.

   (f) Weighting functions have a height of $s_h = 24$ and width of $s_w = 24$. The set of shifts is $\mathcal{H} = \{(2i, 2j) : i = -2, \ldots, 2 \text{ and } j = -2, \ldots, 2\}$.

2. Fractal codes in the training database are created using uniform range block distribution, unrestricted sub-domain block usage, minimum $\alpha = 0$ and maximum $\alpha = 0.5$ and the set of weighting functions $\mathcal{W}_1$ is used. The weighting functions have a height of $s_h = 26$ and width of $s_w = 26$. The set of shifts is $\mathcal{H} = \{(2i, 2j) : i = -2, \ldots, 2 \text{ and } j = -2, \ldots, 2\}$.

3. Fractal codes in the training database are created using quad-tree range block distribution, unrestricted sub-domain block usage, minimum $\alpha = 0$ and maximum $\alpha = 0.5$ and the set of weighting functions $\mathcal{W}_1$ is used. The weighting functions have a height of $s_h = 26$ and width of $s_w = 26$. The set of shifts is $\mathcal{H} = \{(2i, 2j) : i = -2, \ldots, 2 \text{ and } j = -2, \ldots, 2\}$.

4. Fractal codes in the training database are created using quad-tree range block distribution, unrestricted sub-domain block usage, minimum $\alpha = 0$ and maximum $\alpha = 0.95$ and the set of weighting functions $\mathcal{W}_1$ is used. The weighting functions have a height of $s_h = 26$ and width of $s_w = 26$. The set of shifts is $\mathcal{H} = \{(2i, 2j) : i = -2, \ldots, 2 \text{ and } j = -2, \ldots, 2\}$.

5. Fractal codes in the training database are created using HV-partitioned range block distribution, unrestricted sub-domain block usage, minimum $\alpha = 0$ and maximum $\alpha = 0.5$ and the set of weighting functions $\mathcal{W}_1$ is used. The weighting functions have a height of $s_h = 26$ and width of $s_w = 26$. The set of shifts is $\mathcal{H} = \{(2i, 2j) : i = -2, \ldots, 2 \text{ and } j = -2, \ldots, 2\}$.

6. Fractal codes in the training database are created using uniform range block distribution, maximum sub-domain block usage of six (see Section 3.5.3.1), minimum $\alpha = 0$ and maximum $\alpha = 0.5$ and the set of weighting functions $\mathcal{W}_2$ is used. The weighting functions have a height of $s_h = 26$ and width of $s_w = 26$. The set of shifts is $\mathcal{H} = \{(-4, 0), (4, 0), (0, 0), (0, -4), (0, 4)\}$.

7. Fractal codes in the training database are created using uniform range block distribution, unrestricted sub-domain block usage, minimum $\alpha = 0$ and maximum $\alpha = 0.5$ and the set of weighting functions $\mathcal{W}_2$ is used. The weighting

functions have a height of $s_h = 26$ and width of $s_w = 26$. The set of shifts is $\mathcal{H} = \{(-4, 0), (4, 0), (0, 0), (0, -4), (0, 4)\}$.

8. Fractal codes in the training database are created using uniform range block distribution, maximum sub-domain block usage of six (see Section 3.5.3.1), minimum $\alpha = 0$ and maximum $\alpha = 0.5$ and the set of weighting functions $\mathcal{W}_1$ is used. The weighting functions have a height of $s_h = 26$ and width of $s_w = 26$. The set of shifts are $\mathcal{H} = \{(2i, 2j) : i = -2, \ldots, 2 \text{ and } j = -2, \ldots, 2\}$. Manually located eye positions are used.

9. Fractal codes in the training database are created using uniform range block distribution, maximum sub-domain block usage of six (see Section 3.5.3.1), minimum $\alpha = 0$ and maximum $\alpha = 0.5$. The standard FND as described by Equation (3.6) is used. The set of shifts is $\mathcal{H} = \{(0, 0)\}$, that is, shifting is not used. Manually located eye positions are used.

10. The eigenfaces method is used. The maximum number of principal components is used in the eigenspace representation. This meant using 599 principal components for Configuration I and 799 principal components for Configuration II. The faces are normalised using manually located eye positions.

11. The eigenfaces method is used. One hundred principal components are used in the eigenspace representation for both configurations. The faces are normalised using manually located eye positions.

Setting 7 is equivalent to category 1 setting B of the verification system. Settings 1 to 11 are used for Configuration I experiments. Table 5.30 shows the recognition rate results in terms of the system identifying the correct face as the top match. Settings 1b and 1c give the best results in terms of the top matching identity being correct. Setting 1b is chosen for Configuration II experiments, because as Figure 5.23 shows, setting 1b performs better than setting 1c. Figure 5.23 shows the recognition rate as a function of the system returning the correct identity in the first n matches. To avoid over cluttering the graph, only the results for a selection of settings are shown.

Settings 8 and 9 use manually located eye positions. Experiments with these settings are carried out to investigate the performance of the classifier in the case where the eye detector is perfect. For Configuration I, using setting 8 results in a recognition rate of 95.0 % in returning the top rank correctly. This outperforms the 86.7 % obtained with setting 1b, which use the automatic eye detection system R_b. This demonstrates that the weighted FND performs well in classification, but it is still sensitive to misalignments in the normalisation of the face. These misalignments can be due to pose and facial expressions. Particularly, pose that causes out-of-plane rotations have a significant effect on the recognition rate. In-plane rotations are corrected using the eye locations. However, knowing just the eye locations does not allow us to correct for out-of-plane rotations. To do so we require a three dimensional model of the face, locating other features in the face, and then performing out-of-plane rotations that align those features. This type of normalisation is not investigated in this thesis. This sensitivity is more obvious when we examine the results of Configuration II experiments.

Setting 9, which use the standard FND described in earlier chapters, gives a recognition rate of 91.5 % on Configuration I. Setting 8 consistently outperformed this setting, as evident in Figure 5.23. However, Configuration II experiment results show that setting 9 outperforms setting 8. The matching rank results for Configuration II are shown in Figure 5.24. This demonstrates that the standard FND has more invariance to pose and lighting changes than the weighted FND. However, when the faces are aligned properly the weighted FND outperforms the standard FND.

Settings 10 and 11 also use manually located eye positions. Here the standard eigenface approach is used. Setting 10 used the maximum number of principle components, and setting 11 used 100 principle components. Using the maximum number of principle components gives slightly better results than 100 principle components. System settings 8 and 9 consistently perform better than both of these eigenface methods for Configuration I. For Configuration II, setting 9 performs

System Setting	Recognition Rate
1a	86.7 %
1b	87.5 %
1c	87.5 %
1d	87.3 %
1e	86.5 %
1f	87.0 %
2	86.7 %
3	85.8 %
4	85.3 %
5	84.8 %
6	82.5 %
7	81.8 %
8	95.0 %
9	91.5 %
10	87.2 %
11	85.8 %

Table 5.30: Recognition rates for identification experiments using Configuration I. This is the rate at which the top ranking match is identified correctly.

better than the best eigenface method, up to a matching rank of about 20, when the eigenface method begins to have a slight advantage.

Similar to Configuration I results, Figure 5.24 shows that setting 1b has a significantly lower recognition rate due to inaccurate face normalisations, which uses the automatic eye positions returned by the eye detection system R_b. Although a left and right eye detection rate of 96.9 % is obtained, the detection errors compound into recognition errors. Using manually located eye positions we can eliminate the eye detection error rate, and in turn evaluate the performance of the FND based recognition system. Overall, the FND based classification engine still outperforms the widely used and referenced eigenface method. Although the fully automatic system has slightly inferior recognition rates compared to a system using manually located eye positions, it has the advantage of not requiring human intervention. The user interface of our fully automatic face identification system is described in Appendix F.

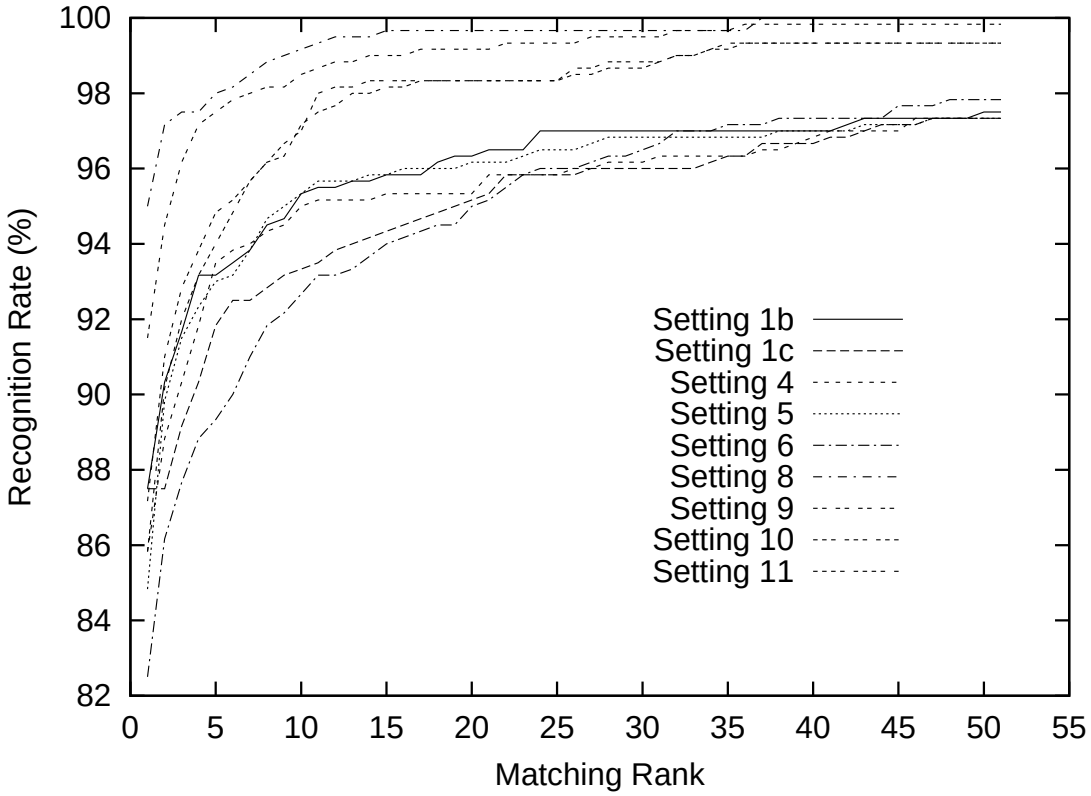


Figure 5.23: Recognition rate versus the rank of a match for a selection of settings when Configuration I is used. A matching rank of n represents a correct identity match in the top n matches.

System Setting	Recognition Rate
1b	78.3 %
8	86.5 %
9	87.5 %
10	85.3 %
11	84.0 %

Table 5.31: Recognition rates for identification experiments using Configuration II. This is the rate at which the top ranking match is identified correctly.

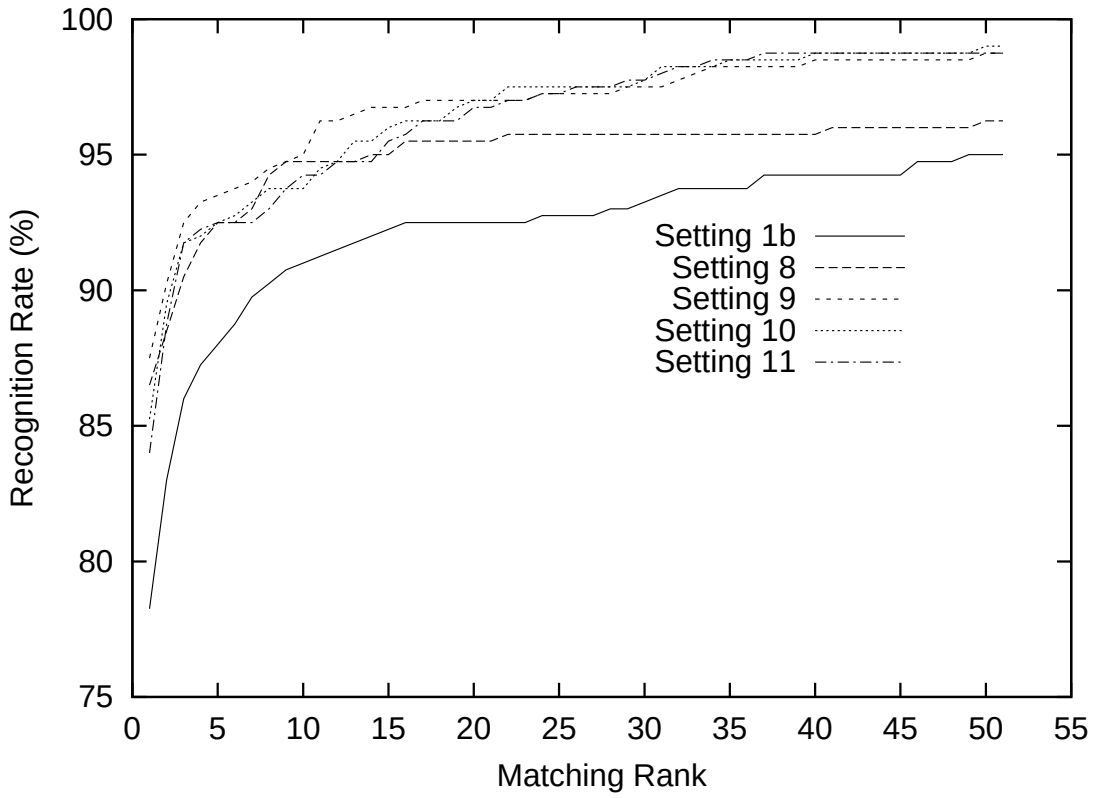


Figure 5.24: Recognition rate versus the rank of a match for a selection of settings when Configuration II is used. A matching rank of n represents a correct identity match in the top n matches.

(a)

(b)

Figure 5.25: Examples of pose variation in faces from the XM2VTS face database. (a) Faces from the Configuration II training set; (b) Faces from the Configuration II evaluation set.

5.5 Voting Scheme Using Normalised Weighted FNDs

The experiments in the previous section using the identification system demonstrate that the weighted FND performs well on training and testing data that are acquired within a short time of each other. However, its performance degrades significantly when the two datasets are obtained after several months. On analysing the two datasets from the XM2VTS database we observe that the main differences are in facial hair, hairstyle, eye glasses, pose and some lighting. As mentioned above, the main factor affecting our system is pose, which introduces out-of-plane rotations. Some images in different poses are shown in Figure 5.25. In this section we propose using normalised weighted FNDs combined with a voting scheme. The weighting functions are normalised to reduce the effects of lighting. The combined voting scheme is likened to a local feature matching process. Firstly, we define

$$
\begin{aligned}
d_{ww}(\mathbf{x},\mathbf{y},b_t,b_b,b_l,b_r,c_1,c_2) \;&=\\
&= \min_{(m,n)\in\mathcal{H}} \left\{ \left[\sum_{i=0}^{I_h}\sum_{j=0}^{I_w} \left[\frac{w(i,j,b_t,b_b,b_l,b_r,c_1,c_2)x(i,j)}{\sqrt{\sum_{k=0}^{I_h}\sum_{l=0}^{I_w}[w(k,l,b_t,b_b,b_l,b_r,c_1,c_2)x(k,l)]^2}} - \right.\right.\\
&\qquad \left.\left. \frac{w(i,j,b_t,b_b,b_l,b_r,c_1,c_2)y_{mn}(i,j)}{\sqrt{\sum_{k=0}^{I_h}\sum_{l=0}^{I_w}[w(k,l,b_t,b_b,b_l,b_r,c_1,c_2)y_{mn}(k,l)]^2}} \right]^2 \right]^{\frac{1}{2}} \right\}
\end{aligned}
\tag{5.15}
$$

where the pixel value at coordinates (k,l) in the image $\mathbf{x}$ is denoted by $x(k,l)$. The pixel value at coordinates (k,l) in the image shifted image $\mathbf{y}_{mn}$ is denoted by $y_{mn}(k,l)$. Also, for an unknown input face $\mathbf{x}$ to be identified, let

$$
\mathcal{V} = \left\{ \arg\min_{\mathbf{x}_i} \{ d_{ww}(f_{\mathbf{x}_i}(\mathbf{x}),\mathbf{x},b_t,b_b,b_l,b_r,c_1,c_2) \} : \forall u, u = (b_t,b_b,b_l,b_r,c_1,c_2) \in \mathcal{W} \right\}
\tag{5.16}
$$

where $\mathbf{x}_i$ is the ith face in the training database. The elements in the set $\mathcal{V}$ are faces from the training database that have been voted as likely matches. Each vote is determined by finding the $\mathbf{x}_i$ that minimises d_{ww} for a particular element in $\mathcal{W}$. There are in total as many votes as there are elements in the set $\mathcal{W}$. We can then derive a score for a face $\mathbf{x}_i$ in the training database that represents the confidence of the match to the unknown face $\mathbf{x}$,

$$
S(\mathbf{x}_i) = \frac{|\{\mathbf{y} : \mathbf{y} = \mathbf{x}_i, \forall \mathbf{y} \in \mathcal{V}\}|}{|\mathcal{W}|}.
\tag{5.17}
$$

The score value $S(\mathbf{x}_i)$ is between zero and one, and is the number of elements in $\mathcal{V}$ that are the same as $\mathbf{x}_i$, relative to the number of elements in $\mathcal{W}$. The $\mathbf{x}_i$ that maximises this score is the top ranking match. To determine whether this match is correct we have to compare the score of the correct entry with the scores of all other entries. Using the scores, we can rank the position of the correct entry relative to other entries. This is performed for the set of test images to obtain a recognition rate at each rank, from which Figure 5.26 is created.

This method is used to perform identification using Configuration II of the XM2VTS database. The aim is to compare the results with the one obtained in the previous

section that use the weighted FND for identification. The following settings are used:

I. Fractal codes in the training database are created using uniform range block distribution, maximum sub-domain block usage of six (see Section 3.5.3.1), minimum $\alpha = 0$ and maximum $\alpha = 0.5$ and the set of weighting functions $\mathcal{W}_1$ is used. The weighting functions have a height of $s_h = 26$ and width of $s_w = 26$. The set of shifts is $\mathcal{H} = \{(0, 0)\}$, that is, shifting is not used. Manually located eye positions are used.

II. Fractal codes in the training database are created using uniform range block distribution, maximum sub-domain block usage of six (see Section 3.5.3.1), minimum $\alpha = 0$ and maximum $\alpha = 0.5$ and the set of weighting functions $\mathcal{W}_1$ is used. The weighting functions have a height of $s_h = 16$ and width of $s_w = 16$. The set of shifts is $\mathcal{H} = \{(0, 0)\}$, that is, shifting is not used. Manually located eye positions are used.

III. Fractal codes in the training database are created using uniform range block distribution, maximum sub-domain block usage of six (see Section 3.5.3.1), minimum $\alpha = 0$ and maximum $\alpha = 0.5$ and the set of weighting functions $\mathcal{W}_1$ is used. The weighting functions have a height of $s_h = 6$ and width of $s_w = 6$. The set of shifts is $\mathcal{H} = \{(0, 0)\}$, that is, shifting is not used. Manually located eye positions are used.

The results for returning the top ranking match correctly using the voting scheme is summarised in Table 5.32. The results in Figure 5.26 show that the voting scheme generally performs better than weighted FNDs (Setting 8) for matching ranks larger than approximately twelve. Below that they had roughly similar average performances. However, weighted FNDs still performed better in terms of returning the top rank correctly. For matching ranks of 15 or less the standard FND outperforms all other methods. Again this shows the standard FND's invariance to pose and illumination, even when the whole face is used for recognition. This is despite the

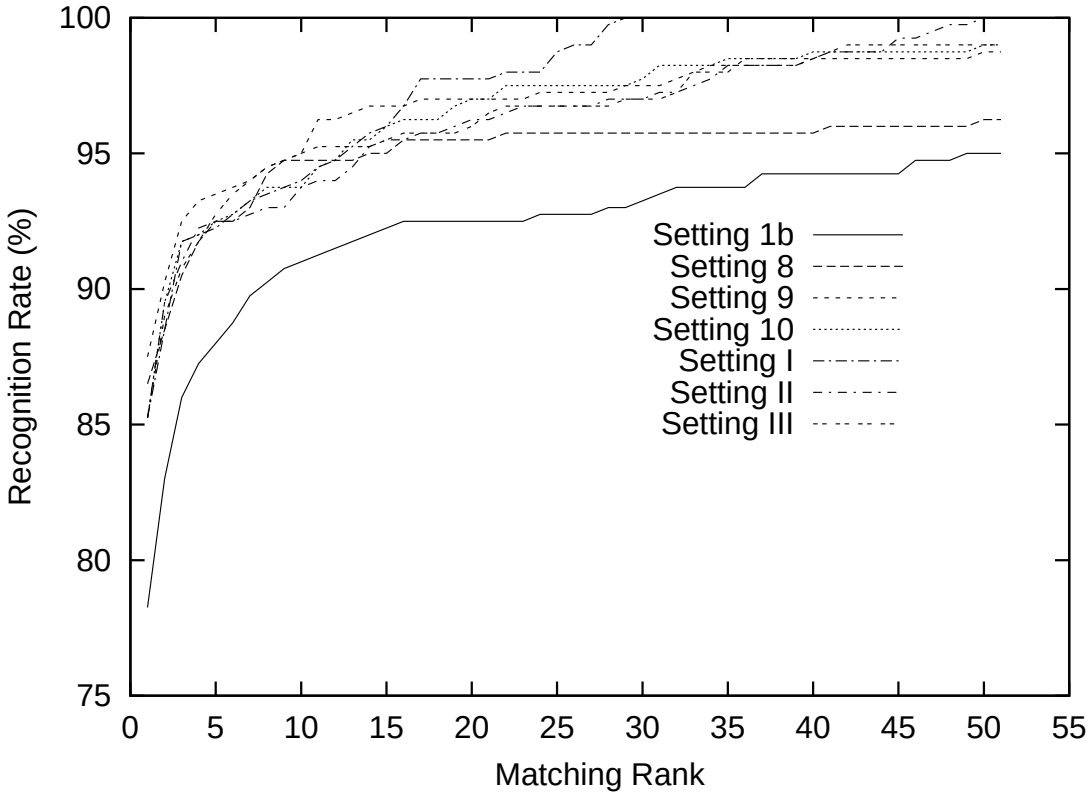


Figure 5.26: Recognition rate versus the rank of a match using Configuration II. The results for the voting scheme using normalised weighted FNDs are compared to those using just weighted FNDs. A matching rank of n represents a correct identity match in the top n matches.

System Setting	Recognition Rate
I	85.3 %
II	85.3 %
III	83.8 %

Table 5.32: Recognition rates for identification experiments using Configuration II and the voting scheme with normalised weighted FNDs. This is the rate at which the top ranking match is identified correctly.

fact that information from the whole face is used. For matching ranks greater than 16 the voting scheme is significantly better than other methods.

We conclude that the standard FND performs better when there are more variations due to pose and illumination, and when we consider larger matching ranks the voting scheme performs better than the standard FND. When faces are more accurately aligned, the weighted FND, with focus on features in the region around the eyes, outperforms the standard FND.

The problem of correcting for pose that results in out-of-plane distortions is not treated in this thesis. This is left for future work. Some possible approaches include the use of face models and the alignment of additional facial features during normalisation. Although we describe and show the inherent invariances of the FND to some out-of-plane distortions, particularly the perspective transformation, we are not able to fully take advantage of it in the experiments using the XM2VTS database. This is due to the fact that the pose differences in the database are more than just mere perspective transformations. The previous chapters note that when faces are not accurately aligned, using fractal codes with slower convergence during the decoding process may give better recognition rates. We perform experiments using a maximum contrast scaling factor α of 0.95, and the results are similar to those described here. Therefore, when datasets like those of Configuration II are used more should be done to make the FND more pose invariant. The task of making the FND more invariant to these types of distortions is left for future work.

5.6 Conclusions

In this chapter we present a complete fully automatic face verification and identification system. The eyes are automatically detected using the system described in Chapter 2, which are then used to normalise the face. The standard FND is modified to use weights and used to perform verification and identification. Manually located eye positions are also used in place of the automatic eye detector. This is performed to isolate the recognition engine from inaccuracies in the eye detector. Both the verification and identification systems show significant improvements as a result of using manually located eye positions. Verification experiments using Configuration I of the XM2VTS database show that our weighted FND method produces results that are comparable to those published in the literature. Results from using Configuration II are not as competitive. We introduce the use of normalised scores, and as a result our verification results improved for both Configuration I and II. Using normalised scores make our system competitive with methods in the literature to which they are compared. To further improve results, we propose an algorithm for calculating client specific thresholds. That is, instead of using a global decision threshold, the decision thresholds are specific for each client. Using client specific thresholds, experiments using the evaluation set of Configuration I show that our method outperformed all methods in the literature to which they are compared. However, our test set results are inferior. We note a problem with asymmetry in the scores obtained using the evaluation and test sets. To demonstrate that it is possible to achieve good results using client specific thresholds, we apply the client specific thresholds algorithm to the test set data. In this case, excellent results are achieved. However, the discovery of a solution to the problem of score asymmetry is left for future work.

The weighted FND is also used for performing facial identification. Better results are obtained by using a different window function to the one used for facial verification. Experiments show that our method performed better than the widely accepted eigenface method. The experiments also show that Configuration I results are better

than those obtained using Configuration II. On comparing the image sets between the two configurations, we conclude that our FND based methods are more sensitive to out-of-plane rotations than other methods in the literature. Nevertheless, our method still performs better than the eigenface method. We also further modify the weighted FND to use a voting scheme. This is shown to give better performance for the identification system at larger matching ranks.

Although our fully automatic verification and identification systems give inferior results compared to the ones using manually located eye positions, they have the advantage of not requiring human intervention. The results in this chapter also demonstrate that face verification and identification are different problems that require different solutions. An algorithm that performs well in verification may not perform well in identification. Facial verification requires different algorithmic specialisations and adaptations to facial identification.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

In this thesis, we present face identification and verification solutions based on fractal image coding. Such solutions require a face detection/localisation as a first step. Therefore, we also present a face localisation system that can locate the two eyes of the face, which then allows the extraction and normalisation of the face for size and in-plane rotations. Our eye detection system utilises a novel representation of the decision function of Support Vector Machines (SVMs) combined with the fast Fourier transform. The decision function of linear and nonlinear SVMs is efficiently represented in terms of what we refer to as ρ and η prototypes. The prototypes can be obtained from any trained SVM. We test the efficiency of this representation by performing identification experiments using the ORL face database. The one-against-all method of classification is used to identify an unknown input face. The SVM for each one-against-all classifier is simplified using our prototypes method and the well-known reduced set method. We then perform identification experiments using the original SVMs, prototype SVMs and reduced set SVMs. The matching scores from the prototype SVMs and reduced set SVMs are compared against the original SVMs, where several statistics are then used to gauge their approximation

accuracy. The results indicate that the approximation accuracy of our prototypes method range from 3 to 175 times better than those obtained using the reduced set method. For the reduced set method to achieve the equivalent approximation accuracy as our method more vectors have to be used, leading to a less efficient representation. Our novel prototypes representation allows us to combine it with the fast Fourier transform to achieve a fast SVM search of an image for potential eye locations. The fast Fourier transform assisted SVM search procedure produces an eye probability map. A range searching algorithm combined with a knowledge base is used to narrow in on the possible eye locations. Two SVMs are created, one for the left and one for the right eye. The SVMs are trained with positive samples representing correct eye locations, and negative samples obtained from randomly sampled non-eye subimages. Accuracy rates of 96.9 % are achieved in locating both eyes correctly. Faces are then extracted and normalised, based on the location of the eyes.

Next, we explore the use of fractal image coding for face recognition. Recognition is performed using a fractal image coding based distance that we called the Fractal Neighbour Distance (FND). Experiments using the FND demonstrate that it has some inherent invariance to translations, scalings, rotations and changes to illumination. In an attempt to quantify the tolerance of the FND to deformations, we perform a set of classification tests using controlled deformations to a set of test images. The deformations used are rotation, scaling, horizontal translation, vertical translation, illumination, and horizontal and vertical perspective transformations. Experiment results indicate that the FND gives reduced recognition error rates across the range of deformations compared to the Euclidean distance. Depending on the type of deformation the improvements in error rate range from 1.7% to 14.1%. We also investigate the effects of various fractal encoding parameters on the recognition rate. In particular, we examine the effects of varying the factors related to the rate of convergence of a fractal decoding process. Under some circumstances, such as when the faces to be compared are not accurately aligned, better recognition rates are obtained with slower convergence. We derive a method of controlling the

convergence rate by changing the contrast scaling factors in a controlled manner, whilst still guaranteeing convergence or eventual convergence. For this we derive a method for calculating the eventual contractivity factor, thus allowing us to control at which iteration the fractal decoding process can be guaranteed to converge. This level of convergence control is novel and has not been possible before. Using this ability to control convergence rates, and at the same time adjusting the limits of the luminance shift factor in the fractal codes, we gain improvements in the recognition rates. A recognition rate of 1.1 % is achieved in identifying faces from the ORL database, and 1.2 % from the Yale database. Of the three range block partitioning methods, using uniform partitioning gives better recognition results because it does not impose a spatial structure like the Quad-tree and HV-partitioning methods. This finding is confirmed in several experiments throughout this thesis.

The Fractal Neighbour Distance (FND) is then combined with an automatic face localisation, extraction and normalisation system to form a complete face recognition system. Furthermore, the FND is modified to use weights. The motivation for this is that previous studies in the literature note discrimination is better in using the region around the eyes. Therefore, the weights are chosen such that focus is around the eye region. Better performance is achieved by using different sets of weights for verification and identification. In experiments using the weighted FND and Configuration I of the XM2VTS database, the verification system is shown to be comparable to current state-of-the-art methods that include elastic graph matching. Configuration II experiments show that our system is sensitive to pose variations that is a result of out-of-plane rotations. The verification experiments using Configuration I give an equal error rate of 5.0 % on the evaluation set. The corresponding false accept error rate and false reject error rate on the test set are 5.0 % and 12.8 % respectively, giving a half-total-error-rate ($HTER$) of 8.9 %. Using Configuration II, an equal error rate of 9.1 % is obtained on the evaluation set. The corresponding false accept error rate and false reject error rate on the test set is 8.6 % and 10.0 % respectively, giving an $HTER$ of 9.3 %. We note the problem where the results obtained on the evaluation and test sets are asymmetric. That is,

the thresholds set using the evaluation data are not optimal on the test data. To alleviate this problem we normalise the matching scores prior to threshold selection and testing. With this modification we achieve an equal error rate of 7.6 % on the evaluation set of Configuration I. The corresponding $HTER$ on the test set is 7.7 %. These results are competitive with some of the current state-of-the-art methods, and is in fact better than the elastic graph matching method from IDIAP, which achieves an equal error rate of 8.0% on the evaluation set and an $HTER$ of 8.3 % on the test set. Again, using normalised scores with the weighted FND, our equal error rate on the evaluation set of Configuration II is 8.3 %, and the corresponding $HTER$ on the test set 8.2 %. Due to sensitivities to pose variations, our results for Configuration II are not as competitive. We then investigate the use of client specific thresholds (CST). Dramatic improvements are obtained as a result. Using un-normalised scores and CSTs, we obtain an $HTER$ of 2.2 % on the evaluation set of Configuration I, which is better than all the methods in the literature against which we compare our results. For Configuration II the $HTER$ is 3.2 %. Results on the test sets are not as competitive, where an $HTER$ of 14.7 % is obtained for Configuration I and 13.4 % for Configuration II. Using normalised scores and CSTs give better test results where an $HTER$ of 9.8 % is obtained for Configuration I and 11.5 % for Configuration II. Here the problem of asymmetry in the evaluation and test set results are evident again. The resolution of this problem is left for future work. To illustrate the optimal performance achievable on the test set, CSTs are applied to the test data. Very competitive results are achieved where an $HTER$ of 3.8 % is obtained for Configuration I and 3.5 % for Configuration II, both using normalised scores. This demonstrates in a practical manner that the weighted FND does in fact produce good class separation.

The XM2VTS database is also used to test our identification system. Various system settings are investigated, which include the use of the standard FND and the weighted FND. The normalised scores version of the weighted FND is also investigated in preliminary experiments. Using the weighted FND with specialised window functions different to the ones used for facial verification, we perform identification

experiments on Configuration I and the system achieves a top rank matching rate of 95.0 %. Using Configuration II the system achieves a top rank matching rate of 87.5 %. In all cases our method still outperforms the widely used and referenced eigenface method. For comparison, using one hundred principal components, the eigenface approach only achieved a top rank matching rate of 85.8 % on Configuration I and 84.0 % on Configuration II. A voting scheme using the normalised version of weighted FNDs is also introduced, achieving significant performance improvements for matching ranks larger than 15.

Our work on facial verification and identification reveal that they are different problems that require different algorithms for optimal performance. For example, our preliminary tests show that the normalised scores version of the weighted FND performed well in facial verification but not identification. The window functions that worked well for identification did not work well for verification. Therefore, facial identification and verification each require their own specialised approaches to achieve optimal performance.

One drawback with our methods is that the fractal decoding process must be carried out for each image in the database, which can be slow if the program code is not optimised. However, we demonstrate that in terms of recognition performance our method is competitive with current state-of-the-art methods, and in some cases performs better than most of them. Our fractal image coding based method is inherently invariant to various types of distortions and illumination changes. To some extent, this inherent invariance can be varied and adapted to different types of images by changing the fractal encoding parameters. Furthermore, new faces can be enrolled into the recognition database without the need for re-training. This is not possible when using methods such as neural networks. Also, our methods are relatively simple to implement. Fractal decoding, as required during recognition, is even simpler to implement than encoding. As such, it can easily be implemented in specialised hardware. Parallel hardware can also be used, since the decoding process for each client in the database can be carried out independently. Furthermore, our method is not based on facial or probabilistic models, so there are no model

parameters to tweak. The only parameters requiring adjustment are those that relate to the fractal encoding process, which directly affect the amount of inherent invariance available. From our investigations and findings we draw up a set of guidelines for selecting the appropriate fractal encoding parameters to achieve better recognition performance.

In summary, we demonstrate that fractal image coding can be applied to the task of human face recognition. Recognition is performed using a fractal image coding based distance. Our method is competitive with, and in some cases better than, current state-of-the-art methods. We show how the method can be integrated into fully automatic face verification and identification systems. We investigate how and why the method can be used for recognition, and from those findings implement changes to further improve the recognition performance. We also demonstrate that the method can still be further improved, but solutions that implement the improvements are left for future work.

6.2 Future Work

At the expense of computation time, our FFT-assisted SVM eye detector could be improved by using multiple vectors to represent the η-prototype, instead of using just one vector. The appropriate simplified decision function representation is just a simple extension of the one vector case. It is likely that the optimisation method for finding the multiple vectors will implement an incremental algorithm. And the optimal coefficients for each vector will most likely be calculated at each incremental step. In addition, the generation of the eye map will need to be modified to incorporate the use of multiple vectors for the η-prototype. The further investigation and implementation of this technique is left for future work. For the SVMs that detect the left and right eyes, more positive and negative training samples could be used. We are aware that some successful SVM based face detection systems, although the results are unpublished, used up to 500,000 training samples. The

samples used have large variations in the lighting conditions. Instead of using raw pixel values as features, Gabor decomposition like the ones described by Smeraldi et al. [SB98, SCB99] could be used as the feature extractor. Gabor decomposition has also been used successfully in elastic graph matching based techniques as a feature extractor.

More complex fractal encoding schemes involving non-rectangular domain and range blocks can be investigated. For example, triangular domain and range blocks could be used. More complex transformations from the domain to range block, which may use transformations other than the affine transformation, should be investigated. The theorems we have presented relating to the control of the eventual convergence of a fractal decoding process also apply to transformations that are non-affine. The method we present for calculating the eventual contractivity factor used a tree-structure. Currently, we can only calculate the tree structure up to six or seven levels before the number of computations required becomes excessive. However, there are redundancies in the tree that can be eliminated, and thus reduce the number of required computations.

A more optimal method of choosing the contrast scaling factor α to recognition rates should be found. Ideally, there should be a minimal effect on the image quality as a result of the modification, typically by reducing the number of changes to α. The effects of the contractivity and eventual contractivity factors on image quality is left for future investigations.

Facial verification and identification results from using Configuration I and Configuration II of the Lausanne Protocol for the XM2VTS face database reveal that our system has a weakness dealing with out-of-plane rotations. For future work, it is desirable to make the FND less sensitive to out-of-plane rotations of the face. This might be possible by using 3D models of the face and aligning the features before performing recognition. Alternatively, instead of a full 3D model of the face, an approximate model can be used such as the one described in Lam and Yan [LY98]. They use a cylindrical model for the head. By detecting the location of the nose

we can approximate the amount of out-of-plane sideways rotation of the head. An approximate gaze angle can be calculated and used to normalise the face. We also suggest that future investigations be carried out on score normalisations. A simple scheme of score normalisation is presented in this thesis, and it is effective in alleviating some of the problems associated with threshold asymmetry in the evaluation and test results. A more sophisticated normalisation approach, possibly applied in steps prior to score generation, may give better results. Particularly for facial verification, client specific thresholding seems to give excellent results. Results obtained using the evaluation sets are very competitive. However, the test results are again skewed by imperfect threshold selection. In spite of this, we demonstrate that excellent results on the test set is possible by applying client specific thresholding to the test set. A sophisticated normalisation scheme, or a different approach to calculating the client specific thresholds, may help in achieving a balanced result on both the evaluation and test sets. Furthermore, using more client evaluation samples should help in determining the optimal thresholds more accurately.

6.3 Other Applications

The FND described in this thesis can also be applied to areas other than face recognition. It can be used to perform object recognition in general. Face recognition is the chosen application because of its difficulty, and a large pool of resources relating to training data and past results are available. This allows us to compare our method with a range of other methods.

Other potential uses of the FND include printed and handwritten character recognition. Practical applications of this include car number plate recognition, recognising characters in scanned documents, and recognising addresses written on an envelope. It can also be applied to other forms of biometrics such as signature verification and fingerprint recognition. Certainly, for the method to be successful some amount of

customisation and specialisation of the FND is required, along the lines and spirit of those we present for facial recognition in this thesis.

Appendix A

Operators of L

The operator $\mathbf{F}_n : \Re^M \rightarrow \Re^{M_d}$ fetches a domain region from the input image for the nth range-region transformation, and is defined as shown in Equation (A.1).

$$\mathbf{F}_n = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \vdots \\ \vdots & 0 & \ddots & 0 \\ 0 & \cdots & 0 & 1 \end{bmatrix} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \vdots \\ \vdots & 0 & \ddots & 0 \\ 0 & \cdots & 0 & 1 \end{bmatrix} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \vdots \\ \vdots & 0 & \ddots & 0 \\ 0 & \cdots & 0 & 1 \end{bmatrix} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \vdots \\ \vdots & 0 & \ddots & 0 \\ 0 & \cdots & 0 & 1 \end{bmatrix} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \vdots \\ \vdots & 0 & \ddots & 0 \\ 0 & \cdots & 0 & 1 \end{bmatrix} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \tag{A.1}$$

This operator consists of unity sub-matrices, each of which fetches a domain sub-region. These sub-matrices can only appear at an integral number of sub-matrix positions horizontally, and each row of the full matrix has only a single '1'. Therefore, partial vertical overlaps of sub-matrices cannot occur, but full overlaps such as that between the two identity submatrices on lines one and five are permitted. This corresponds to the situation where the same domain region is used to transform to different range regions. The input vector has M elements, and the fetched domain

region has M_d elements. Domain regions are not restricted to square or rectangular blocks. Blocks of zero matrices are represented by $\mathbf{0}$, and each can be of any size required.

The next operator decimates a domain vector to reduce it to the size of a range region. That is, $\mathbf{D}_{n,u,t} : \Re^{M_d} \rightarrow \Re^{M_r}$, where n is the transformation number, M_r is the number of elements in the range region, and decimation occurs by a factor u and t in the vertical and horizontal directions respectively. This operator is defined as:

$$\mathbf{D}_{n,u,t} = \begin{bmatrix} \mathbf{w}^T & & & \\ & \mathbf{w}^T & & \\ & & \ddots & \\ & & & \mathbf{w}^T \end{bmatrix} \tag{A.2}$$

where $\mathbf{w} = \begin{bmatrix} w_1 & \cdots & w_{ut} \end{bmatrix}^T$ are weights. For example, a possible set of weights for decimation by sub-sampling is $w_1 = 1$ and $w_i = 0$ for $2 \leq i \leq ut$. Decimation by averaging has weights $w_i = \frac{1}{ut}$ for $1 \leq i \leq ut$.

The decimated domain vector is then multiplied by a contrast scaling factor $\alpha^{(n)}$ and then placed into the appropriate coordinates in the output image vector. The put operator, $\mathbf{P}_n : \Re^{M_r} \rightarrow \Re^{M}$, performs this task:

$$\mathbf{P}_n = \left[\begin{array}{c|cccc|c} & 1 & 0 & & & \\ & 0 & 1 & & & \\ \mathbf{0} & & & 1 & & \mathbf{0} \\ & & & & \ddots & 0 \\ & & & & 0 & 1 \end{array}\right]^T \tag{A.3}$$

Appendix B

Proof of Equation (4.10)

Here the l_2 norm is used, so the contractivity factor for the operator described by Equation (4.3) is given by the square root of the largest eigenvalue of $\mathbf{L}^T\mathbf{L}$ [Lun95]. The proof of Equation (4.10) is based on the framework proposed by Lundheim [Lun95].

The proof begins by examining f-regions. The illustration in Figure B.1 shows an 8 by 8 pixel domain block mapping to a 4 by 4 pixel range block with a decimation factor of two. Each sub-range block is the result of the transformation from a 2 by 2 pixel sub-domain block. These sub-range blocks, under the definitions given above, are also known as f-range regions. Each 2 by 2 pixel sub-domain block transforms to an f-range region, therefore they are referred to as f-domain regions.

The transformations to each f-range region are independent of each other, and can thus be considered as separate transformations. Consequently, the locations of the f-domain regions can be located anywhere. In fact, there is no direct dependence between f-sub-domain regions, so the f-domain region can even consist of scattered or overlapped sub-domain regions. As long as the decimation factor in the horizontal and vertical directions are identical for each transformation to an f-range region we can treat these mappings independently, thus domain and range regions can be of any shape. To use Lundheim's framework with f-domain and f-range regions,

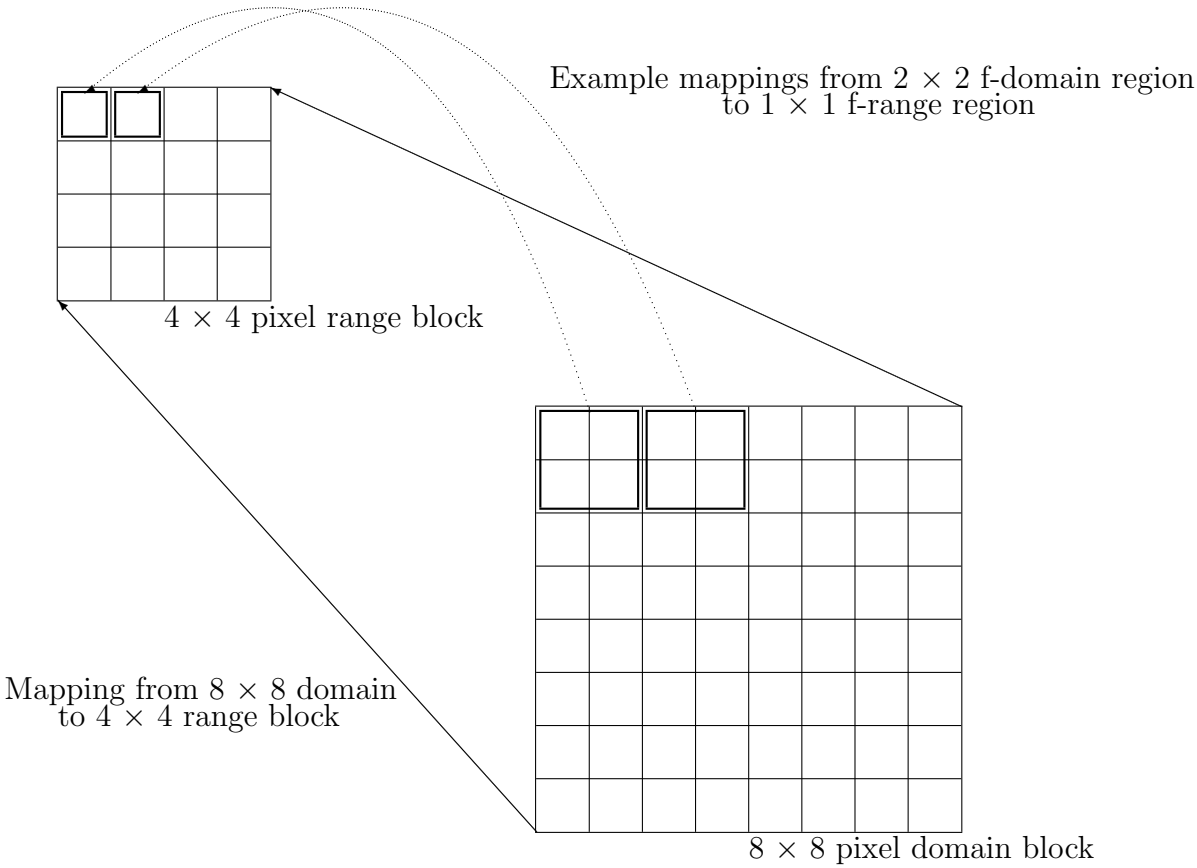


Figure B.1: Illustration of a typical transformation from an 8×8 pixel domain block to a 4×4 pixel range block. The constituent f-domain region to f-range region mapping is also shown, which demonstrates that each f-region mappings can be considered independently of each other.

we need to further subdivide f-range regions into f-sub-range regions, and f-sub-domain regions into f-sub-sub-domain regions (shown in Figure 4.2). Now we have f-sub-domain regions mapping to f-sub-range regions.

With f-sub-range regions, the linear term $\mathbf{L}$ in Equation (4.3) can be expressed as a composite of blocks containing $\alpha^{(i)}\mathbf{w}^T$:

$$\mathbf{L} = \left[\begin{array}{c|c|c|c|c} \alpha^{(1)}\mathbf{w}^T & & & & \\ & & \alpha^{(1)}\mathbf{w}^T & & \\ & & \vdots & & \\ & & & \alpha^{(1)}\mathbf{w}^T & \\ \hline \vdots & & \vdots & & \vdots \\ \hline & \alpha^{(N)}\mathbf{w}^T & & & \\ & \alpha^{(N)}\mathbf{w}^T & & & \\ & & \vdots & & \\ & & & & \alpha^{(N)}\mathbf{w}^T \end{array}\right] \tag{B.1}$$

where $\alpha^{(i)}$ is the contrast scaling factor for the ith f-range region. Each block in $\mathbf{L}$ has the same height and width, where each is equal to the length of a row-wise lexicographically vectorized f-range region. The empty regions in $\mathbf{L}$ are zeroes. Furthermore, every row of the matrix has only one $\alpha^{(1)}\mathbf{w}^T$, but each column can have a multiple of them. Note that each row of a block corresponds to a fetch-decimate-and-multiply-by-α of an f-sub-domain region and placing it in the f-sub-range region corresponding to that row. Permutations of the blocks in $\mathbf{L}$ does not affect the analysis that follows. Therefore, domain and range regions can be of any shape as long as each related domain and range region possess the same shape. From Equation (B.1) the following expression can be derived:

$$\mathbf{L}^T\mathbf{L} = \begin{bmatrix} \xi_1\mathbf{w}\mathbf{w}^T & & & \mathbf{0} \\ & \xi_2\mathbf{w}\mathbf{w}^T & & \\ & & \ddots & \\ \mathbf{0} & & & \xi_N\mathbf{w}\mathbf{w}^T \end{bmatrix} \tag{B.2}$$

where

$$\xi_j = \sum_{i=1}^{N} \rho_{i,j}(\alpha^{(i)})^2 \tag{B.3}$$

where $\rho_{i,j} \geq 0$ is an integer representing the number of times the transformation for the ith f-range region utilizes the jth f-sub-domain region, and N is the total number of f-range regions (pixels). The square matrix $\mathbf{L}^T\mathbf{L}$ is block diagonal. Each block is a scaled version of $\mathbf{w}\mathbf{w}^T$. The eigenvalues of $\mathbf{L}^T\mathbf{L}$ are given by the determinant $|\mathbf{L}^T\mathbf{L} - \lambda\mathbf{I}|$. This determinant can also be calculated by the multiplication of the determinants of the block matrices, that is $|\mathbf{L}^T\mathbf{L}-\lambda\mathbf{I}| = \prod_{j=1}^{N} |\xi_j\mathbf{w}\mathbf{w}^T-\lambda\mathbf{I}|$. It is not difficult to find the determinants $|\xi_j\mathbf{w}\mathbf{w}^T - \lambda\mathbf{I}|$ for all j (see Appendix E). We can then show that $|\mathbf{L}^T\mathbf{L} - \lambda\mathbf{I}|$ possibly have non-zero eigenvalues of $\lambda = \frac{\xi_1}{u^2}, \frac{\xi_2}{u^2}, \ldots, \frac{\xi_N}{u^2}$. Therefore, the contractivity factor is given by:

$$s = \sqrt{\max_{1\leq j\leq N} \frac{\xi_j}{u^2}} = \sqrt{\frac{1}{u^2} \max_{1\leq j\leq N} \sum_{i=1}^{N} \rho_{i,j}[\alpha^{(i)}]^2} \tag{B.4}$$

It is likely that $\xi_i = 0$ for some i because some f-sub-domain regions may remain unused. However, this does not affect Equation (B.4) and Equation (4.10) because we are only interested in the square root of the largest eigenvalue.

The true value of an f-range region is now obtained by applying the decimation function using the values in its sub-range regions. This subdivision of an f-region has corresponded to an increase of image resolution by the decimation factor, so to obtain an image of the original resolution the decimation function must be applied for each pixel, using all the sub-values in that pixel. It has been shown that decimation by discrete spatial contraction has contractivity one [Jac89]. It is also straightforward to show that decimation by sub-sampling also has contractivity one. Therefore, the restoration of the image to its original resolution does not affect the contractivity factor given by Equation (B.4) and Equation (4.10).

Although intuition suggests that $\rho_{i,j}$ is either one or zero, it is in fact an integer value representing the number of times the jth f-sub-domain region transforms to the ith f-range region. This is because from the consideration of independence between f-sub-domain regions an f-range region can use any f-sub-domain more than once. The limiting case occurs when all f-sub-domain regions coincide, which when combined with the decimation operation emulates the case when sub-sampling is used for decimation. This gives rise to an effect we refer to as *apparent decimation*,

which is the decimation that we observe in a domain to range transformation. This occurs only when averaging is chosen as the actual decimation operation, then the choice of the locations of f-sub-domain regions can switch the apparent decimation between sub-sampling and averaging. Note that if the actual decimation used is sub-sampling, then the choice of f-sub-domain regions has no effect on the apparent decimation, which remains as sub-sampling.

The results above apply only for the l_2 norm. However, all norms on finite-dimensional vectors spaces are equivalent. Therefore, the results can be used as contractivity tests for any norm on $\Re^N$.

Despite the preceding discussion about the independence between f-sub-domain regions, some geometrical relationship is required if self-similarity is to be exploited for coding or compression purposes. However, this important relationship can be disregarded for the calculation of the contractivity factor.

Appendix C

Proof of Equation (4.11)

The increase in resolution as described in the proof of Theorem 4.1 by sub-division now occurs by partitioning the fundamental region into u and t regions vertically and horizontally respectively. In total, ut f-sub-domain regions transform to each f-range region, instead of u^2. Equation (4.11) follows simply from replacing occurrences of u^2 with ut.

Appendix D

Proof of Equation (4.12)

The first step in the proof is to subdivide each f-region into u^l and t^l vertical and horizontal f-sub-regions respectively. It can be shown that:

$$R_{j,r} = \mathbf{D}_{i(j,r),u^l,t^l} D_{i(j,r)} \prod_{b=1}^{l} \alpha_{j,r,b} + \gamma^*_{j,r,l} \tag{D.1}$$

where $R_{j,r}$ is the value of the rth f-sub-range region in the jth f-range region, $i(j,r)$ is a two variable function returning the index of the f-sub-domain region that transforms to the corresponding $R_{j,r}$ f-sub-range region, $D_{i(j,r)}$ is the $i(j,r)$th f-sub-domain region, $\mathbf{D}_{i(j,r),u^l,t^l}$ is the decimation operator that decimates an f-sub-domain region and returns a single value since there are exactly $(ut)^l$ elements in $D_{i(j,r)}$, $\alpha_{j,r,b}$ is the contrast scaling factor of the intermediate mapping at the bth iteration in the overall transformation from the $D_{i(j,r)}$ f-sub-domain region to the $R_{j,r}$ f-sub-range region. Furthermore, $\gamma^*_{j,r,l} = \gamma_{j,r,1}\alpha_{j,r,2}\ldots\alpha_{j,r,l} + \gamma_{j,r,2}\alpha_{j,r,3}\ldots\alpha_{j,r,l} + \ldots + \gamma_{j,r,l-1}\alpha_{j,r,l} + \gamma_{j,r,l}$, where $\gamma_{j,r,c}$, for $1 \leq c \leq l$, is the illuminance shift factor of the intermediate mapping at the cth iteration in the overall transformation from the $D_{i(j,r)}$ f-sub-domain region to the $R_{j,r}$ f-sub-range region.

Equation (D.1) suggests that it is possible to reduce l iterations of a fractal code into a single iteration consisting of only f-sub-domain to f-sub-range transformations, where the f-sub-range regions are generated by decimating f-range regions into $u^l \times t^l$ blocks. The contrast scaling factor of this single transformation is the

multiplication of the contrast scaling factors of all intermediate mappings during l iterations originating from an f-sub-domain region and ending at the f-sub-range region.

With this observation an expression for the eventual contractivity factor can be derived using the method described in Appendix B. A slight difference is that the factor $\rho'_{i,j}$ can only take a value of one or zero, rather than an integer valued $\rho_{i,j}$ as in Equation (4.10). The reason is that now we are considering transformations to an f-sub-range region and not an f-range region. An f-sub-range region is single valued and is not subdivided further, and its value is given by the transformation from a single f-sub-domain region. With the assistance of Equation (D.1) and Appendix B, it is relatively straightforward to derive Equation (4.12).

The results above apply only for the l_2 norm. However, all norms on finite-dimensional vectors spaces are equivalent. Therefore, the results can be used as eventual contractivity tests for any norm on $\Re^N$.

Appendix E

Calculating the determinant $|\xi_j \mathbf{w}\mathbf{w}^T - \lambda\mathbf{I}|$

The method to calculate $|\xi_j \mathbf{w}\mathbf{w}^T - \lambda\mathbf{I}|$ is described by Lundheim [Lun95]. It is repeated here for completeness.

An expression is required for the r-th order determinant with the following structure:

$$D_r(a) = \begin{vmatrix} a-\lambda & a & a & \cdots & a \\ a & a-\lambda & a & \cdots & a \\ a & a & \ddots & & a \\ \vdots & & & & \vdots \\ a & a & a & \cdots & a-\lambda \end{vmatrix} \tag{E.1}$$

The second row is subtracted from the first one, and then expressing the determinant via the first row and its determinants of $(r-1) \times (r-1)$ sub-matrices we have,

$$D_r(a) = -\lambda D_{r-1}(a) - \lambda \begin{vmatrix} a & a & a & \cdots & a \\ a & a-\lambda & a & \cdots & a \\ a & a & \ddots & & a \\ \vdots & & & & \vdots \\ a & a & a & \cdots & a-\lambda \end{vmatrix} \tag{E.2}$$

The first row can then be subtracted from all the other rows to give a triangular

matrix whose determinant is $a(-\lambda)^{r-2}$. Therefore, the following recurrence relation is obtained,

$$D_r(a) = a\lambda^{r-1}(-1)^{r+1} - \lambda D_{r-1}(a) \tag{E.3}$$

Then defining $D_0(a) = 1$, and by induction, Equation (E.3) can be expressed as,

$$D_r(a) = (-1)^r \lambda^{r-1}(\lambda - ra) \tag{E.4}$$

The determinant $|\xi_j \mathbf{w}\mathbf{w}^T - \lambda\mathbf{I}|$ has the following form,

$$|\xi_j \mathbf{w}\mathbf{w}^T - \lambda\mathbf{I}| = \begin{vmatrix} \xi_j \left(\frac{1}{u^2}\right)^2 - \lambda & \xi_j \left(\frac{1}{u^2}\right)^2 & \xi_j \left(\frac{1}{u^2}\right)^2 & \cdots & \xi_j \left(\frac{1}{u^2}\right)^2 \\ \xi_j \left(\frac{1}{u^2}\right)^2 & \xi_j \left(\frac{1}{u^2}\right)^2 - \lambda & \xi_j \left(\frac{1}{u^2}\right)^2 & \cdots & \xi_j \left(\frac{1}{u^2}\right)^2 \\ \xi_j \left(\frac{1}{u^2}\right)^2 & \xi_j \left(\frac{1}{u^2}\right)^2 & \ddots & & \xi_j \left(\frac{1}{u^2}\right)^2 \\ \vdots & & & & \vdots \\ \xi_j \left(\frac{1}{u^2}\right)^2 & \xi_j \left(\frac{1}{u^2}\right)^2 & \xi_j \left(\frac{1}{u^2}\right)^2 & \cdots & \xi_j \left(\frac{1}{u^2}\right)^2 - \lambda \end{vmatrix} \tag{E.5}$$

Substituting $a = \xi_j \left(\frac{1}{u^2}\right)^2$ and $r = u^2$ in Equation (E.4) we obtain,

$$|\xi_j \mathbf{w}\mathbf{w}^T - \lambda\mathbf{I}| = (-1)^{u^2} \lambda^{u^2-1}(\lambda - \xi_j \frac{1}{u^2}) \tag{E.6}$$

Therefore, Equation (E.6) has a non-zero eigenvalue of $\lambda = \frac{\xi_j}{u^2}$ only if ξ_j is non-zero.

Appendix F

Face Identification User Interface

The user interface of our fully automatic face identification system is shown in Figure F.1. The system is developed using Microsoft Visual C++, running on the Windows 2000/XP platform. It uses the automatic eye detector described in Chapter 2 for face detection. Face identification is based on the FND. The system also works with multiple faces. On an Athlon XP 2000+ based PC with 1 Gigabyte of DDR-RAM, the system achieves an operational speed of about 10 frames per second.

Figure F.1: User interface of the automatic face identifcation system. Detected eye locations are labelled using the '+' symbol. Extracted faces are shown in the row labelled 'Detected Faces'. Best matching faces in the database are shown in the row labelled 'Matching Suspects'.

Bibliography

[AAM03] A. Ansari and M. Abdel-Mottaleb. 3-d face modeling using two views and a generic face model with application to 3-d face recognition. *Proc. IEEE Conf. Advanced Video and Signal Based Surveillance*, pages 37–44, 2003.

[ALX01] H. Ai, L. Liang, and G. Xu. Face detection based on template matching and support vector machines. *Proc. IEEE Int'l Conf. Image Processing*, 1:1006–1009, 2001.

[AMR02] G. D. Abowd, E. D. Mynatt, and T. Rodden. The human experience. *IEEE Pervasive Computing*, 1(1):48–57, January-March 2002.

[Bal81] D. H. Ballard. Generalizing the hough transform to detect arbitrary shapes. *Pattern Recognition*, 3:110–122, 1981.

[Bar93] M. F. Barnsley. *Fractals everywhere.* Academic Press, New York, 2nd edition, 1993.

[BH93] M. F. Barnsley and L. P. Hurd. *Fractal image compression.* AK Peters Ltd., 1993.

[BHK97] P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman. Eigenfaces vs. fisherfaces: recognition using class specific linear projection. *IEEE Trans. on PAMI*, 19(7), 1997.

[BJ88] M. F. Barnsley and A. E. Jacquin. Application of recurrent iterated function systems to images. *Proc. SPIE 1001*, pages 122–131, 1988.

[BK98] P. Belhumeur and D. Kriegman. What is the set of images of an object under all possible illumination conditions. *Int. J. Computer Vision*, 28(3):245–260, 1998.

[BKP01] L. Buciu, C. Kotropoulos, and I. Pitas. Combining support vector machines for accurate face detection. *Proc. IEEE Int'l Conf. Image Processing*, 1:1054–1057, 2001.

[BLM90] J. Buhmann, M. Lades, and C.v.d. Malsburg. Size and distortion invariant object recognition by hierarchical graph matching. *Proceedings of the International Joint Conference on Neural Networks*, pages 411–416, 1990.

[BMM01] S. Bengio, J. Mariéthoz, and S. Marcel. Evaluation of biometric technology on xm2vts. *European Project BANCA Deliverable D71*, IDIAP-RR 01-21, 2001.

[BP92] R. Brunelli and T. Poggio. Face recognition through geometrical features. *Proc. ECCV 588*, pages 792–800, 1992.

[BP93] R. Brunelli and T. Poggio. Face recognition: features versus templates. *IEEE Trans. on PAMI*, 15(10):1042–1052, October 1993.

[BS88] M. F. Barnsley and A. D. Sloan. A better way to compress images. *Byte Magazine*, 13(1):215–223, January 1988.

[Bur96] C. J. C. Burges. Simplified support vector decision rules. *Proc. Intl. Conf. Machine Learning*, pages 71–77, 1996.

[BYAM99] S. Ben-Yacoub, Y. Abdeljaoued, and E. Mayoraz. Fusion of face and speech data for person identity verification. *IEEE Trans. Neural Networks*, 10(5):1065–1074, 1999.

[CB94] J. L. Crowley and J. M. Bedrune. Integration and control of reactive visual processes. *Proc. Third European Conf. Computer Vision*, 2:47–58, 1994.

[CB97] J. L. Crowley and F. Berard. Multi-modal tracking of faces for video communications. *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pages 640–645, 1997.

[CDCT+01] R. Cowie, E. Douglas-Cowie, N. Tsapatsoulis, G. Votsis, S. Kollias, W. Fellenz, and J. G. Taylor. Emotion recognition in human-computer interaction. *IEEE Signal Processing Magazine*, 18(1):32–80, January 2001.

[CG03] G. Cui and W. Gao. Svms for few examples-based face recognition. *Proc. IEEE Intl. Conf. Acoustics, Speech, and Signal Processing*, 2:381–384, 2003.

[CK02] S. Chandran and S. Kar. Retrieving faces by the pifs fractal code. *Proc. IEEE Workshop on Applications of Computer Vision*, pages 8–12, 2002.

[CN98] D. Chai and K. N. Ngan. Locating facial region of a head-and-shoulders color image. *Proc. Third Int. Conf. Automatic Face and Gesture Recognition*, pages 124–129, 1998.

[Com94] P. Comon. Independent component analysis: A new concept? *Signal Processing*, 36(3):287–314, 1994.

[CR02] S. Cass and M. J. Riezenmen. Improving security, preserving privacy. *IEEE Spectrum*, 39(1):44–49, January 2002.

[CST00] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*. Cambridge University Press, 2000.

[CTB92] I. Craw, D. Tock, and A. Bennet. Finding face features. *Proc. European Conf. Computer Vision*, pages 92–96, 1992.

[CWY95] Q. Chen, H. Wu, and M. Yachida. Face detection by fuzzy pattern matching. *International Conference on Computer Vision*, pages 591–596, 1995.

[Dau90] I. Daubechies. The wavelet transform, time-frequency localization and signal analysis. *IEEE Trans. Information Theory*, 36:961–1005, 1990.

[Dav98] Geoffrey M. Davis. A wavelet-based analysis of fractal image compression. *IEEE Trans. on Image Processing*, 7(2):141–154, February 1998.

[DN96] Y. Dai and Y. Nakano. Face-texture model based on sgld and its application in face detection in a color scene. *Pattern Recognition*, 29(6):1007–1017, 1996.

[DR93] M. J. Donahue and S. I. Rokhlin. On the use of level curves in image analysis. *Image Understanding*, 57:185–203, 1993.

[EC97] K. Etemad and R. Chellappa. Discriminant analysis for recognition of human face images. *Journal of the Optical Society of America A*, 14(8):1724–1733, August 1997.

[EKCS01a] H. Ebrahimpour-Komleh, V. Chandran, and S. Sridharan. Face recognition using fractal codes. *Proc. IEEE Int'l Conf. Image Processing*, 3:58–61, 2001.

[EKCS01b] H. Ebrahimpour-Komleh, V. Chandran, and S. Sridharan. Robustness to expression variations in fractal-based face recognition. *IEEE Int'l Symp. Signal Processing and its Applications*, pages 359–362, 2001.

[EMR00] S. Eickeler, S. Müller, and G. Rigoll. Recognition of jpeg compressed face images based on statistical methods. *Image and Vision Computing Journal*, 18(4):279–287, 2000.

[EWLT02] M. J. Er, S. Wu, J. Lu, and H. L. Toh. Face recognition with radial basis function (rbf) neural networks. *IEEE Trans. Neural Networks*, 13(3):697–710, 2002.

[FBVC01] R. Féraud, O. J. Bernier, J. Viallet, and M. Collobert. A fast and accurate face detector based on neural networks. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 23(1):42–53, January 2001.

[FD00] R. W. Frischholz and U. Dieckmann. Bioid: A multimodal biometric identification system. *IEEE Computer*, pages 64–68, February 2000.

[Fis36] R. A. Fisher. The use of multiple measures in taxonomic problems. *Ann. Eugenics*, 7:179–188, 1936.

[Fis92] Y. Fisher. Fractal image compression. *Siggraph Course Notes, The San Diego Super Computer Center, University of California*, 1992.

[Fis95a] Y. Fisher, editor. *Fractal image compression: theory and application.* Springer-Verlag Inc., New York, 1995.

[Fis95b] Y. Fisher. Fractal image compression with quadtrees. In Y. Fisher, editor, *Fractal Image Compression: Theory and Application*, chapter 3, pages 55–77. Springer-Verlag Inc., 1995.

[GA97] D. B. Graham and N. M. Allinson. Face recognition using virtual parametric eigenspace signatures. *IEE Conf. Publication (Part 1)*, 443:106–110, 1997.

[GAH94a] M. Gharavi-Alkhansari and T. S. Huang. Fractal-based techniques for a generalized image coding method. *Proc. IEEE Int. Conf. Image Processing*, 3:122–126, 1994.

[GAH94b] M. Gharavi-Alkhansari and T. S. Huang. Generalized image coding using fractal-based methods. *Proc. Int. Picture Coding Symp.*, pages 440–443, 1994.

[GAH96] M. Gharavi-Alkhansari and T. S. Huang. Fractal image coding using rate-distortion optimized matching pursuit. *Proc. SPIE: Vis. Commun. Image Process.*, 2727:1386–1393, 1996.

[GBA+03] B. K. Gunturk, A. U. Batur, Y. Altunbasak, M. H. Hayes III, and R. M. Mersereau. Eigenface-domain super-resolution for face recognition. *IEEE Trans. Image Processing*, 12(5):597–606, 2003.

[GBK01] A. S. Georghiades, P. N. Belhumeur, and D. J. Kriegman. From few to many: Illumination cone models for face recognition under variable lighting and pose. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 23(6):643–660, 2001.

[GHF03] Y. Gao, S. C. Hui, and A. C. M. Fong. A multiview facial analysis technique for identity authentication. *Pervasive Computing*, 2(1):38–45, 2003.

[GL02] Y. Gao and M. K. H. Leung. Face recognition using line edge map. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 24(6):764–779, June 2002.

[HFA03] J. Haddadnia, K. Faez, and M. Ahmadi. A fuzzy hybrid learning algorithm for radial basis function neural network with application in human face recognition. *Pattern Recognition*, 36:1187–1202, 2003.

[HFE00] A. Haro, M. Flickner, and I. Essa. Detecting and tracking eyes by using their physiological properties, dynamics, and appearance. *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 1:163–168, 2000.

[HH94] B. Hurtgen and T. Hain. On the convergence of fractal transforms. *Proc. IEEE ICASSP*, 5:561–564, 1994.

[HHC+02] C. Havran, L. Hupet, J. Czyz, J. Lee, L. Vandendorpe, and M. Verleysen. Independent component analysis for face authentication.

Proc. Knowledge-Based Intelligent Information and Engineering Systems, pages 1207–1211, 2002.

[Hil90] F. S. Hill. *Computer graphics.* Macmillan Publishing Company, New York, 1990.

[HKO01] A. Hyvärinen, J. Karhunen, and E. Oja. *Independent Component Analysis.* Wiley-Interscience, 2001.

[HL01] E. Hjelmås and B. K. Low. Face detection: a survey. *Computer Vision and Image Understanding*, 83:236–274, 2001.

[HW99] E. Hjelmås and J. Wroldsen. Recognizing faces from the eyes only. *Proc. 11th Scandinavian Conf. Image Analysis*, 1999.

[IRY96] N. Intrator, D. Reisfeld, and Y. Yeshurun. Face recognition using a hybrid supervised/unsupervised neural network. *Pattern Recognition Letters*, 17:67–76, 1996.

[Jac89] A. E. Jacquin. *A fractal theory of iterated Markov operators with applications to digital image coding.* PhD thesis, Georgia Institute of Technology, 1989.

[Jac90] A. E. Jacquin. A novel fractal block coding technique for digital images. *IEEE ICASSP*, 4:2225–2228, 1990.

[Jac92] A. E. Jacquin. Image coding based on a fractal theory of iterated contractive image transformations. *IEEE Trans. on Image Processing*, 1(1):18–30, January 1992.

[Jac93] A. E. Jacquin. Fractal image coding: a review. *Proceedings of the IEEE*, 81(10):1451–1465, October 1993.

[Jac94] A. E. Jacquin. An introduction to fractals and their applications in electrical engineering. *Journal of the Franklin Institute*, 331B(6):659–680, 1994.

[JKF01] O. Jesorsky, K. J. Kirchberg, and R. W. Frischholz. Robust face detection using the hausdorff distance. *Proc. Third Int. Conf. Audio and Video-based Biometric Person Authentication*, pages 90–95, June 2001.

[JKLM99] K. Jonsson, J. Kittler, Y. P. Li, and J. Matas. Support vector machines for face authentication. *British Machine Vision Conference*, pages 543–553, 1999.

[JM96] P. Juell and R. Marsh. A hierarchical neural network for human face detection. *Pattern Recognition*, 29(5):781–787, 1996.

[Jol86] I. T. Jolliffe. *Principal Component Analysis*. New York: Springer-Verlag, 1986.

[JZY03] X. Jing, D. Zhang, and J. Yang. Face recognition based on a group decision-making combination approach. *Pattern Recognition*, 36(7):1675–1678, 2003.

[Kan77] T. Kanade. *Computer recognition of human faces*. Birkhauser, Basel and Stuttgart, 1977.

[Kap99] L. M. Kaplan. Extended fractal analysis for texture classification and segmentation. *IEEE Trans. on Image Processing*, 8:1572–1585, 1999.

[KGV83] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

[KHS97] A. Z. Kouzani, F. He, and K. Sammut. Fractal face representation and recognition. *IEEE Int'l Conference on Systems, Man and Cybernetics*, 2:1609–1613, 1997.

[KHS99] A. Z. Kouzani, F. He, and K. Sammut. Face image matching using fractal dimension. *Proc. IEEE Int'l Conf. Image Processing*, 3:642–646, 1999.

[KHS00] A. Z. Kouzani, F. He, and K. Sammut. Towards invariant face recognition. *Information Sciences*, 123:75–101, 2000.

[KJK02] K. I. Kim, K. Jung, and H. J. Kim. Face recognition using kernel principal component analysis. *IEEE Signal Processing Letters*, 9(2):40–42, 2002.

[KKL03] M. Kim, D. Kim, and S. Lee. Face recognition using the embedded hmm with second-order block-specific observations. *Pattern Recognition, Article in Press (available on-line)*, 2003.

[KOG01] D. Keren, M. Osadchy, and C. Gotsman. Antifaces: A novel, fast method for image detection. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 23(7):747–761, July 2001.

[Koh90] T. Kohonen. The self-organizing map. *Proc. IEEE*, 78(9):1464–1480, 1990.

[Kom95] John Kominek. Convergence of fractal encoded images. *Proceedings of the Data Compression Conference*, pages 242–251, March 1995.

[KP97] C. Kotropoulos and I. Pitas. Rule-based face detection in frontal views. *Proc. Int. Conf. Acoustics, Speech and Signal Processing*, 4:2537–2540, 1997.

[Kro02] K. L. Kroeker. Graphics and security: Exploring visual biometrics. *IEEE Computer Graphics and Applications*, 22(4):16–21, 2002.

[KS90] M. Kirby and L. Sirovich. Application of the karhunen-loeve procedure for the characterization of human faces. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 12(1):103–108, 1990.

[KTP00] C. Kotropoulos, A. Tefas, and I. Pitas. Morphological elastic graph matching applied to frontal face authentication under well-controlled and real conditions. *Pattern Recognition*, 33(12):1935–1947, 2000.

[KWT87] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *Int. J. Computer Vision*, 1:321–331, 1987.

[KY99] T. Kondo and H. Yan. Automatic human face detection and recognition under non-uniform illumination. *Pattern Recognition*, 32:1707–1718, 1999.

[LBP95] T. Leung, M. Burl, and P. Perona. Finding faces in cluttered scenes using labeled random graph matching. *Int. Conf. Computer Vision*, pages 637–644, 1995.

[LC03] X. Liu and T. Chen. Video-based face recognition using adaptive hidden markov models. *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 1:340–345, 2003.

[LCT03] X. Liu, T. Chen, and S. M. Thornton. Eigenspace updating for non-stationary process and its application to face recognition. *Pattern Recognition*, 36(9):1945–1959, 2003.

[LGTB97] S. Lawrence, C. L. Giles, A. C. Tsoi, and A. D. Back. Face recognition: a convolutional neural network approach. *IEEE Trans. Neural Networks, Special Issue on Neural Networks Pattern Recognition*, 8(1):98–113, 1997.

[LHK02] J. Liu, A. Harris, and N. Kanwisher. Stages of processing in face perception: an meg study. *Nature Neuroscience*, 5(9):910–916, September 2002.

[Liu03] C. Liu. A bayesian discriminating features method for face detection. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 25(6):725–740, 2003.

[LKL97] S. H. Lin, S. Y. Kung, and L. J. Lin. Face recognition/detection by probabilistic decision-based neural network. *IEEE Trans. Neural Networks*, 8(1):114–132, January 1997.

[LLS03] K. Lin, K. Lam, and W. Siu. Spatially eigen-weighted hausdorff distances for human face recognition. *Pattern Recognition*, 36(8):1827–1834, 2003.

[LM98] J. Luettin and G. Maître. Evaluation protocol for the extended m2vts database (xm2vtsdb). *IDIAP Communication*, July 1998.

[LPV01] J. Lu, K. N. Plataniotis, and A. N. Venetsanopoulos. Face recognition using feature optimization and ν-support vector learning. *Proc. IEEE Signal Processing Society Workshop: Neural Networks for Signal Processing*, pages 373–382, 2001.

[LPV03a] J. Lu, K. N. Plataniotis, and A. N. Venetsanopoulos. Face recognition using kernel direct discriminant analysis algorithms. *IEEE Trans. Neural Networks*, 14(1):117–126, 2003.

[LPV03b] J. Lu, K. N. Plataniotis, and A. N. Venetsanopoulos. Face recognition using lda-based algorithms. *IEEE Trans. Neural Networks*, 14(1):195–200, 2003.

[LQLP02] J. Li, R. Y. Qiao, J. Lobb, and G. Poulton. Robust face recognition using multiple eye positions. *Conf. Digital Imaging Computing Techniques and Applications*, pages 236–240, 2002.

[LR03] M. W. Lee and S. Ranganath. Pose-invariant face recognition using a 3d deformable model. *Pattern Recognition*, 36(8):1835–1846, 2003.

[LTC95] A. Lanitis, C. J. Taylor, and T. F. Cootes. An automatic face identification system using flexible appearance models. *Image and Vision Computing*, 13(5):393–401, 1995.

[LTC02] A. Lanitis, C. J. Taylor, and T. F. Cootes. Toward automatic simulation of aging effects on face images. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 24(4):442–455, 2002.

[Lun95] L. Lundheim. A discrete framework for fractal signal modelling. In Y Fisher, editor, *Fractal Image Compression: Theory and Application*, chapter 7, pages 137–152. Springer-Verlag Inc., 1995.

[LVB+93] M. Lades, J. Vorbruggen, J. Buhmann, J. Lange, C.v.d. Malsburg, and R. Wurtz. Distortion invariant object recognition in the dynamic link architecture. *IEEE Trans. Computers*, 42:300–311, 1993.

[LW99] C. H. Lin and J. L. Wu. Automatic facial feature extraction by genetic algorithms. *IEEE Trans. Image Processing*, 8(6):834–845, 1999.

[LW00] C. Liu and H. Wechsler. Evolutionary pursuit and its application to face recognition. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 22(6):570–582, 2000.

[LW01] C. Liu and H. Wechsler. A shape- and texture-based enhanced fisher classifier for face recognition. *IEEE Trans. Image Processing*, 10(4):598–608, 2001.

[LW02] C. Liu and H. Wechsler. Gabor feature based classification using the enhanced fisher linear discriminant model for face recognition. *IEEE Trans. Image Processing*, 11(4):467–476, April 2002.

[LW03] C. Liu and H. Wechsler. Independent component analysis of gabor features for face recognition. *IEEE Trans. Neural Networks*, 14(4):919–928, 2003.

[LY96] K. M. Lam and H. Yan. Locating and extracting the eye in human face images. *Pattern Recognition*, 29(5):771–779, 1996.

[LY98] K. M. Lam and H. Yan. An analytic-to-holistic approach for face recognition based on a single frontal view. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 20(7):673–686, July 1998.

[LYF01] J. H. Lai, P. C. Yuen, and G. C. Feng. Face recognition using holistic fourier invariant features. *Pattern Recognition*, 34:95–109, 2001.

[Man82] B. Mandelbrot. *The fractal geometry of nature*. Freeman, San Francisco, CA, 1982.

[Mar02] A. M. Martínez. Recognizing imprecisely localized, partially occluded, and expression variant faces from a single sample per class. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 24(6):748–763, June 2002.

[MD01] P. McGuire and G. M. T. D'Eleuterio. Eigenpaxels and a neural-network approach to image classification. *IEEE Trans. Neural Networks*, 12(3):625–635, 2001.

[MGR98] S. McKenna, S. Gong, and Y. Raja. Modelling facial colour and identity with gaussian mixtures. *Pattern Recognition*, 31(12):1883–1892, 1998.

[MHJ+00] J. Matas, M. Hamouz, K. Jonsson, J. Kittler, Y. Li, C. Kotropoulos, A. Tefas, I. Pitas, T. Tan, H. Yan, F. Smeraldi, J. Bigun, N. Capdevielle, W. Gerstner, S. Ben-Yacoub, Y. Abdeljaoued, and E. Mayoraz. Comparison of face verification results on the xm2vts database. *Proc. 15th IEEE Int'l Conf. Pattern Recognition*, 4:858–863, 2000.

[MJP00] B. Moghaddam, T. Jebara, and A. Pentland. Bayesian face recognition. *Pattern Recognition*, 33:1771–1782, 2000.

[MKK97] J. Matas, K.Jonsson, and J. Kittler. Fast face localisation and verification. *Proc. British Machine Vision Conf.*, pages 152–161, 1997.

[MM00] D. Maio and D. Maltoni. Real-time face location on gray-scale static images. *Pattern Recognition*, 33:1525–1539, 2000.

[MMK+99] K. Messer, J. Matas, J. Kittler, J. Luettin, and G. Maitre. Xm2vtsdb: The extended m2vts database. *Proc. Second Int'l Conf.*

Audio and Video-based Biometric Person Authentication, pages 72–77, 1999.

[MP97] B. Moghaddam and A. Pentland. Probabilistic visual learning for object representation. *IEEE Trans. on PAMI*, 19(7):696–710, July 1997.

[MT03] K. Ma and X. Tang. Face recognition using discrete wavelet graph. *Proc. IEEE Intl. Conf. Acoustics, Speech, and Signal Processing*, 6:117–120, 2003.

[MWP98] B. Moghaddam, W. Wahid, and A. Pentland. Beyond eigenfaces: Probabilistic matching for face recognition. *Proc. Automatic Face and Gesture Recognition*, pages 30–35, 1998.

[NC96] G. Neil and K. M. Curtis. Scale and rotationally invariant object recognition using fractal transformations. *IEEE Int'l Conference on Acoustics, Speech, and Signal Processing*, 6, 1996.

[NCC96] G. Neil, K. M. Curtis, and M. P. Craven. Shape recognition using a novel fractal technique. *IEEE Int'l Conference on Electronics, Circuits, and Systems*, 2:724–727, 1996.

[NCS+00] M. Negin, T. A. Chmielewski, M. Salganicoff, T. A. Camus, U. M. Cahn von Seelen, P. L. Venetianer, and G. G. Zhang. An iris biometric system for public and personal use. *IEEE Computer*, pages 70–75, February 2000.

[NH98] A. V. Nefian and M. H. Hayes. Face detection and recognition using hidden markov models. *Proc. IEEE Int. Conf. Image Processing*, 1:141–145, 1998.

[NS98] K. Nagao and M. Sohma. Weak orthogonalization of face and perturbation for recognition. *Proc. Computer Vision and Pattern Recognition*, pages 845–852, 1998.

[OFG97] E. Osuna, R. Freund, and F. Girosi. Training support vector machines: An application to face detection. *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pages 130–136, 1997.

[OG99] E. Osuna and F. Girosi. Reducing run-time complexity in support vector machines. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods: Support Vector Learning*, pages 271–283. MIT Press, Cambridge, MA, 1999.

[OL95] G. E. Oien and S. Lepsoy. A class of fractal image coders with fast decoder convergence. In Y Fisher, editor, *Fractal Image Compression: Theory and Application*, chapter 8, pages 153–175. Springer-Verlag Inc., 1995.

[PA96] P. Penev and J. Atick. Local feature analysis: a general statistical theory for object representation. *Network: Computation in Neural Systems*, pages 477–500, March 1996.

[PBJ00] S. Pankanti, R. M. Bolle, and A. Jain. Biometrics: The future of identification. *IEEE Computer*, pages 46–49, February 2000.

[PC00] A. Pentland and T. Choudhury. Face recognition for smart environments. *IEEE Computer*, pages 50–55, February 2000.

[Pen84] A. Pentland. Fractal-based descriptions of natural scenes. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 6(6):661–674, 1984.

[PHPT03] Z. Pan, G. E. Healey, M. Prasad, and B. J. Tromberg. Face recognition in hyperspectral images. *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 1:334–339, 2003.

[PMST94] A. Pentland, B. Moghaddam, T. Starner, and M. Turk. View-based and modular eigenspaces for face recognition. *Proc. IEEE Computer Society Conf. Computer Vision and Pattern Recognition*, pages 84–91, 1994.

[PMWP00] P. J. Phillips, A. Martin, C. L. Wilson, and M. Przybocki. An introduction to evaluating biometric systems. *IEEE Computer*, pages 56–63, February 2000.

[Pou96] A. D. Poularikas, editor. *The Transforms and Applications Handbook*. CRC Press, 1996.

[PTVF92] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C*. Cambridge University Press, 2nd edition, 1992.

[PV96] P. J. Phillips and Y. Vardi. Efficient illumination normalization of facial images. *Pattern Recognition Letters*, 17:921–927, 1996.

[PVY03] S. Palanivel, B. S. Venkatesh, and B. Yegnanarayana. Real time face recognition system using autoassociative neural network models. *Proc. IEEE Intl. Conf. Acoustics, Speech, and Signal Processing*, 2:833–836, 2003.

[PY95] D. Popescu and H. Yan. Fractal-based method for colour image compression. *Journal of Electronic Imaging*, 4(1):23–30, 1995.

[PY97] D. Popescu and H. Yan. A non-linear model for fractal image coding. *IEEE Trans. Image Processing*, 6(3):373–382, 1997.

[RBK95] H. Rowley, S. Baluja, and T. Kanade. Human face detection in visual scenes. *Technical Report CMU-CS-95-158, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA*, July 1995.

[RBK98a] H. Rowley, S. Baluja, and T. Kanade. Neural network-based face detection. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 20(1):23–38, 1998.

[RBK98b] H. A. Rowley, S. Baluja, and T. Kanade. Rotation invariant neural network-based face detection. *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pages 38–44, June 1998.

[RG86] B. Ramamurthi and A. Gersho. Classified vector quantization of images. *IEEE Trans. Commun.*, COM-34(11):1105–1115, November 1986.

[RH97] M. Ruhl and H. Hartenstein. Optimal fractal coding is np-hard. *Proc. DCC'97 Data Compression Conference*, pages 261–270, March 1997.

[RJ93] L. R. Rabiner and B. H. Jung. *Fundamentals of Speech Recognition.* Prentice Hall, 1993.

[RPM98] S. A. Rizvi, P. J. Phillips, and H. Moon. A verification protocol and statistical performance analysis for face recognition algorithms. *IEEE Conf. Computer Vision and Pattern Recognition*, pages 833–838, June 1998.

[RTSB01] S. Romdhani, P. Torr, B. Schölkopf, and A. Blake. Computationally efficient face detection. *Proc. Eighth IEEE Int'l Conf. Computer Vision*, 2:695–700, 2001.

[Ruc96] W. Rucklidge. Efficient visual recognition using the hausdorff distance. *Lecture notes in computers science*, 1173, 1996.

[RWY95] D. Reisfeld, H. Wolfson, and Y. Yeshurun. Context free attentional operators: The generalized symmetry transform. *Int. J. Computer Vision*, 14:119–130, 1995.

[SAA02] A. A. Salah, E. Alpaydin, and Lale Akarun. A selective attention-based method for visual pattern recognition with application to handwritten digit recognition and face recognition. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 24(3):420–425, 2002.

[Sam94] F. S. Samaria. *Face recognition using hidden markov models.* PhD thesis, Trinity College, University of Cambridge, Cambridge, 1994.

[SB98] F. Smeraldi and J. Bigün. Facial feature detection by saccadic exploration of the gabor decomposition. *Proc. IEEE Int'l Conf. Image Processing*, 3:163–167, 1998.

[SB02] F. Smeraldi and J. Bigün. Retinal vision applied to facial features detection and face authentication. *Pattern Recognition Letters*, 23:463–475, 2002.

[SBO+97] T. Satonaka, T. Baba, T. Otsuki, T. Chikamura, and T. H. Meng. Object recognition with luminance, rotation and location invariance. *IEEE Int'l Conf. Image Processing*, 3:336–339, 1997.

[SBS99] B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors. *Advances in Kernel Methods: Support Vector Learning*, Cambridge, MA, 1999. MIT Press.

[SCB99] F. Smeraldi, N. Capdevielle, and J. Bigün. Facial features detection by saccadic exploration of the gabor decomposition and support vector machines. *Proc. 11th Scandinavian Conf. Image Analysis*, 1:39–44, 1999.

[Sed88] R. Sedgewick. *Algorithms*. Addison-Wesley, 2nd edition, 1988.

[SH94] F. S. Samaria and A. C. Harter. Parametrisation of a stochastic model for human face identification. *Proc. 2nd IEEE Workshop on Applications of Computer Vision*, pages 138–142, 1994.

[Sin94] P. Sinha. Learning and using qualitative invariants for recognition. *MIT A.I. Memo*, (1505), November 1994.

[SK87] L. Sirovich and M. Kirby. Low-dimensional procedure for the characterization of human faces. *Journal fo the Optical Society of America A*, 4(3):519–524, 1987.

[SK00] H. Schneiderman and T. Kanade. A statistical method for 3d object detection applied to faces and cars. *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 1:746–751, 2000.

[SK03] M. Savvides and B. V. K. V. Kumar. Efficient design of advanced correlation filters for robust distortion-tolerant face recognition. *Proc. IEEE Conf. Advanced Video and Signal Based Surveillance*, pages 45–52, 2003.

[SMB+99] B. Schölkopf, S. Mika, C. J. C. Burges, P. Knirsch, K. Müller, G. Rätsch, and A. J. Smola. Input space versus feature space in kernel-based methods. *IEEE Trans. Neural Networks*, 10(5):1000–1017, September 1999.

[SRR01] A. Shashua and T. Riklin-Raviv. The quotient image: Class-based re-rendering and recognition with varying illuminations. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 23(2):129–139, 2001.

[SSWB98] B. Schölkopf, A. J. Smola, R. Williamson, and P. Bartlett. New support vector algorithms. *Technical Report NC-TR-98-031, NeuroCOLT Working Group, http://www.neurocolt.com*, 1998.

[Sun96] K. K. Sung. *Learning and Example Selection for Object and Pattern Detection.* PhD thesis, MIT AI Lab, 1996.

[SW91] M. A. Shackleton and W. J. Welsh. Classification of facial features for recognition. *IEEE Conf. Computer Vision and Pattern Recognition*, pages 573–579, 1991.

[SY94] F. Samaria and S. Young. Hmm based architecture for face identification. *Image and Vision Computing*, 12:537–583, 1994.

[Tit02] T. Titsworth. More than face value: Airports and multimedia security. *IEEE Multimedia*, 9(2):11–13, April-June 2002.

[TKP01] A. Tefas, C. Kotropoulos, and I. Pitas. Using support vector machines to enhance the performance of elastic graph matching for frontal face authentication. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 23(7):735–746, July 2001.

[TP91a] M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3(1):71–86, January 1991.

[TP91b] M. Turk and A. Pentland. Face recognition using eigenfaces. *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pages 586–591, 1991.

[TSS+00] J. Terrillon, M. N. Shirazi, M. Sadek, H. Fukamachi, and S. Akamatsu. Invariant face detection with support vector machines. *Proc. International Conference on Pattern Recognition*, 4:210–217, 2000.

[VB98] T. Vetter and V. Blanz. Estimating coloured 3d face models from single images: An example-based approach. *Proc. European Conf. Computer Vision*, pages 499–513, 1998.

[Vin95] J. M. Vincent. Face finding in images. In A. Murray, editor, *Applications of Neural Networks*, chapter 2, pages 35–70. Kluwer Academic Publishers, Netherlands, 1995.

[VJP97] T. Vetter, M. J. Jones, and T. Poggio. A bootstrapping algorithm for learning linear models of object classes. *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pages 40–46, 1997.

[Vot03] D. Voth. Face recognition technology. *IEEE Intelligent Systems*, 18(3):4–7, 2003.

[Vrs95] E. R. Vrscay. A hitchhiker's guide to 'fractal-based' function approximation and image compression. *Math Ties, University of Waterloo*, February and August 1995.

[Way00] J. L. Wayman. Federal biometric technology legislation. *IEEE Computer*, pages 76–80, February 2000.

[WdJ99] B. Wohlberg and G. de Jager. A review of the fractal image coding literature. *IEEE Trans. on Image Processing*, 8(12):1716–1729, December 1999.

[WFKvdM97] L. Wiskott, J. M. Fellous, N. Krüger, and C. v. d. Malsburg. Face recognition by elastic bunch graph matching. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 19(7):775–779, 1997.

[WLS01] K. W. Wong, K. M. Lam, and W. C. Siu. An efficient algorithm for human face detection and facial feature extraction under different conditions. *Pattern Recognition*, 34:1993–2004, 2001.

[WPB+98] H. Wechsler, P. J. Phillips, V. Bruce, F. Fogelman-Soulie, and T. S. Huang, editors. *Face Recognition: From Theory to Applications*, volume 163 of *NATO ASI Series. Series F, Computer and Systems Sciences*, Berlin, 1998. Springer.

[WT03] X. Wang and X. Tang. An improved bayesian face recognition algorithm in pca subspace. *Proc. IEEE Intl. Conf. Acoustics, Speech, and Signal Processing*, 3:129–132, 2003.

[XSZ93] X. Xie, R. Sudhakar, and H. Zhuang. Corner detection by a cost minimization approach. *Pattern Recognition*, 26:1235–1243, 1993.

[YA98] M. H. Yang and N. Ahuja. Detecting human faces in color images. *Proc. IEEE Int. Conf. Image Processing*, 1:127–130, 1998.

[YA99] M. H. Yang and N. Ahuja. Gaussian mixture model for human skin color and its application in image and video databases. *Proc. SPIE: Storage and Retrieval for Image and Video Databases VII*, 3656:458–466, 1999.

[Yan02] M. H. Yang. Face recognition using kernel methods. *Advances in Neural Information Processing Systems 14*, pages 215–220, 2002.

[YC96] K. C. Yow and R. Cipolla. Detection of human faces under scale, orientation and viewpoint variations. *Proc. 2nd Int'l Conf. Automatic Face and Gesture Recognition*, pages 295–300, October 1996.

[YH94] G. Yang and T. S. Huang. Human face detection in complex background. *Pattern Recognition*, 27(1):53–63, 1994.

[YHC92] A. L. Yuille, P. W. Hallinan, and D. S. Cohen. Feature extraction from faces using deformable templates. *Int. Journal of Computer Vision*, 8(2):99–111, 1992.

[YKA02] M. Yang, D. J. Kriegman, and N. Ahuja. Detecting faces in images: A survey. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 24(1):34–58, January 2002.

[YL02] P. C. Yuen and J. H. Lai. Face representation using independent component analysis. *Pattern Recognition*, 35:1247–1257, 2002.

[YS02] A. Yilmaz and M. A. Shah. Automatic feature detection and pose recovery for faces. *The 5th Asian Conf. Computer Vision*, pages 284–289, January 2002.

[Yui91] A. L. Yuille. Deformable templates for face recognition. *Journal of Cognitive Neuroscience*, 3(1):59–70, 1991.

[YY03] J. Yang and J. Y. Yang. Why can lda be performed in pca transformed space? *Pattern Recognition*, 36:563–566, 2003.

[ZP96] X. Zhang and C. Podilchuk. Face location and recognition. *Proc. SPIE*, 2657:252–262, 1996.

[ZPZP00] D. Zhang, H. Peng, J. Zhou, and S. K. Pal. A novel face recognition system using hybrid neural and dual eigenspaces methods. *Image and Vision Computing Journal*, 18(4):279–287, 2000.

[ZS03] L. Zhang and D. Samaras. Face recognition under variable lighting using harmonic image exemplars. *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 1:19–25, 2003.

[ZYL97] J. Zhang, Y. Yan, and M. Lades. Face recognition: eigenface, elastic matching, and neural nets. *Proc. IEEE*, 85(9):1423–1435, 1997.

Publications

Journal Papers

1. T. Tan and H. Yan. Object recognition based on fractal neighbor distance. *Signal Processing*, 81:2105–2129, 2001.

2. T. Tan and H. Yan. The fractal neighbor distance measure. *Pattern Recognition*, 35:1371–1387, 2002.

3. T. Tan and H. Yan. Face recognition using the weighted fractal neighbor distance. *IEEE Trans. Systems, Man, and Cybernetics: Part C*, in press.

International Conference Papers

1. T. Tan and H. Yan. Analysis of the contractivity factor in fractal based face recognition. *IEEE Int'l Conf. Image Processing*, 3:637–641, October 1999.

2. T. Tan and H. Yan. Face recognition by fractal transformations. *IEEE Int'l Conference on Acoustics, Speech, and Signal Processing*, 6:3537–3540, March 1999.

3. T. Tan and H. Yan. Determining and controlling convergence in fractal image coding. *IEEE Int'l Conference on Image Processing*, 2:187–190, September 2000.

4. T. Tan and H. Yan. Object recognition using fractal neighbor distance: eventual convergence and recognition rates. *International Conference on Pattern Recognition*, 2:785–788, September 2000.

5. J. Matas, M. Hamouz, K. Jonsson, J. Kittler, Y. Li, C. Kotropoulos, A. Tefas, I. Pitas, T. Tan, H. Yan, F. Smeraldi, J. Bigun, N. Capdevielle, W. Gerstner, S. Ben-Yacoub, Y. Abdeljaoued, and E. Mayoraz. Comparison of face verification results on the xm2vts database. *Proc. 15th IEEE Int'l Conf. Pattern Recognition*, 4:858–863, 2000.

Local Conference Papers

1. T. Tan and H. Yan. Face recognition using the theory of fractal image coding. *Visual Information Processing*, 34–38, November 1998.

2. T. Tan and H. Yan. Human face recognition using fractal image coding. *Visual Information Processing*, 50–54, December 1997.