Facial Recognition, Eigenfaces and Synthetic Discriminant Functions

Neil Muller

Dissertation presented for the Degree of Doctor of Philosophy
at the University of Stellenbosch.
Promoter: Prof. B. Herbst

November 15, 2000

Declaration

I, the under signed, hereby declare that the work contained in this dissertation is my own original work and that I have not previously in its entirety or in part submitted it at any university for a degree

Signature:_____

Date:_____

Abstract

In this thesis we examine some aspects of automatic face recognition, with specific reference to the eigenface technique. We provide a thorough theoretical analysis of this technique which allows us to explain many of the results reported in the literature. It also suggests that clustering can improve the performance of the system and we provide experimental evidence of this. From the analysis, we also derive an efficient algorithm for updating the eigenfaces. We demonstrate the ability of an eigenface-based system to represent faces efficiently (using at most forty values in our experiments) and also demonstrate our updating algorithm.

Since we are concerned with aspects of face recognition, one of the important practical problems is locating the face in a image, subject to distortions such as rotation. We review two well-known methods for locating faces based on the eigenface technique. These algorithms are computationally expensive, so we illustrate how the Synthetic Discriminant Function can be used to reduce the cost. For our purposes, we propose the concept of a linearly interpolating SDF and we show how this can be used not only to locate the face, but also to estimate the extent of the distortion. We derive conditions which will ensure a SDF is linearly interpolating. We show how many of the more popular SDF-type filters are related to the classic SDF and thus extend our analysis to a wide range of SDF-type filters. Our analysis suggests that by carefully choosing the training set to satisfy our condition, we can significantly reduce the size of the training set required. This is demonstrated by using the equidistributing principle to design a suitable training set for the SDF. All this is illustrated with several examples.

Our results with the SDF allow us to construct a two-stage algorithm for locating faces. We use the SDF-type filters to obtain initial estimates of the location and extent of the distortion. This information is then used by one of the more accurate eigenface-based techniques to obtain the final location from a reduced search space. This significantly reduces the computational cost of the process.

Opsomming

In hierdie tesis ondersoek ons sommige aspekte van automatiese gesigsherkennin g met spesifieke verwysing na die sogenaamde eigengesig ("eigenface") tegniek. 'n Deeglike teoretiese analise van hierdie tegniek stel ons in staat om heelparty van die resultate wat in die literatuur verskyn te verduidelik. Dit bied ook die moontlikheid dat die gedrag van die stelsel sal verbeter as die gesigte in verskillende klasse gegroepeer word. Uit die analise, herlei ons ook 'n doeltreffende algoritme om die eigegesigte op te dateer. Ons demonstreer die vermoë van di e stelsel om gesigte op 'n doeltreffende manier te beskryf (ons gebruik hoogstens veertig eigegesigte) asook ons opdateringsalgoritme met praktiese voorbeelde.

Verder ondersoek ons die belangrike probleem om gesigte in 'n beeld te vind, veral as rotasie– en skaalveranderinge plaasvind. Ons bespreek twee welbekende algoritmes om gesigte te vind wat op eigengesigte gebaseer is. Hierdie algoritme s is baie duur in terme van numerise berekeninge en ons ontwikkel 'n koste-effekt iewe metode wat op die sogenaamde "Synthetic Discriminant Functions" (SDF) gebaseer is. Vir hierdie doel word die begrip van lineêr interpolerende SDF's ingevoer. Dit stel ons in staat om nie net die gesig te vind nie, maar ook 'n skatting van sy versteuring te bereken. Voorts kon ons voorwaardes aflei wat verseker dat 'n SDF lineêr interpolerend is. Aangesien ons aantoon dat baie van die gewilde SDF-tipe filters aan die klassieke SDF verwant is, geld ons resultate vir 'n hele verskeidenheid SDF-tipe filters. Ons analise toon ook dat 'n versigtige keuse van die afrigdata mens in staat stel om die grootte van die afrigstel aansienlik te verminder. Dit word duidelik met behulp van die sogenaamde gelykverspreidings beginsel ("equidistributin g principle") gedemonstreer.

Al hierdie aspekte van die SDF's word met voorbeelde geïllustreer.

Ons resultate met die SDF laat ons toe om 'n tweestap algoritme vir die vind van 'n gesig in 'n beeld te ontwikkel. Ons gebruik eers die SDF-tipe filters om skattings vir die posisie en versteuring van die gesig te kry en dan verfyn ons hierdie skattings deur een van die teknieke wat op eigengesigte gebaseer is te gebruik. Dit lei tot 'n aansienlike vermindering in die berekeningstyd.

Contents

List of Figures

Chapter 1

Introduction

1.1 Background

The need for cheap, robust personal identifications systems in the modern technological society is critical. As society becomes more dependent on automated systems, it is becoming increasingly important to control access to these systems. Whereas identification systems of the past relied heavily on human involvement, this practice is rapidly becoming obsolete—it simply does not provide adequate protection where large sums of money are involved. And automated, electronic transactions are becoming the norm.

There are basically two different types of automated personal identification. The first relies on things like passwords, access codes, identification cards and so on. They typically make use of something the individual *knows* or *owns*. These systems are relatively easy to circumvent, as is testified by the massive credit card fraud world-wide, to mention only one example. The second method makes use of one or more *biometric* properties of the individual. These methods are based on some inherent property of the individual; something you *are*. The computational power and specialised sensor hardware available today have led to an explosion of automated, biometric identification systems. Some well-known systems include fingerprints, retina or iris scans and dynamic signature verification. More exotic systems make use of vein patterns or bodily odour, aptly called the bloodhound!

There is no doubt the most familiar and most natural identification system for humans is the human face. In fact, humans have honed their facial recognition skills to such an extent that a special part of the brain is ded-

icated to this task only. One should heed the warning: if a special part of the brain is dedicated to facial recognition the problem is far from easy, as any person with ambitions for developing a complete system is bound to discover very quickly. It is certainly much harder to get it right, and less reliable than some of the systems mentioned above. A natural question then arises: why should one even consider developing automated facial recognition systems, why not simply stick to iris scans or fingerprints? A clear answer is not forthcoming. However, since it is *the* natural way of identification for humans, it is the indicated identification system in situations where human intervention is possible or required. Although the human system is far from foolproof—for some horror stories where the human system has caused disaster, the reader should consult [27]—it remains a system which is trusted to an extraordinary degree. One should also realise that facial recognition is a passive system, i.e. no cooperation from the individual is required, he or she will be identified no matter what (deliberate disguises excluded). The only systems mentioned above that share this property are the iris– and retina scans and the bodily odour, all systems that require considerably more expensive equipment. In the final analysis, scientists and engineers will always find the challenge irresistible.

1.2 The Facial Recognition Problem

The facial recognition problem is simply stated: Given an image or images of a person in the database, can one tell whether an arbitrary image is of the same person or not? Easily stated but not easily answered. After a moment's reflection it becomes clear that the problem is a challenging one. First of all, different images of the same person can differ in many different ways. Let us consider some of it. In the first place, facial expressions, hair styles, beards, mustaches and makeup which sometimes resemble deliberate disguise, are forever changing. Then one has to deal with equipment variables such as different lighting conditions. Here it is important to bear in mind that a gray scale image on film or CCD, is actually an image, not of a face itself, but of the interaction of light with the physical structure of the face. Therefore, if the lighting changes, either in intensity or direction, it can affect the image in a profound manner. There are several common approaches to dealing with lighting variation. These include compensating for the lighting variation,

using light invariant features (i.e. edges), or modeling the object under many lighting conditions. In some applications, a graph-based description of the object is used, but it is difficult to describe a face with enough detail for recognition using this approach. It is difficult to define light invariant features on a face (edges are too unstable with respect to minor out of plane rotation) and modeling the face under different lighting conditions requires taking images of many individuals under all the lighting variations we would expect to encounter — obviously a difficult task. So the usual approach in face recognition is to try and compensate for the lighting variations. The techniques used range from various forms of intensity normalisation to much more complex light source estimation procedures (which usually also involve some degree of modeling the lighting variations). For instance, in [15], the authors devote a entire chapter (some 30 pages) to an algorithm for dealing with extreme lighting variations.

The position, scale and orientation of an image is also of crucial importance— a system should be able to compensate for images at different scales, in different positions and at different angles. Two types of rotations are possible, in-plane rotation where the image is full frontal but the head is tilted to the side, and out-of-plane rotation where the image is taken from the side. It is relatively easy to compensate for scale, position and in-plane rotation. Out-of-plane rotation is hard and even impossible once the face is rotated to the extent that important facial information become obscured.

Other practical considerations include locating the face in the image, in the first place. This can be particularly hard against a cluttered background.

Let us say we have solved all these practical problems, most likely by assuming some cooperation from the individual. Then there remains the problem of the actual facial comparison. Some similarity measure is required which will detect features which can reliably distinguish one person from another while ignoring the possible variations, such as lighting changes, varying background, in-plane rotation, scale variation, expression changes and natural changes over time (such as changing hair-styles, etc). At this point one runs into the additional complication of the sheer size of the images. Even at a coarse resolution, we will seldom deal with images of less than 100×100 pixels in size, requiring the manipulation of at least 10 000 bytes (for gray scale images). And in most situations, we will want to work at a much finer resolution and consequently deal with images that are considerably larger.

Thus any recognition algorithm must not only be accurate and robust, but must also be able to deal efficiently with this problem.

There are several approaches to facial recognition that have been investigated, including "Fisherfaces" — which use Discriminant Analysis to distinguish individuals, "ZN-faces" — which uses elastic graph matching and eigenfaces. See [15] for more details and examples. In this thesis we concentrate on the latter.

The idea behind eigenfaces can be viewed as a mathematical "identi-kit" where some set of basic images is constructed which is then combined to form the individual faces. One of the key features of the method is the relatively small number of basic images—the so-called eigenfaces—that are required, something of the order of one hundred, irrespective of the resolution of the images. Since we only need to deal with the representation of a face in terms of these basic images the amount of data is dramatically reduced. The construction of these basic images will be described in detail in Chapter 2.

One of the most active groups in the field of face recognition using the eigenface approach is the Vision and Modeling Group of the Media Lab at MIT (see [55]). Their system, one of the most successful in the literature, has achieved a recognition rate of 97% on a test set of 2000 images, using only twenty eigenfaces.

Let us briefly describe how they overcome some of the more serious problems mentioned above. A maximum likelihood algorithm (which we will discuss in detail in section 4.2) is used to locate the faces and at the same time correct for scale and in-plane rotation. The algorithm also relies heavily on the eigenface representation of faces, but uses only ten eigenfaces. It is accurate, but computationally expensive (in a sense that should become clear, this is almost inevitable). Out-of-plane rotation is not considered. Lighting is handled by normalising contrast to match some predetermined model. While this is reasonably successful for small lighting variations, it can fail badly when confronted with extreme variations such as a light source placed to the side of the individual. Background variations (and incidentally problems with changing hair-styles) are removed by cropping the image to leave just the central section of the face. A multi-levelled approach is used for recognition, which uses not only the entire face, but also local features such as the eyes, nose and mouth. This is achieved by extending the eigenface

idea to these local features, i.e. the nose, mouth and eyes are also represented in terms of basic eigenfeatures, a set of features for each of the eyes, the nose and the mouth. Incidentally, this approach using local features also compensates to a large extent for expression changes.

In the next section we give an overview of the contributions of this thesis, in particular in the light of what is known about the eigenface technique.

1.3 Thesis Outline

The main emphasis of this thesis is to make a contribution towards the practical implementation, as well as a better theoretical understanding of the eigenface approach for facial recognition. From the remarks above, the reader should be aware that the development of a robust practical system is a huge undertaking, and this is not attempted in this thesis. Our more modest aim is to make a contribution to the some of the more detailed implementation issues.

Also note that it is difficult to compare our results with the other systems mentioned as we do not have access to all the details about these systems. There is for instance comparatively little information available about the lighting normalisation used by the MIT system and also little information available about the exact method used to compare two images. As shown in the FERET tests (see [40]), these implementation issues can have a significant effect on the performance of the system. The lack of an easily available, common test and training set also makes comparison difficult. Virtually every system in the literature is tested on a locally available test sets, which makes comparison difficult. It is therefore also difficult to assess how various systems scale to large training sets (this is especially difficult in the case of Fisherfaces [1], where the article uses very few faces).

We commence with an overview of the basic eigenface technique. This is followed with a rather detailed mathematical analysis which we believe clarifies some of the implementation issues mentioned in the literature, and at the same time point out some of its limitations. For example, it is remarkable that such a small number of eigenfaces suffice for both the location as well as the recognition of a face. From our analysis, we feel we can explain not only this, but several of the other results frequently quoted about eigenfaces. For instance, we provide a solid mathematical justification for the frequently

stated result that the lower order eigenfaces contain less individual detail than the higher order eigenfaces. We also show how that, provided the initial training set is suitable, the system should be stable when confronted with new faces.

One of the interesting results we obtain is that the stability and efficiency of the system depends on how tightly clustered the images are around the mean. This suggests that it may be possible to improve the performance of a recognition system by sub-dividing the training set into several independent sets of images, each of which is more tightly clustered than the original set.

Of course, before any attempt can be made to apply the technique, we need to ensure we can actually find the face. As alluded to above, the maximum likelihood approach is robust, but expensive. The expense is mainly due to the fact that a search has to be conducted for faces at different rotations and angles. This is described in detail in section 4.2. In a quest to decrease the computational cost, we examine the Synthetic Discriminant Function and its extensions (in Chapter 5), a family of fast location algorithms that have been used for various problems, such as radar imaging. The idea is to use these fast methods to provide an estimate of the scale, rotation and location of the facial image, thereby reducing the size of the search space for the maximum likelihood method. Thus we propose a two-stage algorithm which uses the fast filter-based algorithm to provide an initial estimate for a more accurate method.

In order to employ the SDF in this role, a thorough theoretical analysis is required. In particular, we needed to find a way of estimating the scale and rotation from the filter output. We were able to derive a sufficient condition which the images used to design the filter must satisfy for this to be possible.

We also derived a relationship between the SDF and some of the more popular variants based on the SDF which provides some valuable insight into why the variants behave as they do, but also allows us to extend the condition found for the SDF to these variants. From this, we are able to make informed judgments about which filters to use for our purposes. We also analyse the MACH filter, which although based on ideas introduced from the SDF, is not a true SDF variant, and provide insights into situations where this filter is expected to perform well.

We also relate the SDF to the so-called Feature Space Trajectories. As far as we are aware, no-one has shown this relationship before. We conclude

our analysis by showing how the results we have obtained allow us to use existing mathematical techniques to choose an optimal training set for the filter.

We illustrate aspects of using the SDF and related filters in Chapter 6 and provide some experimental results for our two-stage technique in Chapter 7. We feel the results we obtain here show that this method can significantly reduce the computational cost of searching for a face in an image.

Then we investigate the clustering idea, suggested by the perturbation analysis. We used a fairly large number of images taken from a database gathered of students for student identification cards. As will be discussed in detail, these images are far from ideal. Since they are of students, they are rather more homogeneous than we would expect to encounter in a full system.

First a clustering based on the eigenface approach is described. Then a series of experiments is performed (discussed in Chapter 8). These focus on how such a clustering of the images affects the ability of the system to reconstruct images. We find that clustering the images is beneficial for this, since we can obtain the same quality of reconstruction using significantly less eigenfaces in each cluster. Unfortunately, the clusters separate on large-scale variations, which includes lighting variation. Obviously sub-dividing images purely on lighting is not a good idea for a recognition system, so we investigate clustering a subset of this database with has considerably reduced lighting variation, and find that clustering is beneficial here as well.

All the experimental results where generated using MATLAB. Since the eigenface technique is based entirely on tools from linear algebra, the simplicity with which these can be coded in MATLAB made it the obvious choice. All experiments were run on a Pentium-II 300MHz machine with 128MB memory, running Linux.

In Appendix A, we provide a brief summary of some of the standard mathematical results we use during this thesis. Since we introduce a lot of terms related to both eigenfaces and the SDF, we provide a summary of the results and terms used in Appendixes B and C respectively.

1.4 Our Goals and Contribution

In this thesis, we look to address two major questions. Firstly, can we improve on our understanding of the eigenface technique, and secondly, is it possible to design an algorithm which will locate the face accurately, yet is computationally less expensive than the common eigenface-based approaches?

Our attempts to answer these questions lead to several other issues that need to be examined, such as: Is there any practical benefit to clustering faces? Can we successfully update the eigenfaces to accommodate a new face? Can we provide testable conditions that will ensure that the Synthetic Discriminant approach is acceptable?

In addressing this issues, we believe the following to be novel contributions:

- We provide a through theoretical analysis of the eigenface technique in terms of matrix perturbation theory.
  - As a result, we derive an efficient updating algorithm for this result. This updating algorithm is a improvement on one we presented earlier (see [31]) in that the conditions on the faces are significantly relaxed.
  - We also derive a theoretical indication that clustering may well have a significant impact on the performance of an eigenface based system. We provide some limited experimental evidence that this is indeed the case for reconstructing faces.
- We propose the concept of a linearly interpolating Synthetic Discriminant Function, which allows us to estimate the extent of the distortion,
- We derive a sufficient condition that will ensure that an SDF is linearly interpolating
  - We demonstrate how the well-known equidistributing principle from numerical analysis can be combined with our sufficient condition as a method for designing the training set for the SDF.
- We relate the SDF to another distortion estimating procedure, namely Feature Space Trajectories.

- We show all the major variants of the SDF can be related to the original SDF and thus extend our sufficient condition to include these variants
- We also provide some analysis of the Maximum Average Correlation Height filter, and show how its performance is related to how widely spread the data is.
- We propose and demonstrate the performance of a two-stage algorithm to locate faces, which first uses the SDF to estimate the location of the face and then uses an eigenface-based algorithm to refine this estimate.

Chapter 2

Eigenfaces: Background and Analysis

2.1 Introduction to Eigenfaces

The eigenface idea first appeared in a paper by Sirovich and Kirby [47]. Since then it has been studied by a large number of different groups (see [36] or [41]). Most prominent has been the MIT Media Lab, who have produced a large number of articles on the technique (see [52], [53] or [38] as well as several later articles). This has become one of the standard techniques for face recognition and has been one of the most successful techniques used in the FERET test (see [40]). Not surprisingly, it has formed the starting point of several extended methods for face recognition (such as Fisherfaces[1] [1]). Almost all the work done, however, has given only quantitative analyses of the effectiveness of this technique. One of the aims of this thesis is to provide some measure of qualitative analysis.

2.2 Calculating the Eigenfaces

Mathematically all $D \times D$ images form an D^2 dimensional vector space. This is just another way of saying that one has to specify D^2 pixel values in order to display the image on a computer screen. In the case of a 8-bit gray scale

[1] The idea in Fisherfaces is to use Discriminant Analysis to improve the performance of an eigenface-based recognition scheme, especially with respect to lighting variations. However, there is not much information available about how the system scales to large datasets.

image all pixel values vary between 0 and 255, indicating the different shades of gray. The value of D determines the resolution of the image. Even for images with a coarse resolution the value of D may easily exceed 100; it is clear that the dimension of the image space is high. On the other hand, if we are only interested in facial images, the question arises: Do facial images constitute a lower dimensional sub-space? If so, how do we determine a basis for this sub-space? The answer is given in terms of the Singular Value Decomposition.

We assume the images are represented by D^2 dimensional random vectors, $\boldsymbol{F}$. When necessary to refer to specific faces, it will be denoted by $\boldsymbol{F}_n$ where n is an index indicating the different images. We assume that these vectors are randomly distributed around a mean, $E(\boldsymbol{F})$, where $E(\cdot)$ denotes the expected value, according to some unspecified distribution function. It will be useful to introduce a measure of the spread around this mean,

$$\rho^2 = E\left(\|\boldsymbol{F} - E(\boldsymbol{F})\|_2^2\right). \tag{2.2.1}$$

Assuming that ρ is finite, a Chebyshev-type theorem can be proved following the lines of the familiar one-dimensional version (see for example [43]). According to this theorem the probability of a face being within Q "standard deviations", ρ, from the mean is better than $1 - \frac{1}{Q^2}$. More specifically, the theorem states that

$$P\left(\|\boldsymbol{F} - E(\boldsymbol{F})\|_2 \leq Q\rho\right) \geq 1 - \frac{1}{Q^2}.$$

In order to acknowledge the possibility that some faces may deviate to such an extent from the mean that they become unrecognisable as faces, we allow only those faces that fall within a certain distance from the mean, $E(\boldsymbol{F})$. Thus we consider only those faces satisfying,

$$\|\boldsymbol{F} - E(\boldsymbol{F})\|_2^2 < Q^2\rho^2 \tag{2.2.2}$$

where Q is chosen not too large, but large enough to accommodate the majority of faces. This is related to the usual Sum of Squared Distances (SSD) measure used in automatic clustering techniques or in classification

Figure 2.1: Two normalized faces in the training set.

problems (of which we will say more in section 8.3.2.1).

Due to the wide variety of faces, it is impossible to study every individual face and another way of extracting the information characterizing faces is required. Thus we collect a set of facial images—the training set—that contains all the features encountered in all faces satisfying (2.2.2). Obviously, the training set has to be developed with care. Most importantly, it has to be representative of all faces that will be presented to the system; the training set has to be a good random sample of the all facial images. If the training set consists of M faces, then since the dimension of the image space, D^2, is usually very large, typically $M < D^2$. In practice, because of the difficulty of constructing a large training set, M is usually considerably less than D^2.

Instead of working with the images as a 2-dimensional array, it is more convenient to convert the 2-dimensional images by concatenating the columns (or rows) to form a 1-dimensional column vector of dimension D^2. Thus the faces in our training set are given by M D^2 dimensional vectors: $\boldsymbol{F}_n,\ n = 1, ..., M$. In order for this procedure to make sense, it is important that the images in the training set are standardized with respect to size, orientation and intensity. This means that we have to ensure that all the faces are of the same size, at the same angle (upright is good) and have the same lighting intensity. These are non-trivial image processing tasks that for the purposes of this discussion we will assume have already been done. We will return to the problem of normalising scale and angle in more detail later.

For illustrative purposes we use a small set of facial images collected mainly from [57] and [56]. The faces, at resolution of 112×112 pixels, were separated in a training and test set. The interested reader is encouraged to visit the original web sites. Figure 2.1 shows two normalized images in our training set.

Realizing that the mean face captures essential (general) facial information the next step is to calculate an average face $\boldsymbol{A}$, which serves to approximate the mean face. The average face is defined in the standard way as

$$\boldsymbol{A} = \frac{1}{M}\sum_{n=1}^{M} \boldsymbol{F}_n. \tag{2.2.3}$$

The normalized deviations from the average face are given by

$$\boldsymbol{X}_j = \frac{\boldsymbol{F}_j - \boldsymbol{A}}{d} \tag{2.2.4}$$

where we introduce the normalization d, given by

$$d^2 = \frac{1}{M}\sum_{n=1}^{M} \|\boldsymbol{F}_n - \boldsymbol{A}\|_2^2 \,. \tag{2.2.5}$$

Note that d is an approximation for the "standard deviation" ρ, defined in (2.2.1). Although a normalization is necessary for the computation of the eigenfaces, this specific choice is a matter of convenience. This choice facilitates the analysis in section 2.3. In the literature the normalization $\|\boldsymbol{X}_n\|_2 = 1$ is often employed. In comparison, the normalization (2.2.4) implies that

$$\frac{1}{M}\sum_{n=1}^{M} \|\boldsymbol{X}_j\|^2 = 1. \tag{2.2.6}$$

Said differently, if we define the random numbers $t_n := \|\boldsymbol{X}_n\|_2^2$, their average is one.

In some cases, the average is not subtracted and the variation described by the average is left to the eigenvectors. This has the advantage that certain aspects of the analysis are then simpler as the average doesn't have to be considered as a separate entity. All the images, however, are represented by vectors with only non-negative elements and they generally lie far from the origin. Subtracting the average means we treat it as a uniform bias on all the images, which we consider a more useful way of dealing with the images, especially as the average represents a large deviation from the origin. Furthermore, subtracting the average ensures that the $\boldsymbol{X}_n$'s have a

Figure 2.2: The average face for our training set.

Figure 2.3: The caricatures of the faces of figure 2.1.

zero mean, which is a useful property for some of the algorithms based on eigenfaces.

Another argument for subtracting the mean is that we are looking to characterise the variation between faces — not between general images. If we do not subtract the mean, the first singular value describes the average variation from the origin, which does not have much meaning in terms of the variation amongst the faces.

Figure 2.2 shows the average face calculated for our training set according to (2.2.3).

The non-normalized versions of (2.2.4) are sometimes referred to as the caricatures of the faces and denoted by

$$\boldsymbol{C}_j = \boldsymbol{F}_j - \boldsymbol{A}. \tag{2.2.7}$$

The caricatures for the two faces shown in figure 2.1 are shown in figure 2.3. Note how the individual characteristics are retained, even enhanced, by the caricatures.

We now need to find a basis for the space spanned by the $\boldsymbol{X}_j$'s. These orthonormal basis vectors will be the basic building blocks used to reconstruct

any face the system may encounter, whether it is in the training set or not. Thus we are looking for an orthonormal basis of the sub-space spanned by faces in the training set. If we define the matrix X with columns given by the faces (represented as column vectors, of course) in the training set, i.e.

$$X = \frac{1}{\sqrt{M}} \left[\begin{array}{cccc} \boldsymbol{X}_1 & \boldsymbol{X}_2 & \cdots & \boldsymbol{X}_M \end{array} \right] \tag{2.2.8}$$

where we normalise by $\frac{1}{\sqrt{M}}$ as a matter of convenience, then we are looking for an orthonormal basis of the column space of X. Accordingly we calculate the singular value decomposition (SVD) of X,

$$X = U\Sigma V^T \tag{2.2.9}$$

where U and V are orthonormal matrices and Σ is a diagonal, $D^2 \times M$ matrix containing the singular values of X. It is useful to recall the following properties of the singular value decomposition:

a. The singular values, σ_j, (the diagonal elements of Σ) are non-negative, ordered in decreasing order of magnitude. Let us suppose that there are exactly r non-zero singular values. Clearly $r \leq M$ since we assume that $M < D^2$. It is not difficult to show that X may be written as the sum of r rank-one matrices

$$X = \sum_{j=1}^{r} \sigma_j \boldsymbol{u}_j \boldsymbol{v}_j^T, \tag{2.2.10}$$

where the $\boldsymbol{u}_j$ and $\boldsymbol{v}_j$ are the columns of U and V respectively (note that we assume that we are dealing with real matrices throughout, since X is trivially real). This result implies that the zero singular values may be ignored since they carry no "information". Of course, since we assume the faces in the training set to be independent, none of the singular values will be zero. This does not, however, prevent a large number of singular values from becoming very small. In figure 2.4, we plot the graph of the singular values for our small demonstration training set (105 faces).

The most relevant feature of figure 2.4 is the fact that there is initially

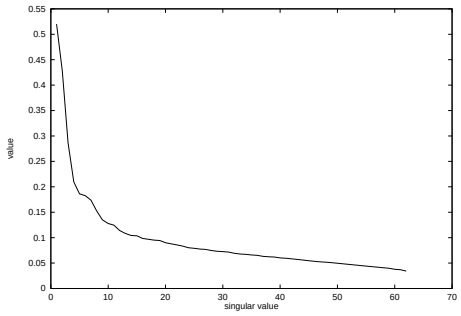


Figure 2.4: Graph of the singular values.

is rapid drop in the magnitude of the singular values, which are close to zero about 20 singular values. The following theorem indicates that singular values close to zero can be ignored. Specifically, if X_ν is the approximation of X obtained by keeping the first ν terms in (2.2.10), i.e.

$$X_\nu = \sum_{j=1}^{\nu} \sigma_j \boldsymbol{u}_j \boldsymbol{v}_j^T, \tag{2.2.11}$$

it is well known that (see e.g. [51])

$$\|X - X_\nu\|_2 = \sigma_{\nu+1}. \tag{2.2.12}$$

In this expression $\sigma_{\nu+1} = 0$ if $\nu = r$. It can be shown that X_ν is the best approximation to X in the L^2 sense over all $D^2 \times M$ matrices of rank less or equal to ν.

Thus if $\sigma_{\nu+1}$ is sufficiently small it is safe to keep only ν singular values. Henceforth we keep only ν non-zero singular values where ν is chosen large enough so that the error in the approximation (2.2.12) is negligible.

Referring back to figure 2.4, it would appear that all the faces in our training set can be adequately represented by about 20 eigenfaces. If the training set used for these experiments were really representative, the results summarized in figure 2.4 would indicate that all faces can be adequately represented by about 20 eigenfaces. Since our training set is not really representative, this is of course not the case. 40 eigenfaces are frequently used in the literature (see for example [47] or [53]).

Experiments with a heterogeneous population such as that of South Africa, indicates that a number of eigenfaces slightly in excess of 100 should suffice (see [3]). On the other hand, the results reported in [55] use only 20 eigenfaces for a success rate of about 95–98% accuracy in an *identification* problem. There is no doubt that an eigenface representation of facial images is highly efficient—the dimension of the facial sub-space is almost certainly less than 100, rather than D^2 as for a general $D \times D$ image.

b. It is easily seen from (2.2.9) that

$$XX^TU = U\Sigma\Sigma^T \text{ and } X^TXV = V\Sigma\Sigma^T. \tag{2.2.13}$$

Thus the columns of U are the eigenvectors of the symmetric matrix XX^T and similarly for V. The singular values are the positive square roots of the eigenvalues of XX^T or X^TX.

c. Since we assume only ν (essentially) non-zero singular values, the SVD may be written as

$$X \approx X_\nu = U_\nu \Sigma_+ V_\nu^T \tag{2.2.14}$$

where $\Sigma_+ = \text{ diag } (\sigma_1 \cdots \sigma_\nu)$, U_ν is an $D^2 \times \nu$ matrix consisting of the first ν columns of U and V_ν is a $M \times \nu$ matrix consisting of the first ν columns of V.

d. The columns of U form an orthonormal basis for the column space of X. From (2.2.14), we can see that the columns of U_ν forms a orthonormal basis for the subset of the column space which we consider important (i.e. the column space of X_ν).

Recall that we are interested in finding a basis for the sub-space spanned by all the facial images. Since the training set is supposed to be representative of all facial images, this means that a basis for the sub-space spanned by the training set should suffice. We have just observed that the columns of U_ν (denoted by $\boldsymbol{u}_j, \quad j = 1, \ldots, \nu$) provide an orthonormal basis for the columns space of X_ν. The $\boldsymbol{u}_j$ are the so-called eigenfaces. The sub-space spanned by these vectors is sometimes referred to as the "face space".

It may be appropriate to briefly say something about the calculation of the SVD. The main difficulty of course, is the fact that X is a very large matrix, due to its D^2 rows. This sort of decomposition occurs in many pattern recognition problems. The usual way in the pattern recognition literature (see for example [12]) of calculating the SVD in this sort of problem is to calculate V by solving the $M \times M$ eigenvalue problem given in (2.2.13). This is manageable since M is in general much smaller than D^2. Once V (and so V_ν) is known, U_ν is obtained from the SVD as

$$XV_\nu = U_\nu \Sigma_+.$$

Normalizing the ν columns of the product on the left hand side, gives the orthonormal basis we are looking for.

Although this procedure is often recommended for the calculation of the eigenfaces, there is a loss of information due to rounding error when $X^T X$ is formed (see [51] or [14]). The smallest singular values are the worst affected by this loss of information. Since small values of σ_j are neglected, in practice difficulties are rarely encountered. It is worth noting, however, that stable algorithms, exploiting the fact that $D^2 \gg M$, are available—at a greatly increased computational cost. Most standard algorithms to solve the eigenvalue problem are $O\left(M^3\right)$, while algorithms to solve the SVD problem are $O\left(\left(D^2\right)^2 M\right)$ (although, since we don't need all the columns of U, it is possible to use one of the truncated SVD algorithms, which are typically $O\left(\left(D^2\right) M^2\right)$). Since $D^2 \gg M$ (in almost all cases $M < 10D$), there is a considerable saving using the eigenvector based approach as opposed to the SVD. Equally important, the large size of D^2 can also impose considerable memory overhead on the SVD approach, which is considerable reduced using the smaller eigenvalue problem[2].

There is a useful geometric interpretation of the SVD that should be included in this discussion (see for example [54] or [51]). Given the faces in the training set, $\boldsymbol{X}_j$, let us try and find the maximum deviation, λ_1, as well as the direction of maximum deviation, $\boldsymbol{u}_1$, between the faces in the training

[2]Since the database calculation is not time-critical, in a practical implementation it is the memory overhead that is the deciding factor.

set, defined by,

$$\lambda_1 = \max_{\|\boldsymbol{u}_1\|_2=1} = \frac{1}{M}\sum_{n=1}^{M}\left(\boldsymbol{u}_1^T\boldsymbol{X}_n\right)^2. \tag{2.2.15}$$

Introducing the (biased) sample covariance matrix

$$C = \frac{1}{M}\sum_{j=1}^{M}\boldsymbol{X}_j\boldsymbol{X}_j^T \tag{2.2.16}$$

we rewrite (2.2.15) as

$$\lambda_1 = \max_{\|\boldsymbol{u}_1\|_2=1} = \boldsymbol{u}^T C\boldsymbol{u}_1. \tag{2.2.17}$$

We now look for the maximum deviation orthogonal to $\boldsymbol{u}_1$. Call this direction $\boldsymbol{u}_2$ and let λ_2 denote the amount of deviation in this direction. Similarly we define the pairs $(\boldsymbol{u}_3, \lambda_3)$, etc. It is not difficult to show using Lagrange multipliers (see [19]) that these quantities are exactly the eigenvalues and eigenvectors of the covariance matrix C (see [54] for an alternative proof). Thus we need to solve the eigenvalue problem,

$$C\boldsymbol{u} = \lambda\boldsymbol{u}. \tag{2.2.18}$$

Note that the covariance matrix C satisfies $C = XX^T$ with X given by (2.2.8). Thus the singular values are the positive square roots of the variances in the principle directions defined by the columns of U_ν or equivalently, the first ν eigenvectors of the covariance matrix C. Note that the eigenfaces are consistently denoted by $\boldsymbol{u}_j$. The singular values are given in terms of the eigenvalues of C by $\lambda_j = \sigma_j^2$.

It is instructive to display the eigenvectors of the covariance matrix visually. Figure 2.5 shows the first, third, twentieth and fortieth eigenvectors displayed as a two dimensional image.

For obvious reasons these are called eigenfaces and the complete set become the basic building blocks from which any arbitrary face will be constructed in future. An interesting feature of these eigenfaces is that the first two are general with few individual characteristics. The last two, on the

(a) First eigenface. (b) Third eigenface.

(c) Twentieth eigenface. (d) Fortieth eigenface.

Figure 2.5: Some of the eigenfaces.

other hand, display significant detailed information. This will be explained by the analysis of the next section.

2.3 An Analysis of the Eigenface Technique

The construction of the eigenfaces ensures that all the faces in the training set are well-represented by the eigenfaces. If the training set is truly representative then all faces presented to the system should be well-represented by the eigenfaces. In this section we derive properties of the facial sub-space that will ensure an efficient representation by the eigenfaces.

Our basic approach is that of perturbation theory. Consider a face that is not part of the training set. Projecting this face onto the sub-space spanned by the eigenfaces results in a projection error that must be small for the face to be well described. Of course for faces inside the training set the projection error will be close to zero, and the only reason why the error is not exactly

zero is because we discarded the smallest singular values. Imagine that a face that was not part of the training set is now included in the training set. Instead of trying to estimate the projection error, we study the changes in the eigenvalues and eigenvectors of the covariance matrix. Presumably, if the projection error is small, the inclusion of the new face in the training set causes only a small perturbation in the covariance matrix C and its eigenvalues and eigenvectors. Our goal is to find conditions ensuring that these perturbations are indeed small.

2.3.1 Small Perturbations of Symmetric Matrices.

Adding a new face to the training set presumably results in a small perturbation in the covariance matrix. The exact nature of the perturbation will be investigated in subsequent sections. To what extent does a small perturbation of a symmetric matrix affect the eigenvalues and eigenvectors? To answer this, we consider the perturbed eigenvalue problem

$$(C + \delta C)\,(\boldsymbol{u}_l + \delta \boldsymbol{u}_l) = (\lambda_l + \delta\lambda_l)(\boldsymbol{u}_l + \delta \boldsymbol{u}_l) \tag{2.3.1}$$

where δC is the known change in C, $\delta\lambda_l$ denotes the unknown change in λ_l and $\delta \boldsymbol{u}_l$ the unknown change in $\boldsymbol{u}_l$. Note that $\delta \boldsymbol{u}_l$ will be small only in the case of single eigenvalues. Consequently, these results are only valid for single eigenvalues.

Expanding (2.3.1) and assuming the perturbations to be small, we keep first order terms to obtain

$$(C - \lambda_l I)\,\delta \boldsymbol{u}_l = (-\delta C + \delta\lambda_l I)\,\boldsymbol{u}_l. \tag{2.3.2}$$

For a solution to exist, the right hand side must lie in the column space of $C - \lambda_l I$. Thus, by the Fredholm alternative, it is orthogonal to the null-space[3] of $C - \lambda_l I$. But the null-space of $C - \lambda_l I$ is the eigenvector $\boldsymbol{u}_l$. Since the eigenvectors are orthonormal, it follows that

[3]Since C is symmetric, so is $C - \lambda_l I$, and consequently the left and right null-spaces are identical.

$$\boldsymbol{u}_l^T \left(-\delta C + \delta\lambda_l I\right) \boldsymbol{u}_l = 0$$

or

$$\delta\lambda_l = \boldsymbol{u}_l^T \delta C \boldsymbol{u}_l. \tag{2.3.3}$$

This is an easily computable expression for the change in a single eigenvalue under a symmetric perturbation of symmetric matrices. Substituting this back into (2.3.1), one can use the pseudo-inverse to calculate the change in the eigenvectors. In practice this is not easily done, again because of the size of the matrix C ($D^2 \times D^2$).

Although these results give computable formulas for the first order corrections, more precise formulas are needed in order to analyse the system. Fortunately there is a wealth of material available for the perturbation of symmetric matrices (see for example [49] or [54]). Our first task is to obtain an expression for the perturbation, δC, in the covariance matrix due to the additional face in the training set. In the next section we show that the perturbation is symmetric and of rank one.

2.3.2 Deriving an Expression for the Perturbation

As mentioned previously, the problem of an arbitrary perturbation of a symmetric matrix has received a great deal of attention. We are, however, not dealing with an arbitrary perturbation. The fundamental assumption in our analysis is that the perturbation of the covariance matrix is caused by the addition of a *facial* image, a fact that is reflected in assumption (2.2.2). We now proceed to investigate the effect of the addition of an additional face—satisfying assumption (2.2.2)—to the training set.

Instead of dealing directly with the covariance matrix C, it is convenient to multiply C by the size of the training set (M) to obtain the renormalised matrix K, i.e.

$$K = MC.$$

Note that K has the same eigenvectors as C but its eigenvalues, μ, are scaled according to $\mu = M\lambda$ where λ is an eigenvalue of C.

If a new face is added to the training set, the matrix K is perturbed to become,

$$\widetilde{K} := K + \delta K = \sum_{n=1}^{M+1} \frac{\left(\boldsymbol{F}_n - \widetilde{\boldsymbol{A}}\right)\left(\boldsymbol{F}_n - \widetilde{\boldsymbol{A}}\right)^T}{\widetilde{d}^2}. \tag{2.3.4}$$

where

$$\widetilde{d}^2 = \frac{1}{M+1}\sum_{n=1}^{M+1}\left\|\boldsymbol{F}_n - \widetilde{\boldsymbol{A}}\right\|_2^2 \text{ and } \widetilde{\boldsymbol{A}} = \boldsymbol{A} + \delta\boldsymbol{A}. \tag{2.3.5}$$

$\boldsymbol{A}$ is the same average face as before and

$$\widetilde{\boldsymbol{A}} = \boldsymbol{A} + \delta\boldsymbol{A} = \frac{1}{M+1}\sum_{n=1}^{M+1}\boldsymbol{F}_n.$$

It follows that

$$\delta\boldsymbol{A} = \frac{1}{M+1}\left(\boldsymbol{F}_{M+1} - \boldsymbol{A}\right).$$

From (2.3.5) it follows in a straightforward manner that

$$\widetilde{d}^2 = d^2\left(1 - \frac{1}{M+1} + \frac{M}{d^2\left(M+1\right)^2}\left\|\boldsymbol{F}_{M+1} - \boldsymbol{A}\right\|_2^2\right). \tag{2.3.6}$$

Thus we can write

$$\widetilde{d}^2 = d^2\left(1+\epsilon\right)$$

where

$$\epsilon = -\frac{1}{M+1} + \frac{M}{d^2\left(M+1\right)}\left\|\boldsymbol{F}_{M+1} - \boldsymbol{A}\right\|_2^2. \tag{2.3.7}$$

If M is large enough and assumption (2.2.2) is satisfied, then $|\epsilon| < 1$,

allowing us to use a Taylor expansion in (2.3.4). Keeping first order terms in ϵ leads to

$$\delta K = \widetilde{\boldsymbol{X}}_{M+1} + \widetilde{\boldsymbol{X}}_{M+1}^T - \epsilon K, \tag{2.3.8}$$

where

$$\widetilde{\boldsymbol{X}}_{M+1} = \frac{1}{\widetilde{d}} \left(\boldsymbol{F}_{M+1} - \widetilde{\boldsymbol{A}} \right).$$

Although ϵ is a quantity that is easily calculated in practice, assumption (2.2.2) also implies that

$$\begin{aligned} \epsilon &< -\frac{1}{M+1} + \frac{MQ^2}{(M+1)^2} \\ &= \frac{1}{(M+1)^2} \left(M(Q^2-1) - 1 \right). \end{aligned}$$

Thus, if $Q = 1 + O(\frac{1}{M})$, ϵ is a second order in $\frac{1}{M}$ and can be ignored in the expression for δK. For this to be true one needs to make more stringent assumptions on the distribution of the faces around their mean. In particular, one has to assume that they are tightly clustered. Although we do not wish to impose these more stringent conditions, our experiments indicate that, for our test cases anyway, ϵ may indeed become sufficiently small that it may be ignored in practice.

2.3.3 Stability of the Eigenvalues

The main result of the previous section is that the perturbation in K is given by

$$\delta K = -\epsilon K + \varphi \varphi^T \tag{2.3.9}$$

where $\varphi\varphi^T$ is a symmetric rank one matrix with normalized coefficients, given by

$$\varphi\varphi^T = \widetilde{\boldsymbol{X}}_{M+1}\widetilde{\boldsymbol{X}}_{M+1}^T.$$

The first term on the right hand side of (2.3.9) introduces a uniform bias in the unperturbed eigenvalues without changing their relative magnitudes. Thus we concentrate on the contribution from the rank one perturbation.

If we denote the only non-zero (positive) eigenvalue of $\varphi\varphi^T$ by ξ and convert back to the original covariance matrix C, it follows readily (for the details the reader is referred to [54]) that the perturbed eigenvalues of C satisfy

$$\widetilde{\lambda}_j - \lambda_j = \frac{1}{M} m_j \xi, \tag{2.3.10}$$

where $0 \leq m_j \leq 1$ and $\sum_j m_j = 1$. Intuitively this is what one would expect– since ξ is positive, the addition of a face can only *increase* the magnitudes of the eigenvalues. Thus it tends to increase the degrees of freedom, i.e. the number of eigenfaces required. The magnitude of the perturbation depends of course on the magnitude of ξ, given by

$$\begin{aligned} \xi &= \widetilde{\boldsymbol{X}}_{M+1}^T \widetilde{\boldsymbol{X}}_{M+1} && (2.3.11)\\ &= \frac{1}{\widetilde{d}^2} \left\| \boldsymbol{F}_{M+1} - \widetilde{A} \right\|_2^2 && (2.3.12)\\ &< Q^2, && (2.3.13) \end{aligned}$$

where the last step is a result of assumption (2.2.2). The relative change in the eigenvalue is given by

$$\frac{\widetilde{\lambda}_j - \lambda_j}{\lambda_j} = \frac{m_j \xi}{M \lambda_j}.$$

Thus we need

$$\xi \ll M\lambda_j \tag{2.3.14}$$

to ensure a small relative change in the eigenvalues.

An immediate consequence of (2.3.10) is that the eigenvalues will con-

verge if the training set is enlarged with randomly distributed facial images satisfying condition (2.2.2). Although the convergence is of the order of $\frac{1}{M}$ the relative change in the perturbation also depends on the distribution of the faces, i.e. on the value of Q—for small values of Q, hence of ξ, the larger the number of eigenvalues that remain virtually unaffected by the perturbation and the more efficient the system becomes. In addition, since (2.3.13) and (2.3.14) require that

$$Q^2 \ll M\lambda_j, \tag{2.3.15}$$

it follows that we are able to accommodate reasonably large deviations from the mean for large values of M. Thus (2.3.15) provides considerable freedom in the distribution of the faces—we do not all need to look alike for the system to function efficiently!

2.3.4 Stability of the Eigenvectors

We now need to investigate how the eigenfaces are affected by the perturbation (2.3.9) in the covariance matrix. In this case the first term has no effect on the eigenfaces and can be ignored. The perturbation in the eigenvectors is most conveniently given in terms of the gap between the eigenvalue associated with the relevant eigenvector and the rest of spectrum of C (see for example [10]), defined by

$$\text{gap}(\lambda_i, C) = \min_{j \neq i} |\lambda_j - \lambda_i| .$$

If θ denotes the acute angle between the original ($\boldsymbol{u}_i$) and perturbed eigenfaces ($\widetilde{\boldsymbol{u}}_i$), then

$$\frac{1}{2}\sin 2\theta \leq \frac{(\xi/M)}{\text{gap }(\lambda_i, C)}, \tag{2.3.16}$$

provided that gap $(\lambda_i, C) > 0$. Also note that we are dealing with a perturbation in the original covariance matrix C, hence the presence of the factor $\frac{1}{M}$. Thus the perturbation in the eigenfaces are totally determined by the distribution of the eigenvalues. In particular, in the case of multiple eigenvalues, the gap is zero, so the change in the eigenvectors can be very large.

The efficiency of the eigenface approach depends on the fact that many of the eigenvalues are either zero or very close to zero, implying that very small gaps cannot be avoided. Eigenvectors associated with these small eigenvalues will be very sensitive to the smallest perturbation in the covariance matrix. We, however, are interested in the eigenfaces, and they are associated with nonzero eigenvalues. Since this implies finite gaps, (2.3.16) ensures a convergence result: the change in the eigenvectors decreases as $\frac{1}{M}$. As in the case of the eigenvalues, the size of the perturbation also depends on Q, the spread of the facial images round their mean. A tighter clustering results in smaller perturbations.

Of course this discussion is based on the perturbation of the individual eigenfaces whereas we really are only interested in the invariant subspace spanned by the eigenfaces. A classical result due to Stewart (see for example [50] or [14]) states that the sensitivity of a sub-space depends entirely on the separation of the eigenvalues associated with the different sub-spaces. This emphasises the need for a limited number of eigenfaces. Too many eigenfaces—those associated with small eigenvalues—may lead to an ill-defined gap in the eigenvalues. Consequently the invariant sub-space spanned by the eigenfaces become sensitive to small perturbations and less robust. It does make sense to use a small number of eigenfaces, as demonstrated by the system developed at the MIT Media Laboratory [55].

Expression (2.3.16) allows for an illuminating interpretation of the meaning of the eigenfaces. Referring back to figure 2.4, one notices that the gap between the largest singular values, and thus the largest eigenvalues, and the rest of the spectrum is significant. The eigenfaces associated with these singular values are therefore not sensitive to the addition of an additional face to the training set. One concludes that these eigenfaces carry general face-like information without much detail about individual characteristics, clearly illustrated by the first two eigenfaces in figure 2.5. On the other hand, the size of the gap decreases for the higher order singular values. The corresponding eigenfaces are therefore more sensitive to the addition of an additional face. One would therefore expect that these eigenfaces contain more detailed information about an individual, as is clearly illustrated by the last two eigenfaces in figure 2.5.

This fact has been exploited in [30] to develop efficient and robust algorithms for finding facial images in general scenes. Basically all that is

required to determine whether a sub-image is a face is to consider its representation by the few of the lowest eigenfaces. Since one is only interested in detecting the presence of a face, only the general characteristics are important and the individual characteristics described by the higher order eigenfaces are irrelevant.

Recalling our remark above concerning the robustness of the system, it is clear that a balance is required. On the one hand, the higher order eigenfaces are required in order to distinguish individual features. On the other hand, the inclusion of eigenfaces associated with small eigenvalues, decreases the robustness of the system.

2.3.5 Concluding Comments

The clustering condition (2.2.2) we use places, through the value of Q, a limit on the distribution of the faces that is acceptable to the system. Condition (2.3.15) on Q indicates that the distribution of the faces needs not be particularly tightly clustered for the eigenface representation to work well. Equation (2.3.16), however, indicates that a smaller value for Q (and hence ξ) allows for larger gaps between the eigenfaces for the same magnitude of perturbation. Thus, the tighter the clustering about the mean, the fewer eigenfaces required, and the more efficient the representation.

Although the experimental evidence indicates that faces satisfy the clustering condition sufficiently well for the eigenface technique to be successful (i.e. Q not too large), there are many other potential applications which this will not be true. In such cases, it is possible to apply an automatic clustering algorithm to sub-divide the images into smaller sets, each of which individually satisfies the appropriate condition. In earlier work, we showed such an approach allows us to use a "eigenletter"-based system for Optical Character Recognition (see [34] or [32] for details).

The case for clustering faces is not as strong. Not surprisingly, experiments using an arbitrary sub-division based on race, gender and other such features (see [3]) show no improvement in performance. On the other hand, the analysis indicates that tightly clustered images are more efficiently represented. Thus it is worth examining whether there are any significant benefits to be gained from dividing facial images into a small number of clusters. Subdividing a large database will also have significant computational benefits when searching for a matching face (see [45]). We return to this topic

later (see Chapter 8).

2.4 Updating the Eigenfaces

In the previous section we noted that the addition of a new face to the training set will have negligible effect provided a clustering condition is satisfied (we needed ξ or Q to be small). In the construction of the training set, one can easily imagine a face that is not well represented by the eigenfaces calculated from the existing training set, i.e. ξ is not small enough so that the perturbations can be ignored. In this case it becomes necessary to change the training set (to include this new face), which implies changing the eigenfaces. Of course it is always possible to do a full recalculation of the eigenfaces, however, it is possible to derive a more efficient algorithm that *updates* rather than *recalculates* the eigenfaces. We presented a similar algorithm in [31], but that was derived under considerably more stringent assumptions, requiring, amongst other conditions, that the faces satisfied a Gaussian distribution.

Keeping in mind that we are only interested on facial images, characterized by the clustering condition (2.2.2), it follows that the perturbation of the covariance matrix will have the same form as before, given by (2.3.9). The main difference being that the perturbation is no longer small enough so that its effect on the (higher order) eigenfaces may be ignored. Another issue that is important, is the presence of the first term $(-\epsilon K)$ on the right hand side of (2.3.9). Before it was largely ignored because it has no influence on the eigenfaces and introduces only a systematic bias in the eigenvalues. Since we do not want to assume that it is negligibly small, it must be taken into account in our calculation of the actual eigenvalues.

Recall from (2.3.9) that the perturbed covariance matrix is given by

$$\widetilde{K} = (1-\epsilon)\, K + \varphi\varphi^T = (1-\epsilon)\, K + \widetilde{\boldsymbol{X}}_{M+1}\widetilde{\boldsymbol{X}}_{M+1}^T. \tag{2.4.1}$$

Although this expression provides us with a formula for the perturbation in the covariance matrix, it is large, of dimension $D^2 \times D^2$ to be precise. The computational cost can again be reduced significantly by noting that, since we are only dealing with ν significant eigenfaces, K is only of rank ν. In that case K can be written as

$$K \approx \mu_1 \boldsymbol{u}_1 \boldsymbol{u}_1^T + \mu_2 \boldsymbol{u}_2 \boldsymbol{u}_2^T + \cdots + \mu_\nu \boldsymbol{u}_\nu \boldsymbol{u}_\nu^T$$

where $\mu_n = M\lambda_n$. Using (2.4.1) we obtain

$$\begin{aligned} K + \delta K &\approx (\mu_1 - \epsilon)\boldsymbol{u}_1 \boldsymbol{u}_1^T + (\mu_2 - \epsilon)\boldsymbol{u}_2 \boldsymbol{u}_2^T + \cdots + (\mu_\nu - \epsilon)\boldsymbol{u}_\nu \boldsymbol{u}_\nu^T + \varphi\varphi^T \\ &= SS^T \end{aligned}$$

where

$$S = \left[\begin{array}{cccc} \left(\sqrt{(\mu_1 - \epsilon)}\right)\boldsymbol{u}_1 & \cdots & \left(\sqrt{(\mu_\nu - \epsilon)}\right)\boldsymbol{u}_\nu & \varphi \end{array} \right].$$

Again the properties of the SVD allow us to obtain the eigenfaces from the eigenvectors of

$$L = S^T S.$$

Since the eigenfaces ($\boldsymbol{u}$'s) are orthonormal, it follows easily that

$$L = \left[\begin{array}{cccc} (\mu_1 - \epsilon) & \cdots & 0 & \sqrt{\mu_1 - \epsilon}\boldsymbol{u}_1^T\varphi \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & (\mu_\nu - \epsilon) & \sqrt{\mu_\nu - \epsilon}\boldsymbol{u}_\nu^T\varphi \\ \sqrt{\mu_1 - \epsilon}\boldsymbol{u}_1^T\varphi & \cdots & \sqrt{\mu_\nu - \epsilon}\boldsymbol{u}_\nu^T\varphi & \varphi^T\varphi \end{array} \right].$$

This matrix is only $(\nu + 1) \times (\nu + 1)$ and has a structure that can be exploited numerically. It is straightforward to obtain estimates for the change in the eigenvalues from (2.3.3). The corresponding eigenvectors of L can then be calculated using any of the standard numerical methods. The extremely sparse structure of L allows for an exceptionally efficient implementation of most popular methods for calculating eigenvectors, typically $O(\nu + 1)$. For large databases, it can be more efficient to repeatedly update the eigenfaces rather than calculate the full set (see our earlier comments (Section 2.2) on calculating the eigenfaces), however this method gives a good approximation to the eigenfaces, rather than the true solution. The error will grow over re-

peated applications of the algorithm. Since the calculation is not a repeated process, the computational cost is comparatively unimportant in our experiments, so we have chosen to calculate the eigenfaces directly throughout. For very large databases the greater efficiency of this updating algorithm can however become significant, provided the error does not become significant.

Chapter 3

Experiments with Eigenface

3.1 An Important Point

As discussed in the Introduction, we are not trying to create a complete recognition system. Thus, we will not conduct any recognition experiments. The ability of the eigenface-based systems to recognise people is based on the compact representation of the face provided by the eigenfaces, so we will focus on experiments which demonstrate this.

Because we have not focused on recognition, our database is also unsuited to recognition experiments. Testing recognition performance requires a careful experimental setup. Both the test and training sets used have to be chosen so that the variations that we would expect to encounter are well represented. We also need to ensure that all the images are carefully labelled. Furthermore, a large test set is required. Acquiring such a database of images is a long term process and also requires a well-defined image acquisition process. We do not currently have access to such a database and are unfortunately not in a position to create our own.

Also, there are many different ways of comparing two images. For instance, in the FERET study [40], comparison rules ranging from a simple Euclidean distance to the multi-levelled eigenfeatures approach of the MIT Media Lab [28] were used for eigenface-based methods, so providing results which will reflect real-world performance is somewhat difficult.

3.2 Using Eigenfaces to Represent Images

We define the eigenface representation as the best possible approximation to the face in terms of the eigenfaces. As in the previous chapter, we will use ν eigenfaces, denoted as $\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_\nu$. Then, given a set of numbers $y_1, y_2, \ldots y_\nu$ so that

$$\widetilde{\boldsymbol{F}} = y_1\boldsymbol{u}_1 + y_2\boldsymbol{u}_2 + \ldots + y_\nu\boldsymbol{u}_\nu + \boldsymbol{A}$$

minimises

$$\epsilon\left(\boldsymbol{F}\right) = \left\|\boldsymbol{F} - \widetilde{\boldsymbol{F}}\right\|_2, \tag{3.2.1}$$

these y's are called the eigenface coefficients and $\widetilde{\boldsymbol{F}}$ is called the eigenface representation.

Obviously, given the vector

$$\boldsymbol{y} = \begin{bmatrix} y_1 & y_2 & \ldots & y_\nu \end{bmatrix}$$

then

$$\widetilde{\boldsymbol{F}} = U_\nu\boldsymbol{y} + \boldsymbol{A}$$

where U_ν is given by (2.2.14). So we only need to store the ν eigenface coefficients to completely describe $\widetilde{\boldsymbol{F}}$. Since ν is small (for these examples, ν is 40 or less), this representation is vastly more efficient than storing the original image.

Furthermore, we can easily see that the error in the representation $\Delta\boldsymbol{F} = \boldsymbol{F} - \widetilde{\boldsymbol{F}}$ is orthogonal to all the $\boldsymbol{u}_i$'s. Also, since U_ν has orthonormal columns, it follows that $U_\nu^T U_\nu$ is the identity matrix, so

$$\boldsymbol{y} = U_\nu^T\left(\widetilde{\boldsymbol{F}} - \boldsymbol{A}\right).$$

But $\boldsymbol{F} = \widetilde{\boldsymbol{F}} + \Delta\boldsymbol{F}$ and $U_\nu^T\left(\Delta\boldsymbol{F}\right) = 0$, so it follows immediately that

$$\boldsymbol{y} = U_\nu^T\left(\boldsymbol{F} - \boldsymbol{A}\right). \tag{3.2.2}$$

Because (3.2.2) is a projection, we lose information, but if $\boldsymbol{F}$ is a face sim-

ilar to the original training set, then $\epsilon(\boldsymbol{F})$ is small, so this loss of information is not significant.

This also suggests that calculating $\epsilon(\boldsymbol{F})$ is a useful measure of how good the representation is. We define $\boldsymbol{C_F} = \boldsymbol{F} - \boldsymbol{A}$ and $\widetilde{\boldsymbol{C_F}} = \widetilde{\boldsymbol{F}} - \boldsymbol{A}$. Then (3.2.1) becomes $\epsilon(\boldsymbol{F}) = \left\| \boldsymbol{C_F} - \widetilde{\boldsymbol{C_F}} \right\|_2$ (since the $\boldsymbol{A}$ terms cancel). Since this measure requires calculating $\widetilde{\boldsymbol{C_F}}$, it is computationally expensive. Calculating $\|\boldsymbol{C_F}\|_2$ is easy since we obtain $\boldsymbol{C_F}$ in calculating the representation, and calculating the length involves only D^2 multiplications. To calculate $\left\|\widetilde{\boldsymbol{C_F}}\right\|_2$ requires that we multiply by the matrix $U_\nu U_\nu^T$, which is a much more expensive operation. But $\boldsymbol{C_F} - \widetilde{\boldsymbol{C_F}}$ is orthogonal to $\widetilde{\boldsymbol{C_F}}$. Thus by Pythagoras' theorem, we can write

$$\epsilon^2(\boldsymbol{F}) = \|\boldsymbol{C_F}\|_2^2 - \left\|\widetilde{\boldsymbol{C_F}}\right\|_2^2.$$

From our definitions, it follows that

$$\begin{aligned} \left\|\widetilde{\boldsymbol{C_F}}\right\|_2^2 &= \left\|\widetilde{\boldsymbol{F}} - \boldsymbol{A}\right\|_2^2 \\ &= \|U_\nu \boldsymbol{y}\|_2^2 \\ &= \boldsymbol{y}^T U_\nu^T U_\nu \boldsymbol{y}. \end{aligned}$$

But, as mentioned earlier, $U_\nu^T U_\nu$ is the identity matrix. Therefore

$$\left\|\widetilde{\boldsymbol{C_F}}\right\|_2^2 = \|\boldsymbol{y}\|_2^2.$$

So $\epsilon(\boldsymbol{F})$ is given by

$$\epsilon^2(\boldsymbol{F}) = \|\boldsymbol{C_F}\|_2^2 - \|\boldsymbol{y_F}\|_2^2, \tag{3.2.3}$$

an expression which is considerably easier to evaluate.

$\epsilon(\boldsymbol{F})$ is commonly known as the "distance from face space" or DFFS measure. It is sometimes scaled by $\|\boldsymbol{C_F}\|_2$, to convert it to the relative error: $\epsilon_r = \frac{\epsilon(\boldsymbol{F})}{\|\boldsymbol{C_F}\|_2}$.

Obviously obtaining the representation involves a single matrix multiplication, i.e. $O\left(\nu D^2\right)$, so it is fairly inexpensive, especially since ν is small. For recognition systems, the cost of comparing two representations depends entirely on the method of comparison used as well as the method for search-

Figure 3.1: Face in training set.

ing the database.. Minimising the distance between representations costs $O(\nu)$, but a simple linear search of a database of size N will require N comparisons. In applications where the database is very large, this is impractical. For very large databases, clustering can again be beneficial since it will reduce the number of comparisons needed considerably.

3.3 A Representation Example

We consider the face shown in figure 3.1. This face is a second image of someone in the training set (i.e. although this image itself is not in the training set, it is very similar to one that is), so we would expect it to be well described by the eigenfaces. Figure 3.2 shows the face represented by progressively more eigenfaces. As we can see, using 10 eigenfaces, we capture not much more than the basic outline, while the detail improves dramatically when we increase the number of eigenfaces to 20. Going from 20 to 40 eigenfaces gives only a comparatively small improvement. This provides visual confirmation of our earlier conclusions.

3.4 Faces Do Not Occupy a Linear Sub-Space

It is worth emphasizing that, although the calculations in this chapter have involved treating the faces as a linear sub-space, this is not the case. The condition (2.2.2) is important. The faces may lie within a linear sub-space, but they do not form a linear sub-space themselves. To demonstrate, consider figure 3.3. This is clearly not a face. It was, however, constructed by choosing random coefficients for the eigenfaces (controlled so that the resulting gray scale values are between 0 and 255), so it is perfectly described by

(a) 10 eigenfaces (b) 20 eigenfaces (c) 40 eigenfaces

Figure 3.2: Various reconstructions of figure 3.1.

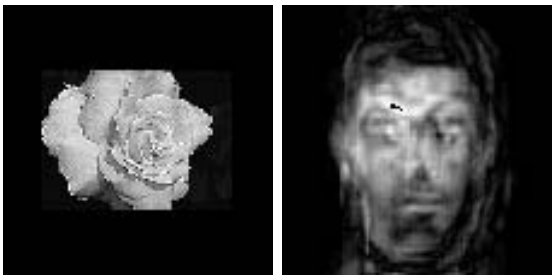
Figure 3.3: Non-face described by the eigenfaces.

the eigenfaces. It merely fails to satisfy (2.2.2), being far from the average. Thus a small projection error is not sufficient evidence to conclude that the image is a face, so we need the additional clustering condition.

On the other hand, we cannot ignore the projection error. The linear space within which the faces lie is a subset of all possible images. We can project any image onto this space, but we are only ensured a good reconstruction if the image is sufficiently similar to the training set to be close to our estimated "face space". Consider the rose shown in figure 3.4(a). Since this differs markedly from a face, we would expect a large error in the reconstruction, and indeed this is what we observe in figure 3.4(b) (reconstructed using 40 eigenfaces).

3.5 An Example of Updating the Eigenfaces

Earlier, we described an algorithm for efficiently recalculating the eigenfaces. To demonstrate it in practice, we present a simple test case. For this section,



(a) The rose. (b) And its reconstruction.

Figure 3.4: A rose is not a face.

(a) Origianl face (b) Reconstruction

Figure 3.5: A face not well described by the training set.

we have used 30 eigenfaces for the reconstructions. We use an initial training set without a single example of a person with a beard or glasses. If we then try and represent the face shown in figure 3.5(a), we should not be surprised that the reconstruction (shown in figure 3.5(b)) is rather poor.

The new image, however, is sufficiently close that its essential features are described (unlike the case with the rose). The reconstruction has more or less the correct basic shape and manages to describe most of shading correctly. Thus it would seem a good candidate for the updating algorithm. Updating the eigenfaces as described earlier gives the reconstruction shown in figure 3.6(a).

Obviously, we need to examine how this compares with the eigenfaces obtained from a full recalculation. For comparison, the recalculation of the new face using the fully recalculated eigenfaces is shown in figure 3.6(b). As

(a) Updated algorithm

(b) Full recalculation

Figure 3.6: Reconstruction using new eigenfaces.

(a) Updated eigenfaces

(b) Fully recalculated eigenfaces

Figure 3.7: A face in the original training set.

can be seen, there is virtually no visible difference between figures 3.6(a) and 3.6(b). The difference between the two images doesn't exceed 1 gray scale level (approximately 0.3% error) at any point.

How does the recalculation affect a face already in the training set? In figure 3.7, we show one of the faces from figure 2.1 represented using the updated and fully recalculated eigenfaces respectively. Again there is virtually no visible difference. Indeed, the difference between the reconstructions using the fully recalculated eigenfaces and the updated ones is never more than 0.1 of a gray scale level for images in the original training set (approximately 0.03% error) in this case (with round off errors in displaying the image, this can grow to 1 gray scale level) . Clearly the updated approximation is very close.

We have had good results repeating the updating process several times[1], indicating that the error grows quite slowly, but without tighter assumptions on the nature of the new face, we cannot provide useful bounds on the rate of error growth.

[1]We have tried adding up to 10 faces to a small set (around 50) of faces with no visible drop in performance. Experiments with the "eigenletter"-based OCR system mentioned earlier (see [34]) also indicate that the error grows slowly.

Chapter 4

Locating the Region of Interest

4.1 Distance From Face Space

The simplest method for locating a region of interest (in our case the face) in an image is simple template-based matching, which takes the correlation between the image and a template and places the location at the resulting maximum. Although this is fast, it is not very accurate as a single template can describe only a small fraction of the variation we would expect to encounter. The eigenfaces from our training set should describe this variation, so one would expect a technique based on the eigenfaces to perform better.

The first eigenface-based method used for locating the face in an image is to find the region in the image that has the smallest projection error, known as the "distance from face space" or DFFS algorithm (see [53] and [37]). This completely ignores the fact that the face space is not a linear sub-space, and fails badly when confronted with images like figure 3.3. In situations where there is only one face in the image, however, it does perform reasonably well. Not surprisingly, it is more expensive than the template-based search. If we use the projection based on P eigenfaces, we need P correlations to find the eigenface coefficients at each point, plus additional multiplications to calculate the projection error, making it at least P times more expensive than template matching.

4.2 A Maximum Likelihood Algorithm

We now describe a statistical estimate, developed by Moghaddam and Pentland (see [30] or [29]), of how face-like an image is which can be used to find the location of a face in a general image. We assume that the facial images are samples from a multivariate Gaussian (normal) distribution in the total image space Ω. Since the matrix C is defined as the covariance matrix of the $\boldsymbol{X}_n$'s, it follows from the definition of the multivariate Gaussian distribution that

$$\mathrm{P}\left(\boldsymbol{X}|\Omega\right) = \frac{\mathrm{e}^{-\frac{1}{2}\left(\boldsymbol{X}^T C^{-1}\boldsymbol{X}\right)}}{(2\pi)^{\frac{N}{2}}\,|C|^{\frac{1}{2}}}. \tag{4.2.1}$$

where N is the dimension of our image space, i.e. $N = D^2$ ($\approx 100^2$ in our earlier examples), and $\boldsymbol{X} = \frac{\boldsymbol{F}-\boldsymbol{A}}{d}$ for some image $\boldsymbol{F}$.[1]

A sufficient statistic for this probability is

$$\mathrm{d}\left(\boldsymbol{X}\right) = \boldsymbol{X}^T C^{-1}\boldsymbol{X}\,.$$

Furthermore, we know that C can be written as

$$C = U^T \Lambda U$$

where U is defined as containing *all* the eigenvectors (i.e. is of full rank). Then

$$\begin{aligned}\mathrm{d}(\boldsymbol{X}) = \boldsymbol{X}^T C^{-1}\boldsymbol{X} &= \boldsymbol{X}^T U^T \Lambda^{-1} U\boldsymbol{X} \\ &= \boldsymbol{y}^T \Lambda^{-1}\boldsymbol{y}.\end{aligned}$$

But Λ^{-1} is diagonal (since Λ is diagonal) so the expression simplifies to

$$\begin{aligned}\mathrm{d}(\boldsymbol{X}) &= \sum_{i=1}^{N}\frac{y_i^2}{\lambda_i} \\ &= \sum_{i=1}^{P}\frac{y_i^2}{\lambda_i} + \sum_{i=P+1}^{N}\frac{y_i^2}{\lambda_i}\end{aligned}$$

[1] We use the notation $\boldsymbol{X}$ conditional on Ω ($\mathrm{P}\left(\boldsymbol{X}|\Omega\right)$) to indicate that we are dealing with the images in relation to the entire image space, and not the subspace described by the eigenfaces.

where P is the number of eigenvectors we keep for reconstruction purposes. Since we are dealing with a location problem, the required accuracy is less than that used for reconstruction, thus typically $P < \nu$.

This allows us to write

$$\mathrm{P}\left(\boldsymbol{X}|\Omega\right) = \mathrm{P}_{F}\left(\boldsymbol{X}|\Omega\right)\mathrm{P}_{\widetilde{F}}\left(\boldsymbol{X}|\Omega\right) \tag{4.2.2}$$

where

$$\mathrm{P}_{F}\left(\boldsymbol{X}|\Omega\right) = \frac{\mathrm{e}^{-\frac{1}{2}\left(\sum_{i=1}^{P}\frac{y_i^2}{\lambda_i}\right)}}{(2\pi)^{\frac{P}{2}}\prod_{i=1}^{P}\lambda_i^{\frac{1}{2}}}$$

and

$$\mathrm{P}_{\widetilde{F}}\left(\boldsymbol{X}|\Omega\right) = \frac{\mathrm{e}^{-\frac{1}{2}\left(\sum_{i=P+1}^{N}\frac{y_i^2}{\lambda_i}\right)}}{(2\pi)^{\frac{N-P}{2}}\prod_{i=P+1}^{N}\lambda_i^{\frac{1}{2}}}.$$

Here $\mathrm{P}_{F}(\boldsymbol{X}|\Omega)$ is the true distribution in "face space", while $\mathrm{P}_{\widetilde{F}}(\boldsymbol{X}|\Omega)$ is the distribution in the space orthogonal to "face space" (i.e. $F \perp \widetilde{F}$).

This measure $\mathrm{d}(\boldsymbol{X})$ requires that we store all the eigenvectors and eigenvalues, not just those we would usually keep for reconstruction. We therefore look for an approximation $\widehat{\mathrm{d}}\left(\boldsymbol{X}\right)$, which uses only the available eigenfaces. Accordingly, consider

$$\widehat{\mathrm{d}}(\boldsymbol{X}) = \sum_{i=1}^{P}\frac{y_i^2}{\lambda_i} + \frac{1}{\rho}\left(\sum_{j=P+1}^{N} y_j^2\right)$$

where ρ is chosen to minimise the error.

Since $\boldsymbol{y}$ represents the image exactly (because U is of full rank), it follows from $\boldsymbol{y} = U\boldsymbol{X}$ and the orthonormality of U that

$$\left\|\boldsymbol{X}\right\|_2 = \left\|\boldsymbol{y}\right\|_2.$$

Therefore

$$\begin{aligned}\epsilon^2(\boldsymbol{X}) &= \|\boldsymbol{y}\|_2^2 - \sum_{i=1}^{P} y_i^2 \\ &= \sum_{i=1}^{N} y_i^2 - \sum_{i=1}^{P} y_i^2 \\ &= \sum_{i=P+1}^{N} y_i^2\end{aligned}$$

and

$$\widehat{\mathrm{d}}(\boldsymbol{X}) = \sum_{i=1}^{P} \frac{y_i^2}{\lambda_i} + \frac{1}{\rho}\left(\epsilon^2(\boldsymbol{X})\right).$$

We can now estimate $\mathrm{P}(\boldsymbol{X}|\Omega)$ as the product of two independent Gaussian distributions. Substituting $\widehat{\mathrm{d}}(\boldsymbol{X}|\Omega)$ into (4.2.1), it follows that

$$\begin{aligned}\widehat{\mathrm{P}}(\boldsymbol{X}|\Omega) &= \left[\frac{\mathrm{e}^{-\frac{1}{2}\left(\sum_{j=1}^{P}\frac{y_j^2}{\lambda_j}\right)}}{(2\pi)^{\frac{P}{2}}\left(\prod_{i=1}^{P}\lambda_i^{\frac{1}{2}}\right)}\right] \times \left[\frac{\mathrm{e}^{-\frac{\epsilon^2(\mathbf{X})}{2\rho}}}{(2\pi\rho)^{\frac{N-P}{2}}}\right] \\ &= \mathrm{P}_F(\boldsymbol{X}|\Omega)\,\widehat{\mathrm{P}}_{\widetilde{F}}(\boldsymbol{X}|\Omega), \qquad (4.2.3)\end{aligned}$$

where $\widehat{\mathrm{P}}_{\widetilde{F}}(\boldsymbol{X}|\Omega)$ is the estimate for $\mathrm{P}_{\widetilde{F}}(\boldsymbol{X}|\Omega)$.

We need to choose ρ so that this estimate for $\mathrm{P}(\boldsymbol{X}|\Omega)$ is as accurate as possible. Moghaddam and Pentland calculated the optimal value of ρ by minimising

$$\mathrm{J}(\rho) = E\left(\log\frac{\mathrm{P}(\boldsymbol{X}|\Omega)}{\widehat{\mathrm{P}}(\boldsymbol{X}|\Omega)}\right).$$

This measure comes from information theory and is known as the Kullman-Leibler divergence, also referred to as the informational divergence or information for discrimination. It provides a measure of the error made by assuming that the true distribution is $\widehat{\mathrm{P}}(\boldsymbol{X}|\Omega)$ when it is actually $\mathrm{P}(\boldsymbol{X}|\Omega)$. It can be shown that $\mathrm{J}(\rho)$ is always non-negative and that the larger $\mathrm{J}(\rho)$, the less the correspondence between the distributions, and thus the less accurate the results obtained by using $\widehat{\mathrm{P}}(\boldsymbol{X}|\Omega)$ instead of $\mathrm{P}(\boldsymbol{X}|\Omega)$ (see [7]

for more details). Thus minimising $\mathrm{J}(\rho)$ should optimise our approximate distribution $\widehat{\mathrm{P}}(\mathbf{X}|\Omega)$.

Substituting (4.2.2) and (4.2.3) into the definition of $\mathrm{J}(\rho)$, we see that

$$\begin{aligned}
\mathrm{J}(\rho) &= E\left(\log \frac{\mathrm{P}_{\widetilde{F}}(\boldsymbol{X}|\Omega)}{\widehat{\mathrm{P}}_{\widetilde{F}}(\boldsymbol{X}|\Omega)}\right) \\
&= E\left(\log\left(\frac{\exp\left(-\frac{1}{2}\sum_{i=P+1}^{N}\frac{y_i^2}{\lambda_i}\right)}{(2\pi)^{\frac{N-P}{2}}\prod_{i=P+1}^{N}\sqrt{\lambda_i}} \times \frac{(2\pi\rho)^{\frac{N-P}{2}}}{\exp\left(-\frac{1}{2}\sum_{i=P+1}^{N}y_i^2\right)}\right)\right) \\
&= E\left(\log\left(\frac{\rho^{\frac{N-P}{2}}\exp\left(-\frac{1}{2}\sum_{i=P+1}^{N}\frac{y_i^2}{\lambda_i}\right)}{\exp\left(-\frac{1}{2}\sum_{i=P+1}^{N}y_i^2\right)\prod_{i=P+1}^{N}\lambda_i^{\frac{1}{2}}}\right)\right) \\
&= E\left(\log\left(\left(\prod_{i=P+1}^{N}\frac{\rho}{\lambda_i}\right)^{\frac{1}{2}}\left(\mathrm{e}^{-\frac{1}{2}\left(\sum_{i=P+1}^{N}\left(\frac{y_i^2}{\lambda_i}-\frac{y_i^2}{\rho}\right)\right)}\right)\right)\right) \\
&= E\left(\frac{1}{2}\left(\sum_{i=P+1}^{N}\log\frac{\rho}{\lambda_i}\right) - \frac{1}{2}\left(\sum_{i=P+1}^{N}\left(\frac{y_i^2}{\lambda_i}-\frac{y_i^2}{\rho}\right)\right)\right) \\
&= \frac{1}{2}\left(\sum_{i=P+1}^{N}\left(\log\frac{\rho}{\lambda_i} - \left(\frac{E\left(y_i^2\right)}{\lambda_i} - \frac{E\left(y_i^2\right)}{\rho}\right)\right)\right). \qquad (4.2.4)
\end{aligned}$$

Since the mean of the $\boldsymbol{X}$'s is zero, it can easily be shown that

$$\begin{aligned}
E\left(y_i^2\right) &= \mathrm{Var}\left(y_i\right) \\
&= \lambda_i.
\end{aligned}$$

where the last equality follows because the eigenfaces diagonalise the covariance matrix. Thus (4.2.4) becomes

$$\begin{aligned}
\mathrm{J}(\rho) &= \frac{1}{2}\sum_{i=P+1}^{N}\left(\log\frac{\rho}{\lambda_i} - \left(\frac{\lambda_i}{\lambda_i} - \frac{\lambda_i}{\rho}\right)\right) \\
&= \frac{1}{2}\sum_{i=P+1}^{N}\left(\frac{\lambda_i}{\rho} - 1 + \log\frac{\rho}{\lambda_i}\right)
\end{aligned}$$

and consequently

$$\frac{\delta \mathrm{J}(\rho)}{\delta \rho} = \frac{1}{2} \sum_{i=P+1}^{N} \left(-\frac{\lambda_i}{\rho^2} + \frac{1}{\rho} \right).$$

To minimise $\mathrm{J}(\rho)$, we require

$$\sum_{i=P+1}^{N} \left(1 - \frac{\lambda_i}{\rho} \right) = 0$$

which amounts to

$$\rho = \frac{1}{N-P} \sum_{i=P+1}^{N} \lambda_i. \tag{4.2.5}$$

Thus ρ is simply the average of the eigenvalues we ignore—clearly a reasonable choice. Since we can only calculate at most M non-zero eigenvalues, we assume for simplicity that $\lambda_i = 0 \, \forall \, i > M$.

The final form of $\widehat{\mathrm{d}}(\boldsymbol{X})$ is therefore given by

$$\begin{aligned} \widehat{\mathrm{d}}(\boldsymbol{X}) &= \sum_{i=1}^{P} \frac{y_i^2}{\lambda_i} + \frac{N-P}{\sum_{i=P+1}^{N} \lambda_i} \left(\sum_{i=P+1}^{N} y_i^2 \right) \\ &= \sum_{i=1}^{P} \frac{y_i^2}{\lambda_i} + \frac{N-P}{\sum_{i=P+1}^{N} \lambda_i} \epsilon^2(\boldsymbol{X}). \end{aligned} \tag{4.2.6}$$

Thus rather than needing all N eigenvectors and eigenvalues, we need only P eigenvectors and eigenvalues and ρ, which can easily be calculated when we calculate the eigenvalues.

We now have enough information to calculate the estimated probability $\widehat{\mathrm{P}}(\boldsymbol{X}|\Omega)$. To convert this to the maximum likelihood estimator, we define $\boldsymbol{x}_{ij}$ as a section of the image centred at (i, j). This is treated as a full size image. Our likelihood estimate is $S_{ij} = \widehat{\mathrm{P}}(\boldsymbol{x}_{ij}|\Omega)$. The maximum likelihood then represents the region which we identify as containing the face. We can handle a multi-scale search by varying the size of $\boldsymbol{x}_{ij}$.

Since $\widehat{\mathrm{d}}(\boldsymbol{X})$ is a sufficient statistic for $\widehat{\mathrm{P}}(\boldsymbol{X}|\Omega)$, we need only calculate $\widehat{\mathrm{d}}(\boldsymbol{X})$, rather than the entire probability expression (this is standard practice, see e.g. [43]). Since we are looking for that region which has the highest

probability of being a face, we want to maximise $\widehat{\mathrm{P}}\left(\boldsymbol{X}|\Omega\right)$. We note that

$$\widehat{\mathrm{P}}\left(\boldsymbol{X}|\Omega\right) = \frac{\mathrm{e}^{-\frac{\widehat{\mathrm{d}}(\boldsymbol{X})}{2}}}{(2\pi)^{\frac{N}{2}}\left(\prod_{i=1}^{P}\lambda_i^{\frac{1}{2}}\right)\left(\rho^{\frac{N-P}{2}}\right)}. \tag{4.2.7}$$

This is obviously a maximum if $\widehat{\mathrm{d}}\left(\boldsymbol{X}|\Omega\right)$ is a minimum. So to maximise $\widehat{\mathrm{P}}\left(\boldsymbol{X}|\Omega\right)$, we need to minimise $\widehat{\mathrm{d}}\left(\boldsymbol{X}\right)$.

As mentioned earlier, since we are only dealing with the most basic face-like characteristics of an image, we do not need the same degree of accuracy here as for the reconstruction of the faces. Since the first 10 eigenfaces describe the basic shape of the face, we can use $P = 10$ without adversely affecting performance.

Although the image in figure 3.3 is constructed from the eigenfaces, it does not satisfy the normality condition. Thus it will not be mistaken for a face when searching through an image.

4.3 Efficiency of the Location Algorithms

While the accuracy of these algorithms can only be quantified by experimental evidence, we can at least analyse the numerical efficiency. For simplicity we only consider multiplications and divisions.

For the analysis, it is useful to define the $.*$ product (a notation inspired by MATLAB) of two N-dimensional vectors as the N-dimensional vector formed by multiplying each element of $\boldsymbol{a}$ with the corresponding element of $\boldsymbol{b}$, i.e.

$$\boldsymbol{a} \mathbin{.*} \boldsymbol{b} = \begin{bmatrix} a^1b^1 & a^2b^2 & \dots & a^Nb^N \end{bmatrix}. \tag{4.3.1}$$

It is worth noting that the sum of the elements of $\boldsymbol{a} \mathbin{.*} \boldsymbol{b}$ equals $\boldsymbol{a}^T\boldsymbol{b}$.

To use either of the "Distance From Face Space" or the maximum likelihood algorithms, we need to calculate the eigenface coefficients at each point of the image. This can be achieved by taking the correlation between the image and $\boldsymbol{u}_j$ for each j, an operation performed efficiently in the Fourier domain. Given two vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ of length N, the Fourier transform of the correlation is given by $\widehat{\boldsymbol{a} \circ \boldsymbol{b}} = \widehat{\boldsymbol{a}}^* \mathbin{.*} \widehat{\boldsymbol{b}}$ (see A.2.5). Thus the calculation

of the correlation requires 2 Fourier Transforms, N multiplications and 1 inverse Fourier Transform. The Fast Fourier Transform is of $O(N \log N)$, as is its inverse, while the multiplication step is trivially of $O(N)$, so the correlation operation is $O(N \log N)$. As mentioned in Appendix A.2, using the Fast Fourier Transform assumes that the function is periodic. To avoid any problems this may cause, we pad the images with zeros. This merely increases the constant involved in the Fast Fourier Transform, so the order is still $O(N \log N)$.

4.3.1 Distance From Face Space

The DFFS algorithm calculates $\epsilon(\boldsymbol{F})$ (3.2.3) at each point on the image being searched. Thus

$$\begin{aligned} \epsilon(\boldsymbol{F}) &= (\boldsymbol{F}-\boldsymbol{A})^T(\boldsymbol{F}-\boldsymbol{A}) - \boldsymbol{y}^T\boldsymbol{y} \\ &= \boldsymbol{F}^T\boldsymbol{F} - 2\boldsymbol{F}^T\boldsymbol{A} + \boldsymbol{A}^T\boldsymbol{A} - \boldsymbol{y}^T\boldsymbol{y} \end{aligned}$$

and

$$\begin{aligned} \boldsymbol{y} &= U^T(\boldsymbol{F}-\boldsymbol{A}) \\ &= U^T\boldsymbol{F} - U^T\boldsymbol{A}. \end{aligned}$$

We note that $c_1 = \boldsymbol{A}^T\boldsymbol{A}$ and $\boldsymbol{c_2} = U^T\boldsymbol{A}$ are constant and can be precalculated. To calculate $U^T\boldsymbol{F}$ for each point in the image requires taking the correlation with each column of U. Thus if we are dealing with P columns, we require P correlations.

Similarly, calculating $\boldsymbol{F}^T\boldsymbol{A}$ requires an additional correlation. Calculating $\boldsymbol{F}^T\boldsymbol{F}$ requires a bit more care, since we need the L_2 norm of a particular section of the image centred on a given point. This can be accomplished by calculating $(\boldsymbol{F} .* \boldsymbol{F}) \circ \boldsymbol{o}$ where $\boldsymbol{o}$ is a vector which one over the area of the image we are interested in and zero elsewhere.

Combining these steps, we see that

$$\begin{aligned} O(\text{DFFS}) &= O(N) + O(N \log N) + O(N \log N) + O(P \times N \log N) \\ &= O((P+2) \times N \log N). \end{aligned}$$

4.3.2 Maximum Likelihood Algorithm

The maximum likelihood algorithm requires calculating

$$d(\boldsymbol{X}) = \sum_{i=1}^{P} \frac{y_i^2}{\lambda_i} + \frac{N-P}{\sum_{i=P+1}^{N} \lambda_i} \epsilon^2(\boldsymbol{X}),$$

at each point, so its order of complexity is the work needed by the DFFS algorithm and the work needed to evaluate the $\sum_{i=1}^{P} \frac{y_i^2}{\lambda_i}$ term. From the previous section, we know that $O(\text{DFFS}) = O((P+2) \times N \log N)$, and that calculating the y_i's was $O(N \log N)$ for each case. So evaluating the $\sum_i \frac{y_i^2}{\lambda_i}$ term is $O(P \times N \log N)$. Thus we can see that the maximum likelihood algorithm is also $O(P \times N \log N)$, but has a larger constant than the DFFS measure.

In each case, however, this analysis is true only for a single scale and orientation. If we are trying to find an image at several different scales and orientations, the effect is multiplicative. Thus if we are searching at M scales and L rotations, the total search time is $O(L \times M \times P \times N \log N)$, which grows very quickly if our initial images are not well controlled.

Chapter 5

The Synthetic Discriminant Function

5.1 Background and Notation

As was pointed out in the previous chapter, the high computational cost of searching for a face through multiple scales and rotations makes the various eigenface based techniques unwieldy. Therefore a more efficient method for dealing with the problem is required. There has been a substantial amount of work done on similar problems, using a wide variety of methods, including various modified Fourier transforms (see [44]), numerous methods based on different wavelet transforms (e.g. in [6] and [4] various methods based on Gabor wavelets are discussed) and the so-called feature space trajectories (see [35]). We examine a method based on correlation filters, known as the Synthetic Discriminant Function or SDF.

Before we can describe the method, we provide a re-cap of the basic theory of the SDF and its variants. We will also provide a thorough theoretical analysis of these filters which, to the best of our knowledge, is novel. A somewhat less complete analysis is given in [33].

We will use the following notation

M	Number of images in the training set.
$\mathcal{X}_i$	i'th 2-D image in the training set.
$\widehat{\mathcal{X}}_i$	Fourier transform of $\mathcal{X}_i$.

$\boldsymbol{x}_i$	$\mathcal{X}_i$ as a column vector (of dimension D).
$\widehat{\boldsymbol{x}}_i$	Fourier transform of $\boldsymbol{x}_i$.
x_i^j	j'th component of $\boldsymbol{x}_i$.
X	Matrix $\left[\begin{array}{cccc} \boldsymbol{x}_1 & \boldsymbol{x}_2 & \cdots & \boldsymbol{x}_M \end{array} \right]$.
$\mathrm{diag}\,(\boldsymbol{x}_i)$	Diagonal matrix with the elements of $\boldsymbol{x}_i$ on the diagonal.
a^*	Complex conjugate of a.
A^H	Hermitian of A.
$\boldsymbol{h}$	Desired filter.
$\boldsymbol{h} * \boldsymbol{x}_i$	Convolution between $\boldsymbol{h}$ and $\boldsymbol{x}_i$.
$\boldsymbol{h} \circ \boldsymbol{x}_i$	Correlation between $\boldsymbol{h}$ and $\boldsymbol{x}_i$.

Additionally, given two D-dimensional vectors $\boldsymbol{a}$ and $\boldsymbol{b}$, we define (from MATLAB) the .* product as before (4.3.1), i.e.

$$\boldsymbol{a} \mathbin{.*} \boldsymbol{b} = \left[\begin{array}{cccc} a^1 b^1 & a^2 b^2 & \ldots & a^D b^D \end{array} \right].$$

We note that $\boldsymbol{a} \mathbin{.*} \boldsymbol{b} = \mathrm{diag}\,(\boldsymbol{a})\, \boldsymbol{b}$.

5.2 The Synthetic Discriminant Function

The Synthetic Discriminant Function (SDF) was first proposed by Hester and Casasent in [16]. Since then there has been a great deal of additional material published on the SDF and its variants. Much of the material for this review is based on the tutorial paper by Kumar and Mahalanobis [23], although we have added considerable detail.

Assuming we are given a set of vector $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_M$, recall that the correlation function between a $\boldsymbol{x}_i$ and a filter $\boldsymbol{h}$ is defined as

$$\sigma_i\,(t) = \boldsymbol{x}_i \circ \boldsymbol{h} = \sum_{j=1}^{D} \left(x_i^j \right)^* h^{j+t}.$$

In SDF's we constrain the values of the different correlation functions at the origin to some preset values, i.e.

$$\begin{aligned} v_i = \sigma_i(0) &= \sum_{j=1}^{D} x_i^j h^j \\ &= \boldsymbol{x}_i^T \boldsymbol{h}, \ i = 1, \ldots, M \end{aligned}$$

where we drop the conjugate since the $\boldsymbol{x}_i$'s are all real vectors.

Since we do this for all M images in our training set, we can rewrite this as a matrix equation

$$X^T \boldsymbol{h} = \boldsymbol{v} \tag{5.2.1}$$

where $\boldsymbol{v}$ is the vector (of length M) of desired responses. But $M \ll D$, so the system has infinitely many solutions. We choose the unique solution in the column space of X, i.e. $\boldsymbol{h} = X\boldsymbol{\gamma}$ for some vector $\boldsymbol{\gamma}$. It follows that

$$\boldsymbol{h} = X\left(X^T X\right)^{-1} \boldsymbol{v}. \tag{5.2.2}$$

It is worth noting that we assume that the columns of X are independent.

In the literature, the v_i's are usually chosen to be all equal (typically 1), but this is not required, an important point we shall return to later.

5.3 The Aim of the SDF

As discussed earlier, the problem is to locate the region of interest in an image in a manner that is independent of scale and rotation.

As is well known, a simple manner of finding a region of interest is to take the correlation of the image and some model. The maximum of the correlation function is then centred on the region that is most similar to the model. This, however, is not robust when the image can include scale and rotation variations.

The SDF is designed to deal with distortions represented by the training set. These distortions can be anything, but for our purposes, will generally be scale and rotation changes.

The SDF attempts to create a filter that is robust against scale and rotation variations by constraining the output to the specified values. The

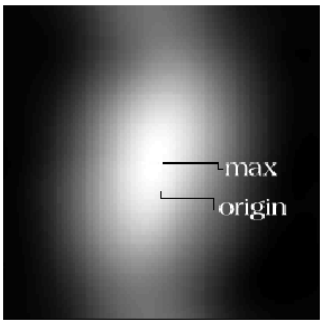


Figure 5.1: Typical correlation plane for the SDF.

aim of the constraints is to control the variation in the output caused by changes in scale and rotation. Typically the output is constrained to be 1 for all the images in the training set, and it is then hoped that the value at the centre of the region of interest (which in keeping with the usual practice in the literature, we will refer to as the origin[1]) will be close to 1 regardless of any variations in scale and rotation. It is unlikely to be exactly 1 as there are invariably numerous sources of minor variation which are impossible to remove completely.

5.4 Problems with the Classic SDF

The major problem with the classic SDF is that it only constrains the output at a single point, the origin. No attempt is made to ensure that this is a maximum. Thus frequently the actual maximum does not occur at the origin. Typically, the correlation plane (output of the correlation function considered as a 2-D image) for the SDF looks like figure 5.1, where the maximum and the origin do not coincide. Since in a traditional correlation filter, the location is found at the maximum, this behaviour is undesirable.

Furthermore, it is not sufficient to merely look for a value similar to the constrained value, since the maximum is not located near the origin, we tend to find a entire ring with the same value. For instance, figure 5.1 was obtained from a filter where the value at the origin was constrained to be 1 (since this is the correlation plane for an image not in the training set, but very similar, the value at the origin actually turns out to be ≈ 1.00015 in

[1] This should not be confused with the centre of the image. Generally these are different points in the image, although in some of our examples they coincide.

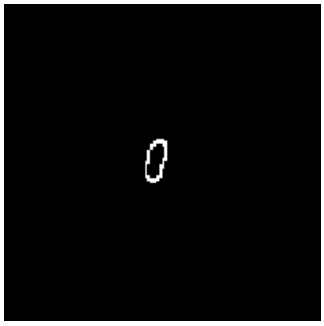


Figure 5.2: Section of correlation plane ≈ 1.

this case). Figure 5.2 shows the section of the resulting correlation plane with values between 0.999 and 1.001. From this we can easily conclude that, without some prior information, it would be impossible to locate the constrained value.

Despite these problems, the SDF forms the basis for an entire class of filters with similar aims, and which, in addition, try to ensure that the maximum will occur at the origin.

5.5 Some Extensions to the Classic SDF

Several extensions have been proposed for the SDF. For a review of most of these see [21] and [23]. In this section we discuss some of the more significant variants.

5.5.1 Minimum Variance SDF's

One of the earliest variations is known as the MVSDF (Minimum Variance Synthetic Discriminant Function) (see Kumar [20] for more information). We assume that there is some additional noise, described by $\boldsymbol{n}$, in the images, i.e. $\boldsymbol{x}_i = \widetilde{\boldsymbol{x}}_i + \boldsymbol{n}$ where $\widetilde{\boldsymbol{x}}_i$ is the noiseless image. We assume that the noise process has a mean of zero and that the covariance matrix is given by C. The output of the filter (at the origin) is

$$\sigma_i(0) = \boldsymbol{h}^T \boldsymbol{x}_i = \boldsymbol{h}^T \widetilde{\boldsymbol{x}}_i + \boldsymbol{h}^T \boldsymbol{n}$$

and it can be easily shown that the variance in the output, $\sigma_i(0)$, is given by

$$\rho^2 = \boldsymbol{h}^T C \boldsymbol{h}.$$

The MVSDF is the filter that minimises this variance subject to the condition in (5.2.1). By minimising the variance in the output, we should reduce the impact of $\boldsymbol{n}$ as much as possible. The effect of variations such as background can thus by minimised if a suitable choice for C is used.

We use Lagrange multipliers, and therefore minimise

$$\begin{aligned} f(\boldsymbol{h}, \boldsymbol{\lambda}) &= \boldsymbol{h}^T C \boldsymbol{h} - \boldsymbol{\lambda}^T \left(X^T \boldsymbol{h} - \boldsymbol{v} \right) \\ &= 2 \sum_{i=1}^{D} \sum_{j=i+1}^{D} C^{ij} h^i h^j + \sum_{i=1}^{D} C^{ii} \left(h^i\right)^2 - \sum_{i=1}^{M} \lambda_i \left(-v_i - \sum_{j=1}^{D} x_i^j h^j \right) \end{aligned}$$

since C is symmetric. Thus

$$\frac{\delta f}{\delta h^n} = 2 \sum_{i=1}^{D} C^{in} h^i - \sum_{i=1}^{M} \lambda_i x_i^n = 0.$$

This is a system of D equations. We can rewrite this as the following vector equation:

$$2C\boldsymbol{h} - X\boldsymbol{\lambda} = 0$$

and so

$$\boldsymbol{h} = C^{-1} X \left(\frac{1}{2} \boldsymbol{\lambda} \right) \tag{5.5.1}$$

where $\boldsymbol{\lambda} = \begin{bmatrix} \lambda_1 & \lambda_2 & \dots & \lambda_M \end{bmatrix}$.

Since the constraint (5.2.1) must be satisfied, we multiply by X^T to

obtain

$$\begin{aligned} X^T \boldsymbol{h} &= X^T C^{-1} X \left(\frac{1}{2}\boldsymbol{\lambda}\right) \\ \boldsymbol{v} &= X^T C^{-1} X \left(\frac{1}{2}\boldsymbol{\lambda}\right) \\ \frac{1}{2}\boldsymbol{\lambda} &= \left(X^T C^{-1} X\right)^{-1} \boldsymbol{v}. \end{aligned}$$

Substituting this into (5.5.1), we see that

$$\boldsymbol{h} = C^{-1} X \left(X^T C^{-1} X\right)^{-1} \boldsymbol{v}. \tag{5.5.2}$$

The MVSDF has two problems. Like the conventional SDF, the constrained value is not required to be a maximum and it also introduces the matrix C, which is of size $D \times D$, so it may be difficult to estimate and can be computationally expensive to invert. It does, however, introduce the idea of imposing some additional constraint on the SDF, which is the basis of most of the more popular variants.

5.5.2 Minimum Average Correlation Energy Filters

The MACE filter tries to minimise the correlation energy over all the images in the training set. This will suppress all values in the correlation plane except at the origin, which is constrained to be the selected value. This should completely eliminate the problem of the maximum not lying at the origin (see [25]). From the correlation theorem (A.2.5), we know that the Fourier transform of the correlation surface between $\boldsymbol{x}_i$ and the filter is given by $\widehat{\boldsymbol{x}}_i^* \,.*\, \widehat{\boldsymbol{h}} = \operatorname{diag}(\widehat{\boldsymbol{x}}_i)^* \, \widehat{\boldsymbol{h}}$. Thus the energy in the Fourier domain is

$$E_i = \widehat{\boldsymbol{h}}^H \operatorname{diag}(\widehat{\boldsymbol{x}}_i) \operatorname{diag}(\widehat{\boldsymbol{x}}_i)^H \widehat{\boldsymbol{h}}.$$

We minimise the average of the E_i's. After summing over i and dividing by M, it is easy to show that the average correlation energy can be written as

$$E_{ave} = \widehat{\boldsymbol{h}}^H B \widehat{\boldsymbol{h}}$$

where

$$\boldsymbol{b} = \frac{1}{M}\sum_{i=1}^{M}(\widehat{\boldsymbol{x}}_i^* \ .*\ \widehat{\boldsymbol{x}}_i)$$

and B is a diagonal matrix with the elements of $\boldsymbol{b}$ on the diagonal (i.e. $B = \text{diag}(\boldsymbol{b})$). Alternatively, $B = \frac{1}{M}\sum_{i=1}^{M}\text{diag}(\widehat{\boldsymbol{x}}_i)^*\,\text{diag}(\widehat{\boldsymbol{x}}_i)$. The MACE filter minimises E_{ave} subject to (5.2.1). From the definition of the Fourier transform (see (A.2.2)), it follows that the Fourier transform can be treated as a matrix multiplication where $F^{-1} = \frac{1}{D}F^H$. From this it follows that (5.2.1) can be rewritten as

$$X^T F^{-1} F\boldsymbol{h} = X^T F^H \frac{1}{D} F\boldsymbol{h} = \widehat{X}^H \frac{1}{D}\widehat{\boldsymbol{h}} = \boldsymbol{v}.$$

So, in the Fourier domain, (5.2.1) becomes

$$\widehat{X}^H\widehat{\boldsymbol{h}} = D\boldsymbol{v}. \tag{5.5.3}$$

Thus we can apply the same analysis used for the MVSDF in the Fourier domain to show that

$$\widehat{\boldsymbol{h}} = B^{-1}\widehat{X}\left(\widehat{X}^H B^{-1}\widehat{X}\right)^{-1} D\boldsymbol{v}. \tag{5.5.4}$$

It is important to note that we now have an expression for the Fourier transform of $\boldsymbol{h}$.

Although the MACE filter does try to ensure that the maximum occurs at the origin, in many cases it does not generalise well to images outside of the training set and so is seldom particularly successful.

5.5.3 Mean Squared Error SDF's

The Mean Squared error SDF tries to constrain the correlation plane to a given surface (see for example [24]). The idea is to ensure that the maximum will occur at the origin by forcing the correlation surface to a suitable shape (such as a Gaussian surface) without being as restrictive as the MACE filter.

Let $s(x, y)$ be the desired shape of the correlation plane, then $\widehat{\boldsymbol{s}}$ is the Fourier transform of the vector form $\boldsymbol{s}$. In the Fourier domain, the difference between the correlation of the i'th image with the filter and the desired

surface $s(x,y)$ is $\widehat{\boldsymbol{e}}_i = \operatorname{diag}(\widehat{\boldsymbol{x}}_i)^* \widehat{\boldsymbol{h}} - \widehat{\boldsymbol{s}}$. The mean square error can be written as

$$\begin{aligned}
\text{MSE} &= \frac{1}{M}\sum_{i=1}^{M} \widehat{\boldsymbol{e}}_i^H \widehat{\boldsymbol{e}}_i \\
&= \frac{1}{M}\sum_{i=1}^{M} \left(\operatorname{diag}(\widehat{\boldsymbol{x}}_i)^* \widehat{\boldsymbol{h}} - \widehat{\boldsymbol{s}}\right)^H \left(\operatorname{diag}(\widehat{\boldsymbol{x}}_i)^* \widehat{\boldsymbol{h}} - \widehat{\boldsymbol{s}}\right) \\
&= \widehat{\boldsymbol{h}}^H \left[\frac{1}{M}\sum_{i=1}^{M} \operatorname{diag}(\widehat{\boldsymbol{x}}_i)^* \operatorname{diag}(\widehat{\boldsymbol{x}}_i)\right] \widehat{\boldsymbol{h}} + \widehat{\boldsymbol{s}}^H \widehat{\boldsymbol{s}} \\
&- \widehat{\boldsymbol{h}}^H \left(\frac{1}{M}\sum_{i=1}^{M} \operatorname{diag}(\widehat{\boldsymbol{x}}_i)\right) \widehat{\boldsymbol{s}} - \widehat{\boldsymbol{s}}^H \left(\frac{1}{M}\sum_{i=1}^{M} \operatorname{diag}(\widehat{\boldsymbol{x}}_i)^*\right) \widehat{\boldsymbol{h}} \qquad (5.5.5) \\
&= \widehat{\boldsymbol{h}}^H B \widehat{\boldsymbol{h}} + \widehat{\boldsymbol{s}}^H \widehat{\boldsymbol{s}} - \widehat{\boldsymbol{h}}^H \widehat{\boldsymbol{p}} - \widehat{\boldsymbol{p}}^H \widehat{\boldsymbol{h}}
\end{aligned}$$

where B is the same matrix as in the MACE filter and $\widehat{\boldsymbol{p}} = \frac{1}{M}\sum_{i=1}^{M} \operatorname{diag}(\widehat{\boldsymbol{x}}_i)\, \widehat{\boldsymbol{s}}$. The MSE SDF minimises the MSE subject to (5.2.1). If we use Lagrange multipliers, it is not too difficult to show that after taking the partial derivatives, the desired filter must satisfy the following vector equation:

$$B\widehat{\boldsymbol{h}} - \widehat{\boldsymbol{p}} = \widehat{X}\boldsymbol{\lambda}.$$

Then, by applying the same method used to solve (5.5.1), it is easy to show that

$$\widehat{\boldsymbol{h}} = B^{-1}\widehat{\boldsymbol{p}} + B^{-1}\widehat{X}\left(\widehat{X}^H B^{-1}\widehat{X}\right)^{-1}\left(D\boldsymbol{v} - \widehat{X}^H B^{-1}\widehat{\boldsymbol{p}}\right). \qquad (5.5.6)$$

5.5.4 Optimal Mean Square Error SDF's

The MSE SDF shapes the filter to give some desired output. The question is what is the best shape to use. To find this, we minimise the MSE with respect to $\widehat{\boldsymbol{s}}$. We obtain the gradient of the MSE from (5.5.5) and so we need to solve

$$\nabla_{\widehat{\boldsymbol{s}}}(\text{MSE}) = \widehat{\boldsymbol{s}}^H - \widehat{\boldsymbol{h}}^H \left[\frac{1}{M}\sum_{i=1}^{M} \operatorname{diag}(\widehat{\boldsymbol{x}}_i)\right] = 0.$$

If we define $\widehat{A} = \frac{1}{M}\sum_{i=1}^{M} \operatorname{diag}(\widehat{\boldsymbol{x}}_i)$, it follows from the above that the

optimum choice for $\widehat{\boldsymbol{s}}$ is

$$\widehat{\boldsymbol{s}} = \widehat{A}^* \widehat{\boldsymbol{h}}.$$

Substituting this into the expression for the MSE, we see that

$$\begin{aligned} \mathrm{MSE}_{\mathrm{opt}} &= \frac{1}{M} \sum_{i=1}^{M} \left(\mathrm{diag}\left(\widehat{\boldsymbol{x}}_i\right)^* \widehat{\boldsymbol{h}} - \widehat{A}^* \widehat{\boldsymbol{h}} \right)^H \left(\mathrm{diag}\left(\widehat{\boldsymbol{x}}_i\right)^* \widehat{\boldsymbol{h}} - \widehat{A}^* \widehat{\boldsymbol{h}} \right) \\ &= \widehat{\boldsymbol{h}}^H \left[\frac{1}{M} \sum_{i=1}^{M} \left(\mathrm{diag}\left(\widehat{\boldsymbol{x}}_i\right) - \widehat{A} \right)^* \left(\mathrm{diag}\left(\widehat{\boldsymbol{x}}_i\right) - \widehat{A} \right) \right] \widehat{\boldsymbol{h}} \\ &= \widehat{\boldsymbol{h}}^H S \widehat{\boldsymbol{h}} \end{aligned}$$

where

$$S = \frac{1}{M} \sum_{i=1}^{M} \left(\mathrm{diag}\left(\widehat{\boldsymbol{x}}_i\right) - \widehat{A} \right)^* \left(\mathrm{diag}\left(\widehat{\boldsymbol{x}}_i\right) - \widehat{A} \right). \tag{5.5.7}$$

Clearly S is a diagonal matrix. Deriving the filter in this case is identical to the case of deriving the MACE filter earlier and so the optimal MSE SDF is

$$\widehat{\boldsymbol{h}} = S^{-1} \widehat{X} \left(\widehat{X}^H S^{-1} \widehat{X} \right)^{-1} D \boldsymbol{v}. \tag{5.5.8}$$

5.5.5 The Minimum Noise and Correlation Energy Filter

The Minimum Noise and Correlation Energy filter re-introduces the notion of a noise term from the MVSDF, but also minimises the average correlation energy (see [42]). The simplest case minimises the average correlation energy with the addition of an appropriate noise factor. The noise is modeled as affecting the Fourier transform of the image and is assumed to be independent of the images in the training set. Thus the training set and the noise are uncorrelated. The covariance matrix of the noise is assumed to be diagonal (typically the noise is modeled as Gaussian, with a diagonal covariance matrix). Thus we define $\widehat{C}$ as the covariance matrix (in the Fourier domain). We then minimise

$$\mathrm{MINACE} = \widehat{\boldsymbol{h}}^H T \widehat{\boldsymbol{h}}$$

where

$$T = B - b\widehat{C}$$

subject to the constraints in (5.2.1). Here B is the same matrix as used in the MACE filter and b is some constant determined by the constraints of the problem. It determines the relative importance of the noise. The larger b, the greater the effect of the noise that must be compensated for. If $b = 0$, we are back at the original MACE filter. Clearly T is diagonal. Typically T is constrained to be non-negative. By the same argument used to solve the MACE filter, the solution here is

$$\widehat{\boldsymbol{h}} = T^{-1}\widehat{X}\left(\widehat{X}^H T^{-1}\widehat{X}\right)^{-1} D\boldsymbol{v}. \tag{5.5.9}$$

There are some variations on the MINACE filter (for example see [5]), which depend on slightly different specifications of the noise and B, but they all lead to filters of the same form with T a diagonal matrix. This filter has been successful in locating targets against noisy backgrounds.

5.5.6 Concluding Comment

It is worth noting that all the variants on the SDF have very similar structures. Most are of the form

$$\boldsymbol{h} = W^{-1}X\left(X^T W^{-1}X\right)^{-1}\boldsymbol{v} \tag{5.5.10}$$

or the equivalent expression in the Fourier domain. Thus we can easily identify that the additional matrix W plays a major role in determining the behaviour of the filter.

This point is also made in [2] and the authors provide some information about various possibilities for constructing W, but there is little theoretical analysis of the properties of W available in the literature.

5.6 The Maximum Average Correlation Height Filter

The Maximum Average Correlation Height (MACH) filter is not a true extension of the SDF, but, as it grows out of the concepts introduced in the optimal MSE-SDF (see [26] for more details), it is typically described as a related filter in the literature.

5.6.1 The Average Similarity Measure

Recalling the optimal MSE SDF, we note that it is the filter that minimises

$$Q = \widehat{\boldsymbol{h}}^H S \widehat{\boldsymbol{h}}, \tag{5.6.1}$$

where S is given by (5.5.7), subject to the constraint (5.2.1). What is the significance of Q? It can be rewritten as

$$\begin{aligned} Q &= \widehat{\boldsymbol{h}}^H \left(\frac{1}{M} \sum_{i=1}^{M} \left(\operatorname{diag}\left(\widehat{\boldsymbol{x}}_i\right) - \widehat{A} \right)^* \left(\operatorname{diag}\left(\widehat{\boldsymbol{x}}_i\right) - \widehat{A} \right) \right) \widehat{\boldsymbol{h}} \\ &= \frac{1}{M} \sum_{i=1}^{M} \left(\left(\operatorname{diag}\left(\widehat{\boldsymbol{x}}_i\right) - \widehat{A} \right)^* \widehat{\boldsymbol{h}} \right)^H \left(\left(\operatorname{diag}\left(\widehat{\boldsymbol{x}}_i\right) - \widehat{A} \right)^* \widehat{\boldsymbol{h}} \right) \\ &= \frac{1}{M} \sum_{i=1}^{M} \left\| \operatorname{diag}\left(\widehat{\boldsymbol{x}}_i\right)^* \widehat{\boldsymbol{h}} - \widehat{A}^* \widehat{\boldsymbol{h}} \right\|_2^2 \\ &= \frac{1}{M} \sum_{i=1}^{M} \left\| \widehat{\boldsymbol{g}}_i - \widehat{\overline{\boldsymbol{g}}} \right\|_2^2 \end{aligned}$$

where $\widehat{\boldsymbol{g}}_i$ is the Fourier transform of the correlation surface obtained from $\boldsymbol{x}_i$ and $\boldsymbol{h}$. Since $\widehat{A} = \frac{1}{M} \sum_{i=1}^{M} \operatorname{diag}\left(\widehat{\boldsymbol{x}}_i\right)$, $\widehat{\overline{\boldsymbol{g}}}$ is the correlation surface obtained from the average image and $\boldsymbol{h}$. From Parseval's theorem (A.2.3), it follows that

$$Q = \frac{D}{M} \sum_{i=1}^{M} \left\| \boldsymbol{g}_i - \overline{\boldsymbol{g}} \right\|_2^2 . \tag{5.6.2}$$

Thus we can say that Q measures difference between the output correlation surfaces for images in the training set and the correlation surface of the mean image. Thus we refer to this measure as the Average Similarity

Measure (ASM).

5.6.2 Interpretations of the ASM

Equation (5.6.2) is a Sum of Squared Distances (SSD) measure in the correlation space. The SSD is a metric for how tightly clustered a number of observations are (see section 8.3.2.1). Thus the ASM measures how clustered the different correlation surfaces are. Since the filter is a linear transformation, the ASM measures how well this transform maps the images onto a small cluster.

In addition, we note that, from the definition of the ASM (5.6.1), it follows that

$$\begin{aligned} \text{ASM} &= \widehat{\boldsymbol{h}}^H S \widehat{\boldsymbol{h}} \\ &= \widehat{\boldsymbol{h}}^H B \widehat{\boldsymbol{h}} - \widehat{\boldsymbol{h}}^H \widehat{A}^* \widehat{A} \widehat{\boldsymbol{h}} \\ &= \text{ACE} - \widehat{\boldsymbol{h}}^H \widehat{A} \widehat{A}^* \widehat{\boldsymbol{h}} \end{aligned}$$

where we exploit the fact fact that $\widehat{A}^* \widehat{A}$ is real, so $\widehat{A}^* \widehat{A} = \widehat{A} \widehat{A}^*$. Here B is the same diagonal matrix used in the MACE filter.

Thus the ASM is the difference between the average correlation energy of the training set and the correlation energy of the average image.

5.6.3 The ASM as a Filter Design Criterium

Clearly, if a filter has a small ASM, the resulting correlation surfaces will be similar, which should help ensure distortion invariance. Merely minimising the ASM, however, is not sufficient to create a good filter, since it is trivially minimised if we choose $\widehat{\boldsymbol{h}} = 0$. Thus we also need an additional condition to ensure that the surfaces are well-shaped.

Since the correlation surfaces are all similar to the correlation surface from the average image, we add the additional constraint that this correlation surface (from the average image) should have a high peak at the origin.

We can satisfy both these conditions by maximising

$$\begin{aligned} J\left(\widehat{\boldsymbol{h}}\right) &= \frac{\widehat{\boldsymbol{h}}^H \widehat{\boldsymbol{a}} \widehat{\boldsymbol{a}}^H \widehat{\boldsymbol{h}}}{\widehat{\boldsymbol{h}}^+ S \widehat{\boldsymbol{h}}} \\ &= \frac{\left(\widehat{\boldsymbol{a}}^H \widehat{\boldsymbol{h}}\right)^H \left(\widehat{\boldsymbol{a}}^H \widehat{\boldsymbol{h}}\right)}{\widehat{\boldsymbol{h}}^H S \widehat{\boldsymbol{h}}} \end{aligned}$$

where $\widehat{\boldsymbol{a}} = \frac{1}{M}\sum_{i=1}^{M}\widehat{\boldsymbol{x}}_i$, the average of the Fourier transforms of the images in the training set.

Since it is obvious that $J\left(\widehat{\boldsymbol{h}}\right) = J\left(\lambda\widehat{\boldsymbol{h}}\right)$, $\lambda \neq 0$, we need to impose one further constraint on $\widehat{\boldsymbol{h}}$ to ensure uniqueness. A useful choice is simply $\widehat{\boldsymbol{h}}^H\widehat{\boldsymbol{h}} = 1$.

We can fairly easily determine $\widehat{\boldsymbol{h}}$ up to an arbitrary constant by using the gradient

$$\begin{aligned}\nabla J(\boldsymbol{h}) &= \frac{\left(\widehat{\boldsymbol{h}}^H S\widehat{\boldsymbol{h}}\right)\left(\widehat{\boldsymbol{h}}^H\widehat{\boldsymbol{a}}\right)\widehat{\boldsymbol{a}}^H - \left(\widehat{\boldsymbol{h}}^H\widehat{\boldsymbol{a}}\right)\left(\widehat{\boldsymbol{a}}^H\widehat{\boldsymbol{h}}\right)\widehat{\boldsymbol{h}}^H S}{\left(\widehat{\boldsymbol{h}}^H S\widehat{\boldsymbol{h}}\right)^2} \\ &= 0.\end{aligned}$$

Then

$$\begin{aligned}\left(\widehat{\boldsymbol{h}}^H S\widehat{\boldsymbol{h}}\right)\left(\widehat{\boldsymbol{h}}^H\widehat{\boldsymbol{a}}\right)\widehat{\boldsymbol{a}}^H &= \left(\widehat{\boldsymbol{h}}^H\widehat{\boldsymbol{a}}\right)\left(\widehat{\boldsymbol{a}}^H\widehat{\boldsymbol{h}}\right)\widehat{\boldsymbol{h}}^H S \\ \Rightarrow\ \widehat{\boldsymbol{a}} &= \frac{\widehat{\boldsymbol{h}}^H\widehat{\boldsymbol{a}}}{\left(\widehat{\boldsymbol{h}}^H S\widehat{\boldsymbol{h}}\right)} S\widehat{\boldsymbol{h}}\end{aligned}$$

since S is real and diagonal. Therefore

$$\widehat{\boldsymbol{h}} = cS^{-1}\widehat{\boldsymbol{a}}$$

where $c = \frac{\widehat{\boldsymbol{h}}^H S\widehat{\boldsymbol{h}}}{\widehat{\boldsymbol{h}}^H\widehat{\boldsymbol{a}}}$ is a scalar. Using the normalisation $\widehat{\boldsymbol{h}}^H\widehat{\boldsymbol{h}} = 1$, it follows that

$$\widehat{\boldsymbol{h}} = \frac{S^{-1}\widehat{\boldsymbol{a}}}{\left\|S^{-1}\widehat{\boldsymbol{a}}\right\|_2}. \tag{5.6.3}$$

In the literature, $\widehat{\boldsymbol{h}} = S^{-1}\widehat{\boldsymbol{a}}$ is also frequently used. Without normalising $\widehat{\boldsymbol{h}}$, however, it is possible to have extremely high values for some of the elements of $\widehat{\boldsymbol{h}}$ which can lead to numerical problems.

5.7 Using the SDF to Estimate Distortion Parameters

5.7.1 Initial Comments

We are now ready to explain how the SDF and its variants can be used to solve the location problem.

A few comments are in order

- The SDF-based techniques are, by themselves, not sufficiently accurate for our purposes. Correlation-based filters are already not sufficiently accurate and the construction of the SDF from images that cover a wide range of distortions averages the images even further, so exact results are impossible. Provided the SDF approach reduces the initial search space, however, the final refinement can be done by a more accurate (and more computationally expensive) method such as one of the eigenface-based algorithms discussed earlier.

- We can use the SDF and its variants to provide not only estimates of the location in a distortion invariant manner, but also to provide estimates of the extent of the distortion. This is a most useful attribute of the SDF.

- Due to the problems of the classic SDF (see section 5.4), estimating the distortion requires the use of multiple filters. We must first use one of the variants discussed to locate the origin and then use the classic SDF to estimate the distortion parameters.

5.7.2 Linearly Interpolating SDF's

In this section we investigate the properties of the filter required to estimate the distortion parameters.

In the conventional SDF, we constrain the output values for each image in the training set. The detector, however, must locate images at situations not in the training set. Thus we need to establish what happens for these intermediate situations.

In the literature, the constraints are usually chosen so that the filter response is 1 for each image in the training set (i.e. $\boldsymbol{v} = \begin{bmatrix} 1 & 1 & \dots & 1 \end{bmatrix}$). The claim is that for images in intermediate situations, the output will also

be 1. In [22], the authors point out that a simple constant has poor discriminatory power. Images that are linear combinations of the images in the training set will return the same constant, even if they do not represent distortions of interest. They propose that a complex constant of magnitude 1 be used instead. This does not fully exploit the freedom allowed by an arbitrary choice of $\boldsymbol{v}$. We propose a more general concept.

Let us consider the case when we design a filter from a training set where the only variation is due to rotation. We assume that all the images come from the same underlying function which takes the angle as a parameter. Thus $\mathcal{X}_i = \mathcal{X}(\theta_i)$. We define the output at the origin of the correlation surface produced by the filter and an image as σ with $\sigma(\theta_i) = \beta_i$. We then consider an angle between two given angles in the training set,

$$\theta^* = \alpha\theta_i + (1-\alpha)\,\theta_{i+1}$$

where $0 < \alpha < 1$. The most useful filter response is one which allows us to determine θ^*. This is achieved if the filter response changes linearly as the angle changes[2], namely

$$\sigma(\theta^*) = \alpha\beta_i + (1-\alpha)\,\beta_{i+1}.$$

We call filters with this property, linearly interpolating SDF's.

5.7.3 Designing Linearly Interpolating SDF's

In order to derive conditions for linearly interpolating SDF's, we continue with the previous example where the only variation is due to rotation. It is convenient to work in polar coordinates. Then the images are given by $\mathcal{X}_i = \mathcal{X}(\theta + \theta_i, r)$ and

$$\sigma(\lambda) = \iint \mathcal{X}(\theta+\lambda, r)^*\, h(\theta, r)\, r\,dr\,d\theta$$

Earlier, to derive (5.2.2) we chose $h(\theta, r)$ as a combination of the faces

[2]Of course, linearity is not the only possible choice to allow the estimation to work. Any function that is one-to-one over the interval $\alpha \in [0 \ldots 1]$ would work. The simplicity of linear functions, however, is a major advantage in this type of problem.

in the training set. Thus, there is some set of γ_i's so that

$$\begin{aligned} h(\theta, r) &= \sum_{i=1}^{M} \gamma_i \mathcal{X}_i(\theta, r) \\ &= \sum_{i=1}^{M} \gamma_i \mathcal{X}(\theta + \theta_i, r). \end{aligned}$$

Furthermore, we know the values of $\sigma(\theta_i)$, since they are specified by the design of $h(\theta, r)$. Substituting the expression for $h(\theta, r)$ into the formula for $\sigma(\lambda)$ we find that

$$\begin{aligned} \sigma(\lambda) &= \iint \mathcal{X}(\theta + \lambda, r)^* \sum_{i=1}^{M} \gamma_i \mathcal{X}(\theta + \theta_i, r)\, r dr d\theta \\ &= \sum_{i=1}^{M} \gamma_i \iint \mathcal{X}(\theta + \lambda, r)^* \mathcal{X}(\theta + \theta_i, r)\, r dr d\theta \\ &= \sum_{i=1}^{M} \gamma_i c(\lambda - \theta_i) \end{aligned}$$

where

$$c(\beta) = \iint \mathcal{X}(\theta + \beta, r)^* \mathcal{X}(\theta, r)\, r dr d\theta. \tag{5.7.1}$$

We will refer to $c(\beta)$ as the self-correlation function.

For a linearly interpolating SDF, we require that

$$\sigma(\alpha\theta_i + (1-\alpha)\theta_{i+1}) = \alpha\sigma(\theta_i) + (1-\alpha)\sigma(\theta_{i+1})$$

which implies that

$$\begin{aligned} \sum_{j=1}^{M} \gamma_j c(\alpha\theta_i + (1-\alpha)\theta_{i+1} - \theta_j) &= \alpha \sum_{j=1}^{M} \gamma_j c(\theta_i - \theta_j) \\ &+ (1-\alpha) \sum_{j=1}^{M} \gamma_j c(\theta_{i+1} - \theta_j) \ \forall \alpha \in [0,1]. \end{aligned}$$

We would like this condition to hold regardless of the γ_j's. This is satisfied if $c(\beta)$ is linear between the sampled angles. Thus we can ensure that our SDF

is linearly interpolating provided that $c(\beta)$ is a piecewise linear function. Of course, in practice, this is seldom true. By taking a suitable number of samples (i.e. an appropriate training set), however, it is possible to obtain a good approximation to $c(\beta)$ that is piecewise linear.

5.7.4 Multiple Parameters

This generalises directly to multiple parameters. The mathematics just becomes slightly more complicated.

It is also important to note that the choice of parameters is dependent on the nature of the desired invariance. For instance, we can ensure that the filter will be linearly interpolating for translation along the x and y axes separately.

In this situation the filter response is

$$\begin{aligned}
\sigma(x_*, y_*) &= \iint \mathcal{X}(x + x_*, y + y_*)^* \, h(x, y) \, dxdy \\
&= \iint \mathcal{X}(x + x_*, y + y_*)^* \sum_{i,j} \gamma_{ij} \mathcal{X}(x + x_i, y + y_j) \, dxdy \\
&= \sum_{i,j} \gamma_{ij} \iint \mathcal{X}(x + x_*, y + y_*)^* \, \mathcal{X}(x + x_i, y + y_j) \, dxdy \\
&= \sum_{i,j} \gamma_{ij} c(x_* - x_i, y_* - y_j)
\end{aligned}$$

where

$$c(\mu, \rho) = \iint \mathcal{X}(x + \mu, y + \rho)^* \, \mathcal{X}(x, y) \, dxdy.$$

Then, for a linearly interpolating SDF, we require that $c(\mu, \rho)$ be linear in both x and y. This can be generalised to as many parameters as we wish.

5.8 Analysis of the SDF Variants

5.8.1 Relationship to the Ordinary SDF

We noted earlier that almost all the variants we discussed earlier have the same structure, given by (5.5.10) or its Fourier equivalent, a point also made by Brasher, et al in [2]. Thus we consider a filter defined by the following

equation:

$$\widehat{\boldsymbol{h}} = W^{-1}\widehat{X}\left(\widehat{X}^H W^{-1}\widehat{X}\right)^{-1} D\boldsymbol{v}$$

where W^{-1} is a real diagonal matrix with non-negative elements. This covers a range of filters, including the MACH, optimal MSE SDF and MINACE filters. The ordinary MSE SDF (5.5.6) can be expressed in this framework by redefining $\widehat{\boldsymbol{h}}$ and $\boldsymbol{v}$. D is a scaling parameter, necessitated by the properties of the Fourier transform.

Since W^{-1} is a real, nonnegative, diagonal matrix, there exists a diagonal matrix E so that

$$EE^H = W^{-1}.$$

Then

$$\begin{aligned}\widehat{\boldsymbol{h}} &= EE^H\widehat{X}\left(\widehat{X}^H EE^H\widehat{X}\right)^{-1} D\boldsymbol{v} \\ &= E\widehat{Y}\left(\widehat{Y}^H\widehat{Y}\right)^{-1} D\boldsymbol{v}\end{aligned}$$

where

$$\widehat{Y} = E^H\widehat{X}.$$

If we define $\widehat{\boldsymbol{f}} = E^{-1}\widehat{\boldsymbol{h}}$ then

$$\widehat{\boldsymbol{f}} = \widehat{Y}\left(\widehat{Y}^H\widehat{Y}\right)^{-1} D\boldsymbol{v}.$$

This is similar to the original formula for the SDF, except we are now in the Fourier domain.

We already mentioned that we can treat the Fourier transform as multiplication by a matrix F where $F^{-1} = \frac{1}{D}F^H$ (see A.2.2). Then

$$\begin{aligned}\widehat{\boldsymbol{f}} &= FY\left(Y^H F^H FY\right)^{-1} D\boldsymbol{v} \\ &= FY\left(Y^H DF^{-1}FY\right) D\boldsymbol{v} \\ &= FY\frac{1}{D}\left(Y^H Y\right)^{-1} D\boldsymbol{v}\end{aligned}$$

which implies that

$$F^{-1}\widehat{\boldsymbol{f}} = Y\left(Y^H Y\right)^{-1}\boldsymbol{v}$$

and finally that

$$\boldsymbol{f} = Y\left(Y^H Y\right)^{-1}\boldsymbol{v}. \tag{5.8.1}$$

Thus we see that $\boldsymbol{f}$ is a ordinary SDF for the transformed image space Y. This is hardly surprising as all the variants need to satisfy (5.2.1), and they merely add additional constraints. Thus the additional constraints impose an additional transformation on the image space.

In [2], the authors make the point that filters of the form (5.5.10) satisfy the SDF constraint (5.2.1) with additional preprocessing. However, we show that they are exactly equivalent to SDF's in the transformed space.

Of course, the question arises as to how these additional constraint transform the image space. We know that

$$\widehat{Y} = E^H \widehat{X}$$

so

$$FY = E^H \widehat{X}$$

and therefore

$$\begin{aligned} Y &= F^{-1}\left(E^H \widehat{X}\right) \\ &= F^{-1}\left(\widehat{\boldsymbol{e}}^* \;.*\; \widehat{X}\right) \end{aligned}$$

where $E = \text{diag}\ (\widehat{\boldsymbol{e}})$. Since E is a diagonal matrix (since W^{-1} is diagonal), this last step follows immediately from our definition of the $.*$ product. The correlation theorem (A.2.5) states that multiplication by the conjugate in the Fourier domain in equivalent to correlation in the spatial domain, thus, applying it here, we see there is some filter $\boldsymbol{e}$, based on E so that $Y = \boldsymbol{e} \circ X$. Thus, the major variants on the SDF all first filter the input images and then behave as a normal SDF on the transformed image space.

We note that all we needed to derive (5.8.1) was $EE^H = W^{-1}$, a factori-

sation that is possible for any matrix W^{-1} that is Hermitian and has non-negative eigenvalues[3]. The above interpretation of E, however, depends on the fact that W^{-1} be diagonal. More complex matrixes, while still providing SDF's in a transformed image space, do not permit a simple interpretation of the transformation. We are unaware of any major SDF variant that has a non-diagonal matrix for W^{-1}, but (5.8.1) suggests that this might be an area worth investigating.

This result immediately allows us to generalise our linearly interpolating concept to a much larger class of SDF-based filters.

5.8.2 Generalised Linearity Condition

In the case of the SDF, the linearly interpolating condition requires that the self-correlation function,

$$c(\lambda) = \iint \mathcal{X}(\lambda)^* \mathcal{X}(0)\, dA,$$

is piecewise linear. In the discrete case, it becomes

$$c(\lambda) = \boldsymbol{x}(\lambda)^H \boldsymbol{x}(0). \tag{5.8.2}$$

By Parseval's Theorem (A.2.3), it then follows that

$$\begin{aligned} c(\lambda) &= \frac{1}{D}\widehat{\boldsymbol{x}}(\lambda)^H \widehat{\boldsymbol{x}}(0) \\ &= \frac{1}{D}\sum_{k=1}^{D}\left(\widehat{\boldsymbol{x}}(\lambda)^k\right)^* \widehat{\boldsymbol{x}}(0)^k. \end{aligned}$$

For this to be piecewise linear in λ, any non-linear terms in this sum must be negligible.

Given a modified filter of the form

$$\widehat{\boldsymbol{h}} = W^{-1}\widehat{X}\left(\widehat{X}^H W^{-1}\widehat{X}\right)^{-1} D\boldsymbol{v}$$

we showed that this was a SDF in the transformed image space $\boldsymbol{y}$ where

[3]This is a trivial result since it is well known any Hermitian matrix can be factored as $A = Q^H \Lambda Q$ where Λ is a real, diagonal matrix (containing the eigenvalues) and Q is unitary ($Q^H Q = I$).

$EE^H = W^{-1}$ and $\widehat{\boldsymbol{y}} = E^H \widehat{\boldsymbol{x}}$. Thus the self-correlation function becomes

$$\begin{aligned} c(\lambda) &= \frac{1}{D}\widehat{\boldsymbol{y}}(\lambda)^H \widehat{\boldsymbol{y}}(0) \\ &= \frac{1}{D}\widehat{\boldsymbol{x}}(\lambda)^H EE^H \widehat{\boldsymbol{x}}(0) \\ &= \frac{1}{D}\widehat{\boldsymbol{x}}(\lambda)^H W^{-1}\widehat{\boldsymbol{x}}(0). \end{aligned} \tag{5.8.3}$$

In practice, we seldom have a closed expression for $\boldsymbol{x}(\lambda)$, and so have to evaluate $c(\lambda)$ numerically.

5.8.3 MACE Filter

5.8.3.1 Maximum at the Origin

The design of the MACE filter tries to ensure that we have a maximum at the origin since it minimises the average correlation energy subject to constraints at the origin. Alternatively, it can be derived from the MSE SDF by setting the desired shape to be the Dirac delta function. If the output shape is indeed a delta function, the maximum value is trivially at the origin and thus locating the origin is not a problem.

Filtering is, however, a linear operation. Thus it is highly unlikely that the MACE filter will give perfect delta functions as output. We expect to find non-zero values off the origin. Furthermore, it can be shown, that if we choose non-constant $\boldsymbol{v}^j$'s, then we can produce images in the training set which do not have a maximum at the origin.

For example, consider the case of a filter derived from 3 vectors of length 11. The middle 5 values are

$$\boldsymbol{x}_1 = \begin{bmatrix} 0 \\ \vdots \\ 0.64 \\ 0.16 \\ 0.73 \\ 0.27 \\ 0.97 \\ \vdots \\ 0 \end{bmatrix}, \boldsymbol{x}_2 = \begin{bmatrix} 0 \\ \vdots \\ 0.49 \\ 0.28 \\ 0.53 \\ 0.53 \\ 0.15 \\ \vdots \\ 0 \end{bmatrix}, \boldsymbol{x}_3 = \begin{bmatrix} 0 \\ \vdots \\ 0.57 \\ 0.77 \\ 0.9 \\ 0.8 \\ 0.48 \\ \vdots \\ 0 \end{bmatrix}$$

and the rest are zero. We require that the filter response at the origin be 1, 1 and 10 respectively. The resulting correlation outputs (σ) near the origin are (to 2 decimal places)

$$\sigma_1 = \begin{bmatrix} 1.20 \\ 1 \\ 5.59 \end{bmatrix}, \sigma_2 = \begin{bmatrix} 8.97 \\ 1 \\ 2.20 \end{bmatrix}, \sigma_3 = \begin{bmatrix} 7.91 \\ 10 \\ 3.29 \end{bmatrix}.$$

As can be seen, we have the desired response at the origin in all the cases (as we would expect), but it is only a maximum in the last case. From this we see that we cannot use the MACE filter to locate images and estimate distortions without running into the problem of being unable to locate the maximum. We need to choose $\boldsymbol{v}$ constant if we want to be able to locate the origin.

5.8.3.2 Linearity Condition

For the MACE filter, we know that $W = B = \frac{1}{M}\sum_{i=1}^{M} \left(\text{diag}\left(x\right)_i\right)^* \left(\text{diag}\left(x\right)_i\right)$. Substituting this into (5.8.3), we see that

$$c\left(\lambda\right) = \sum_{k=1}^{D} \frac{\left(\widehat{\boldsymbol{x}}\left(\lambda\right)^k\right)^* \widehat{\boldsymbol{x}}\left(0\right)^k}{\sum_{j=1}^{M}\left(\widehat{\boldsymbol{x}_j}^k\right)^* \widehat{\boldsymbol{x}_j}^k}.$$

This is a weighted sum of the earlier expression. The denominator, however, is large when all the images in the training set have significant contributions to that frequency. It is small when most images have little or no contribution to that frequency. It is clear that this emphasises precisely those frequencies where we would expect to observe non-linear behaviour. Thus it is unlikely that this will be piecewise linear. In practice, it is observed that the MACE filter performs poorly for images between those in the training set.

5.8.4 Optimal MSE-SDF

5.8.4.1 Maximum at the Origin

We derived the optimal MSE error by minimising the difference between the correlation surfaces of the images in the training set and the average correlation surface subject to constraints at the origin. The total MSE error

between the correlation surfaces is thus given by

$$\begin{aligned} \text{MSE} &= \widehat{\boldsymbol{h}}^H S \widehat{\boldsymbol{h}} \\ &= \boldsymbol{v}^H \left(\widehat{X}^H S^{-1} \widehat{X} \right)^{-1} \widehat{X}^H S^{-1} S S^{-1} \widehat{X} \left(\widehat{X}^H S^{-1} \widehat{X} \right)^{-1} \boldsymbol{v} \\ &= \boldsymbol{v}^H \left(\widehat{X}^H S^{-1} \widehat{X} \right)^{-1} \boldsymbol{v}. \end{aligned}$$

It fairly obvious that, by choosing arbitrary v_j's, we can make this error as large as we wish. To obtain the desired behaviour, we are constrained in our choice of v_j's. Thus we cannot use the optimal MSE SDF to both locate the origin and estimate distortions in a single step.

On the other hand, if the images resemble the average image closely, then many terms in S are small and $\left(\widehat{X}^H S^{-1} \widehat{X} \right)^{-1}$ will have mainly small values and so the MSE should be small.

The contribution to the MSE due to the differences at the origin of the different output correlation surfaces is given by

$$\text{origin} = \frac{1}{M} \sum_{j=1}^{M} (v_j - \overline{v})^2$$

where $\overline{v} = \sum_{j=1}^{M} v_j$. We immediately can see that, if we choose all the v_j's to be equal, there is no contribution to the error from the origin. Since we wish to minimise the MSE, this suggests that choosing a uniform $\boldsymbol{v}$ has significant advantages for the optimal MSE SDF. Thus, if the mean correlation surface has a maximum at the origin then choosing a uniform $\boldsymbol{v}$ tries to ensure that all the images have a maximum at the origin.

The average correlation surface is given by

$$\begin{aligned} \widehat{\boldsymbol{g}} &= \widehat{A}^H \widehat{\boldsymbol{h}} \\ &= \widehat{A}^H S^{-1} \widehat{X} \left(\widehat{X}^H S^{-1} \widehat{X} \right)^{-1} \boldsymbol{v} \\ &= \widehat{A}^H S^{-1} \widehat{X} \boldsymbol{q}. \end{aligned}$$

This is the correlation of the mean image with some linear combination of the columns of $S^{-1}\widehat{X}$, which are linear transformations of the images in the training set. Furthermore, from the definition of S, it follows that multiplying by S^{-1} emphasises those aspects of $\widehat{X}$ most similar to the mean

image. Thus, if the images are all sufficiently similar, we can be reasonably sure of a maximum at the origin, since we come close to taking the correlation of the mean image with itself, which should have a maximum at the origin.

Of course, sufficiently similar is a vague concept, and in general, we will need to test whether the images satisfy the appropriate conditions.

5.8.4.2 Linearity Condition

For the optimal MSE SDF, the expression for $c(\lambda)$ (5.8.3) becomes

$$c(\lambda) = \sum_{k=1}^{D} \frac{\left(\widehat{\boldsymbol{x}}(\lambda)^k\right)^* \widehat{\boldsymbol{x}}(0)^k}{\sum_{j=1}^{M} \left(\widehat{\boldsymbol{x}_j}^k - \widehat{\boldsymbol{m}}^k\right)^* \left(\widehat{\boldsymbol{x}_j}^k - \widehat{\boldsymbol{m}}^k\right)}.$$

We now divide by the variance of each frequency. Thus we heavily emphasise those frequencies where all the images have similar values, which is where we expect the more linear behaviour. Highly variable frequencies, which we expect to be non-linear, are de-emphasised. Thus we would expect this filter to perform well.

In constructing the filter, however, the estimated variance is used. If our training set is badly sampled, (which can easily occur in practice since the images are frequently selected from some deterministic sequence) it is possible to get unrealistically low variances from the training set. This will over-emphasis frequencies which may actually be very non-linear. Thus we need to be very careful in selecting our training set before using this filter.

5.8.5 MINACE Filter

It follows from (5.8.3) that for the MINACE filter

$$c(\lambda) = \widehat{\boldsymbol{x}}(\lambda) \left(B - b\widehat{C}\right)^{-1} \widehat{\boldsymbol{x}}(0).$$

Thus there is a large a dependence on the choice of C and b and without specifying these it is impossible to make any informative comments on the performance of this filter.

In fact, depending on the choice of C and b, we aren't even assured of a maximum at the origin. For example, choosing $\widehat{C} = B - \mathbf{I}$ and $b = 1$, then $B - b\widehat{C} = \mathbf{I}$ and the MINACE filter reverts to the ordinary SDF, which we know does not necessarily have a maximum at the origin.

For specific problems, however, good results have been achieved using a suitable choice for $\widehat{C}$ (see for example [5]). Unfortunately, unless the problem domain is well specified, choosing $\widehat{C}$ can be hard, so for our purposes, we ignore the MINACE filter.

5.8.6 The MACH Filter

Since the MACH filter is not based on the SDF, its analysis is rather different.

The aim of the MACH filter is to ensure that the output correlation surfaces are as similar as possible. It achieves this by minimising the ASM, so we return to the ASM (5.6.1) for the analysis of how well it achieves this goal. From our earlier work:

$$\begin{aligned}
\text{ASM} &= \frac{1}{M}\sum_{i=1}^{M}\left\|\operatorname{diag}\left(\widehat{\boldsymbol{x}}_i\right)^*\widehat{\boldsymbol{h}} - \widehat{A}^*\widehat{\boldsymbol{h}}\right\|_2^2 \\
&= \frac{1}{M}\sum_{i=1}^{M}\left\|\widehat{\boldsymbol{x}}_i^* \mathbin{.*} \widehat{\boldsymbol{h}} - \widehat{\boldsymbol{a}}^* \mathbin{.*} \widehat{\boldsymbol{h}}\right\|_2^2 \\
&= \frac{1}{M}\sum_{i=1}^{M}\left\|H^*\left(\widehat{\boldsymbol{x}}_i - \widehat{\boldsymbol{a}}\right)\right\|_2^2 \\
&\leq \frac{1}{M}\sum_{i=1}^{M}\|H\|_2^2\,\|\widehat{\boldsymbol{x}}_i - \widehat{\boldsymbol{a}}\|_2^2
\end{aligned}$$

where H is a diagonal matrix with the elements of $\widehat{\boldsymbol{h}}$ on the diagonal and we use the L_2 or spectral matrix norm (largest singular value of H) (see (A.1.1)). Since H is diagonal, this is the largest element on the diagonal and so it follows immediately from our definition of H that

$$\begin{aligned}
\|H\|_2^2 &\leq \left\|\widehat{\boldsymbol{h}}\right\|_2^2 \\
&= 1.
\end{aligned}$$

Then

$$\begin{aligned} \text{ASM} &\leq \frac{1}{M}\sum_{i=1}^{M} \|\widehat{\boldsymbol{x}}_i - \widehat{\boldsymbol{a}}\|_2^2 \\ &\leq \frac{D}{M}\sum_{i=1}^{M} \|\boldsymbol{x}_i - \boldsymbol{a}\|_2^2 \end{aligned} \tag{5.8.4}$$

by Parseval's theorem (A.2.3) where $\boldsymbol{a} = \sum_{i=1}^{M} \boldsymbol{x}_i$ as before. But we recognise (5.8.4) as the conventional SSD measure in the original image space (see section 8.3.2.1). Thus a small ASM requires that conventional SSD must also be small.

Since the SSD is a measure of cluster size, it follows that unless the images for which the filter must be invariant are reasonably tightly clustered, the MACH filter is not expected to perform well.

5.9 Effects of the Background

5.9.1 Effect on Location

In realistic problems, one has to account for background variations. Here we treat it as a perturbation of the image. Thus, given that the correlation surface $\boldsymbol{s}$, given by

$$\boldsymbol{s} = \boldsymbol{x} \circ \boldsymbol{h}$$

has a maximum at the origin, we need to know when

$$\boldsymbol{s}' = (\boldsymbol{x} + \boldsymbol{\epsilon}) \circ \boldsymbol{h}$$

has a similar maximum.

Since correlation is linear, we know that

$$\boldsymbol{s}' = \boldsymbol{x} \circ \boldsymbol{h} + \boldsymbol{\epsilon} \circ \boldsymbol{h}.$$

Thus we can see that if $\boldsymbol{\epsilon}$ is sufficiently small, the perturbation will have no effect. Furthermore, if $\boldsymbol{\epsilon} \circ \boldsymbol{h}$ has the same maximum as $\boldsymbol{x} \circ \boldsymbol{h}$, then it doesn't matter whether $\boldsymbol{\epsilon}$ is small or not. Of course, determining when this will be satisfied requires $\boldsymbol{\epsilon}$ to be known.

This can be seen as a stability result. Small background variations will not unduly affect the location. The effect, however, of large background variations will always be problem specific.

5.9.2 Effect on Estimation

The effect of the background here is different since it is assumed that the location of the origin has already been found. The only concern is how the perturbation affects the response from the origin. For our purposes, since we use only the ordinary SDF for estimation, we only need to concern ourselves with the ordinary SDF in this case. The response is given by

$$\begin{aligned} s_0 &= \boldsymbol{h}^T (\boldsymbol{x} + \boldsymbol{\epsilon}) \\ &= \boldsymbol{h}^T \boldsymbol{x} + \boldsymbol{h}^T \boldsymbol{\epsilon} \end{aligned}$$

In the case of the ordinary SDF, we know that $\boldsymbol{h} = \sum_{i=1}^{M} \alpha_i \boldsymbol{x}_i$ for some set of α_i. If we then split $\boldsymbol{\epsilon} = \boldsymbol{\epsilon}^{\perp} + \boldsymbol{\epsilon}^{\top}$ where $\boldsymbol{x}_i^T \boldsymbol{\epsilon}^{\perp} = 0 \, \forall i$, then

$$s_0 = \boldsymbol{h}^T \boldsymbol{x} + \boldsymbol{h}^T \boldsymbol{\epsilon}^{\top}.$$

Thus if $\boldsymbol{\epsilon}^{\top}$ is sufficiently small, then the estimate is unaffected, regardless of $\boldsymbol{\epsilon}^{\perp}$. Note that we cannot easily apply a similar analysis to the other filters, since they are not simple linear combinations of the $\boldsymbol{x}$'s. There are additional factors which have to be accounted for.

Again, this gives us a stability result, ensuring that small background variations will not have an unduly large effect. It also gives us a filter design criteria. For the estimation problem, the only relevant effect is $\boldsymbol{\epsilon}^{\top}$. Thus we want to design filters for which this is as small as possible. $\boldsymbol{\epsilon}^{\top}$, however, depends on the training set. The more variation (especially in scale) between the images in the training set, the larger $\boldsymbol{\epsilon}^{\top}$ becomes. This suggests that trying to achieve too much with any one filter is a bad idea. This is somewhat different from the point we made in section 5.7.1, where we mentioned that the SDF is inaccurate since it must encompass a large range of distortion, which averages the images. Here we showed that including a large range of distortion increases the sensitivity of the system to background variation. Consequently, when dealing with a large range of distortion, the combined effect of both points is that it is probably better to use several filters, each

describing a smaller range of distortion.

5.10 More About the SDF

There is a great body of existing mathematical techniques and theory which can be applied to the SDF. In this section we wish to discuss two which we think are of particular interest. The first, feature space trajectories, is an alternative method for estimating image distortion. We describe how the two techniques are related. The second technique we discuss is the equidistributing principle, sometimes used to select the nodes for solving differential equations numerically. We describe how this technique can also be used to select the training set for the SDF.

5.10.1 The SDF as a Feature Space Trajectory

5.10.1.1 The Short Version of Feature Space Trajectories

Feature Space Trajectories (FST's) (see [35]) are based on a simple idea. We assume that we have an object from which we can generate observations. From these observations, we can then determine various features of the object. Typically, an observation is an image of the object and the features are then information derived from this image, such as its correlation with some reference image, average intensity, etc.

As we change various parameters of the object, such as its orientation, the observations and thus the features will change. Using FST's, we express the features as a function of the parameter. Thus, if we have 3 features, λ, μ and η and the parameter is orientation, θ, then we can trace the parametric curve given by $\begin{bmatrix} \lambda(\theta) \\ \mu(\theta) \\ \eta(\theta) \end{bmatrix}$. This is a curve in $\mathbb{R}^3$ and is known as the feature space trajectory. Knowing the exact curve, we can recover the angle from the features by finding the value of θ which gives the observed features. Usually a large number of features are used, so the curve lies in $\mathbb{R}^n$. In some cases, this curve is then projected into $\mathbb{R}^3$ to ease analysis.

Generally, we obtain the features experimentally. Thus we do not have the entire curve, just a number of known points on the curve. The key idea in FST's is to approximate the curve by a piecewise linear function. Then, given a set of features which do not lie on the approximated curve, we map it

to the closest point on the curve (closest in the sense of Euclidean distance). This is the same concept we used for the linearly interpolating SDF's.

This only covers the simplest case of the FST. There is a great deal of flexibility in terms of weighting features and dealing with unduly jagged curves. It has also been extended to deal with image classification problems rather than just image location (see [46]). These details, however, are not relevant to this section.

5.10.1.2 SDF Outputs as a FST

There are strong similarities between our concept of linearly interpolating SDF's and the idea of FST's. In fact, we can express the single-parameter SDF as a FST, by simply defining the output of the SDF as a feature. Thus we consider functions of the form $\sigma(\theta)$. For one parameter, the two approaches are identical.

For multiple parameters, we notice an important difference. We have only one feature, while the parameters can change completely independently. For 2 parameters, our feature trajectory is no longer a simple curve, but an entire surface. And given that we only extract a single value from the SDF, there is no way to determine the closest point on the surface. The single feature specifies a line. Thus we cannot expect to determine unique values for several parameters with only one SDF. In the 2 parameter case, to return to a curve and a point, we will need a second feature, so we require another independent SDF.

It is easy to argue from the geometry of the situation that, to determine the values of n parameters, we will need at least n features. Thus to determine several parameters, we will need several independent SDF's.

5.10.2 Choosing the Optimal Training Set for the SDF

We require that the underlying self-correlation function be linear between the sampled points. In many numerical analysis applications, obtaining such a function is not a serious problem since any continuous function can be closely approximated by a piecewise linear function provided the sampled points are suitably chosen. From this it follows that, in theory anyway, we can always design a linearly interpolating SDF, although, if the self-correlation function is highly nonlinear, this may require a large training set. The question then

arises whether there is any method for deciding where to sample the function.

This idea, while theoretically valid, is frequently inapplicable to the SDF as it requires some knowledge of the self-correlation function $c(\alpha)$. In many practical image processing applications, we only have numerical values of $c(\alpha)$ at a given number of sample points, where the sampling is beyond our control. Thus it is not always possible to design a linearly interpolating SDF from the available information, although we can still test for suitability by examining $c(\alpha)$.

For completeness, we describe a well known technique from numerical analysis (often used in solving differential equations numerically) which can be used to select an optimal training set (see for example [39]).

For the purposes of this discussion we will confine ourselves to a function $f(x) \in C^2$ defined on the interval $[0,1]$. Since any continuous function can be rescaled to this interval, there is no loss of generality. We assume we have a a piecewise linear approximation $F(x)$ with nodes

$$0 = x_0 < x_1 < \ldots < x_N = 1$$

(i.e. $F(x_n) = f(x_n)$). We define the maximum norm $\|\bullet\|_i$ on the interval $I_i := [x_{i-1}, x_i]$ as

$$\|g\|_i := \max_{x \in I_i} |g(x)| . \tag{5.10.1}$$

It is well known that the error $e(x) := f(x) - F(x)$ is bounded by

$$\|e\|_i \leq \frac{1}{8} h_i^2 \left\| f''(x) \right\|_i$$

where $h_i = x_i - x_{i-1}$. The global error can be expressed as

$$\|e\|_\infty := \max_{1 \leq i \leq N} \|e\|_i \leq \frac{1}{8} \max_{1 \leq i \leq N} \left(h_i^2 \left\| f''(x) \right\|_i \right) . \tag{5.10.2}$$

We need to place the nodes so that $\|e\|_\infty$ is minimised. In [8], de Boor showed that the right hand side of (5.10.2) is minimised if the nodes are placed to satisfy

$$h_i^2 \left\| f''(x) \right\|_i = k \tag{5.10.3}$$

where k is a constant for all i. Equation (5.10.3) is an example of an equidistributing principle, which places nodes in such a way that some specified quantity is kept constant over each interval. It is clear that (5.10.3) places more nodes in the areas where the second derivative of f is large.

Determining the nodes from (5.10.3) is computationally expensive, so de Boor [8] proposed a simple and efficient approximation. We observe that (5.10.3) is equivalent to

$$h_i \left\| \left| f''(x) \right|^{\frac{1}{2}} \right\|_i = \sqrt{k}.$$

From this, de Boor proposed that the nodes be calculated from

$$\int_{x_{i-1}}^{x_i} \left| f''(x) \right|^{\frac{1}{2}} dx = \widetilde{k}$$

or

$$\int_{x_{i-1}}^{x_i} \left| f''(x) \right|^{\frac{1}{2}} dx = \frac{1}{N} \int_0^1 \left| f''(x) \right|^{\frac{1}{2}} dx. \tag{5.10.4}$$

This gives approximately the same distribution of nodes as (5.10.3). Furthermore it is clear that the constant in (5.10.3) is approximately

$$k \approx \left[\frac{1}{N} \int_0^1 \left| f''(x) \right|^{\frac{1}{2}} dx \right]^2.$$

We can use standard numerical techniques to solve (5.10.4). For further details on the derivation, the reader is referred to [8] or [9]. We present only the simplest version of this technique. Various extensions and improvements are possible. For example, Pereyra and Sewell [39] show how the number of nodes can be calculated as well.

Thus, in situations where we can obtain enough information about $c(\alpha)$ to approximate the second derivative fairly accurately and have complete freedom about where we choose the sample points, we can select an optimal training set for the derivation of the SDF.

Chapter 6

SDF Examples

6.1 Purpose of the SDF

As we mentioned at the start of our discussion of the SDF and its variants, we need a faster method of finding the location of a face than just the eigenface-based approaches, especially if we need to deal with finding faces at a number of different scales and rotations.

We also showed that, provided the images in the training set are suitably chosen, i.e. so that the self-correlation function is piecewise linear, the SDF will allow us to estimate the scale and rotation. This suggests an obvious two-stage algorithm, namely, obtain the estimates via the SDF and use these estimates as starting values for some more accurate algorithm, such as the maximum likelihood algorithm. If our initial estimates are reasonably accurate, this should provide us with a considerable reduction in the search times.

In this section, we provide a few simple examples of how the SDF can be applied, demonstrating its ability to estimate the extent of distortion accurately.

6.2 In Plane Rotation and Scale Changes

6.2.1 Description

Since we are primarily concerned with the problem of locating facial images, we will start with a brief demonstration of how the various SDF-type filters can be used to locate faces and estimate scale and in-plane rotation. For this

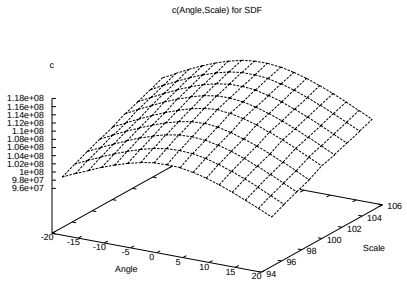


Figure 6.1: Self-correlation function.

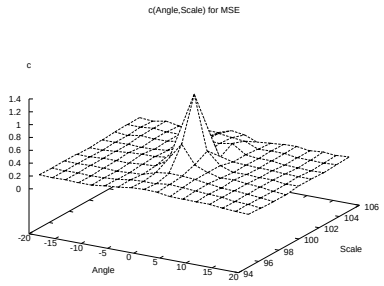


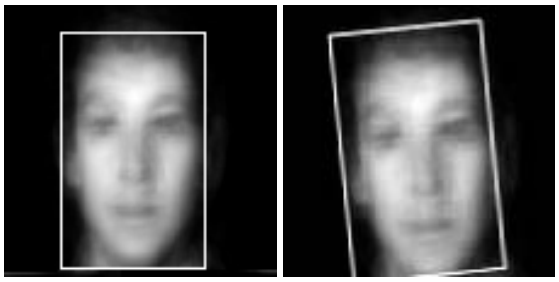
Figure 6.2: Weighed self-correlation function from MSE.

demonstration, we use the same set of images as used in our earlier examples of eigenfaces (see Chapter 3).

A small subset was used to calculate an average face. Rotated and scaled versions of the average face are used as the training set to construct the filters. Two classic SDF's were created to estimate the scale and rotation. An optimal MSE SDF was created for locating images.

Our analysis indicates that we need a piecewise linear self-correlation function. Figure 6.1 shows that the self-correlation surface, calculated by (5.8.2), is close to being piecewise linear. The SDF filters used were created with samples taken at every fourth node on the graph (along each axis). Thus we would expect the SDF approach to perform well.

The weighed self-correlation function from the optimal MSE filter used, calculated by (5.8.3), is shown in figure 6.2. It too is fairly close to linear over most of the range. The central spike is sufficiently broad so that the approximation, based on samples taken at every fourth node on this figure, is sufficiently good for our purposes.



(a) Rotation and scale in training set.

(b) Roation and scale not in the training set.

Figure 6.3: Average face.

6.2.2 Results

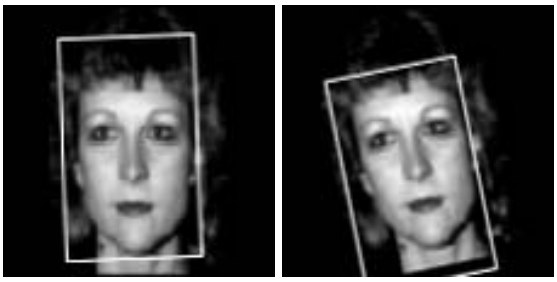
We find that the rotational information is invariably accurate to within a degree of the actual value. The scale estimates are somewhat less accurate, but almost always within a reasonable margin of error (usually about 10% out). Since in any practical implementation, these values are used only as a starting point for more accurate techniques, this behaviour is quite acceptable.

To illustrate the performance of the algorithm, we use the information to draw bounding boxes on a few test cases. In the two cases shown in figure 6.3 and figure 6.4, the image on the left is at a scale and rotation in the training set, while the image on the right is not, consequently the estimated scale and rotation are interpolated.

Figure 6.3 shows the results of testing the filter on the average face. As expected, the estimates for rotation and scale are very accurate.

Figure 6.4 shows the results for one of the images used to create the average face. Not surprisingly, these are less accurate than for the average face, but are still quite acceptable.

Figure 6.5 shows the results for a face quite different from those used to construct the average face. Although the scale estimate is not perfect, the rotation is again very accurate. It clearly serves as a excellent starting point for a more sophisticated procedure. Also, since this image differs significantly from those used in creating the filter, a more representative training set would also improve the results.



(a) Rotation and scale as in the training set.

(b) Rotation and scale not represented in training set.

Figure 6.4: Image similar to average face.

Figure 6.5: Quite different face.

6.3 Out of Plane Rotation

6.3.1 Description

Something that should be stressed is that the SDF can be used to estimate *any* parameter for which the linear interpolating property is satisfied. To demonstrate this graphically, we used the ray tracing package POVRAY (see [58]) to create images with controlled out of plane rotation ranging from 5° to -30°. Figure 6.6 shows two examples from the training set.

We again need the self-correlation function to be piecewise linear. Figure 6.7 plots the self-correlation function while figure 6.8 plots the weighted self-correlation function of the MACE filter. In both cases we show the original as well as the estimates obtained from sampling at every 4 and every 8 nodes. In both cases, the approximations are fairly close, so we expect good results.

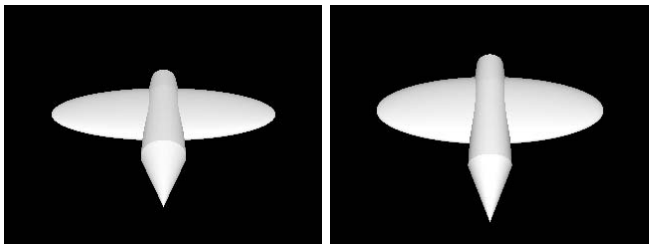


Figure 6.6: Example images in the training set.

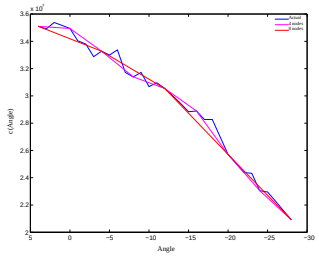


Figure 6.7: Self-correlation function.

6.3.2 Results

Figure 6.9 plots the estimated angle against the true angle sampling at every second node, every 4th node and every 8th node. As expected, the results are very good and the worst error (from using every 8th node) is only 1.2°. In location experiments, the MACE filter successfully locates the image exactly in all cases. Of course, we expect this to decrease if we include background variations.

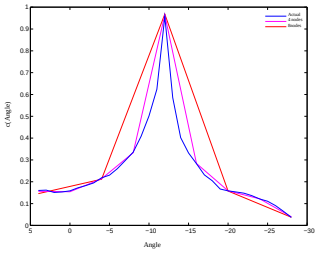


Figure 6.8: Self-correlation function for MACE filter.

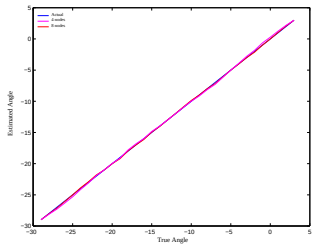
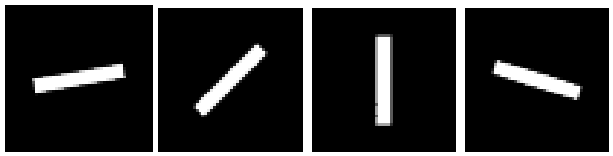


Figure 6.9: Estimated angles plotted against true angle



(a) 10° (b) 45° (c) 90° (d) 163°

Figure 6.10: Typical images.

6.4 A Non-Linear Self-Correlation Function

6.4.1 Description

The images in these experiments use a data-set consisting of rectangles rotated through angles varying from 10° off the horizontal to 170°. The aim is to construct a SDF filter which accurately returns the angle of rotation. Since this is merely a test to demonstrate the capabilities of the SDF filter, we shall ignore the problem of locating the origin of the correlation surface and use our *a priori* knowledge to do this.

Typical examples of the images used are shown in figure 6.10.

The SDF filter is designed to give the correct angle from the input image. We know that the SDF can do this correctly if the self-correlation function $c(\alpha)$ is piecewise linear. In this case the self-correlation function is shown in figure 6.11. It is clearly quite non-linear. By carefully choosing the sampled angles, however, it should be possible to approximate this quite well with a piecewise linear function.

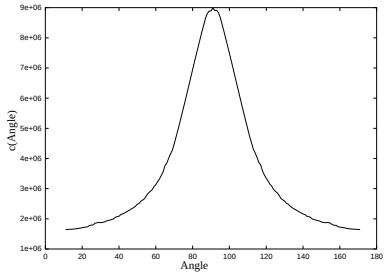


Figure 6.11: $c(\alpha)$ for lines.

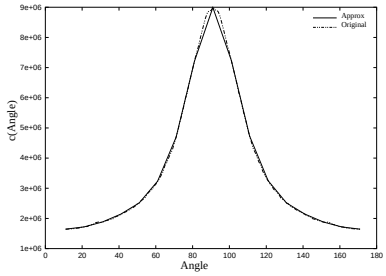


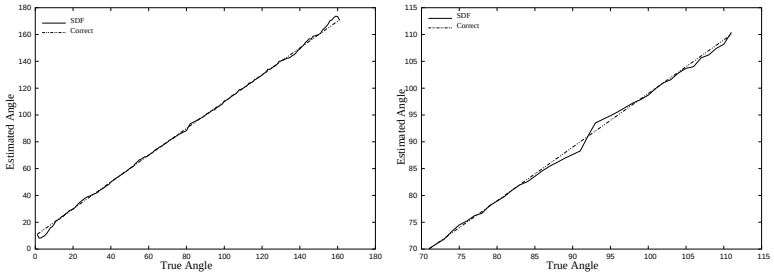
Figure 6.12: Approximation of $c(\alpha)$ using samples taken every 10°.

6.4.2 Results

We demonstrate graphically the results of three different attempts to create an appropriate SDF filter. The first one uses as the training sets sampled at 10 degree intervals. In figure 6.12, we show the approximation to $c(\alpha)$ obtained by this method plotted with the original graph. It is clearly inaccurate, especially near the central peak. In figure 6.13, we plot the identified angle against the true angle. In the magnified display, it is clear that there are quite large errors round the 90 degree mark.

In the second example, we choose a step of five degrees, and display the approximation in figure 6.14 (again with the original $c(\alpha)$) and the results in figure 6.15. Although the results improve, there are still considerable errors around the 90 degree mark.

For the last example, we choose a non-uniform distribution of the nodes, calculated using the algorithm described earlier (in section 5.10.2). We used 25 nodes, a rather ad hoc choice, being chosen as a round figure between the number of nodes for the 10 degree example and the 5 degree example. The average error given by the optimal samples is the smallest of the three. The set of angles used is {10, 12, 15, 20, 30, 45, 60, 70, 80, 84, 86, 88, 90,



(a) Full results. (b) Results around 90°.

Figure 6.13: Results using samples taken every 10°.

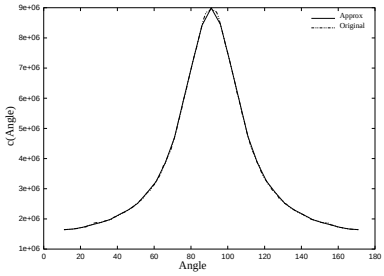
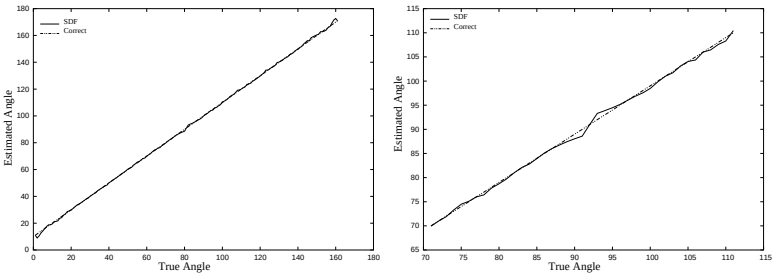


Figure 6.14: Approximation of $c(\alpha)$ using samples taken every 5°.



(a) Full results. (b) Results around 90°.

Figure 6.15: Results using samples taken every 5°.

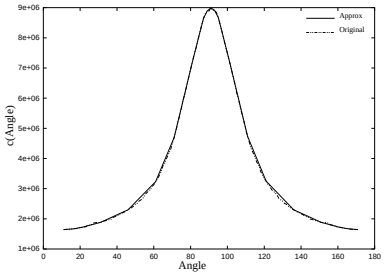
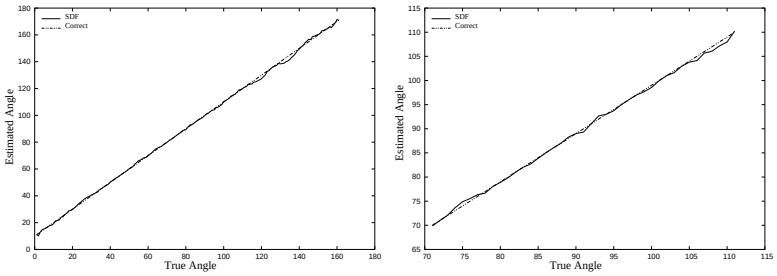


Figure 6.16: Approximation of $c(\alpha)$ using non-uniform sampling.



(a) Full results. (b) Results around 90°.

Figure 6.17: Results using non-uniform sampling.

92, 94, 96, 100, 110, 120, 135, 150, 160, 165, 168, 170}. We note that this emphasises those sections where $c(\alpha)$ has a high curvature, which is what we expect since (5.10.3) involves the second derivative. Linear sections have a low curvature and thus the nodes are well spread out where $c(\alpha)$ is close to linear. The estimate to $c(\alpha)$ (and the original) are shown in figure 6.16 and the results in figure 6.17. In this case, we can see a considerable improvement around the 90 degree mark, as well as an improvement at the ends of the range. The non-uniform choice allows us to focus on the difficult regions.

Chapter 7

Combining Eigenfaces and the SDF

7.1 The Algorithm

As we mentioned earlier, we cannot hope to determine the location exactly using just the SDF. The best we can aim for is to try and reduce the search space before using some more accurate algorithm. To do this, we use 4 filters. The actual procedure consists of the following steps:

1. Determine the location:
   We need to know the origin of the SDF so that we can estimate the parameters. The alternatives that we have tried which may be used for this step are the Optimal MSE SDF, the MACE filter or the MACH filter. Which one should be used depends on the specific problem, as each has its own strengths and weaknesses. For our experiments with faces, we generally used the Optimal MSE SDF, but very similar results were obtained using the MACH filter. The MACE filter, as expected from our analysis, did not perform well in our experiments.
   In our experiments, we design these filters with constant outputs, due to the analysis in Chapter 5. However, in [2], the authors make the point that this can lead to poor discrimination and hence many false alarms. This is not a major factor for our problem, since we will be using a more accurate algorithm later which should eliminate the false alarms.
   Since there tend to be sharp discontinuities near the edge of an image,

which can cause spurious high peaks, we restrict the search for the location to the central subsection of the image (We normally ignore a border of some 30 pixels wide around the edge of the image. Since the faces in our test sets are wider than this, there should be no possibility of our missing the location due to this assumption). We do this for the eigenface location technique as well.

2. Use the SDF to estimate the parameters:
   We are trying to estimate both scale and rotation, so we need to use 2 SDF's. The first to estimate the rotation, the second to estimate the scale.
   We introduce another filter as part of the rather simplistic method we use to compensate for lighting variations (a point we will comment on in more detail in the next section) which models the lighting variation as a simple multiplication of the image elements. In other words, given an image $\boldsymbol{x}$, changing the lighting will be equivalent to multiplying by some scalar β (i.e. the new image $\widetilde{\boldsymbol{x}}$ is $\widetilde{\boldsymbol{x}} = \beta\boldsymbol{x}$). Obviously, if we can obtain β, then since the value at the origin of the filter is $\boldsymbol{h}^T\widetilde{\boldsymbol{x}} = \beta\boldsymbol{h}^T\boldsymbol{x}$, we can divide by β to remove the effect of the lighting.
   The algorithm we use is:

   (a) Apply a filter constructed with $\boldsymbol{v} = \begin{bmatrix} 1 & 1 & \dots & 1 \end{bmatrix}$ to obtain a base value k. If there is no lighting to worry about, in theory $k = 1$. If the lighting has changed, then under our model, $k = \beta$.

   (b) Apply the scale and rotation filters to obtain the outputs p_1 and p_2.

   (c) Use the scaled outputs $\frac{p_1}{k}$ and $\frac{p_2}{k}$ as the actual estimates. If the lighting is a uniform linear scaling of the image pixels and no other variation is present, this will completely nullify the effect on the output. This model is obviously unrealistic, but it gives acceptable results, so we have not tried to improve on it.

3. We now have estimates of the location, scale and rotation. We can thus apply an exhaustive search on a narrow range about these values to determine the final estimates for the scale and location. We refine the search results using the maximum likelihood algorithm discussed

earlier (section 4.2).

7.2 Additional Comments on the Experiments

7.2.1 The Limitations of the SDF

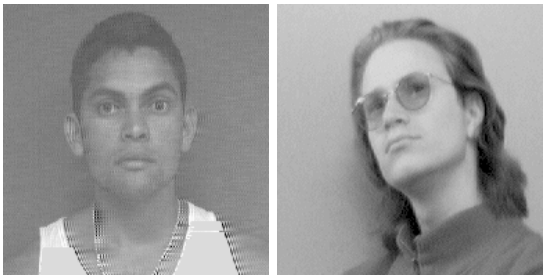
The first important consideration is that the filters we are using have no immunity to lighting variations. For simplicity, we use a simple model of light variation as a linear scaling of all pixel values and compensate for this. This model can only deal with small variations in even lighting. It fails completely when dealing, for example, with directional variations in lighting. Suitable lighting normalisation is a difficult problem. For instance, see [15] for an example of one possible algorithm, which uses a training set of images taken under controlled variations in lighting.

A similar idea can be implemented using SDF filters, by including the lighting variation as another distortion parameter. It requires a carefully constructed training set with the variations one would expect to deal with, which depends on the envisioned application. This approach will also impact on the accuracy of the estimated parameters as it adds extra variation.

7.2.2 The Test Set

The images used for these tests were selected at random from a large database (several thousand images), at a resolution of 200×200 pixels. The images were used to provide pictures for student identification cards. They represent a wide range of different racial groups, but are predominantly of people between 18 and 25. Since many students only had their pictures taken at the start of their academic careers, there is a bias towards the younger end of this range. These are images gathered in a fairly simplistic setup, with no thought to an automated recognition system, so the variation is not well controlled at all, making them quite a difficult test case. In a full system, we would expect to be able to gather better quality images.

The images were gathered by asking the person to sit and look at a digital camera which was then operated by hand. Very little effort was made to ensure that people were at the same distance from the camera, so there is considerable scale variation. Illumination was also not particularly well



(a) Poor contrast between face and background.

(b) Large out of plane rotation.

Figure 7.1: Difficult images in the test set.

controlled.

Images were usually taken with a white wall as background, but due to the lighting variations, there are many images where the contrast between the face and the background is very low. Rotation is also a problem, especially when there is considerable out of plane rotation. Two difficult images are shown in figure 7.1.

7.2.3 Reference Results

In order to assess the accuracy of any given algorithm, we need reference results. To generate these, we chose 50 images as a training set for the eigenfaces. The images were cropped to remove hair and background and the location of the face in these images was normalised by hand. The hand normalisation was improved by iterating the maximum likelihood algorithm over these images. Typical examples are shown in figure 7.2. These images where then used to generate the eigenfaces. These training set images are all upright, but as figure 7.2 shows, the scale normalisation is not perfect.

The reference results were then generated by running the maximum likelihood algorithm on the test set. We made no attempt to classify how accurate these reference results are since our interest is in whether the SDF-based technique is of any use as a starter for the maximum likelihood algorithm, so we are only interested in comparative performance. However, due to the difficult nature of the test set, there are of course cases where the maximum likelihood algorithm gives poor results.

Figure 7.2: Two of the images used to generate the eigenfaces.

(a) Upright test case.

(b) Non upright test case.

Figure 7.3: Example test set images.

7.3 Pure SDF Results

7.3.1 Location tests

We want some indication of how well the SDF-based filters can actually locate the image. For the purposes of this test, we ignored rotation and scale. We chose a test set of 50 images which the maximum likelihood algorithm returned as being upright and of a suitable scale. An example is shown in figure 7.3(a). We constructed an optimal MSE SDF filter which was meant to have a maximum at the origin from the average face and a random selection of 10 images from the eigenface training set. The c-function for the MSE SDF shown is figure 7.4, while not perfectly linear, is sufficiently flat that virtually any sampling of images from the training set will give a "reasonable" linear approximation.

For control purposes, we tested the filter on the entire training set. Not surprisingly, the filter returned virtually perfect results. Only two images in

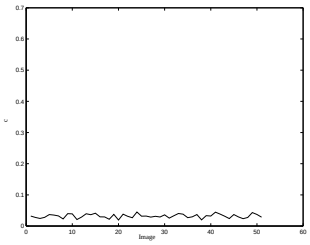


Figure 7.4: c-function for MSE SDF.

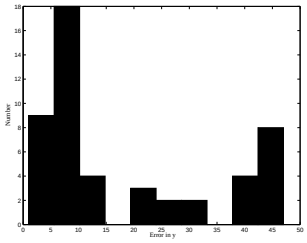


Figure 7.5: Pixel error in the y coordinate.

the training set were not placed at the origin and in those cases the error was less than four pixels in both x and y.

For the test set, the results were less accurate. We observe that our location in the y direction is more accurate than in x, but this is probably due to the greater variation across the height of an image, making false peaks in this direction less likely.

Using the filter, the y coordinate returned was within 10 pixels of the correct location more than 50% of the time and within 30 pixels more than 80% of the time (remember that we are dealing with 200×200 pixel images). A histogram of the error in y is shown in figure 7.5. As can be seen, the most significant peak lies with an error of between 7 and 10 pixels. There are a significant number where the error is unacceptably high, which is due mainly to the poor separation between the background and the region of interest in these cases.

The x coordinate returned was less accurate, being within 20 pixels slightly more than 50% of the time and within 30 pixels 80% of the time, although, as the histogram in figure 7.7 shows, we were in this test case never closer than 10 pixels away.

If we plot the total distance from the true centre, as down in figure

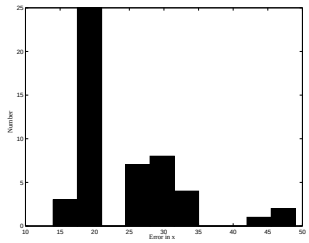


Figure 7.6: Pixel error in the x coordinate.

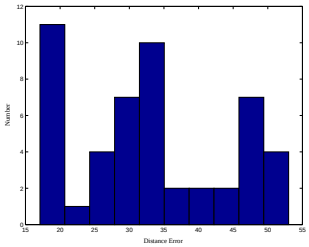


Figure 7.7: Total distance from correct location.

7.7, we see that although we were seldom exact, the error was usually less than a total of 35 pixels (60% of the time). This indicates that when we performed poorly in the x direction, we also usually performed poorly in y. We can also see that there are a significant number of cases where the error is unacceptably high.

Since we are using this as a starting point for the maximum likelihood algorithm, which provides a measure of whether a face is actually present at a given location, we can always fall back to an exhaustive search when we cannot find a suitable region within a given distance of the starting location. In about 60% of the cases, using the SDF results in a significant reduction in the size of the search space.

Furthermore, in many of the cases where we perform badly, there is a smaller peak close to the true location. Thus we can search around the second largest peak more than 35 pixels from the original. If we do this, we see that for more than 80% of the images, we are within 35 pixels of one of these two peaks (as shown in figure 7.8).

Thus the SDF approach frequently gives us a good starting point for the maximum likelihood algorithm.

We should not be surprised that the filter performs worse on this test set

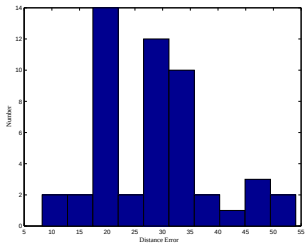


Figure 7.8: Distance from closest of the 2 peaks.

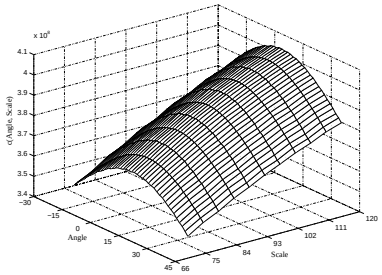


Figure 7.9: Self-correlation function for for SDF.

since it adds considerable variation that was not present in the training set, such as the presence of the background. In a more controlled environment, we would hope that the performance should improve, since this sort of additional variation should be greatly reduced.

Experiments using a MACH filter gave similar results.

7.3.2 Distortion estimation

Of course, the location step is perhaps the least important. What we want is estimates of extent of the distortion (in our case scale and in-plane rotation) of an image.

For this experiment, we constructed a filter using the average face rotated through several angles (from $-30°$ to $30°$) and scales (from 80% to 120%). The self-correlation function for the SDF is shown in figure 7.9. It is close to linear, so we can be optimistic about the results.

The self-correlation function for the optimal MSE SDF is shown in figure 7.10. The majority of the function is fairly linear, with the typical sharp peak. Thus, provided an adequate sampling is used near the peak, we would

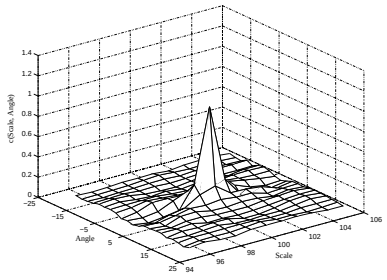


Figure 7.10: Self-correlation function for optimal MSE SDF.

expect good results. In this figure we only show the section of the self-correlation function covering the range from $-20°$ to $20°$ and 95% to 105% to display the width of the peak's base clearly.

For these experiments, we only use the average face to create the filter, and add the additional variation of rotation and scaling. Not surprisingly, the results are less accurate than for the pure location case.

For the images in the training set, we estimate the scale to within 3% and the angle to within 5 degrees. The location is within a distance of 5 pixels.

We use a test set 200 images for which the results of an exhaustive search by the maximum likelihood algorithm were known. An example of a non-upright image is shown in figure 7.3(b).

Based on the results of the previous section, we take the closest of the two largest peaks and consider the estimates obtained from this.

If we consider the location results, shown in figure 7.11, we see that we are within a radius of 35 pixels almost 75% of the time, which compares very favourably with the results when there was no additional distortion imposed. Again, we find that our estimate in the y-direction is better than the estimate in the x-direction.

The estimated angle is shown in figure 7.12, we can see that the estimate angle is within $10°$ degrees about 50% of the time and within $15°$ degrees some 70% of the time. It is worth noting that of the images which are within a radius of 35 pixels of the true location, almost 60% are within $10°$ and 88% are within $15°$ of the true angle.

The results for the scale estimates are shown in figure 7.13. Here we find that the scale estimate is within 5% some 70% of the time and within 10%,

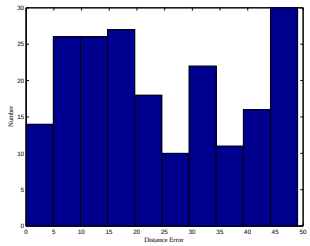


Figure 7.11: Distance of estimated location from the true location.

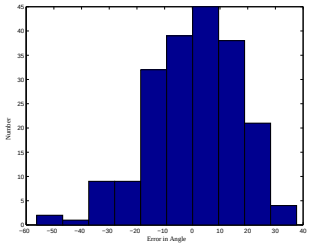


Figure 7.12: Estimated angle.

90% of the time. Of the images where the location is within 35 pixels, our scale estimate is within 5% some 78% of the time.

Again we can see that in many cases, the SDF output will give us a good starting point for the maximum likelihood algorithm.

7.4 Results of the Combined Algorithm

For this experiment, we used the results of the SDF-based filters to start the maximum likelihood algorithm. We are interested in how often the results converge to those of the exhaustive search. From our earlier results we would

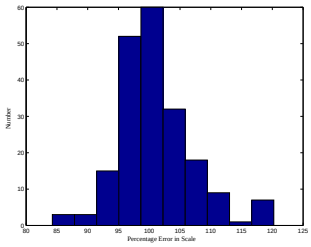


Figure 7.13: Estimated scale.

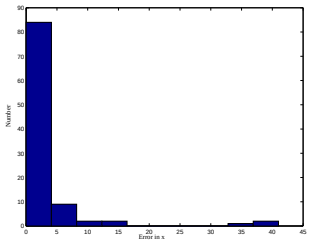
expect this to be around 70% to 75% of the time.

For this experiment, we take a sample of 100 faces (none are in the test set of 200 used earlier) for which the results of the maximum likelihood algorithm are known and apply the algorithm described earlier to it, searching around the two largest peaks. We observe that the run time dropped by factor of around 2.5 compared to that of the maximum likelihood algorithm in this case. (From about 3.8×10^{10} floating point operation in MATLAB to slightly less than 1.5×10^{10} floating point operations). For simplicity, although we restrict the search to a radius of 35 pixels from the peaks, we still calculate the full correlation operation in the maximum likelihood algorithm, which introduces a number of unnecessary operations. Optimising this aspect will become significant when dealing with large images.

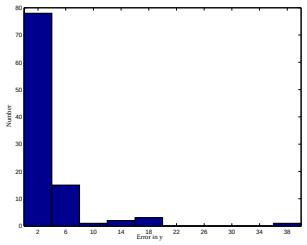
The histograms for the errors in the various parameters are shown in figure 7.14. We find that we get an exact match to the results of the maximum likelihood algorithm 17% of the time, while we are within an acceptable error range (± 2 pixels in x and y, ± 2 degrees in angle and $\pm 5\%$ scale variations) 70% of the time, which is what we would expect based on our earlier results. Since there are numerous sources for numerical error in the process of resizing and rotating the images for the maximum likelihood algorithm, these results are quite acceptable.

In a practical implementation, it is possible to threshold on the output of the maximum likelihood algorithm, which allows us to fall back onto a exhaustive search in those cases where the SDF does not give a good starting point. In these cases, since we would have already covered part of the search space, running time will be only marginally slower than for the pure maximum likelihood algorithm.

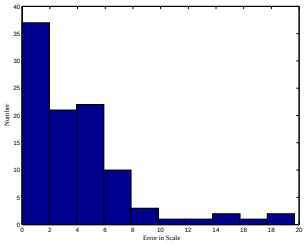
Of the 30 images where there is a large difference between the results of our algorithm and the maximum likelihood algorithm, we note that in 10 of the cases, our algorithm gives what we would describe as visually better results than the exhaustive search, as demonstrated in figure 7.15. This is due to the fact that the SDF can in some cases successfully extrapolate beyond the fixed range imposed in the exhaustive search. However, the fact that we need this extrapolation is a result of the poorly controlled test images and in a practical implementation, we would always expect the exhaustive search to perform better than the SDF-based algorithm, as it does for the remaining 20 images, as shown in figure 7.16.



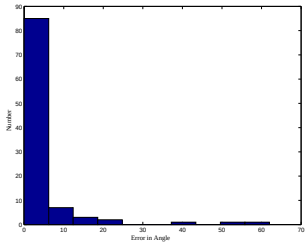
(a) x coordinate.



(b) y coordinate.



(c) Scale.



(d) Angle.

Figure 7.14: Error histograms for full algorithm.

(a) Exhausive search.

(b) SDF-based algorithm.

Figure 7.15: Case where the SDF-based algorithm gives a better result than an exhaustive search.

(a) Exhausive search. (b) SDF-based algorithm.

Figure 7.16: Case where an exhaustive search outperforms the SDF-based algorithm.

Since the maximum likelihood algorithm fails in several cases, this indicates that our test case is rather difficult. That we successfully obtain a significant reduction in the search space for a more than 70% of the images using just the average face to construct the filter is, we feel, testament to the accuracy of the SDF-based approach.

Chapter 8

Clustering and Eigenfaces

8.1 Introduction

In our earlier analysis of the eigenface technique (section 2.3), the results indicated that the robustness of the system depends on how tightly the images are clustered around the mean (based on equations (2.2.2) and (2.3.16)). From this we concluded in section 2.3.5 that it might be beneficial to cluster the images.

In this section, we describe some simple experiments to test if there is any practical gain from clustering faces. Due to the reasons discussed earlier, we make no attempt to consider recognition performance and merely consider the ability of the eigenfaces to reconstruct the image accurately.

In addition to improving the robustness of the system, clustering the image database into smaller sections would allow a hierarchical search for a match, which would reduce the number of comparisons needed in a recognition system (see [45]). This should have significant benefits in practical implementations. The clustering, however, must be stable. If images of the same individual will be placed in different clusters due to the normal variations over time, obviously clustering will not be useful (we are not overly concerned with the effect of extreme variations as these will be always be problematic). Since we do not have a large database of faces with the same individual taken at different times, we cannot address this issue here.

8.2 Experimental Design and Setup

For this experiment we use the images from the database described in Chapter 7. For any experiment involving clustering, a large number of images are required. We started with some 400 images from the database, which we normalised using the SDF-based algorithm. We used a threshold in the maximum likelihood algorithm and a final pass by hand to weed out poorly located images. A simple filtering step was then used to remove most of the background. We are left with 351 images all centred, all very close to upright and close to the same scale.

From these 351 images, we calculate the eigenfaces and observe that to describe 95% of the total variation requires almost 40 eigenfaces and 97% requires 60 eigenfaces.

The question we wish to answer is whether it is possible to find some sub-division of the training set into different clusters which are not too small that allows us to obtain equally good reconstructions with significantly fewer eigenfaces in each cluster. If clustering has no major benefit, we would expect the number of eigenfaces required to drop only slightly. Before we attempt to answer this question, we need to provide some background on automatic clustering.

8.3 Automatic Clustering

8.3.1 Aim

Automatic clustering techniques have received a great deal of attention in a variety of fields, so we do not attempt a complete review of the existing literature. We merely provide a brief description of some of the more popular and best-understood techniques.

8.3.2 The k-means Algorithm

8.3.2.1 Refining an Initial Subdivision

A common idea is to start from some initial sub-optimal division of the data into a specified number of clusters and then refine this until we arrive at the "best possible" division of the data for that number of clusters.

One significant problem is that we need to know the number of clusters from the outset. There are several ways of deciding this, a matter we will discuss in the next section where we describe methods for obtaining the initial clustering.

In order to improve the initial clustering, we require a suitable measure of how good the given clusters are. Since we want clusters that satisfy assumption (2.2.2), an obvious choice is to use the variance within each individual cluster.

The variance measure most suitable for our purposes is based on the sum of the squared distances of the cluster elements from the cluster mean, which is known as the cluster SSD. The sum of all the cluster SSD's is known as the total SSD and gives a measure of the average variance. We wish to partition our training set into N clusters so that the total SSD is a minimum. Let C_i be the set of the elements of the i'th clusters and m_i the number of elements in C_i ($\sum_{i=1}^{N} m_i = M$ where M is the size of the training set). The vectors $\boldsymbol{x}_{ji} \in C_i$ are the individual elements of the i'th cluster[1] where $j = 1, \ldots m_i$. The cluster mean is

$$\overline{\boldsymbol{x}_i} = \frac{1}{m_i} \sum_{j=1}^{m_i} \boldsymbol{x}_{ij}. \tag{8.3.1}$$

The quantity to be minimised, the total SSD, is given by

$$\text{total SSD} = \sum_{i=1}^{N} \sum_{j=1}^{m_i} \|\boldsymbol{x}_{ji} - \overline{\boldsymbol{x}}_i\|_2^2 .$$

In order to find the global minimum for the total SSD we need to consider all possible divisions of the training set. This is computationally very expensive and so we accept a local minimum found by successively refining our initial clustering. The method we use for this is one of the more popular clustering algorithms and is known either as the k-means algorithm [48] or generalised Lloyd algorithm [13].

Since we modify the clusters, we need to know how these changes will

[1] In our case, the $\boldsymbol{x}$'s will be the images.

affect the total SSD. We define e_q to be the SSD of q'th cluster, C_q, i. e.

$$e_q = \sum_{j=1}^{m_q} \|\boldsymbol{x}_{jq} - \overline{\boldsymbol{x}}_q\|_2^2 .$$

From the definition of the mean, we know that

$$\sum_{i=1}^{m} (\boldsymbol{y}_i - \overline{\boldsymbol{y}}) = 0$$

where $\overline{\boldsymbol{y}} = \frac{1}{m} \sum_{i=1}^{m} \boldsymbol{y}_i$, so it follows that

$$e_q = \sum_{i=1}^{m_q} \|\boldsymbol{x}_{iq}\|_2^2 - m_q \|\overline{\boldsymbol{x}}_q\|_2^2 . \tag{8.3.2}$$

Let us now consider a cluster C_j, which contains at least 2 elements. We divide C_j into 2 disjoint and non-empty subsets, C_k and C_p, i.e. C_k is a subset of C_k and C_p is the complement of C_k in C_j. Formally, $C_k \neq \emptyset$, $C_p \neq \emptyset$, $C_j = C_p \bigcup C_k$ and $C_k \cap C_p = \emptyset$.

From (8.3.1) it follows that

$$\begin{aligned} \overline{\boldsymbol{x}}_j &= \frac{1}{m_j} \left(\sum_{i=1}^{m_k} \boldsymbol{x}_{ik} + \sum_{l=1}^{m_p} \boldsymbol{x}_{lp} \right) \\ &= \frac{1}{m_j} \left(m_k \overline{\boldsymbol{x}}_k + m_p \overline{\boldsymbol{x}}_p \right) \end{aligned} \tag{8.3.3}$$

and so

$$\overline{\boldsymbol{x}}_p = \frac{1}{m_j - m_k} \left(m_j \overline{\boldsymbol{x}}_j - m_k \overline{\boldsymbol{x}}_k \right)$$

since $m_p = m_j - m_k$.

From this, we can rewrite e_p (from 8.3.2) as

$$\begin{aligned} e_p &= \sum_{i=1}^{m_p} \|\boldsymbol{x}_{ip}\|_2^2 - m_p \|\overline{\boldsymbol{x}}_p\|_2^2 \\ &= \sum_{i=1}^{m_j} \|\boldsymbol{x}_{ij}\|_2^2 - \sum_{l=1}^{m_k} \|\boldsymbol{x}_{ik}\|_2^2 - m_p \left\| \frac{m_j \overline{\boldsymbol{x}}_j - m_k \overline{\boldsymbol{x}}_k}{m_p} \right\|_2^2 \\ &= (e_j + m_j \|\overline{\boldsymbol{x}}_j\|_2^2) - (e_k + m_k \|\overline{\boldsymbol{x}}_k\|_2^2) - m_p \left(\frac{\|m_j \overline{\boldsymbol{x}}_j - m_k \overline{\boldsymbol{x}}_k\|_2^2}{m_p^2} \right) \\ &= e_j - e_k + m_j \|\overline{\boldsymbol{x}}_j\|_2^2 - m_k \|\overline{\boldsymbol{x}}_k\|_2^2 - \frac{1}{m_j - m_k} \|m_j \overline{\boldsymbol{x}}_j - m_k \overline{\boldsymbol{x}}_k\|_2^2 . \end{aligned}$$

Since

$$\|m_j \overline{\boldsymbol{x}}_j - m_k \overline{\boldsymbol{x}}_k\|_2^2 = \left(m_j \|\overline{\boldsymbol{x}}_j\|_2\right)^2 + \left(m_k \|\overline{\boldsymbol{x}}_k\|_2\right)^2 - 2 m_j m_k \overline{\boldsymbol{x}}_j^T \overline{\boldsymbol{x}}_k$$

it follows that

$$\begin{aligned} e_p &= e_j - e_k + \left(m_j - \frac{m_j^2}{m_j - m_k} \right) \|\overline{\boldsymbol{x}}_j\|_2^2 + \left(\frac{m_k^2}{m_j - m_k} - m_k \right) \|\overline{\boldsymbol{x}}_k\|_2^2 \\ &\quad + \frac{2 m_j m_k}{m_j - m_k} \overline{\boldsymbol{x}}_j^T \overline{\boldsymbol{x}}_k \\ &= e_j - e_k - \frac{m_j m_k}{m_j - m_k} \|\overline{\boldsymbol{x}}_j - \overline{\boldsymbol{x}}_k\|_2^2 . \end{aligned}$$

Similarly, we can show that

$$e_j = e_p + e_k + \frac{m_p m_k}{m_p + m_k} \|\overline{\boldsymbol{x}}_p - \overline{\boldsymbol{x}}_k\|_2^2 . \tag{8.3.4}$$

Let us consider the case where $m_k = 1$ (i.e.. where C_k has only 1 element, $\boldsymbol{x}_{1k}$). Then, from the above results,

$$\overline{\boldsymbol{x}}_p = \frac{m_j \overline{\boldsymbol{x}}_j - \boldsymbol{x}_{1k}}{m_j - 1}$$

$$\overline{\boldsymbol{x}}_j = \frac{m_p \overline{\boldsymbol{x}}_p + \boldsymbol{x}_{1k}}{m_p + 1}$$

$$e_p = e_j - \frac{m_j}{m_j - 1} \|\overline{\boldsymbol{x}}_j - \boldsymbol{x}_{1k}\|_2^2$$

$$e_j = e_p + \frac{m_p}{m_p + 1} \|\overline{\boldsymbol{x}}_p - \boldsymbol{x}_{1k}\|_2^2 .$$

These results describe how the cluster mean and the cluster SSD will change if we add or remove an element from the cluster.

The k-means algorithm uses these results to determine how the total SSD would change if an element is moved to another cluster. We determine the change for moving an element to each other cluster and move the element only if the move decreases the total SSD. If there are several possible moves, we will choose the move which leads to the greatest reduction in the total SSD.

Let us consider $\boldsymbol{x}_{ir}$, which is currently assigned to cluster C_r. For all the clusters C_s, $s \neq r$, we calculate $\frac{m_s}{m_s+1} \|\overline{\boldsymbol{x}}_s - \boldsymbol{x}_{ir}\|_2^2$, the contribution to the total SSD of placing $\boldsymbol{x}_{ir}$ in cluster C_s. If

$$\frac{m_s}{m_s + 1} \|\overline{\boldsymbol{x}}_s - \boldsymbol{x}_{ir}\|_2^2 \geq \frac{m_v}{m_v + 1} \|\overline{\boldsymbol{x}}_v - \boldsymbol{x}_{ir}\|_2^2 \ \ \forall\, j$$

(i.e. adding $\boldsymbol{x}_{ir}$ to cluster C_v has the smallest contribution to the total SSD) and

$$\frac{m_r}{m_r - 1} \|\overline{\boldsymbol{x}}_r - \boldsymbol{x}_{ir}\|_2^2 > \frac{m_v}{m_v + 1} \|\overline{\boldsymbol{x}}_v - \boldsymbol{x}_{ir}\|_2^2$$

(i.e. by moving $\boldsymbol{x}_{ir}$ from C_r to C_v, we will decrease the total SSD) then we move $\mathbf{x}_{ir}$ to C_v. After the re-assignment, the total SSD becomes

$$\text{New total SSD} = \text{old total SSD} - \frac{m_r}{m_r - 1} \|\overline{\boldsymbol{x}}_r - \boldsymbol{x}_{ir}\|_2^2 + \frac{m_v}{m_v + 1} \|\overline{\boldsymbol{x}}_v - \boldsymbol{x}_{ir}\|_2^2 .$$

This leads to the largest possible reduction in the total SSD that can be obtained by moving $\boldsymbol{x}_{ir}$.

In the k-means algorithm, we consider each element of the training set and, unless it is in a cluster of size 1 (to prevent us creating any empty clusters), we test it against all the other clusters, moving it to a different cluster if that decreases the total SSD. We repeat this process until there are no more reassignments which reduce the total SSD. Thus the running

time for a single pass through the training set depends on the number of elements in the set and the number of clusters, while the total running time also depends on the initial sub-division. Also this algorithm only finds a local minimum for the SSD, which depends on the initial clustering.

8.3.2.2 Choosing the Initial Subdivision

For our description of the k-means algorithm we assumed that the number of clusters to be used is already known. Choosing the number of clusters to use for the algorithm is frequently the least well defined part of any clustering application. Factors that must be considered usually include the total SSD, the separation between clusters and the size of the smallest (or largest) cluster. Quantifying the effects of all the factors is seldom easy.

A large number of clustering algorithms exist which do not require that the number of clusters is known. Generally these either repeatedly merge small clusters or sub-divide large clusters. These algorithms can be designed to stop once a suitable trade-off between the factors of interest has been found. However, this approach is frequently slow and usually gives a non-optimal clustering which is then refined by means of the k-means algorithm (e.g. the Pairwise Nearest Neighbour algorithm from [11]). Due to the computational complexity of these approaches, we choose to ignore them for the purposes of this experiment.

In cases where the final refinement is done using the k-means algorithm, various ad-hoc methods for selecting the number of clusters can also be used. Using some predetermined rule, various numbers of clusters are tried, providing estimates from which the final number of clusters can be determined. While several techniques (see [48]) are available, we briefly describe two, namely random allocation and the so-called leader principle.

Random allocation is as simple as the name suggests. For a given number N, we randomly select N distinct vectors from the training set. These form our initial cluster means. We then proceed to cluster the training set based on these, assigning each element to the cluster with the closest cluster mean, updating the cluster means and the total SSD as we progress through the data. Since it is a random scheme, it is usual to repeat it several times for each N and use the average results.

The leader principle works by assigning the N initial cluster means so that they are as widely spread as possible. This can easily be accomplished

by the following algorithm.

```
Select the 2 elements in the training set that are
   furthest apart.
For J running from 3 to N
   Select element J so that its average distance from
   the J-1 already selected elements is as large as
   possible.
```

After selecting the N starting points, the clustering proceeds as for random allocation. This method has the advantage that the clustering obtained is stable for N, so no repetition is needed, but the process of selecting the initial cluster means makes it computationally more expensive.

Both methods give upper bounds on the total SSD obtained from the k-means algorithm. They cannot give precise information on cluster sizes, but do provide an indication of what range of sizes can be expected.

The results of either a random allocation or the leader principle can be used to start the k-means algorithm. Both methods are used in the literature. The leader principle is preferred in some applications as the results are stable, but in neither case are we assured that the final answer is optimal.

8.4 Experimental Results

Due to the size of the images, we do all the calculations for the clustering algorithms using the projection of the images onto the eigenfaces. However, the projection of two vectors being close to each other is no guarantee that the two original vectors are actually close, as shown in figure 8.1. Thus we use 200 eigenfaces for these calculations, which describe more than 99.7% of the information about the images so that any error introduced by the projection should be small enough to be safely ignored.

In practice, this clustering step will only be done once, when we design the system. Any new images will than be placed into the existing clusters.

8.4.1 Choosing the Number of Clusters

As described earlier, we need to choose the number of clusters. For our purposes, we require that the total SSD be small and that the smallest

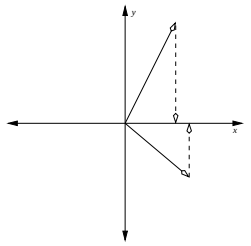


Figure 8.1: Distant vectors whose projections (onto x) are close.

cluster still has enough elements that we can sensibly apply the eigenface technique.

We use both the leader principle and random allocation with the number of clusters ranging from 1 to 10. For the random allocation, we repeat the experiment 20 times on each occasion. In figure 8.2, we plot the average SSD and average smallest cluster size for the random case as well as the SSD and the smallest cluster size for the leader principle.

We see that the SSD drops off fairly rapidly until about 5 clusters. The size of the smallest cluster, however, becomes unacceptably small for more than 4 clusters. Thus the optimal number would seem to be either 3 or 4 clusters. This means that the population of the training set is rather homogeneous, which is what we would expect since there is little variation in age.

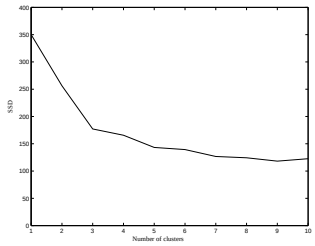
8.4.2 After k-means Refinement

We performed a k-means refinement on all the initial clusterings obtained in the previous section. The total SSD figures quoted are accurate to three decimal places.

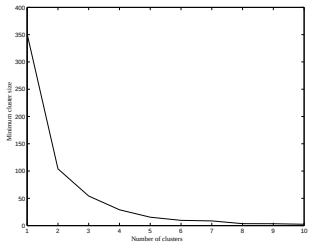
In the case of three clusters, we obtain the best results from refining one of the random allocations[2], giving a total SSD of 135.868, as opposed to starting from the leader principle, which gives a final SSD of 142.275. The size of the smallest cluster in this case is 48.

In the case of 4 clusters, we obtain the smallest SSD by refining the leader principle. The total SSD is 116.497. But the smallest cluster has only

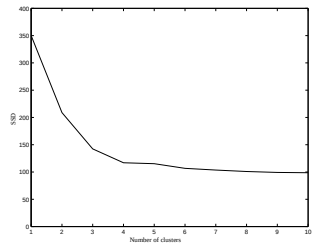
[2]Of the 20 initial random clusterings, 17 converged to this value. While we cannot confirm that this clustering is in fact optimal, it is better than the results obtained from refining the leader principle. Thus it is an example where refining from the leader principle definitely does not lead to the optimal clustering.



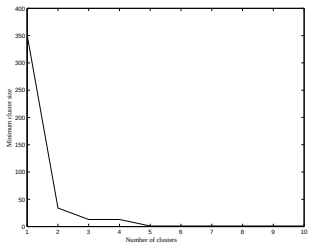
(a) Average SSD for random allocation.



(b) Aveage minimum cluster size for random allocation.



(c) SSD for leader principle.

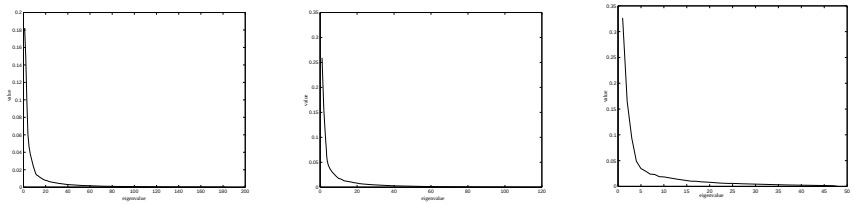


(d) Minimum cluster size for leader principle.

Figure 8.2: Results for initial clustering.

13 elements, which is too small for the eigenface technique to be applied. Starting from one of the random allocations, in three cases we obtain results which are identical to those we obtain from the leader principle. Of the rest, the sub-division with the largest value for the smallest cluster size has an SSD of 129.745, with the smallest cluster containing 48 elements. This, however, is clearly not an optimal sub-division.

It is interesting to compare the best 4-cluster sub-division to the best 3-cluster sub-division. The two largest clusters in each case are virtually identical. In the 3 cluster case, they each contain only 1 image not found in the 4-cluster sub-division. Thus we can see that the 4-cluster sub-division essentially divides the smallest cluster in the 3-cluster sub-division. We also note that in the 4-cluster sub-division where the smallest cluster has 48 elements, this small cluster is identical to the smallest cluster in our best 3-cluster sub-division.



(a) eigenvalues for cluster 1 (size=183).

(b) eigenvalues for cluster 2 (size=120).

(c) eigenvalues for cluster 3 (size=48).

Figure 8.3: Eigenvalues (σ^2) for clustered sub-division.

Thus we chose to use three clusters for our experiment. The final cluster sizes were 183, 120 and 48.

If we examine the clusters, we notice that the smallest cluster differs from the others mainly in illumination. As mentioned, the images were taken under varying lighting conditions. The majority of the images are moderately dark, but there are a handful of fairly bright images, These bright images all fall into the smallest cluster. This is not surprising considering that several papers (see [38]) have found that the first few eigenfaces generally describe the majority of the illumination variance. However, it does mean that we cannot generalise the results of this experiment to a more controlled training set. It also illustrates that in practice, one cannot ignore the problem of varying illumination. There is no immediate obvious visual difference between the two largest clusters, although the majority of the black faces fall into the largest cluster.

8.4.3 Eigenface Representation of the Final Subdivision

Since we have a good sub-division into 3 clusters, we calculate the eigenfaces for each cluster independently. Since we wish to consider the variation amongst the faces of each class individually, we treat the faces assigned to each class as separate datasets for the calculation of the eigenfaces. It is possible to use the SVD to characterise the between-class variation , but that is of no interest to us here. We plot the eigenvalues obtained from each cluster in figure 8.3.

The same fast drop-off we observed in figure 2.4 applies in all 3 cases.

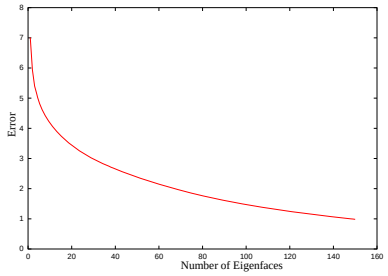


Figure 8.4: Mean grey scale error (full set of 351 images).

Of interest is how many eigenfaces it takes to describe each cluster. To describe 95% of the variation in cluster 1 requires 70 eigenfaces. To describe 95% of the variation in cluster 2 requires 50 eigenfaces and to describe 95% of the variation in cluster 3 requires 30 eigenfaces. It is important to realise, however, that these values are not comparable with those for the full set of images as we are describing different variations. Rather than describing 95% of the total variation, we are talking about the variation within any given cluster, which is not the same thing.

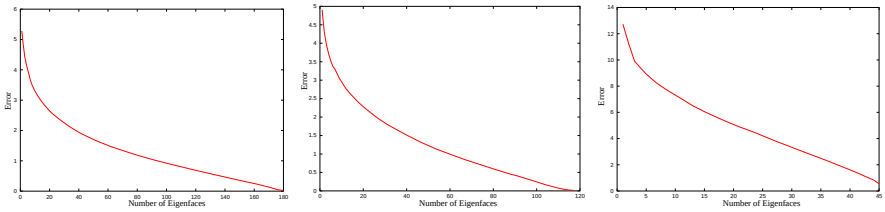
We need to compare the quality of the reconstruction in the clustered case against that of the full set of eigenfaces. This is again an image comparison problem. As described in the Introduction, this can be a difficult problem. Here, however, we are faced with a much more limited comparison problem since if the reconstruction were perfect, then the images would be identical (within numerical accuracy anyway). There are a large number of measures we can use to determine how good a reconstruction is in this situation. Because it is fairly simple to compute, we use the mean absolute grey scale error of the reconstruction. Given the original image $\boldsymbol{F}$ and the reconstruction $\widetilde{\boldsymbol{F}}$, the absolute grey scale error vector is

$$\boldsymbol{E} = \left| \boldsymbol{F} - \widetilde{\boldsymbol{F}} \right| \tag{8.4.1}$$

and the mean error M_e is

$$M_e = \frac{1}{D} \sum_{i=1}^{D} E^i. \tag{8.4.2}$$

We plot the mean absolute grey scale error as a function of the number of eigenfaces used. In figure 8.4, we display the error for the full set of images,



(a) Cluster 1 (183 images).

(b) Cluster 2 (120 images).

(c) Cluster 3 (48 images).

Figure 8.5: Mean grey scale error (for the three clusters).

while in figure 8.5, we display the graphs for each of the three clusters. We see that for the full set, we need around 75 eigenfaces to get a mean grey scale error of less than 2. However, to achieve the same result in the clustered case we need 38, 26 and 37 eigenfaces respectively. This is in total more eigenfaces than required for the full case[3], but the number of eigenfaces required for each individual cluster is significantly less than that for the total training set, which shows some promise for the clustering approach. Since the representation in any of the clusters is considerably more compact than the representation using the full set, we would also expect any search through the database to be considerably faster in the clustered case.

It is of interest to note how poorly the smallest cluster performs in this case. Our earlier conclusion was that the performance of the eigenface method depended on how tightly clustered the images were. Since the smallest cluster is divided in the 4-cluster case, it is obviously the least tightly clustered of the three, so its poor performance is not surprising and agrees with our earlier analysis. This again indicates the problems involved in selecting a good training set. In this case, we have a small set of images which obviously describe quite a large fraction of the total variation of the entire training set. This is clearly undesirable.

[3]However, as with any simple metric, the mean error does not consider the variation in the errors, which is smaller in the clustered case.

8.5 Reduced Illumination Variation Experiment

8.5.1 Experiment Setup

In the previous example, one cluster was determined almost exclusively by the variation in illumination. This occurred because we did not consider the problems of normalising illumination. However, a sub-division based on illumination variation is not particularly useful in a recognition system. Also, this cluster, which was the smallest, had an unacceptably high variance for the number of images in the cluster, indicating that the training set does not describe this section of the "face space" well at all. Thus, to approximate the situation where we have uniform illumination more accurately, we simply remove all the images in the smallest cluster in the previous experiment. Although the remaining images still have variations in illumination, it has been considerably reduced.

As before, we experiment with several cluster sizes, using the same methods described earlier, to obtain a 4-clusters division. This divides the 303 remaining images into clusters of 98, 92, 60 and 53 images respectively.

It is difficult to determine any particular characteristics from the different clusters. However, we do note that the 53 image cluster consists almost entirely of black faces, while the 92 image cluster contains no black faces. The other two clusters are fairly mixed in terms of race, but almost all the images of men with beards are found in the largest cluster (although these are less than half of the images in that cluster[4]). The value of clustering all the bearded faces together is unclear, especially as we do not have any images of the same faces without beards, so we cannot determine whether this is likely to cause any problems.

8.5.2 Results

As before, we look at the mean grey scale error. The graph for the full set is shown in figure 8.6, while those for the four clusters are shown in figure 8.7.

Using the full set of eigenfaces, we require around 70 eigenfaces to obtain an error of less than 2 grey scale levels. To achieve the same results in the clustered case we require 25, 24, 18 and 21 eigenfaces respectively. This is a considerable reduction in the number of eigenfaces required for each cluster,

[4]There are very few images in the database of bearded men. This is probably because the majority of the images are of first year students.

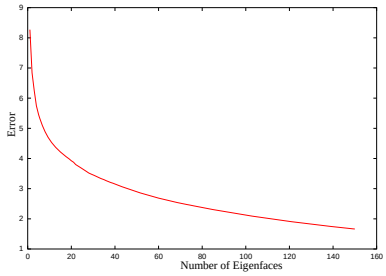


Figure 8.6: Mean grey scale error (full set of 303 images).

which we feel indicates that the clustering is beneficial. Again, the reduced number of eigenfaces should also make searching for a match much easier.

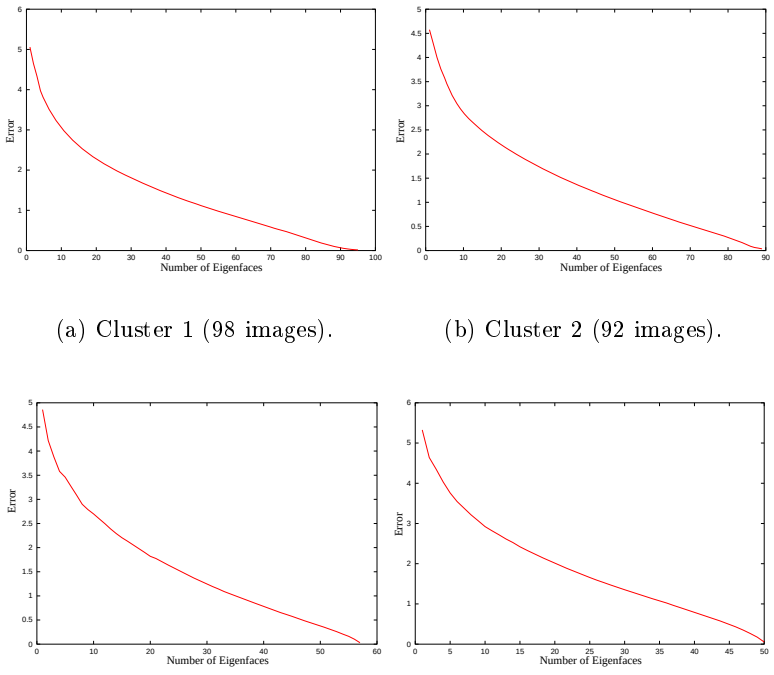
We also observe that the four clusters in this case provide much more uniform performance than in the previous experiment. This is an indication that the clusters here are all reasonably tight.

8.6 Conclusions

For our experiments, we have shown there is a benefit to using a clustering scheme in terms of the error in the reconstruction. It is worth noting that here we have considered the mean error. If we consider the maximum element of $\boldsymbol{E}$ (8.4.1), then we observe that although the mean error for the full set and the clustered case may be the same, the maximum is always smaller in the clustered case, which is hardly surprising considering that the clustered case has a greatly reduced range.

However, the training set used is not yet sufficiently representative for us to generalise these results. Furthermore, since we are talking here about the mean error in the reconstruction, it is unclear what impact this will have on recognition performance. The more compact representation will benefit searches through the database. The sub-division of the database will also have benefits for searching for a match, but until more work is done, it is unclear whether such a sub-division of the database will cause any serious problems for a recognition system.

There are also many practical considerations that need to be addressed before a recognition system using clustering can be designed. In particular, we need to examine how the cluster to which an image is assigned will vary over time. This criteria will have a considerable impact on the reliability of



(a) Cluster 1 (98 images). (b) Cluster 2 (92 images).

(c) Cluster 3 (60 images). (d) Cluster 4 (53 images).

Figure 8.7: Mean grey scale error (for the four clusters).

such a system. We do not, however, have access to a large database of faces which has a significant number of images of the same individual taken at different times, nor is there one publicly available, a problem which needs to be addressed before we can examine this further.

Chapter 9

Conclusions

The motivation behind the work presented in this thesis is provided by the eigenface approach to facial recognition and face verification. The key idea is that the eigenfaces form a basis for the linear subspace containing all the facial images. This basis is efficiently calculated using the Singular Value Decomposition. These eigenfaces are calculated once and for all from a training set of representative faces. Although conceptually very simple, there are several interesting questions one can ask. For instance, isn't is likely that a truly representative training set will become excessively large? After all, it has to present all possible facial images. What are the chances that an unknown face, not encountered before will be adequately represented by the fixed set of eigenfaces? In an attempt to find answers to these questions and others, we considered the situation where a single face is added to the training set. A standard result for symmetric perturbations of symmetric matrices states that the size of the perturbation in the eigenfaces depend on the separation of the singular values. Provided the facial images satisfy the clustering condition (2.2.2) and the training set is well-chosen, we showed that the system is stable with respect to the addition of new faces. More precisely, we showed that under these conditions, the addition of a new face does not have a significant effect on the choice of the eigenfaces. In addition, since the lower order eigenvalues are more widely separated, these are more stable, explaining for example why the MIT system performs well using only the first twenty eigenfaces. This, of course also provides an alternative explanation for the fact that the lower order eigenfaces provide more general face information than the higher order ones. Finally, the perturbation theory also

indicates that the stability should improve if the faces are more tightly clustered, suggesting that clustering the faces into sub-sections should improve performance.

This idea was tested experimentally in Chapter 8. Using a standard automatic clustering algorithm, namely k-means refinement, to separate the faces into clusters based on their eigenface representation, we demonstrated that there is a significant improvement in the compactness of the representations in the clustered case. This happens without any loss in accuracy. In the first experiment we were able to reduce the representation from 75 eigenfaces to between 21 and 38 eigenfaces depending on the cluster. The main drawback is that the clustering also tends to separate on large-scale variations, such as lighting variations. This is clearly not a good idea—we do not want the same person to end up in different clusters simply because of lighting variations. We therefore also conducted experiments simulating situations where lighting conditions can be more carefully controlled, or situations where different lighting conditions can be corrected for. Not surprisingly, our experiments with a data set with reduced variation in lighting again shows a significant improvement in reconstruction performance. In this case we reduced the representation from 70 eigenfaces to between 18 and 25 eigenfaces depending on the cluster. It should be emphasised that these results are for reconstruction; it is not yet clear what impact this approach will have on recognition performance and further work is needed on this aspect. It also illustrates how important lighting variations are for practical implementations.

Another benefit derived from the perturbation analysis is that it leads to an efficient algorithm for updating the eigenfaces if a new face is added to the training set. Our experiments clearly demonstrated that our updating algorithm can successfully improve the description of a new face dramatically with only a small impact on the representation of the faces already described by the system. In our experiments, the difference between the reconstructions obtained from recalculating the eigenfaces and updating them was only 0.3% for the new face and less than 0.03% for faces in the database.

One of the many practical problems that need to be addressed by a facial recognition system is the one of locating the face in the image. Accordingly, we discussed two eigenface-based systems for locating a face in a scene. We provided a comprehensive description of the more accurate maximum likelihood based method and briefly described the so-called “Distance

From Face Space" algorithm. We also examined the computational cost of each method. The "Distance From Face Space" algorithm is fairly accurate, but the maximum likelihood algorithm is more robust. Both methods become prohibitively expensive when it becomes necessary to search through a wide range of distortions such as in-plane rotation and scale changes.

In order to reduce the computational cost, and at the same time retain the robustness of the maximum likelihood method, we examined the Synthetic Discriminant Function, as a means to obtain estimates of the distortions, in detail. This necessitated a thorough analysis of the SDF. We introduced the idea of a linear interpolating SDF's and showed how it can be used to estimate distortion parameters. From this, we derived a sufficient condition, which can be experimentally tested, which will ensure that a SDF is linearly interpolating. It is based on what we refer to as the self-correlation function and it is this function that needs to be piecewise linear. Maybe the best illustration of this idea is to be found in the application of the equidistributing principle from approximation theory to construct a good training set for the SDF. Although this idea can only be applied in situations where one has complete control over the construction of the training set—a situation that is rarely encountered in practice—it does illustrate the validity of the linear interpolating ideas, as well as explaining why SDF's often need a very large amount of training data.

We demonstrated the ability of the SDF to estimate distortions in several examples. Of particular note, we demonstrated how the SDF can also estimate unusual distortions such as out of plane rotation. This is potentially very useful with the growing interest in using 3D techniques for face recognition.

Fortunately these results can be extended to most of the popular SDF variants, since it was demonstrated how they can be described as SDF's in a transformed image space. In addition it also explains some of the observed behaviour of these SDF variants, such as poor performance of the MACE filter on images outside it training set.

Although not directly derived from the SDF, the Maximum Average Correlation Height (MACH) Filter, is still closely related to the SDF. We were able to show how the performance of this filter depends on the images in the training set being reasonably tightly clustered.

Finally we also showed how the SDF's are also connected to the Feature

Space Trajectory method of calculating image distortion.

The developments described above led us to proposed a two-stage algorithm for locating the face in an image, where we first use a SDF-based approach to obtain initial estimates to the distortion and then use the maximum likelihood algorithm to locate the face very accurately. This algorithm requires using 4 filters to estimate the distortion, and therefore only four fast Fourier transforms, a considerable saving over the much more expensive eigenface-based algorithms. Using a filter based only on the average face, our experiments on quite a difficult test set showed that in 70% of the cases, we are able to estimate the scale to within 10%, the rotation to within 15° and the location to within 35 pixels of that given by the maximum likelihood algorithm. Refining these results by the maximum likelihood algorithm allowed us to achieve results very close to those returned by an exhaustive search. In these cases, we gain a significant improvement in the running time for locating a face.

Of course, there is considerable scope for further work. As mentioned several times, lighting variation is a major problem and there is considerable scope for improvement in the current solutions. An obvious one is to move over to a 3-D representation of a face, but this requires either using multiple cameras and solving the associated stereo problem or integrating a range finder into the system. Both possibilities introduce various practical problems and also increase the cost of the system, so more investigation is needed into the potential benefit. Another recent suggestion (see [59]) is to use a modified edge detection scheme to obtain features that are invariant under lighting, yet robust against small changes in movement.

Other areas of future work include examining how the error grows when using the updating algorithm repeatedly for large databases and also the possibilities of using the updating algorithm to select an optimal training set for the eigenfaces from a large database.

Apart from many practical implementation details that need to be addressed, we believe that there is considerable scope for improvement in the filter-based approach. Areas that still should be investigated are whether a suitably designed MINACE-style filter can significantly improve performance, especially against cluttered backgrounds and whether it is possible to improve performance by using more filters covering a reduced range of variation. Another possible area of research is whether using complex out-

puts, as suggested in [22] will have any major influence on the results.

Appendix A

Mathematical Background

For convenience, we provide a brief overview of some of the basic mathematical results and definitions used this thesis.

A.1 Linear Algebra

Eigenfaces are one of the many applications of linear algebra. We assume the following basic concepts in linear algebra are known: vectors, matrices and vector multiplication. We use the notation $\boldsymbol{x}^T$ to indicate the transpose and $\boldsymbol{x}^H$ to indicate the Hermitian (transpose and complex conjugate).

For our purposes, the following results are important:

Symmetric Matrix: A matrix X where $X^T = X$.

Hermitian Matrix: $X^H = X$. The complex analogue of symmetric.

Orthogonal vectors: Two vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ where $\boldsymbol{a}^T\boldsymbol{b} = 0$.

Orthonormal vectors: Vector $\boldsymbol{q}_1$ and $\boldsymbol{q}_2$ which are orthogonal and normalized, i.e. $\boldsymbol{q}_1^T\boldsymbol{q}_1 = \boldsymbol{q}_2^T\boldsymbol{q}_2 = 1$.

Orthogonal Matrix: Matrix Q satisfying $Q^TQ = I$ (where I is the identity matrix), i.e. its columns form an orthonormal set of vectors.

Unitary Matrix: Matrix Q satisfying $Q^HQ = I$. The complex analogue of orthogonal.

L_2 **Norm:** We use the L_2 norm in almost all cases. For a vector $\boldsymbol{x}$, $\|\boldsymbol{x}\|_2^2 = \boldsymbol{x}^T\boldsymbol{x}$ (in the case of a complex vector $\|\boldsymbol{x}\|_2^2 = \boldsymbol{x}^H\boldsymbol{x}$). For a matrix

A, the L_2 norm (also known as the spectral norm) is given by $\|A\| = \sup_{\|\boldsymbol{x}\|_2 \neq 0} \frac{\|A\boldsymbol{x}\|_2}{\|\boldsymbol{x}\|_2}$ or $\|A\|_2 \geq \frac{\|A\boldsymbol{x}\|_2}{\|\boldsymbol{x}\|_2} \forall \|\boldsymbol{x}\| \neq 0$. It can be shown that $\|A\|_2$ is the largest singular value (A.1.4) (see SVD).

Eigenvectors: Given a square matrix X, it eigenvectors are those vectors which satisfy the relationship $X\boldsymbol{u} = \lambda\boldsymbol{u}$ for some scalar λ.

Eigenvalue: In the above relationship, the eigenvalue associated with the eigenvector $\boldsymbol{u}$ is the scalar λ.

.* product: Following MATLAB, we definition the .* product of two D-dimensional vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ as the D-dimensional vector $\boldsymbol{a}$.* $\boldsymbol{b} = \begin{bmatrix} a^1b^1 & a^2b^2 & \dots & a^Db^D \end{bmatrix}$.

SVD Factorisation of an $m \times n$ matrix A so that $A = U\Sigma V^T$ where U and V are both orthogonal matrices (or unitary when A is complex). Σ is a diagonal matrix of the same size as A with the singular values along the diagonal. The columns of the $n \times n$ matrix V are eigenvectors of A^TA and the columns of the $m \times m$ matrix U are eigenvectors of AA^T. The singular values are given by $\sigma_i^2 = \lambda_i$ where the λ_i's are the eigenvalues of AA^T and A^TA. $\Sigma = \text{diag}\ (\sigma_1, \sigma_2, \dots)$. The singular values are non-negative.

We used the identity $\|A\boldsymbol{x}\|_2 \leq \|A\|_2 \|\boldsymbol{x}\|_2$. This follows immediately from the definition above, namely $\|A\| = \sup_{\|\boldsymbol{x}\|_2 \neq 0} \frac{\|A\boldsymbol{x}\|_2}{\|\boldsymbol{x}\|_2}$. We need to show that this is equivalent to the largest singular value. We consider only the case of a real matrix as the complex case is a simple extension.

From the SVD, $A = U\Sigma V^T$. We assume that σ_1 is the largest singular value and $\sigma_1 > 0$. The case $\sigma_1 = 0$ is trivially true since it implies A is the zero matrix. Since V is orthogonal and of size $n \times n$, we can expand $\boldsymbol{x}$ in terms of the columns of V. Thus $\boldsymbol{x} = \alpha_1\boldsymbol{v}_1 + \alpha_2\boldsymbol{v}_2 + \ldots + \alpha_n\boldsymbol{v}_n = V\boldsymbol{\alpha}$. From the orthogonality of V, it follows that $\|\boldsymbol{x}\|_2^2 = \sum_{j=1}^n \alpha_j^2$.

From the definition of the L_2-norm,

$$\frac{\|A\boldsymbol{x}\|_2}{\|\boldsymbol{x}\|_2} \leq \|A\|_2$$

where $\boldsymbol{x} \neq 0$. We square both sides of the inequality:

$$\begin{aligned} \frac{\|A\boldsymbol{x}\|_2^2}{\|\boldsymbol{x}\|_2^2} &\leq \|A\|_2^2 \\ \frac{\|A\boldsymbol{x}\|_2^2}{\sum_{j=1}^n \alpha_j^2} &\leq \|A\|_2^2 . \end{aligned} \tag{A.1.1}$$

The numerator of the left hand side can be rewritten as

$$\begin{aligned} x^T A^T A x &= \left(\sum_{j=1}^n \alpha_j \boldsymbol{v}_j^T\right) V\Sigma\Sigma^T V^T \left(\sum_{j=1}^n \alpha_j \boldsymbol{v}_j\right) \\ &= \boldsymbol{\alpha}^T V^T V \Sigma\Sigma^T V^T V \boldsymbol{\alpha} \\ &= \boldsymbol{\alpha}^T \Sigma\Sigma^T \boldsymbol{\alpha} \\ &= \sum_{j=1}^n \sigma_j^2 \alpha_j^2 \end{aligned} \tag{A.1.2}$$

since $\Sigma\Sigma^T$ is diagonal. The inequality becomes

$$\frac{\sum_{j=1}^N \sigma_j^2 \alpha_j^2}{\sum_{j=1}^n \alpha_j^2} \leq \|A\|_2^2 . \tag{A.1.3}$$

We define the set L so that $\alpha_j \neq 0 \,\forall\, j \in L$ and $\alpha_j = 0 \,\forall\, j \notin L$. Equality is obviously only possible if the σ_j's multiplying the non-zero α_j's are equal, i.e $\sigma_j = \sigma \,\forall\, j \in L$. Since σ_1 is the largest singular value, obviously the right hand side of (A.1.1) is maximised when $\sigma = \sigma_1$. Therefore (A.1.3) becomes

$$\begin{aligned} \frac{\sigma_1^2 \sum_{j\in L} \alpha_j^2}{\sum_{j\in L} \alpha_j^2} &= \|A\|_2^2 \\ \sigma_1 &= \|A\|_2^2 \end{aligned} \tag{A.1.4}$$

and the proof is complete.

A.2 Fourier Analysis

The continuous Fourier Transform of a function f (in 1 dimension) is defined as

$$\hat{f}(\omega) = \int_{-\infty}^{\infty} f(t) e^{-i\omega t} dt \tag{A.2.1}$$

with the inverse transform being defined as

$$f(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \hat{f}(\omega) e^{i\omega t} d\omega.$$

The discrete Fourier transform of a vector $\boldsymbol{f} = [f_1, f_2, \ldots, f_D]$ is the vector $\widehat{\boldsymbol{f}} = \left[\widehat{f}_1, \widehat{f}_2, \ldots, \widehat{f}_D\right]$ where the elements are given by

$$\widehat{f}_{m+1} = \sum_{n=0}^{D-1} f_{n+1} e^{-2\pi i \frac{mn}{D}}, m \in [0, D-1] \tag{A.2.2}$$

and the discrete inverse transform is

$$f_{n+1} = \frac{1}{D} \sum_{m=0}^{D-1} \widehat{f}_{m+1} e^{2\pi i \frac{mn}{D}}, n \in [0, D-1].$$

It is clear that the discrete Fourier transform can be written as a matrix multiplication:

$$\widehat{\boldsymbol{f}} = F\boldsymbol{f}$$

with F a $D \times D$ matrix with F_{mn}, the element in the m'th row and n'th column, given by

$$F_{mn} = e^{-2\pi i \frac{(m-1)(n-1)}{D}}.$$

It also follows immediately that $F^{-1} = \frac{1}{D} F^H$.

Our choice of constants used for the transform and the inverse transform are commonly used, although it is possible to define the transform and inverse transform with different constants. However this has no effect on our analysis in any way.

One of the results we use frequently is Parseval's theorem. It is usually

stated as

$$\int_{-\infty}^{\infty} f(t) f(t)^* dt = \frac{1}{2\pi} \int_{-\infty}^{\infty} \hat{f}(\omega) \hat{f}(\omega)^* d\omega.$$

The discrete version is

$$\|\boldsymbol{f}\|_2^2 = \frac{1}{D} \left\| \widehat{\boldsymbol{f}} \right\|_2^2. \tag{A.2.3}$$

An important result is the convolution theorem. Recall that in the continuous case convolution is defined as

$$f(x) * g(x) = \int_{-\infty}^{\infty} f(t) g(t-x) dt$$

and the discrete case

$$\boldsymbol{h}^j = (\boldsymbol{f} * \boldsymbol{g}) = \sum_{i=1}^{D} \boldsymbol{f}^i \boldsymbol{g}^{j-i}.$$

The convolution theorem can be stated as

$$\widehat{f * g} = \widehat{f}\widehat{g} \tag{A.2.4}$$

and

$$\widehat{fg} = \widehat{f} * \widehat{g}.$$

Since we have been dealing almost exclusively with filters, we are more interested in the related correlation theorem. Recall that in the continuous case, correlation is defined as

$$f(x) \circ g(x) = \int_{-\infty}^{\infty} f^*(t) g(t+x) dx$$

and in the discrete case

$$\boldsymbol{h}^j = (\boldsymbol{f} \circ \boldsymbol{g}) = \sum_{i=1}^{D} \left(\boldsymbol{f}^i\right)^* \boldsymbol{g}^{j+i}.$$

The correlation theorem is

$$\widehat{f \circ g} = \widehat{f}^* \widehat{g} \tag{A.2.5}$$

and

$$\widehat{f^* g} = \widehat{f} \circ \widehat{g}.$$

This approach is frequently used to calculate correlations and convolutions, but it assume that the vector is periodic, i.e. $f_{D+j} = f_j$. This can cause problems especially near the end of the vector. A common solution is to pad the vector with zeros. It is easy to see that these problems will be avoided by padding with $\frac{D}{2}$ zeros, i.e. we use vectors of the form

$$\widetilde{f}_j = \begin{cases} 0 & j \leq \frac{D}{2} \\ f_{j-\frac{D}{2}} & D + \frac{D}{2} \geq j > \frac{D}{2}. \\ 0 & j > D + \frac{D}{2} \end{cases}$$

This does however increase the computational cost.

Appendix B

Summary of Eigenface Terms

Training set: Representative set of images from which the eigenfaces are constructed. We denote images in the training set as $\boldsymbol{F}_1, \boldsymbol{F}_2, \ldots, \boldsymbol{F}_M$ where M is the number of images in the training set and $\boldsymbol{F}_i$ is the vector representation of the i'th image in the training set.

Average face: The average of the elements of the training set. $\boldsymbol{A} = \frac{1}{M}\sum_{i=1}^{M} \boldsymbol{F}_i$.

Caricature: The difference between an image in and the average face. $\boldsymbol{C}_i = \boldsymbol{F}_i - \boldsymbol{A}$.

d: Constant used to normalise the images. $d^2 = \frac{1}{M}\sum_{i=1}^{M} \|\boldsymbol{F}_i - \boldsymbol{A}\|_2^2$. This is an approximation to the measure of spread we defined in (2.2.1).

Normalised Images: Caricature divided by normalised constant. $\boldsymbol{X}_j = \frac{\boldsymbol{F}_j - \boldsymbol{A}}{d}$.

X: Matrix constructed from the normalised images. $X = \frac{1}{\sqrt{M}}\begin{bmatrix} \boldsymbol{X}_1 & \boldsymbol{X}_2 & \ldots & \boldsymbol{X}_M \end{bmatrix}$.

Covariance matrix: The matrix $C = XX^T$.

SVD: Singular Value Decomposition. Matrix factorisation used to construct the eigenfaces.

Eigenfaces: Set of orthonormal vectors used to decompose the face. A small number of eigenfaces should capture the majority of the information about the faces. They are obtained from the SVD of X. The eigenfaces are a sub-set of the basis vectors of the sub-space spanned by the training set. They are also the eigenvectors with the greatest

eigenvalues of the covariance matrix, hence the name. We represent the eigenfaces as $\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_\nu$ where we keep ν eigenfaces.

"Face space": The sub-space spanned by the eigenfaces.

Eigenface representation: Given $\boldsymbol{F}$, the image of a face, the representation is the set of numbers $\boldsymbol{y} = \begin{bmatrix} y_1 & y_2 & \ldots & y_\nu \end{bmatrix}$ so that $y_1\boldsymbol{u}_1 + y_2\boldsymbol{u}_2 + \ldots + y_\nu\boldsymbol{u}_\nu + \boldsymbol{A} = \widetilde{\boldsymbol{F}} \approx \boldsymbol{F}$. This is calculated by $\boldsymbol{y} = U_\nu^T(\boldsymbol{F} - \boldsymbol{A})$ where $U_\nu = \begin{bmatrix} \boldsymbol{u}_1 & \boldsymbol{u}_2 & \ldots & \boldsymbol{u}_\nu \end{bmatrix}$. $\boldsymbol{y}$ minimises $\epsilon(\boldsymbol{F}) = \left\| \boldsymbol{F} - \widetilde{\boldsymbol{F}} \right\|$.

Eigenface reconstruction: The image constructed from the eigenface representation. $\widetilde{\boldsymbol{F}} = U_v\boldsymbol{y} + \boldsymbol{A}$.

DFFS: Distance From Face Space. The L_2 distance between an image and its eigenface reconstruction (i.e. the information lost by projecting onto the eigenfaces). Calculated as $\text{DFFS}^2 = \epsilon(\boldsymbol{F})^2 = \|\boldsymbol{F} - \boldsymbol{A}\|_2^2 - \|\boldsymbol{y}\|_2^2$.

Maximum Likelihood Algorithm: An algorithm based on modeling the distribution of the faces in the face space with a Gaussian distribution. Provides a more accurate estimate for the location than the DFFS measure.

Appendix C

Summary of SDF Terms

$\mathcal{X}_i$**:** i'th 2-D image in the training set.

$\boldsymbol{x}_i$**:** Vector representation of i'th image in training set, $\mathcal{X}_i$.

$\widehat{\boldsymbol{x}}_i$**:** Fourier transform of $\boldsymbol{x}_i$.

diag $(\boldsymbol{x}_i)$**:** Diagonal matrix with the elements of $\boldsymbol{x}_i$ on the diagonal.

X**:** Matrix $X = \left[\begin{array}{cccc} \boldsymbol{x}_1 & \boldsymbol{x}_2 & \ldots & \boldsymbol{x}_M \end{array} \right]$ where there are M images in the training set.

$\boldsymbol{h}$**:** The filter constructed. The desired response is given by $\boldsymbol{v}$.

B**:** Matrix $B = \frac{1}{M} \sum_{i=1}^{M} \text{diag}\,(\widehat{\boldsymbol{x}}_i)^* \,\text{diag}\,(\widehat{\boldsymbol{x}}_i)$.

S**:** Matrix $S = \frac{1}{M} \sum_{i=1}^{M} \left(\text{diag}\,(\widehat{\boldsymbol{x}}_i) - \widehat{A}\right)^* \left(\text{diag}\,(\widehat{\boldsymbol{x}}_i) - \widehat{A}\right)$ where $\widehat{A}$ is the average of the diag $(\widehat{\boldsymbol{x}}_i)$'s.

SDF: Synthetic Discriminant Function. Initial filter with distortion invariant capabilities. Calculated as $\boldsymbol{h} = X \left(X^T X\right)^{-1} \boldsymbol{v}$.

MACE: Minimum Average Correlation Energy Filter. SDF-based filter which tries to provide an output surface with minimum energy. Calculated as $\widehat{\boldsymbol{h}} = B^{-1} \widehat{X} \left(\widehat{X}^H B^{-1} \widehat{X}\right)^{-1} D\boldsymbol{v}$.

MINACE: Minimum Noise and Correlation Energy Filter. SDF-based filter that minimises correlation energy and provides invariance to some known additional noise with covariance matrix C. Calculated as $\widehat{\boldsymbol{h}} = T^{-1} \widehat{X} \left(\widehat{X}^H T^{-1} \widehat{X}\right)^{-1} D\boldsymbol{v}$ where $T = B - bC$ and b is a parameter chosen when designing the filter.

MVSDF: Minimum Variance SDF. Simple extension to the SDF that adds invariance to some known noise with covariance matrix C. Calculated as $\boldsymbol{h} = C^{-1}X\left(X^TC^{-1}X\right)^{-1}\boldsymbol{v}$.

MSE SDF: Minimum Squared Error SDF. SDF-based filter that constrains the output plane to be as similar as possible to some specified function ($s(x,y)$). Calculated as $\widehat{\boldsymbol{h}} = B^{-1}\widehat{\boldsymbol{p}}+B^{-1}\widehat{X}\left(\widehat{X}^HB^{-1}\widehat{X}\right)^{-1}\left(D\boldsymbol{v}-\widehat{X}^HB^{-1}\widehat{\boldsymbol{p}}\right)$ where $\widehat{\boldsymbol{p}} = \frac{1}{M}\sum_{i=1}^{M}\mathrm{diag}\left(\widehat{\boldsymbol{x}}_i\right)\widehat{\boldsymbol{s}}$.

Optimal MSE SDF: Optimal MSE SDF. Variation of the MSE SDF that chooses a surface so that the actual squared error for the surface is minimised. Calculated as $\widehat{\boldsymbol{h}} = S^{-1}\widehat{X}\left(\widehat{X}^HS^{-1}\widehat{X}\right)D\boldsymbol{v}$.

ASM: Average Similarity Measure. Measure used to determine how similar output correlation surface are to the correlation form the average surface. Design criteria for the MACH filters. Defined as ASM $= \widehat{\boldsymbol{h}}^HS\widehat{\boldsymbol{h}}$.

MACH: Maximum Average Correlation Height Filter. Filter designed to minimise the ASM while at the same time having a high correlation peak with the mean filter. From the initial definition, it is only determined to a constant, so usually an additional constraint is required. (i.e. $\left\|\widehat{\boldsymbol{h}}\right\|_2 = 1$). Calculated as $\widehat{\boldsymbol{h}} = \frac{S^{-1}\widehat{\boldsymbol{a}}}{\|S^{-1}\widehat{\boldsymbol{a}}\|_2}$ where $\widehat{\boldsymbol{a}}$ is the Fourier transform of the mean image.

c-function: See self-correlation function.

Self-correlation function: Function defined by the inner product between the image and a distorted variation of itself. Defined as $c(\lambda) = \boldsymbol{x}(\lambda)^H\boldsymbol{x}(0)$.

Linearly interpolating SDF: SDF-type filter $\boldsymbol{h}$ that satisfies the following: Given $\boldsymbol{h}^T\boldsymbol{x}(\lambda_0) = \beta_0$, $\boldsymbol{h}^T\boldsymbol{x}(\lambda_1) = \beta_1$ and $\lambda = \alpha\lambda_0 + (1-\alpha)\lambda_1$ then $\boldsymbol{h}^T\boldsymbol{x}(\lambda) = \alpha\beta_0 + (1-\alpha)\beta_1$ for any $0 \leq \alpha \leq 1$. Filters with a piecewise linear self-correlation function are linear interpolating.

SSD: Sum of Squared Distances. Measure of cluster size common in automatic clustering applications. SSD $= \sum_{i=1}^{m}\|\boldsymbol{x}_i - \overline{\boldsymbol{x}}\|^2$ where $\overline{\boldsymbol{x}}$ is the mean of the $\boldsymbol{x}_i$'s and m is the number of elements in the cluster (see Chapter 8).

Bibliography

[1] P. N. Belhumeur, J. P. Hespanha and D. J. Kriegman. Eigenfaces vs. Fisherfaces: Recognition using class specific linear projection. In *1996 European Conference on Computer Vision*, 1996.

[2] J. D. Brasher, C. F. Hester, D. W. Lawson and R. Sims. Multi-stage, higher-order sdf filters. *SPIE*, 1297:103–109, 1990.

[3] J.D. Brink, C. Nieuwoudt and E.C. Botha. Facial image compression using the Karhunen-Loève transform. In *Proceedings of the Ninth Annual South African Workshop on Pattern Recognition*, 1998.

[4] David Casasent. Gabor wavelets and fusion for distortion-invariant multi-class object detection. In *Proc. SPIE*, volume 2491, 1995.

[5] David Casasent, Voon-Yim Choo and Geogery House. Advanced and orthogonal MINACE filter sets: Initial detection results. In *Proc. SPIE*, volume 2237, May 1994.

[6] David Casasent, John-Scott Smokelin and Anqi Ye. Wavelet and Gabor transforms for detection. *Optical Engineering*, 31(9):1893–1898, September 1992.

[7] T. M. Cover and J. A. Thomas. *Elements of Information Theory.* John Wiley & Sons, New York, 1994.

[8] Carl de Boor. Good approximation by splines with variable knots II. In *Conference on the Numerical Solution of Differential Equations*, number 363 in Lecture Notes in Mathematics, pages 12–20. Springer-Verlag, 1973.

[9] Carl de Boor. *A Pratical Guide to Splines*, pages 44–47,180–196. Number 27 in Applied Mathematical Sciences. Springer-Verlag, 1978.

[10] J. W. Demmel. *Applied Numerical Linear Algebra.* SIAM, Philadelphia, 1997.

[11] W. Equitz. A new vector quantization clustering algorithm. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 37(10), October 1989.

[12] K. Fukunaga. *Statistical Pattern Recognition - 2nd Edition.* John Hopkins University Press, Baltimore, 1989.

[13] A. Gersho and R. Gray. *Vector Quantization and Signal Compression.* Kluwer Academic Publishers, Boston, 1992.

[14] G. H. Golub and C. F. Van Loan. *Matrix Computation - 2nd Edition.* Academic Press, Boston, 1990.

[15] Peter W. Hallinan, Gaile G. Gordon, A. L. Yuille, Peter Giblin and David Mumford. *Two- and Three-Dimensional Face Patterns of the Face.* A K Peters, Ltd, Natick, Massachusetts, 1999.

[16] C. F. Hester and D. Casasent. Multivariant techniques for multiclass pattern recognition. *Applied Optics*, 19:1758–1761, 1980.

[17] IEEE. *14th International Conference on Pattern Recognition*, 1998.

[18] IEEE. *15th International Conference on Pattern Recognition*, 2000.

[19] I. T. Jolliffe. *Principal Component Analysis.* Springer-Verlag, New York, 1986.

[20] B. V. K. Vijaya Kumar. Minimum variance synthetic discriminant functions. *JOSA*, A 3:1579–1584, 1986.

[21] B. V. K. Vijaya Kumar. Tutorial survey of composite filters designs for optical correlators. *Applied Optics*, 31:4773–4801, 1992.

[22] B. V. K. Vijaya Kumar, J. D. Brasher, C. F. Hester, G. Srinivasan and S. Bollapragada. Role of the constraint values in synthetic discriminant function filter design. *SPIE*, 1959:23–31, 1993.

[23] B. V. K. Vijaya Kumar and A. Mahalanobis. Recent advances in distortion-invariant correlation filter design. In *Proc. SPIE*, volume 2490, 1995.

[24] B. V. K. Vijaya Kumar, A. Mahalanobis, Sewoong Song, S. R. F. Sims and J. F. Epperson. Minimum squared error synthetic discriminant functions. *Optical Engineering*, 31(5):915–922, May 1992.

[25] A. Mahalanobis, B. V. K. Vijaya Kumar and D. Casasent. Minimum average correlation energy filters. *Applied Optics*, 26:3633 – 3640, 1987.

[26] A. Mahanobis, B. V. K. Vijaya Kumar, S. Song, S. R. F. Sims and J. F. Epperson. Unconstrained correlation filters. *Applied Optics*, 33:3751–3759, 1994.

[27] Daniel McNeill. *The Face.* Penguin Books, London, England, 1998.

[28] B. Moghaddam and A. Pentland. An automatic system for model-based coding of faces. In *IEEE Data Compression Conference, Snowbird, Utah*, 1995.

[29] B. Moghaddam and A. Pentland. Probablistic visual learning for object detection. In *The 5th International Conference on Computer Vision, Cambridge, MA.*, 1995.

[30] B. Moghaddam and A. Pentland. A subspace method for maximum likelihood target detection. In *IEEE International Conference on Image Processing, Washington, DC.*, 1995.

[31] N. Muller and B. Herbst. Building a representative training set based on eigenimages. In *14th International Conference on Pattern Recognition* [17], pages 1846–1848.

[32] N. Muller and B. Herbst. The use of eigenpictures for optical character recognition. In *14th International Conference on Pattern Recognition* [17], pages 1124–1126.

[33] N. Muller and B. Herbst. On the use of SDF-type filters for distortion invariant image location. In *15th International Conference on Pattern Recognition* [18], pages 530–533.

[34] Neil Muller. Image recognition using the eigenpicture technique. Master's thesis, University of Cape Town, 1997.

[35] L. Neiburg. *Feature Space Trajectory Pattern Classifier.* PhD thesis, Carnegie Mellon University, 1996.

[36] A. J. O'Toole, H. Abdi, K. Deffenbacher and Valentin D. Low-dimensional representation of faces in higher dimensions of the face space. *J. Opt. Soc. Am. A.*, 10(3):405–410, 1993.

[37] A. Pentland, B. Moghaddam and T. Starner. View based and modular eigenspaces for face recognition. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 1994.

[38] A. Pentland, T. Starner, N. Etcoff, A. Masoiu, O. Oliyide and M. Turk. Experiments with eigenfaces. Technical Report Perceptual Computing Group Technical Note No. 194, MIT Media Laboratory, 1992.

[39] V. Pereya and E. G. Sewell. Mesh selection for discrete solution of boundary problems in ordinary differential equations. *Numerical Mathematics*, 23:261–268, 1975.

[40] J. Phillips, H. Moon, P. Rauss and S. Rizvi. The FERET September 1996 database and evaluation procedure. In *Proceedings of First International Conference and Video-based Biometric Person Authentication*, 1997.

[41] D. W. Purnell, C. Nieuwoudt and E. C. Botha. Face recognition using the Karhunen-Loéve transform. In *Proceedings of the Seventh Annual South African Workshop on Pattern Recognition*, pages 7–11, 1996.

[42] Gopalan Ravichandran and David Casasent. Minimum noise and correlation energy optical correlation filter. *Applied Optics*, 31(11), April 1992.

[43] John A. Rice. *Mathematical Statistics and Data Analysis (2nd Edition)*. Duxbury Press, Belmont, California, 1995.

[44] Joseph Rosen and Joseph Shamir. Scale invariant pattern recognition with logarithmic radial harmonic filters. *Applied Optics*, 28(2):240–244, January 1989.

[45] Thomas Ruggles. Comparison of biometric techniques, 1996. URL http://biometric-consulting.com/bio.htm.

[46] Micheal A. Sipe. *Feature Space Trajectory Methods for Active Object Recognition*. PhD thesis, Carnegie Mellon University, 1999.

[47] L. Sirovich and M. Kirby. Low-dimensional procedure for the characterization of human faces. *J. Opt. Soc. Am. A.*, 4(3):519–524, 1987.

[48] H. Späth. *Cluster Analysis Algorithms for Data Reduction and Classification of Objects.* Ellis Horwood, 1980.

[49] G. W. Stewart and Ji-guang Sun. *Matrix Perturbation Theory.* Academic Press, Boston, 1990.

[50] G.W. Stewart. Error and perturbation bounds for subspaces associated with certain eigenvalue problems. *SIAM Review*, 15:727–764, 1973.

[51] L. N. Trefethen and D. Bau III. *Numerical Linear Algebra.* Society for Industrial and Applied Mathematics, Philadelphia, 1997.

[52] M. Turk and A. Pentland. Representing faces for recognition. Technical Report No. 132, Vision and Modelling Group, MIT, 1990.

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

[54] J. H. Wilkinson. *The Algebraic Eigenvalue Problem.* Clarendon Press, Oxford, 1965.

[55] Vision and Modeling Group of the MIT Media Laboratory. URL: http://vismod.www.media.mit.edu/vismod/.

[56] Face database compiled by the Media Laboratory, MIT. URL: ftp://whitechapel.media.mit.edu/pub/images/.

[57] ORL database of faces. AT&T Laboratories, Cambridge, UK. URL: http://www.cam-orl.co.uk/facedatabase.html.

[58] POVRAY Home Page. URL: http://www.povray.org/.

[59] Alper Yilmaz and Muhittin Gökmen. Eigenhill vs. Eigenface and Eigenedge. In *15th International Conference on Pattern Recognition* [18], pages 831–834.