Studies in Optical Super-Resolution

Alistair Sutherland

Doctor of Philosophy

University of Edinburgh

1989

Contents

Abstract

This thesis is about the application of the singular value decomposition to bandlimited optical images in order to increase the resolution. Previous workers have considered only one-dimensional systems or two-dimensional systems which are separable. This thesis extends the technique to the non-separable case. An algorithm is presented for the computation of the singular functions of an optical system where object and pupil differ in shape. Numerical examples are given of both the eigenvalue decomposition and the singular value decomposition applied to images in the presence of noise. The singular value decomposition is shown to be superior. Finally the singular value decomposition is applied to real optical images generated on an optical bench using incoherent illumination.

Chapter One

Introduction to Super-resolution

Section 1.1 Introduction

This thesis is about one particular technique of achieving "super-resolution". Super-resolution is an attempt to improve the resolving power of optical systems (e.g. telescopes, microscopes, cameras etc) by using digital processing.

In this introductory chapter, we shall set super-resolution in its context. We shall see first of all that it is a typical member of a set of problems known as "inverse problems" and then go on to trace the historical development of ideas about super-resolution up to the present day.

Section 1.2 General characteristics of super-resolution

1.2.1 Definition of resolution

Consider the following schematic drawing of a generalised imaging system. See fig. 1.1.

The optical system is looking at an "object" - this could be a specimen sitting on a glass slide in the case of a microscope, or a cluster of stars in the case of an astronomical telescope - and it is forming an "image" of it. In this thesis, the "object" and "image" will be described by two-dimensional functions which represent either the amplitude or intensity distribution of the wave-front across planes close to the object and image respectively.

In general, the image will be similar but not identical to the object. Large scale features of the object will tend to be reproduced better than small-scale features. In fact there are usually features of the object which are so small that they do not appear in the image at all. "Resolution" has traditionally been defined as a measure of the size of the smallest feature of the object which appears in the image. For example, the famous Rayleigh criterion - (which is just one of several measures of resolution) - defines the size of the smallest feature as being the distance from the

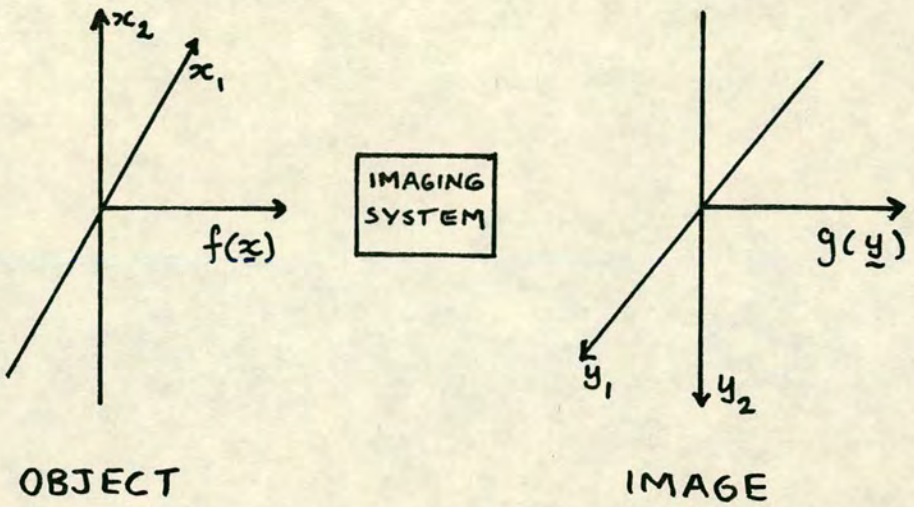


FIG. 1.1

maximum of an image of a point source to its first minimum. However, the Rayleigh criterion was invented before the time of image detectors which could yield accurate measurements of the 2D intensity variation across the image - (the Rayleigh criterion is based on the light-detecting ability of the human eye) - and before the time of digital computers which could process this data. Nowadays, we need not regard the Rayleigh criterion as being fundamentally unbeatable. Therefore, we can now introduce the idea of "super-resolution" - by which we mean the attempt to increase the resolution of images beyond the Rayleigh limit by use of digital processing.

1.2.2 Inverse problems and *a priori* information

An alternative way to view super-resolution is to consider it as an attempt to "reconstruct the object from the image". In other words, we try to put back the features that were lost in the transition from object to image and so find the original object. The problem is that there may be many different possible objects which could

give rise to the same (or a very similar image). We must decide which out of this set, is the original object. Each of the possible objects is different from the others but the differences are due to only the small-scale features - the large-scale features being the same from object to object. Since the small scale features are lost in the transition, the images all end up looking identical. It is therefore impossible to decide which of the possible objects is the true one.

However, we can improve our chances of estimating the correct object, if we have some *a priori* knowledge about the object. This knowledge can be of a very simple kind eg. we may know that the object has a certain degree of smoothness, or that it is non-negative (eg. light intensity distributions are always non-negative.) We can then discard from the set of possible objects, any object which does not comply with our a priori information. Quite often this can greatly reduce the solution set.

However, even with *a priori* information, we may still be left with a class of possible objects which give rise to images which are not identical but are very similar. A consequence of this is that a small error in our measurement of the image will result in a large error in our estimate of the object. Therefore, we must try to make our measurements as accurately as possible.

These characteristics - the large set of possible solutions, the amplification of small errors and the need for *a priori* information - are typical of a set of problems known as "inverse problems" - which are common in many fields besides optics. Other examples include, the recovery of a three-dimensional object from its two-dimensional projection (which is a problem solved by the brain from data provided by our eyes) or the estimation of the interior structure of the Earth from seismic data.

We shall try to quantify these characteristics more precisely later in the Chapter.

Section 1.3 Fourier theory of imaging

1.3.1 The imaging equation

Our first step in making our ideas about super-resolution more precise, is to provide a quantitative relationship between the image and the object. This relationship

is given by the Fourier theory of imaging. This theory is very well known and is well-described in many texts, especially Goodman [31], but we shall summarise the main elements here, in order to establish the terms we shall use later in the thesis.

Firstly we must say a little more about the functions, $f(\mathbf{x})$ and $g(\mathbf{y})$ which were introduced in fig. 1.1. These are two-dimensional functions of the vectors $\mathbf{x} = (x_1, x_2)$ and $\mathbf{y} = (y_1, y_2)$. $\mathbf{x}$ and $\mathbf{y}$ represent displacement in the object and image planes respectively. They are both measured in the same arbitrary units which I shall call "space units". In the case of coherent illumination (e.g. in microscopy), $f(\mathbf{x})$ and $g(\mathbf{y})$ could represent amplitude distributions and could therefore both be complex. In the case of incoherent illumination (e.g. astronomy), they could represent intensity distributions and would therefore both be real. For simplicity, we shall restrict our attention to either fully coherent or fully incoherent illumination although the theory can be extended to partially coherent systems. See Frieden [10]

According to Fourier theory the object, $f(\mathbf{x})$, and the image, $g(\mathbf{y})$, can be related by the following equation.

$$g(\mathbf{y}) \quad = \quad \int_{\text{all space}} f(\mathbf{x}) h(\mathbf{y} - \mathbf{x})\, d\mathbf{x} \qquad 1.3.1$$

where $h(\mathbf{y})$ is known as the *Point Spread Function* and is defined to be the image of a delta-function object. This equation is very general and describes most optical systems - coherent, incoherent, partially coherent - including some systems with aberrations. The only condition is that the system should be linear and space-invariant.

The integral on the RHS is known as a convolution integral and is sometimes written as

$$g \quad = \quad f \star h \qquad 1.3.2$$

Fourier theory allows us to express 1.3.2 in a way which we shall later find to be more convenient. In order to do this we must first define an integral operator known as the Fourier transform operator. The Fourier transform operator and its inverse are

defined as follows

$$\begin{aligned} F(\mathbf{k}) &= \int_{\text{all space}} f(\mathbf{x}) \exp(i\mathbf{k}\cdot\mathbf{x})\, d\mathbf{x} \\ f(\mathbf{x}) &= \frac{1}{(2\pi)^2} \int_{\text{all space}} F(\mathbf{k}) \exp(-i\mathbf{x}\cdot\mathbf{k})\, d\mathbf{k} \end{aligned} \qquad 1.3.3$$

The function, $F(\mathbf{k})$, is known as the Fourier transform of $f(\mathbf{x})$. The Fourier transform contains exactly the same information as the original function. The information is merely displayed in a different way. The Fourier transform is a two-dimensional function of the vector $\mathbf{k} = (k_1, k_2)$. $\mathbf{k}$ is known as the spatial frequency and is measured in units which are the reciprocal of the space units used to measure $\mathbf{x}$ and $\mathbf{y}$. I shall call these units frequency units. The value of the Fourier transform at $\mathbf{k}$ is the complex amplitude of the Fourier component of $f(\mathbf{x})$ with spatial frequency $\mathbf{k}$.

Now, according to the convolution theorem we have the following relationship.

$$G(\mathbf{k}) = F(\mathbf{k})H(\mathbf{k}) \qquad 1.3.4$$

where G(k), F(k) and H(k) represent the Fourier transforms of g(y), f(x) and h(y) as defined by 1.3.3. Thus we can see that the relationship between the Fourier transforms is much simpler than that between the original functions. This simplicity allows us a much greater insight into the imaging process. In fact, it allows us to break the imaging process down into three stages.

1) Fourier transform the object, f(x), to obtain F(k).

2) Multiply F(k) by H(k) to obtain G(k).

3) Inverse Fourier transform G(k) to obtain g(y).

We can now re-write equn. 1.3.1 as follows

$$g(\mathbf{y}) = \frac{1}{(2\pi)^2} \int_{\text{all space}} \exp(-i\mathbf{y}\cdot\mathbf{k})\, H(\mathbf{k}) \int_{\text{all space}} f(\mathbf{x}) \exp(i\mathbf{k}\cdot\mathbf{x})\, d\mathbf{x}\, d\mathbf{k} \qquad 1.3.5$$

I shall call equn. 1.3.5 the imaging equation and the operator defined on the RHS the imaging operator. I shall refer back to this equation many times throughout this

thesis. In chapter 2 I shall base a mathematical model of an imaging system on 1.3.5 and then implement it on a computer by representing each of the three stages as a subroutine. In chapter 4 I shall base an experimental model of an imaging system on 1.3.5 in which each of the three stages is represented as a component on an optical bench.

The imaging equation is very general and should describe any linear space-invariant imaging system. We can therefore hope that either of the models mentioned in the previous paragraph should be equally general and that the results derived from them should be typical of any linear space-invariant optical system.

1.3.2 The transfer function

Please notice, from equation 1.3.4 that all the information about the optical system is contained in the function, H(k). This function is called the "Transfer Function" and is, in general, complex. Transfer functions vary greatly from one system to another but they all have one characteristic in common: they are uniformly zero outside same closed boundary. This boundary is determined by the physical dimensions of the system e.g. in a microscope the boundary will be circular and the diameter will depend on the diameter of the smallest lens in the system.

The region within this closed boundary is known as the "pass-band" or sometimes the "pupil". In a one-dimensional system the pass-band becomes an interval, and the width of this interval is called the "band-width". A function whose Fourier transform is zero outside some closed boundary is known as a "band-limited" function. Therefore the image, g(y), is a band-limited function.

We can think of the Transfer Function as being like a "mask" which is placed over the Fourier plane. Those frequencies within the mask are transmitted (albeit with some modification) those outside are not transmitted at all. The information carried by those frequencies would therefore appear to be lost. But it is precisely these outer frequencies which carry the information about the small-scale features of the object. Therefore, the loss of these frequencies leads to the loss of the small-scale

information: in other words, it leads to the finite resolution of the system. Therefore, the resolution limit of the system depends on the cut-off boundary in the Fourier plane.

The last statement implies that if we are to achieve super-resolution we must somehow recover some of the lost frequencies outside the cut-off boundary. In other words, we must extrapolate the Fourier transform outside the cut-off boundary. For this reason, super-resolution is sometimes known as "band-width extrapolation".

One might at first think that if H(k) differed from unity within the cut-off boundary then this too would have an effect on the resolution. However, in principle, this effect can be eliminated if we know H(k). We could proceed as follows

1) We Fourier transform the image, g(y), to obtain G(k).

2) We calculate F(k) within the cut-off boundary by

$$F(\mathbf{k}) \quad = \quad \frac{G(\mathbf{k})}{H(\mathbf{k})} \qquad 1.3.6$$

Thus, in principle, we could always recover F(k) within the cut-off boundary provided we know H(k) (and it is usually possible to measure H(k) for most optical systems). This process is known as "deconvolution". However, in practice, this process can often be quite difficult to carry out especially when H(k) falls to small values, because then division by H(k) will be unstable and any noise in the data will be amplified. We shall meet this problem in chapter 4. But in the rest of this thesis, except where otherwise stated, I shall consider only those systems where H(k) is sufficiently large that this problem does not occur, i.e. I shall assume that the Fourier transform is accurately known within the pass-band.

1.3.3 Analytic continuation

We saw in the last section that super-resolution was equivalent to retrieving some of the lost frequencies outside the cut-off boundary. At first sight, this would seem to be impossible. As we saw in the section on inverse problems, there must be a large set of possible objects which could give rise to the same image. The Fourier

transform of each of these objects would be the same within the pass-band but could differ widely outside it. We are left with the problem of selecting which of this set is the true object. In most cases this is impossible.

But, as we saw in the section on inverse problems, if we have some *a priori* information we can reduce the class of possible objects. There are many different types of *a priori* information which we could use but for most of this thesis I shall concentrate on only one type:- that we know that the object is "space-limited". A space-limited function is a function which is zero outside some closed boundary. Thus, for example, the transfer function, H(k), is a space-limited function because it is zero outside the pass-band. The idea of using this type of *a priori* information was first suggested in 1955 by Toraldo di Francia [15] who pointed out that the Fourier transform of a space-limited function is "analytic". An analytic function is defined in complex variable theory as a function which obeys the Cauchy-Riemann relations. It can be shown that all analytic functions have the following property: if an analytic function is known over a finite, bounded area then it is determinable everywhere. The process of calculating the function outside the known area is called "analytic continuation".

Toraldo argued that if we know in advance that the object is space-limited then its Fourier transform, F(k), must be analytic and so, even though we only know that part of F(k) contained within the pass-band, we should be able to calculate the rest of F(k) using the analytic continuation property. We should thus be able to reconstruct the object, f(x), with, in principle, infinite resolution.

This whole thesis is based on Toraldo's theorem. Henceforward I shall assume that we have the *a priori* information that the object is space-limited and we know the region within which it lies.

1.3.4 The band-limited image of a space-limited object

For the rest of this thesis I shall consider only objects which are space-limited and their band-limited images. We can now modify the imaging equation (1.3.5) to

incorporate the information that both the object and the transfer function are space-limited functions.

$$g(\mathbf{y}) \quad = \quad \frac{1}{(2\pi)^2} \int_{\Omega} \exp(-i\mathbf{y}\cdot\mathbf{k}) \int_{X} f(\mathbf{x}) \exp(i\mathbf{k}\cdot\mathbf{x})\, d\mathbf{x}\, d\mathbf{k} \qquad 1.3.7$$

I have replaced the limits on the two integrals. In equn. 1.3.5 the region of integration was all space, but in equn.1.3.7. the regions of integration are the finite bounded regions X and Ω. X is the region within which we know the object to lie- this constitutes our *a priori* information. Ω is the boundary of the pass-band of the optical system. The two Fourier integrals in equn. 1.3.7. are known as "finite Fourier transforms" because their regions of integration are finite rather than infinite.

Throughout most of this chapter we shall consider one-dimensional imaging systems, because that is what most previous literature has been concerned with up till now. In 1D equn. 1.3.7. reduces to

$$g(y) \quad = \quad \frac{1}{(2\pi)} \int_{-\Omega}^{\Omega} \exp(-iky) \int_{-x_0}^{x_0} f(x) \exp(ikx)\, dx\, dk \qquad 1.3.8$$

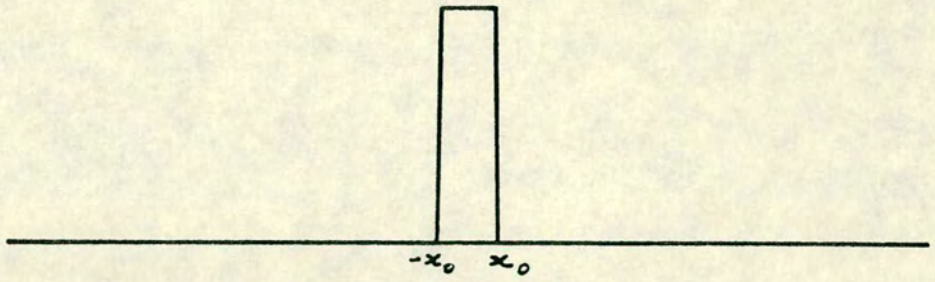
where the object lies in the interval $[-x_0, x_0]$ and the pass-band is the interval $[-\Omega, \Omega]$.

In this section we shall look at some of the general characteristics of images generated by this equation. The key parameter of such an image is a dimensionless number called the "space-bandwidth product" and which is given the symbol, "c". It is defined by

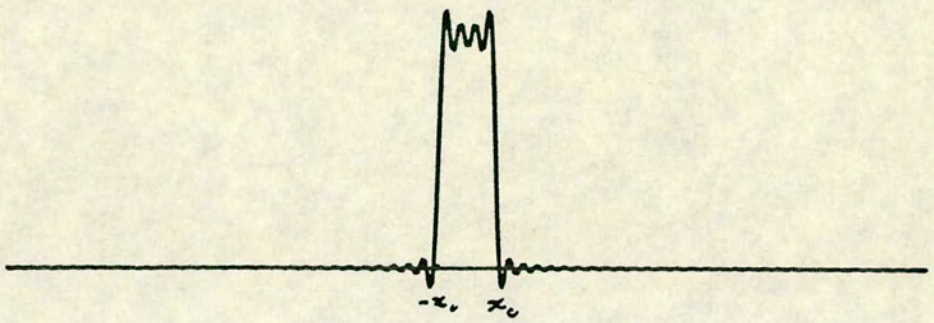
$$c \quad = \quad x_0\Omega \qquad 1.3.9$$

Fig. 1.2 shows three such images at different values of c.

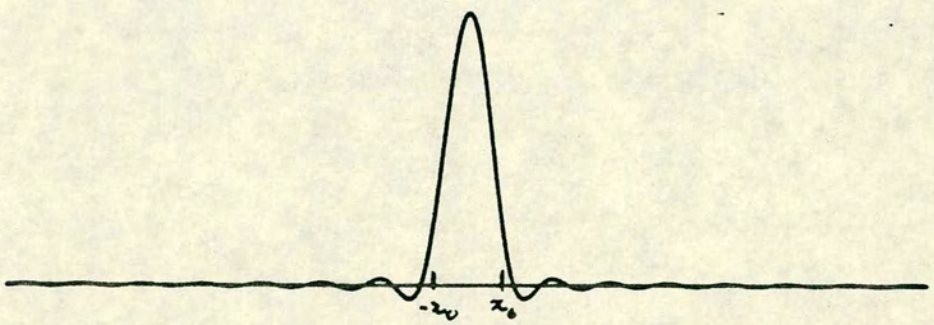
In the top-most image $c = \infty$, in other words the image is not band-limited at all and so is identical to the original object. We could call this the "geometrical image" since it is the image which would be produced according to the rules of geometrical optics. In this case the object is a top-hat function.



geometrical image



$C = 4\pi$



$C = \pi$

FIG. 1.2

The bandlimited image of a space-limited object

The second image is of the same object but this time $c = 4\pi$. The degradation due to the finite bandwidth is obvious, but please also note that the image now extends over all space. Most of the energy is still concentrated around the geometrical image region but a significant proportion is scattered over the rest of image space.

The third image has $c = \pi$. The image is further degraded and more of it is spread outside the region of the geometrical image.

The main consequence of the finite band-width is that the information about the object is pushed out of the the geometrical image region and spread over the whole image plane. If we are to restore the object as much as possible we must make use of all of this information, rather than just that contained within the geometrical image. The significance of this fact will become clear when we come to compare the eigenvalue decomposition with the singular value decomposition in section 1.5.

Section 1.4 Prolate spheroidal functions

1.4.1 Summary

Although Toraldo had proved that band-width extrapolation was possible he had not shown exactly how it could be carried out. The first indication as to how this might be done appeared in 1961 when Slepian and Pollak published a paper on the Fourier transform properties of "prolate spheroidal wave functions". These functions had originally been discovered as the solutions to the Wave Equation in prolate spheroidal co-ordinates. A comprehensive treatment of them was published by Flammer [18] in 1957 which included methods of computation. But, it was not until Slepian, Landau and Pollak began to study them that their peculiar Fourier transform properties were discovered. They are all stated and proved by Slepian and Pollak [5] and also by Frieden [11] and I shall summarise them here.

1.4.2 Definition

The functions $\psi_n(x)$ are defined by:

$$\int_{-x_0}^{x_0} \psi_n(x) \exp(ikx)\, dx \quad = \quad i^n (2\pi\lambda_n x_0/\Omega)^{\frac{1}{2}} \psi_n(kx_0/\Omega) \qquad 1.4.1$$

This states that the functions are invariant to a finite Fourier transform.

In other words, by taking the part of $\psi_n(x)$ contained within the finite interval $[-x_0, x_0]$ and Fourier transforming it, we obtain the function over all space.

The numbers λ_n are the "linear prolate eigenvalues" .

1.4.3 Eigenfunctions of the 1D imaging operator

If we now apply a finite inverse Fourier transform to both sides of equation 1.4.1 we obtain

$$\frac{1}{2\pi}\int_{-\Omega}^{\Omega}\exp(-iky)\,dk\int_{-x_0}^{x_0}\psi_n(x)\exp(ikx)\,dx \quad = \lambda_n\psi_n(y) \qquad 1.4.2$$

The operator on the LHS of equn. 1.4.2. is the 1D imaging operator which we met in equn. 1.3.8. Thus, equation 1.4.2 describes a 1D imaging system with bandwidth 2Ω which is looking at an object of width $2x_0$. The functions, $\psi_n(x)$, can be regarded as eigen-images of this system in the sense that they pass through the system unaltered apart from the scaling factor, λ_n.

The space-bandwidth product, c, of this system is as defined in equn.1.3.9. The $\psi_n(x)$ and λ_n are functions of c as well as of x. So they should really be written as $\psi_n(x,c)$ and $\lambda_n(c)$ but I have suppressed this for simplicity. But the reader should bear in mind throughout this thesis that the $\psi_n(x)$ and λ_n depend on c, the space-bandwidth product.

1.4.4 Dual orthogonality

The most striking property of the $\psi_n(x)$ is that they are orthogonal over two different regions. The ψ_n are orthogonal over both the interval $[-x_0, x_0]$ and the interval $(-\infty, \infty)$, i.e. both the following relationships hold

$$\int_{-x_0}^{x_0}\psi_m(x)\,\psi_n(x)\,dx \quad = \quad \lambda_n\delta_{mn} \qquad 1.4.3$$

$$\int_{-\infty}^{\infty}\psi_m(x)\,\psi_n(x)\,dx \quad = \quad \delta_{mn} \qquad 1.4.4$$

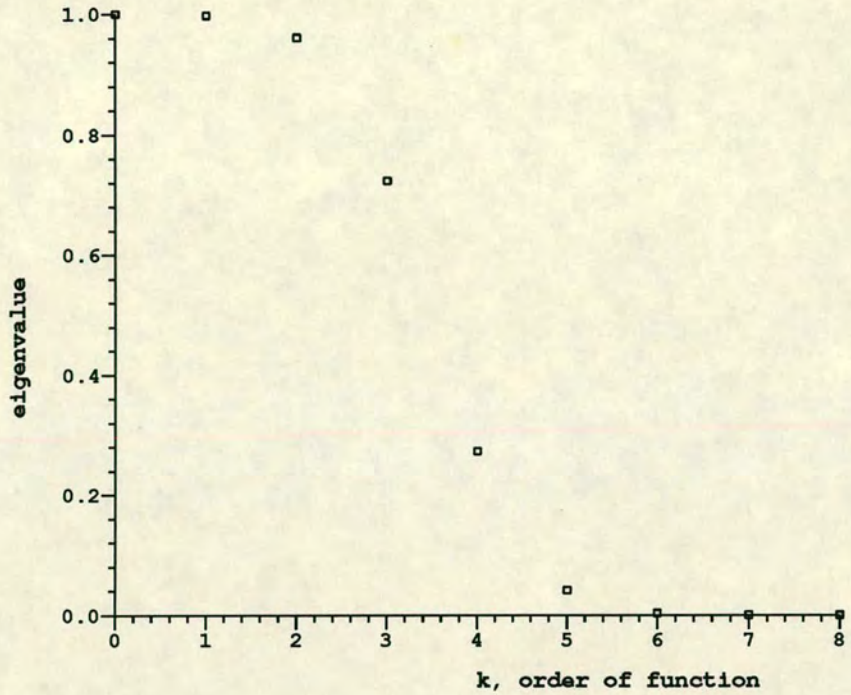


FIG. 1.3

The eigenvalue spectrum for $c = 2\pi$

N.B. the $\psi_n(x)$ form a complete set over all functions on $[-x_0, x_0]$. They also form a complete set over those functions on $(-\infty, \infty)$ whose Fourier transforms are contained in the region $[-\Omega, \Omega]$.

1.4.5 Symmetry properties

Due to symmetry properties of equn.1.4.2 the $\psi_n(x)$ must have either odd or even symmetry about the origin. In fact, successive eigenfunctions have alternate symmetry.

1.4.6 The eigenvalue spectrum

Fig. 1.3 shows a graph of λ_k against k, for $c = 2\pi$.

The eigenvalues for all values of c have the following properties

a)they are all < 1

b)they are all > 0

c)low-order eigenvalues are almost equal to 1 but the value falls drastically after

$k = 2c/\pi$

d)above $k = 2c/\pi$ lambda tends asymptotically to zero.

These properties can all be seen fig. 1.3.

1.4.7 Bandwidth extrapolation properties

Slepian and Pollak suggested that the PSFs could be used for band-width extrapolation. This was developed by the following authors: Harris[9], Barnes[8], Frieden[10], Pask[14], McCutchen[13], Rushforth and Harris[12], Saleh[19]and Bershad[35]. All these writers give similar accounts but I shall not repeat them here because they are all subsumed by the account given by Bertero and Pike [1] which I shall repeat in detail in section 1.5. I shall make a few general points here.

The problem encountered by all of these authors is that the resolution can only be increased at the expense of an increase in noise in the reconstructed object. We saw that this was a typical characteristic of inverse problems in section 1.2.

All of these authors take only a theoretical approach; they sketch out a possible method of carrying out super-resolution but they do not actually go ahead and do it. Frieden [10] gives a simulation but this is only of an object which he has chosen carefully so it can be treated analytically. None of the authors discuss the difficulties of putting super-resolution into practice, eg sampling and truncation. They only address 1D imaging systems or 2D systems which can be separated into two 1D systems.

Section 1.5 The eigenvalue decomposition and singular value decomposition

1.5.1 Summary

The fullest and most general account of super-resolution using prolate spheroidal functions is that given by Bertero and Pike. I give their argument in detail because it subsumes the arguments of all the previous authors. Whereas most of the previous treatments were restricted to the one-dimensional, continuous case Bertero and Pike's includes that case and the multi-dimensional discrete case as well. The argument

of chapter 2 of this thesis depends on this generalisation to the multi-dimensional, discrete case.

Bertero and Pike achieve this generality by introducing a concept known as a "singular value decomposition" (SVD). This concept underlies the treatments given by previous authors but was not explicitly stated. SVD is an extension of the concept of an eigenvalue decomposition (EVD). Bertero and Pike begin their first paper [1] by showing that an EVD can achieve super-resolution and then going on to extend that idea to a SVD. Finally they compare the results of the two decompositions and show that SVD is significantly better in the presence of noise.

One of the themes of this thesis will also be a comparison between SVD and EVD. I shall give numerical examples of the two decompositions in chapter 3.

I shall recount Bertero and Pike's argument here. I shall present it as they presented it in their first paper making reference to only the 1D continuous, coherent case but as we shall see in chapter 2 it can be easily generalised to other cases.

1.5.2 Eigenvalue decomposition

The essence of the eigenvalue decomposition is to represent the imaging process by an operator and then find its eigenfunctions and values. We can then use these to invert the operator.

We have already met the 1D imaging operator in equn. 1.3.8. Therefore let us define an operator, A, as follows

$$g(y) \quad = (Af)(y) \quad = \quad \frac{1}{2\pi} \int_{-\Omega}^{\Omega} \exp(-iky)\, dk \int_{-x_0}^{x_0} f(x) \exp(ikx)\, dx \qquad 1.5.1$$

where f(x) is the object and g(y) is the image. This is very similar to the operator defined in 1.3.8. The only difference is that we have restricted g(y) to lie in the same interval as f(x), $[-x_0, x_0]$. In other words we are only considering the geometrical image region and ignoring the rest of the image which is scattered outside it. The reason for this is that the concept of an eigenfunction requires that the operator maps the same set of functions onto itself. In this case that set is $L^2[-x_0, x_0]$, the set of

all complex valued functions on the interval $[-x_0, -x_0]$. We require both f(x) and g(y) to belong to this set.

We can write equation 1.5.1 as follows

$$\mathbf{Af} \quad = \quad \mathbf{g} \qquad 1.5.2$$

Now, we wish to find the eigenfunctions of A, defined by

$$\mathbf{Au}_k \quad = \quad \lambda_k \mathbf{u}_k \qquad 1.5.3$$

This is clearly very similar to equation 1.4.2. So the $u_k(x)$ must be closely related to the prolate spheroidal functions $\psi_n(x)$. In fact,

$$u_k(x) \quad = \quad \lambda_k^{\frac{1}{2}} \psi_k(x) \qquad 1.5.4$$

where λ_k are the prolate spheroidal eigenvalues which we met in equn. 1.4.1. The scaling factor arises because $\psi_n(x)$ are normalised over $(-\infty, \infty)$ but $u_k(x)$ are normalised over $[-x_0, x_0]$.

Now, because A is self-adjoint its eigenfunctions are complete and orthogonal over $L^2[-x_0, x_0]$ and its eigenfunctions are real. Furthermore, because A is positive definite its eigenvalues are all strictly positive. Hence, we can represent both the object and image as linear combinations of the $u_k(x)$.

$$\mathbf{f} \quad = \quad \sum_{k=0}^{\infty} f_k \mathbf{u}_k \qquad \mathbf{g} \quad = \quad \sum_{k=0}^{\infty} g_k \mathbf{u}_k \qquad 1.5.5(a),(b)$$

where f_k and g_k are given by

$$f_k \quad = \quad \int_{-x_0}^{x_0} f(x) u_k(x)\, dx \qquad g_k \quad = \quad \int_{-x_0}^{x_0} g(y) u_k(y)\, dy \qquad 1.5.5(c),(d)$$

Now, let us apply operator A to f as expressed in 1.5.5

$$\begin{aligned} \mathbf{Af} \quad &= \quad \mathbf{A}\left(\sum_{k=0}^{\infty} f_k \mathbf{u}_k\right) \quad \text{using 1.5.5(a)} \\ &= \quad \sum_{k=0}^{\infty} f_k \mathbf{A}\mathbf{u}_k \\ &= \quad \sum_{k=0}^{\infty} f_k \lambda_k \mathbf{u}_k \quad \text{using 1.5.3} \end{aligned}$$

But

$$\begin{aligned} \mathbf{Af} &= \mathbf{g} \quad \text{from 1.5.2} \\ &= \sum_{k=0}^{\infty} g_k \mathbf{u}_k \quad \text{from 1.5.5(b)} \end{aligned}$$

Therefore, equating co-efficients gives us

$$f_k = \frac{g_k}{\lambda_k} \qquad 1.5.6$$

which is a relationship between the object co-efficients, f_k, and the image co-efficients, g_k. It is this relationship which allows us to reconstruct the object from the image, or, in other words, to bring about super-resolution. We can, in principle, measure all the image co-efficients g_k and then calculate the object co-efficients, f_k, from them using 1.5.6. Once we know all the f_k, our knowledge of the object is complete and so we have completely reconstructed the object from the image.

But, the problem is that 1.5.6. involves division by λ_k and as we saw in section 1.4.6. some of the λ_k are $<< 1$ and so the division is very unstable. Any error in our measurement of g_k will be amplified by the factor $1/\lambda_k$ which could be very large for higher order eigenfunctions. See fig. 1.3. So we must truncate the summation in equn. 1.5.5(a) at some value of k where the eigenvalue, λ_k is not too small. This value of k will depend on the signal-to-noise ratio in the image. We shall return to this topic in chapter 3.

1.5.3 Singular value decomposition

Bertero and Pike then asked themselves whether it was possible to do better in the presence of noise than the eigenvalue decomposition allowed. It occurred to them that they were not using all the information available to them in the image plane. As we saw EVD restricts its attention to the geometrical image region whereas the information about the object is spread over the whole image plane. This led Bertero and Pike to turn to the concept of a "singular value decomposition". A singular value decomposition is similar to an eigenvalue decomposition except that the operator is

allowed to map between two different sets of functions. For example, let us define a new operator, K

$$g(y) \quad = (Kf)(y) \quad = \quad \frac{1}{2\pi}\int_{-\Omega}^{\Omega} \exp(-iky)\,dk \int_{-x_0}^{x_0} f(x)\exp(ikx)\,dx \qquad 1.5.7$$

Now, the only difference between K and the previous operator, A, is that y now lies on the interval $(-\infty, \infty)$ rather than $[-x_0, x_0]$. But x still lies on the interval $[-x_0, x_0]$. Thus K maps from the set $L^2[-x_0, x_0]$ to the set $L^2[-\infty, \infty]$.

The singular functions and values are defined by the two coupled equations.

$$\begin{aligned} \mathbf{K}\ \mathbf{u}_k &= \alpha_k \mathbf{v}_k \\ \mathbf{K}^T \mathbf{v}_k &= \alpha_k \mathbf{u}_k \end{aligned} \qquad 1.5.8(a),(b)$$

where $u_k(x)$ are defined over interval $[-x_0, x_0]$ and $v_k(y)$ over $(-\infty, \infty)$. 1.5.8 is the equivalent of 1.5.3. Note that, whereas in the eigenfunction case we have only one set of functions, the $u_k(x)$, we now have two sets, the $u_k(x)$ and the $v_k(y)$.

Now, if we premultiply 1.5.8 (a) by $\mathbf{K}^T$ we obtain

$$\mathbf{K}^T\mathbf{K}\mathbf{u}_k \quad = \quad \alpha_k^2 \mathbf{u}_k \qquad 1.5.9$$

Thus, the $u_k(x)$ are eigenfunctions of $\mathbf{K}^T\mathbf{K}$ with eigenvalues α_k^2. Now, Bertero and Pike go on to show that

$$\mathbf{K}^T\mathbf{K} \quad = \quad \mathbf{A} \qquad 1.5.10$$

Thus, the $u_k(x)$ are actually identical to the eigenfunctions of A which is why we have given them the same symbol, $u_k(x)$. As before, they are orthogonal and complete over the interval $[-x_0, x_0]$. Furthermore, we must have

$$\alpha_k \quad = \quad \lambda_k^{\frac{1}{2}} \qquad 1.5.11$$

Similarly, we can show that

$$\mathbf{K}\mathbf{K}^T\mathbf{v}_k \quad = \quad \alpha_k^2 \mathbf{v}_k \qquad 1.5.12$$

So, the $v_k(y)$ are eigenfunctions of operator $\mathbf{K}\mathbf{K}^T$. This is a self-adjoint operator so $v_k(y)$ are also orthogonal. N.B. The $v_k(y)$ are complete only over the set of functions on $(-\infty, \infty)$ which have bandwidth, 2Ω. They are not complete over all functions on $(-\infty, \infty)$.

Due to the orthogonality and completeness properties of the $u_k(x)$ and the $v_k(y)$ we can express f and g as follows

$$\mathbf{f} = \sum_{k=0}^{\infty} f_k \mathbf{u}_k \qquad \mathbf{g} = \sum_{k=0}^{\infty} g_k \mathbf{v}_k \qquad 1.5.13(a),(b)$$

where f_k and g_k are given by

$$f_k = \int_{-x_0}^{x_0} f(x) u_k(x)\, dx \qquad g_k = \int_{-\infty}^{\infty} g(y) v_k(y)\, dy \qquad 1.5.13(c),(d)$$

Notice the differences between the range of integration in 1.5.13(c) and the range of integration in 1.5.13(d).

Now, in a way similar to that in the eigenfunction case, we can apply operator K to f as defined in 1.5.13(a).

$$\begin{aligned} \mathbf{K}\mathbf{f} &= \mathbf{K}\left(\sum_{k=0}^{\infty} f_k \mathbf{u}_k\right) \\ &= \sum_{k=0}^{\infty} f_k \mathbf{K}\mathbf{u}_k \\ &= \sum_{k=0}^{\infty} f_k \alpha_k \mathbf{v}_k \quad \text{using 1.5.8 (a)} \end{aligned}$$

But, we also have

$$\begin{aligned} \mathbf{K}\mathbf{f} &= \mathbf{g} \quad \text{from 1.5.7} \\ &= \sum_{k=0}^{\infty} g_k \mathbf{v}_k \quad \text{from 1.5.13(b)} \end{aligned}$$

So by equating co-efficients of $v_k(y)$ we have

$$f_k = \frac{g_k}{\alpha_k} \qquad 1.5.14$$

which is the equivalent of 1.5.6. But from 1.5.11 we have

$$f_k = \frac{g_k}{\lambda_k^{\frac{1}{2}}} \qquad 1.5.15$$

So again, as in 1.5.6., we have a relationship between the image co-efficients, g_k, and the object co-efficients, f_k. Again this relationship allows us to reconstruct the object from the image. But this time we must divide by $\lambda^{\frac{1}{2}}$ whereas in 1.5.6. we had to divide by λ. And since $\lambda < 1$, $\lambda^{\frac{1}{2}} > \lambda$ and so the division is more stable than in equn. 1.5.6. Thus SVD is less sensitive to noise than EVD. This is essentially because SVD makes use of all the information in the image plane rather than just that in the geometrical image region. This proves the main conclusion of Bertero and Pike's first paper: that SVD is superior to EVD in the presence of noise.

Bertero and Pike go on to discuss in more detail the effects of noise on the SVD algorithm and derive an expression for the smallest singular value which can be used in the presence of a given level of noise. We shall discuss this in Chapter 3 when we look at my own simulations in the presence of noise.

1.5.4 w-functions

At this point it is convenient to introduce a third set of functions the w_k. These exist in the Fourier plane and are defined over the region $[-\Omega, \Omega]$ which is the pass-band of the system. They are defined by

$$w_k(k) = \int_{-\infty}^{\infty} v_k(y) \exp(iky)\, dy \qquad 1.5.16$$

i.e. they are the Fourier transforms of the $v_k(y)$. They are complete and orthogonal over $[-\Omega, \Omega]$.

We could repeat the above analysis, substituting $w_k(k)$ for $v_k(y)$ and substituting the Fourier transform, G(k), for the image, g(y). This would be precisely equivalent to the foregoing analysis. Thus we can equally well carry out SVD in either the Fourier plane using the $w_k(k)$ or in the image plane using the $v_k(y)$.

This is because the Fourier transform and the image contain exactly the same amount of information, since the one is easily obtainable from the other.

The reason I introduce the $w_k(k)$ is that it is usually more convenient to work with them than with the $v_k(y)$, since they occupy a smaller spatial region and so they occupy less space inside the computer. If we ever wish to compute the $v_k(y)$, it is usually better to compute the w_k first, and then obtain the $v_k(y)$ from them by inverse Fourier transforming them. See subsection 2.3.8.

Section 1.6 Miscellaneous topics

1.6.1 Computation of prolate spheroidal functions

Methods for the computation of prolate spheroidal functions are summarised in Frieden's review paper of 1971 [11]. All the methods he describes are based on deriving linear expansions of the PSFs in terms of Bessel functions. The coefficients can be derived from first principles and are tabulated in various standard works (e.g. Flammer [18]). Frieden suggests that it might be possible to compute the functions as the eigenfunctions of 1.2.2 but that had not been tried up until that time. He suggests this might avoid "extensive computation".

In their third paper [3] Bertero and Pike give a computational algorithm which is based on calculating the eigenfunctions of the operator A. They use a technique known as "the power method followed by Hotelling's deflation". This is described in more detail in Chapter 2.

However, the method Pike and Bertero adopt of calculating the elements of operator A is rather obscure. I have adopted what I regard as a much simpler and more obvious method, and this is the main subject of Chapter 2.

1.6.2 Two-dimensional prolate spheroidal functions

So far we have discussed only one-dimensional imaging and one-dimensional prolate spheroidal functions. The extension to two-dimensions is straight-forward if our 2D imaging system can be separated into two 1D systems. For example, if the

object and pupil are both square. Then we can form 2D PSFs by multiplying together 1D PSFs as follows

$$\Psi_{ij}(x,y) \quad = \quad \psi_i(x)\psi_j(y) \qquad 1.6.1$$

If the object and pupil are both circular, the problem is more difficult but it can still be solved. This was the subject of Slepian's paper in 1964 [6]. The eigenfunctions of this problem are known as Generalised Prolate Spheroidal Functions (GPSW). Bertero and Pike also treat these two cases.

Systems where the object and the pupil are both different and arbitrary were first treated by Fedotowsky and Boivin [30] in 1972. These functions can be defined as eigenfunctions of the equation:

$$\lambda_{ij}\Psi_{ij}(\mathbf{y}) \quad = \quad \frac{1}{(2\pi)^2}\int_{\Omega}\exp(-i\mathbf{y}\cdot\mathbf{k})\int_{X}\Psi_{ij}(\mathbf{x})\exp(i\mathbf{k}\cdot\mathbf{x})\,d\mathbf{x}\,d\mathbf{k} \qquad 1.6.2$$

where X is the closed, 2D region in which the object lies and Ω is the closed 2D region which represents the pupil. Ω and X are different and arbitrary. The integral operator on the RHS of equn. 1.6.2. is the 2D imaging operator which we met in equn. 1.3.7.

Fedotowsky and Boivin extended many of the properties of the 1D PSFs to these 2D functions eg. dual orthogonality etc. They also extended the symmetry properties of the PSFs to 2D. Please recall from section 1.4.5 that the PSFs must have either odd or even symmetry about the origin. Fedotowsky and Boivin showed that if the regions X and omega both have symmetry about the origin then the 2D eigenfunctions must also have either odd or even symmetry about the origin. We shall find this fact very useful in chapter 2 because it helps to reduce the computational requirements for calculating the 2D eigenfunctions.

Fedotowsky and Boivin also said that "several practical methods for calculating and using these functions have been developed and will be presented in a subsequent paper". But, as far as I know, no subsequent paper has ever appeared. So far no

paper has appeared from any source on the computation of 2D PSFs where object and pupil differ in shape.

In Chapter 2 I shall present my algorithm for computing such functions which I claim to be original and the first practical algorithm which is capable of doing just this.

1.6.3 Coherent and incoherent illumination

Bertero and Pike's second paper is about incoherent illumination. In this case the transfer function, H(k), is no longer a simple binary function. (In fact, it is the auto-correlation of the corresponding transfer function for coherent illumination). See fig. 1.4. This presents us with a choice: we can either use the method suggested by Frieden [10] or that suggested by Bertero and Pike [2]. Frieden suggests that we deconvolve the data by dividing G(k) by H(k), as described in section 1.3.2, and then proceed as for coherent images using the singular functions described above and using the deconvolved image for vector $\mathbf{g}$.

However, Bertero and Pike adopt a different approach. They seek to find the singular functions of the following equation.

$$g(y) \quad = \quad \frac{1}{2\pi} \int_{-2\Omega}^{2\Omega} \exp(-iky) H(k)\, dk \int_{-x_0}^{x_0} f(x) \exp(ikx)\, dx \qquad 1.6.3$$

which includes the transfer function. They then apply these functions directly to the incoherent image without deconvolving it. The eigenvalues of this equation tend to follow a curve typical of that shown in fig 1.5. which shows the eigenvalues for a space-bandwidth product of 2π. Compare this with fig. 1.3 which shows the eigenvalues for a coherent system with the same space-bandwidth product. The coherent eigenvalues are larger than the corresponding incoherent ones.

It is not immediately obvious whether either of these techniques will be better than the other. On the one hand, Frieden's technique involves division by H(k) which as we saw in section 1.3.2 can be unstable when H(k) approaches zero as it does in this case. On the other hand, Bertero and Pike's method involves division

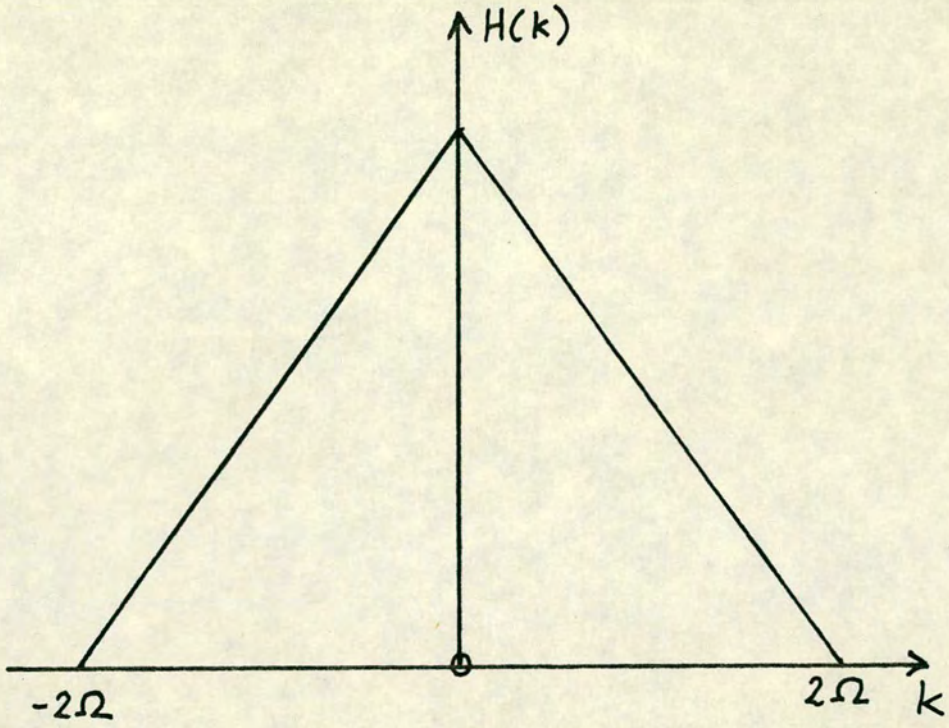


Fig 1.4

The one-dimensional, incoherent, diffraction-limited transfer function

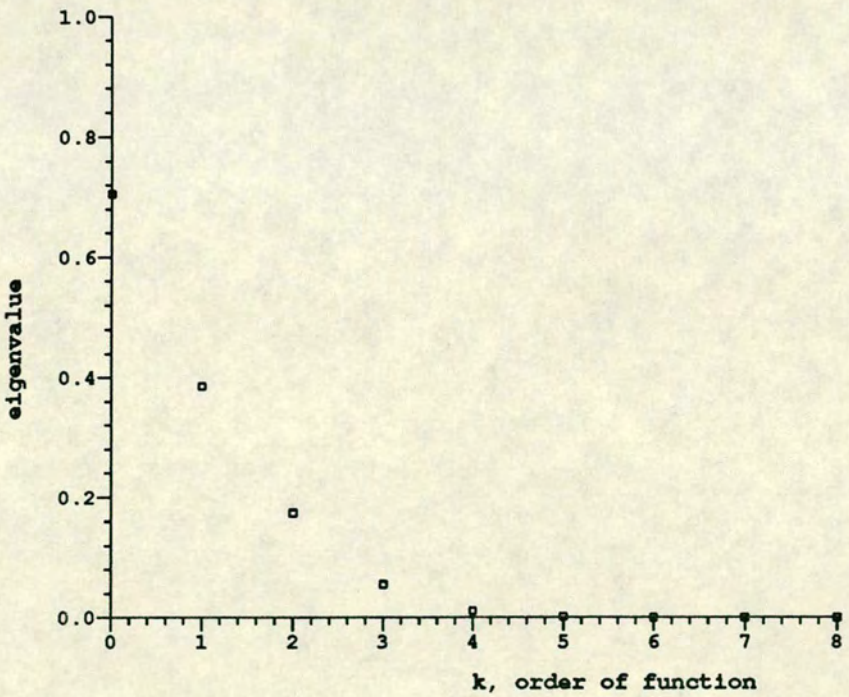


Fig 1.5

The eigenvalues of equn. 1.6.3 with $c = 2\pi$

by smaller eigenvalues and so this introduces instability. It is possible that these two effects might cancel out. In chapter four where experimental incoherent images are considered I use both and compare the results. In the appendix I give a numerical simulation which compares the results of the two methods in the presence of Gaussian additive noise.

1.6.4 Restriction of SVD to low space-bandwidth products

Bertero and Pike [1] give a graph of the expected increase in resolution due to SVD against c, the space-bandwidth product, for different signal-to-noise ratios. This graph is reproduced in fig. 1.6. From this graph we can see that at a typical signal-to-noise ratio, say 100, SVD gives a significant increase in resolution only at values of c less than ~ 10. Cox and Shepherd [20] confirm this behaviour. This is a serious limitation on SVD and we shall discuss its implications in chap. 5 when we compare the performance of SVD with other super-resolution techniques.

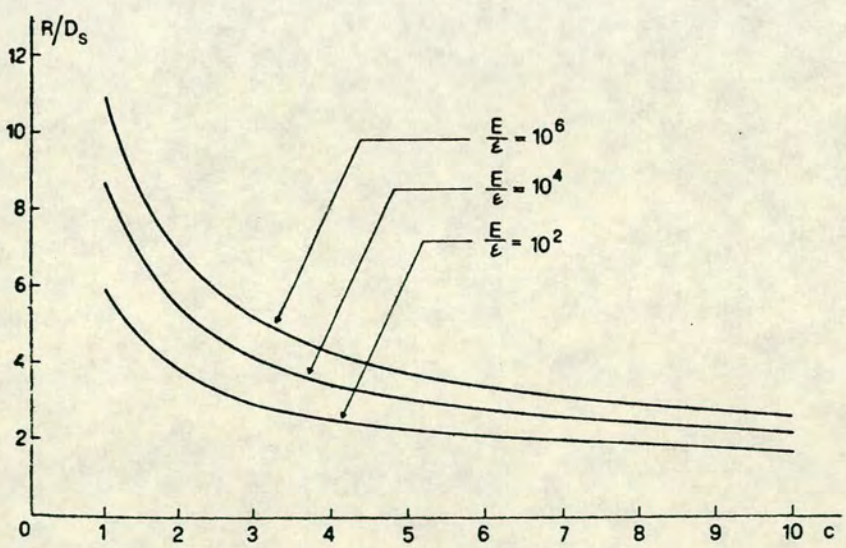


Fig. 1.6

A graph of super-resolution gain against c

1.6.5 Experimental work

So far I have been able to find only one paper which describes the application of super-resolution to real experimental images, by Walker in 1984 [38]. All the other papers give only computer simulations. Walker describes a 1D example where the data is collected by a photo-multiplier scanned through a deliberately blurred image of three boxes. The illumination is coherent. He uses the Gerchberg technique [39] to achieve super-resolution.

The photo-multiplier has the advantage that it gives high signal-to-noise ratio and good dynamic range, but it is slow and awkward to gather data in that it has to be scanned mechanically across an image. It would be very tedious to acquire a 2D image in this way, since the photomultiplier would have to be scanned both horizontally and vertically.

The use of coherent light means that the intensity values must be square-rooted to obtain the amplitude values and then assigned a phase value - either positive or negative. (Walker knows in advance that the object is real and therefore that the image must be real and so can either be positive or negative). The necessity to do this does not arise with incoherent illumination.

Chapter Two

Computing the Singular Functions

2.1 Introduction

One of the main objectives of my PhD was to compute truly two-dimensional analogues of the prolate spheroidal functions. These functions are defined by the equation 1.6.2

$$\lambda_{ij}\Psi_{ij}(\mathbf{y}) \quad = \quad \frac{1}{(2\pi)^2}\int_{\Omega}\exp(-i\mathbf{y}\cdot\mathbf{k})\int_{X}\Psi_{ij}(\mathbf{x})\exp(i\mathbf{k}\cdot\mathbf{x})\,d\mathbf{x}\,d\mathbf{k} \qquad 1.6.2$$

where X is the region within which the object lies and Ω is the region within which the pupil lies. When X and Ω are identical this equation becomes separable and the functions become relatively easy to compute but, if they are different the equation is inseparable and must be treated as a completely two-dimensional problem.

At first sight the problem would seem to be too huge to be soluble in any reasonable time. For example, suppose we wished to compute the functions on a 64x64 grid of sample points, then the integral operator defined in equation 1.6.2 would have 64x64x64x64 = 16 million elements. To store such an operator would require 64Mbytes of memory. It is probably considerations such as this that have led previous researchers to abandon the problem.

However, a closer examination shows that it is not so daunting as it might seem. For a start the operator has a huge amount of redundancy- many of the elements are the same. Being a convolution operator it depends only on the difference (y-x) rather than the individual values of y and x. So we need not store the whole array.

In fact, we shall see that the best way to represent the operator is not as an array but as a sequence of subroutines, in which the Fourier transform operators in equn. 1.6.1 are represented by Fourier transform subroutines. We therefore need to store only the Fourier coefficients, which for the above example could be stored in a 64x64 complex array.

Even further reductions in storage and CPU time can be made if the regions X and Ω have some symmetry. For instance, if they both have quarter-turn symmetry (as for example the square and circle do) then we can reduce storage and CPU by a factor of 4.

Looked at in this way, the problem becomes a lot more tractable. I have succeeded in solving this problem for one particular case: when X is a square and Ω is a circle. My method should be easy to generalise to other cases especially when both regions share certain symmetry properties. In principle it can be extended to regions of completely arbitrary shape. The remainder of this chapter describes the algorithm used to calculate these functions. I claim that this algorithm is original and as far as I know the first to solve this problem.

Section 2.2 A discrete mathematical model of an imaging system

2.2.1 Sampling, truncation and noise

The key to solving equn. 1.6.2 computationally is to develop a mathematical model of an optical system, which retains all the essential features of the system and yet can be implemented inside a computer. In chapter 1 we began to build such a mathematical model. Our first step was to describe both the object and the image by two-dimensional complex functions and then to introduce Fourier theory and show how this provides a simple relationship between the object and the image. However, that model was rather abstract and idealised; - all the functions were continuous and some were defined over all space i.e $(-\infty, \infty)$. In the real, physical world we cannot measure such functions. In this chapter I want to make the model more like a real physical system and to include all the characteristics that an experimenter might meet in the laboratory.

There are two basic sets of characteristics which we must include. The first is that we cannot measure continuous functions; we can only measure a function at a set of discrete sample points. Secondly, we cannot measure an infinite function, we can only measure a function over a finite range or, in other words, the function must

be truncated.

The experimenter may have some choice as to how far apart he places his sample points and as to where he places his truncation boundary. We shall discuss the best choice of these parameters for a given system. We shall make one restriction, however, we shall only consider rectangular sampling arrays.

What parameters are necessary to define our mathematical model completely? To answer this, let us recall from chapter 1 that the imaging process can be broken down into three stages;

1. The object, f(x) is Fourier transformed to give, F(k).
2. F(k) is multiplied by H(k) to give G(k).
3. G(k) is inverse Fourier transformed to give g(y), the image.

We are restricting our attention to diffraction-limited systems, therefore H(k) is just a binary function, which has the value unity inside a certain closed boundary and zero outside. Remember, that the object, f(x), is also contained within a closed boundary - the a priori information on which our SVD algorithm is based. Thus, our optical system is defined completely once we know these two boundaries - one in object space and one in Fourier space.

We can think of our optical system as consisting of three planes - or "spaces" - the object plane (or space), the Fourier plane and the image plane. Any function which we deal with must exist in one of these planes. As we said in the opening paragraph of this section, all functions must be sampled and truncated. Therefore, we must choose where we are going to place the sample points and the truncation boundaries in each of the three planes. The sample points and boundaries need not be the same in each plane. However, there is a certain relationship between two neighbouring planes. This relationship is provided by the rules for Fourier transforming sampled functions. We shall discuss this in the next section.

Section 2.2.2 The discrete fourier transform (DFT)

The rules governing the Fourier transforms of discrete functions are discussed

fully by Newland[32] and Brigham[33]. I shall summarise their accounts here. I shall make it one-dimensional for simplicity.

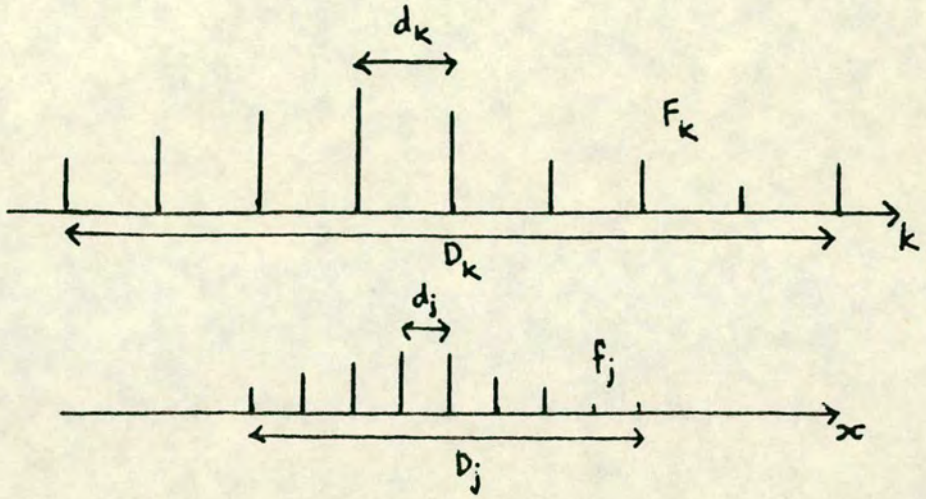


Fig. 2.1 A discrete function and its Fourier transform

Fig. 2.1 shows a discrete function, f_j, and its discrete Fourier transform (DFT), F_k. Both functions are sampled- in other words they have values only at a discrete set of points- and both are truncated- in other words they are only defined over a finite range. Let the truncation distances in each plane be D_k and D_j respectively, and the sample spacing in each plane d_k an d_j. Then, the rules of the DFT demand the following relationships

$$d_j = \frac{2\pi}{D_k} \qquad d_k = \frac{2\pi}{D_j} \tag{2.2.1}$$

Thus, the sample spacing in one plane determines the truncation spacing in the other and vice versa. Clearly, the smaller the values of d_k and d_j the closer the model approximates to the continuous case.

The total number of sample points within one truncation cell in either plane is given by

$$\frac{D_k}{d_k} = \frac{D_j}{d_j} = N \tag{2.2.2}$$

and is the same in both planes.

We can now define the discrete Fourier transform pair as follows

$$\begin{aligned} F_k &= \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} f_j \exp(2\pi ijk/N) \\ f_j &= \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} F_k \exp(-2\pi ijk/N) \end{aligned} \qquad 2.2.3$$

It is easy to show that these two discrete operators are inverses of one another.

Section 2.2.3 Choosing the sample spacings in the three planes

In section 2.2.2 we saw that there was a relationship between the sample spacing in one plane and the truncation spacing in its Fourier transform and vice versa. In the light of this relationship how can we choose the best sample spacings for our mathematical model of an imaging system?

Let us work in one dimension for simplicity. Let the sample spacings in the object plane, the Fourier plane and the image plane be d_O, d_F and d_I respectively and the truncation spacings be D_O, D_F and D_I in each of the three planes respectively. Let the object lie in the region $[-x_0, x_0]$ and the pupil in $[-\Omega, \Omega]$ The geometrical image lies in the same region as the object.

We can see at once that D_O and D_I must be equal. This is because by equn. 2.2.1 they are both given by

$$D_O = D_I = \frac{2\pi}{d_F} \qquad 2.2.4$$

Thus, the truncation spacings are the same in both the object and the image planes.

Furthermore, the sample spacings in the object and image planes must also be the same. This is because the operator, A, maps the sample points contained within the object boundary onto the sample points contained within the geometrical image boundary. If we are to calculate the eigenfunctions of A, then clearly the sample points in the image and object planes must be in a one-to-one correspondence.

Thus we have both

$$D_O = D_I \quad \text{and} \quad d_O = d_I \qquad 2.2.5$$

and, once we have determined these, the sample spacing and truncation spacing in the Fourier plane are also determined via equn. 2.2.1. Therefore, the model really only has two free parameters, and, once these have been determined, the whole model is determined.

What is the best way of choosing these parameters? I think that they should be chosen so that both D_O and D_F are large with respect to the object and pupil size respectively. This is primarily so that truncations in the Fourier and image planes do not overlap to a large extent. But, it also ensures that our model is an adequate representation of a band-limited system and furthermore, that it incorporates our a priori information that the object is space-limited.

The only way we can ensure that our model is a band-limited system is to make sure that a large part of the Fourier transform is thrown away, and the only way to do that is to make D_F large compared to $2\,\Omega$, the bandwidth.

Furthermore, the only way we can represent a space-limited object is to force it to sit in the centre of a large field of zeros, and this implies D_O is large compared with $2x_0$.

If we choose our parameters so that our model meets these two conditions then we can be sure that it be a reasonable approximation to a real optical imaging system. In the next section I shall give some typical numerical values for these parameters.

Section 2.2.4 A particular example of a discrete model

Let us look at a typical example of a discrete mathematical model of an imaging system. Let us assume that the object is square with corners at $(\pm 1, \pm 1)$ space units and that the pupil is a circle of radius π frequency units.

It will make the computing simpler if we make both object and pupil occupy roughly similar numbers of sample points. So let us place both the object and the

pupil on 64x64 grids. Of course, the object being square will completely cover the grid, but the pupil, being circular, will leave some points uncovered. The sample spacing in the object plane is thus

$$d_O = \frac{1}{32} \qquad 2.2.6$$

and the sample spacing in the Fourier plane is

$$d_F = \frac{\pi}{32} \qquad 2.2.7$$

The total number of points per truncation cell is given by

$$N = \frac{D_O}{d_o} = \frac{2\pi}{d_O d_F} = \frac{2\pi}{(1/32)(\pi/32)} = 2048 \qquad 2.2.8$$

We thus have to imagine the object as sitting in field of 2048x2048 points of which all are zero except the central grid of 64x64 on which the object sits. Similarly, the DFT of this object occupies 2048x2048 points but out of these only those contained within the pupil are let through to form the image.

2.3 Expressing A and K as subroutines

2.3.1 The discrete equivalents of A and K

We can now find discrete two-dimensional equivalents of the operators, A and K, defined in equns. 1.5.1 and 1.5.7. We can write A as follows

$$\mathbf{A} = \mathbf{OF^T HFO} \qquad 2.3.1$$

where F represents the DFT operator defined in subsection 2.2.2 and $\mathbf{F}^T$ represents its inverse. O is an operator which sets to zero all sample values outside the object region but leaves those inside unchanged and H is an operator which sets to zero all sample values outside the pupil but leaves those inside unchanged. Operator A maps the sample points inside the object region onto the sample points inside the geometrical image region.

Now operator K is defined as

$$\mathbf{K} = \mathbf{F}^{\mathrm{T}}\mathbf{HFO} \qquad 2.3.2$$

and it maps the points inside the object region onto all image space, i.e. all the points contained within the truncation cell in image space.

With A and K thus defined, the whole analysis given in section 1.5 carries over to the multi-dimensional, discrete case. We can thus define u_k and v_k as given in equns. 1.5.3 and 1.5.8. The u_k are now defined over the object region and the v_k are now defined over the truncation cell in image space. For example, in the discrete model described in subsection 2.2.4 the u_k would be vectors with 64x64 elements and the v_k would be vectors with 2048x2048 elements.

We have defined the operators A and K in the discrete, multi-dimensional case, in the rest of this section we shall consider how to express these operators as sequences of computer subroutines. We shall first consider the subroutine which carries out the DFT and then show how it can be combined with other subroutines to form A or K.

Section 2.3.2 The finite Fourier transform routine

In the previous section we used a 2048x2048 DFT as the basis for our mathematical model. But we really only used the central 64x64 elements out of those 2048x2048. For instance, the object occupied only 64x64, the rest of the 2048x2048 were set to zero. Similarly, the pupil occupied fewer than 64x64 sample points, the rest of the 2048x2048 were thrown away.

Now what is the best way to carry out the DFT computationally? The first thing that comes to mind is to use a standard Fast Fourier Transform (FFT) routine. But the problem here is that we would have to supply such a routine with a 2048x2048 array. Such an array would occupy 16Mbytes of storage. A huge amount of CPU time would be spent moving data in and out of memory. The routine would be very slow. Furthermore, it would compute all 2048x2048 elements of the DFT when we really only need the central 64x64. The routine thus spends most of its time computing values which we are just going to throw away.

Clearly this is not an efficient solution to our problem. The solution which I eventually chose was to write my own routine, which I call a "Finite Fourier Transform routine." This routine accepts as input the central 64x64 elements of a 2048x2048 array- (the rest of the array elements are assumed to be zero.) It then computes the central 64x64 elements of the 2048x2048 DFT. This is obviously much more efficient since it never deals with arrays larger than 64x64. The values it does calculate, however, are identical to those that would be calculated by a 2048x2048 FFT (if we could get such a routine to work.) We have merely calculated a small segment of such a routine.

The formula for such a routine is given by

$$F_{k_1 k_2} = \frac{1}{2048} \sum_{j_1=1}^{64} \sum_{j_2=1}^{64} f_{j_1 j_2} \exp\left(\frac{2\pi i (j_1 - 32.5)(k_1 - 32.5)}{2048}\right) \exp\left(\frac{2\pi i (j_2 - 32.5)(k_2 - 32.5)}{2048}\right)$$

where I have moved the zero order to lie at the centre of the 64x64 array at the point (32.5,32.5). This is because optical systems usually have their pupils centered on the zero order. Notice that the denominator in the exponent is still 2048, not 64, this is what characterises it as a 2048 DFT , whereas the limit on the summation is 64. This is what characterises it as a finite Fourier transform, we are only calculating 64 elements out of 2048.

This formula can be simplified if we know that the function to be transformed has either odd or even symmetry about the horizontal and vertical axes and that it is real. For instance, if the function is real and it has even symmetry about both axes then we know its Fourier transform must also be real and have even symmetry about both axes. We can therefore replace the complex exponentials by cosines.

$$F_{k_1 k_2} = \frac{1}{2048} \sum_{k_1=1}^{64} \sum_{k_2=1}^{64} f_{j_1 j_2} \cos\left(\frac{2\pi (j_1 - 32.5)(k_1 - 32.5)}{2048}\right) \cos\left(\frac{2\pi (j_2 - 32.5)(k_2 - 32.5)}{2048}\right)$$

Similarly, if the function has odd symmetry about both axes then we can replace the complex exponentials by sines, and if it has odd symmetry about one axis and even about the other than we replace one exponential by a cosine and the other by a sine. We shall find the ability to do this useful in subsection 2.3.5. when it will be necessary to find eigenfunctions with certain symmetry properties.

Section 2.3.3 Expressing operator A as a sequence of subroutines

We can now express operator, A, as a sequence of subroutines. For instance, for the particular case of the square object and circular pupil, which we discussed in subsection2.2.4 we can call the following subroutines

```
CALL FINFT(U)
CALL PUPIL(U)
CALL FINFT(U)
```

where U is a 64x64 array which initially contains the object sample data. FINFT is the finite Fourier transform routine and PUPIL is a routine which sets to zero all sample values outside the pupil region.

Thus, the first subroutine calculates a 64x64 segment in the centre of the Fourier transform. The second subroutine, PUPIL, throws away all sample values of this 64x64 segment outside the pupil and the third subroutine calculates the bandlimited image within the geometrical image region.

If the object had not been contained within a square region but in a region of some other shape we could add an extra subroutine to either end of this sequence, thus

```
CALL OBJECT(U)
CALL FINFT(U)
CALL PUPIL(U)
CALL FINFT(U)
CALL OBJECT(U)
```

where OBJECT sets to zero all elements of the 64x64 array outside the object

region.

In the separable case where object and pupil are both contained within square regions we can simply call

CALL FINFT(U)

CALL FINFT(U)

Thus A is expressed by a sequence of subroutines, not by an array of matrix elements. It is important to realise this when we come to consider the algorithm which computes the eigenfunctions and values.

2.3.4 Power Method and Hotelling's Deflation

In this section we look at the algorithm used to compute the eigenfunctions and singular functions. There are no algorithms available for computing singular functions directly but there are algorithms to compute eigenfunctions. As we have seen, the singular functions u_k are the eigenfunctions of the operator $A = K^T K$. Therefore, we must construct an algorithm to find the eigenfunctions of this operator. There are many such algorithms available, but I decided to choose the simplest, and therefore the easiest to program, even though it is probably the least efficient. The algorithm I chose was the "Power Method" followed by "Hotelling's deflation". This is the algorithm used by Bertero, Pike et al.[3]. The method is fully described by Ralston[34].

A further reason for rejecting the more sophisticated algorithms available is that these algorithms assume that the operator is specified by a matrix or array, whereas, in our case, it is specified by a series of subroutines. There is no one array which completely describes our operator. Thus, it is not possible, as some of the more sophisticated algorithms require, to select a given set of array elements from the array that specifies the operator. On the other hand, in the case of our simple algorithm, it is very easy to adapt it to cope with an operator which is specified by subroutines rather than a single array. This is because, it is essentially an iterative process.

In order for the algorithm to work we need to know that the eigenvalues are

widely spaced and in decreasing numerical order. We know this is true for small values of c in the one-dimensional case. See subsection. 1.4.6. It may not be true in the two-dimensional case where degenerate eigenvalues are possible. I shall deal with that problem below.

The Power method is based on the following fact; if an arbitrary vector, $\mathbf{x}_0$, say, is repeatedly multiplied by an operator, $\mathbf{A}$, then the product vector, $\mathbf{x}_1$, will eventually converge on the eigenvector of $\mathbf{A}$ with the largest eigenvalue - the so-called dominant eigenvector. We can then find the eigenvector with the second highest eigenvalue by using Hotelling's deflation. This involves forming the operator $\mathbf{A}_2$ specified by

$$\mathbf{A}_2 = \mathbf{A} - \lambda_1 \mathbf{u}_1 \mathbf{u}_1^T \qquad 2.3.5$$

where $\mathbf{u}_1$ is the dominant eigenvector, which has now been normalised. $\mathbf{A}_2$ has all the same eigenvectors as $\mathbf{A}$ apart from $\mathbf{u}_1$ which has eigenvalue 0 when multiplied by $\mathbf{A}_2$. Therefore, we can now apply the Power method to $\mathbf{A}_2$ and so find $\mathbf{u}_2$, the dominant eigenvector of $\mathbf{A}_2$ and the sub-dominant eigenvector of A. We can then repeat the deflation process by forming $\mathbf{A}_3$

$$\mathbf{A}_3 = \mathbf{A}_2 - \lambda_2 \mathbf{u}_2 \mathbf{u}_2^T \qquad 2.3.6$$

whose dominant eigenvector is now $\mathbf{u}_3$. In principle, we could repeat this process until we had formed all the eigenvectors of A, but in practice we can only calculate the top few eigenvectors before the accumulation of round-off noise causes the algorithm to fail to converge.

How do we adapt this algorithm so that it can cope with an operator that is specified by subroutines rather than an array? It is easy to calculate the dominant eigenvector, $\mathbf{u}_1$, simply by repeatedly calling the appropriate subroutines but how can we carry out the deflation process? The text-books suggest that we should form arrays $\mathbf{A}_2$, $\mathbf{A}_3$ etc. but clearly we cannot do this if our original operator is not expressed by an array in the first place. Our solution is to add an extra subroutine

onto the ones already specifying the original operator. Thus, for example, if we wish to multiply a vector $\mathbf{x}_0$ by $\mathbf{A}_2$ we can carry this out in three stages

$$\begin{aligned} \mathbf{A}_2\mathbf{x}_0 &= (\mathbf{A} - \lambda_1 \mathbf{u}_1 \mathbf{u}_1^T)\mathbf{x}_0 \\ &= \mathbf{A}\mathbf{x}_0 - \lambda_1 \mathbf{u}_1 \mathbf{u}_1^T \mathbf{x}_0 \end{aligned} \qquad 2.3.7$$

The first stage is to multiply $\mathbf{x}_0$ by A to give a vector $\mathbf{x}_{temp}$. This can be done by the subroutines we have already described. The second stage is to form the scalar product, g, say, of $\mathbf{u}_1$ and $\mathbf{x}_0$.

$$g = \mathbf{u}_1^T \mathbf{x}_0 \qquad 2.3.8$$

This forms part of the deflationary stage of the above equation. The third stage is to multiply $\mathbf{u}_1$ by λ_1 and g and subtract it from $\mathbf{x}_{temp}$, thus

$$\mathbf{x}_1 = x_{temp} - \lambda_1 g \mathbf{u}_1 \qquad 2.3.9$$

These last two stages are carried out by our new subroutine DEFLATE. If we wish to form the operators $\mathbf{A}_3$, $\mathbf{A}_4$ etc. we simply repeat the last two stages using the eigenvectors $\mathbf{u}_2$, $\mathbf{u}_3$ etc.

2.3.5 Symmetry and degenerate eigenfunctions

With 2D functions we can have degenerate or near-degenerate eigenvalues. This can create a problem for our algorithm since it works best when the eigenvalues are widely spaced.

The reason for the degeneracy is clear if we consider the separable case. Here the eigenvalues are products of 1D eigenvalues. Thus, if the 2D eigenvalue is λ_{ij} and the 1D values are λ_i and λ_j we have

$$\lambda_{ij} = \lambda_i \lambda_j \qquad 2.3.10$$

But, we also have

$$\lambda_{ji} = \lambda_j \lambda_i \qquad 2.3.11$$

where λ_{ij} and λ_{ji} are the eigenvalues of two different eigenfunctions $\mathbf{u}_{ij}$ and $\mathbf{u}_{ji}$. (One is merely the other rotated through 90 degrees, but, nevertheless, they are different functions). Thus, we have degenerate eigenvalues.

This creates a problem for the algorithm because it will not know onto which eigenfunction to converge. But we can solve this problem as follows. When we have degenerate eigenfunctions, any linear combination of them is also an eigenfunction with the same eigenvalue. Thus, we can form two new eigenfunctions as follows

$$\begin{aligned} \mathbf{u}_S &= \mathbf{u}_{ij} + \mathbf{u}_{ji} \\ \mathbf{u}_A &= \mathbf{u}_{ij} - \mathbf{u}_{ji} \end{aligned} \qquad 2.3.12$$

where the subscripts "S" and "A" stand for "symmetric" and "anti-symmetric". this is because $\mathbf{u}_S$ will be symmetric about the leading diagonal and $\mathbf{u}_A$ will be anti-symmetric. The reason I choose these particular combinations is that it is possible to force our algorithm to converge only on eigenfunctions which have one particular symmetry property eg. symmetric or anti-symmetric about the leading diagonal. The algorithm will thus converge on only one of the degenerate pair and we shall have no problems with closely-spaced eigenvalues.

How do we force the algorithm to do this? We have to add yet another subroutine to those which describe our operator. This subroutine forces its input vector to become either symmetric or anti-symmetric by replacing the values on one side of the diagonal by their images under reflection in the diagonal (having been multiplied by ± 1 as appropriate). This subroutine is called REFL and is called as follows.

```
CALL REFL(U)
CALL OBJECT(U)
CALL FINFT(U)
CALL PUPIL(U)
CALL FINFT(U)
CALL OBJECT(U)
CALL REFL(U)
```

As mentioned before, it is possible to force the DFT routines to assume certain symmetries about the horizontal and vertical axes. This is described by equation 2.3.4 where we can choose the function to be symmetric by making the Fourier kernel into a cosine function rather than a complex exponential or we can make it anti-symmetric by choosing the kernel to be a sine function. This allows us to limit further the number of functions onto which the algorithm can converge. Thus, for example, we could force the algorithm to converge only on functions which were symmetric about both axes and the leading diagonal. In fact, there are six sets of functions which we can force the algorithm to converge on. These are shown in the table below.

Horizontal	Vertical	Diagonal
S	S	S
S	S	A
A	A	S
A	A	A
A	S	-
S	A	-

In the last two cases, where the functions have opposite symmetry about the two axes, it is not possible for them to have any particular symmetry about the leading diagonal.

We know that all the eigenfunctions must belong to one of these sets because of the symmetry properties of the object and pupil. As Fedotowsky and Boivin[30] showed, if the object and pupil are symmetric about the origin (as the square and circle are) then the eigenfunctions must be either symmetric or antisymmetric about the origin. See subsection. 1.6.3. The above six sets are the only sets allowed by this rule.

If the object and pupil did not have any particular symmetry then computing their eigenfunctions would be very much harder using this algorithm, although, in principle, it could be done, given enough CPU time. This problem should be the

subject of further investigation.

Forcing the algorithm to converge on only one set has several advantages. Firstly, it reduces the error in each function because the different sets are calculated independently and therefore the accumulation of round-off error from one function to the next is less. Furthermore, the error is reduced because values which are supposed to be equal are forced to be exactly equal. Secondly, it reduces CPU time because we only need to calculate one quadrant of each function - the rest can be generated by symmetry operations.

2.3.6 Speed of Convergemce

The rate at which the Power method converges depends on how closely spaced the eigenvalues are. If the eigenvalues are widely spaced the algorithm will converge very rapidly on each eigenfunction in turn. If two eigenvalues are very close together the algorithm will take much longer to converge on the larger of the two. The latter tends to be the case for values of $c > 4$, because, in this region the top eigenvalues are very close together - all being nearly unity - as can be seen from figure 1.3 in Chapter 1. In this case we are forced to let the algorithm run for many iterations - perhaps as much as 2-3,000. This larger number of iterations is only necessary for the top one or two eigenfunctions. The rest require only a smaller number - perhaps 100 - because they are spaced further apart.

2.3.7 Accuracy of Eigenfunctions

The accuracy of the computed eigenfunctions will decrease as the eigenvalue decreases. This is partially because errors in the previous eigenfunctions will accumulate and be passed on to subsequent eigenfunctions but mainly because the smaller eigenvalues are very much less than 1.. For example, we often find eigenvalues of the order of 10^{-7}. Let us consider what happens to this eigenfunction when it is multiplied by the operator A. Let us suppose that initially all the elements of the vector are of the order of unity. After the multiplication, they are all of the order 10^{-7}. This reduction has come about mainly as a result of the subtraction of very

nearly equal numbers. These numbers are stored only to a certain degree of precision. For example, a REAL variable in FORTRAN is stored to 8 significant figures and a DOUBLE PRECISION variable is stored to 16 significant figures. Let us suppose, two nearly equal numbers, both of the order of unity, are subtracted and the difference between them is found to be 10^{-7}, how accurately will this difference be known? If the original numbers were stored to 8 significant figures the different will be known to only 1 significant figure accuracy. If the original numbers were stored to 16 significant figures, the different will be known to 16 - 7 = 9 significant figures of accuracy. We can expect roughly the same degrees of accuracy in our estimation of our eigenvalues. Thus, for an eigenfunction stored as DOUBLE PRECISION and with an eigenvalue of 10^{-7} we can expect roughly 9 significant figures of accuracy.

2.3.8 Computing the singular functions

We have now written an algorithm which can calculate the eigenfunctions $\mathbf{u}_k$. What is the best way to calculate the singular functions $\mathbf{v}_k$? The first way that might occur to us is to use equn 1.5.8(a)

$$\mathbf{K}\mathbf{u}_k = \alpha_k \mathbf{v}_k \qquad 1.5.8(a)$$

But, as we have seen above, this will result in an increase in the error in the $\mathbf{v}_k$, especially if λ_k is very much less than 1. Therefore, this is an inherently inaccurate way of estimating $\mathbf{v}_k$.

A possible way is to calculate the $\mathbf{v}_k$ directly as the eigenfunctions of $\mathbf{K}\mathbf{K}^T$. See equn.1.5.12. But this would involve a great deal more CPU time than calculating $\mathbf{u}_k$ because the $\mathbf{v}_k$ normally occupy many more sample points.

The best way to solve this problem is to calculate the functions, $\mathbf{w}_k$, which were introduced in subsection 1.5.4. These functions exist in the Fourier plane and are defined over the pupil region. They are related to the $\mathbf{v}_k$ by equn. 1.5.16. We can rewrite 1.5.16 as follows

$$\mathbf{v}_k = \mathbf{F}^T \mathbf{w}_k \qquad 2.3.13$$

It is easy to show that the $\mathbf{w}_k$ are eigenfunctions of the operator

$$\mathbf{B} = \mathbf{H}\mathbf{F}^T\mathbf{O}\mathbf{F}\mathbf{H} \qquad 2.3.14$$

We can express B using the same subroutines which we used to express A, but they are called in a different order.

CALL PUPIL(U)

CALL FINFT(U)

CALL OBJECT(U)

CALL FINFT(U)

CALL PUPIL(U)

We can then calculate the eigenfunctions of this operator just as we did for operator, A. These eigenfunctions are the $\mathbf{w}_k$. We can then calculate the $\mathbf{v}_k$ from them using equn. 1.5.16.

2.3.9 Computing the incoherent singular functions

In section 1.6.3 it was pointed out that for incoherent images the transfer function was no longer binary and that it was possible to define a new set of singular functions using equn. 1.6.3.

$$g(y) = \frac{1}{2\pi}\int_{-2\Omega}^{2\Omega}\exp(-iky)H(k)\,dk\int_{-x_0}^{x_0}f(x)\exp(ikx)\,dx \qquad 1.6.3$$

This equation incorporates the non-binary transfer function. The transfer function for a one-dimensional, diffraction-limited incoherent system is shown in fig. 1.4.

It is easy to adapt the algorithm described in this chapter to compute these incoherent singular functions. The operator $\mathbf{K}$ is defined as before by equn 2.3.2

$$\mathbf{K} = \mathbf{F}^\mathbf{T}\mathbf{H}\mathbf{F}\mathbf{O} \qquad 2.3.2$$

except that $\mathbf{H}$ now represents multiplication by the non-binary transfer function. The $\mathbf{u}_k$ can now be computed as eigenfunctions of the operator

$$\mathbf{K}^T\mathbf{K} = \mathbf{O}^T\mathbf{F}^T\mathbf{H}^T\mathbf{H}\mathbf{F}\mathbf{O} \qquad 2.3.15$$

We can represent this operator by sequence of subroutines similar to that we used to represent the operator **A** in section 2.3.3

CALL OBJECT(U)

CALL FINFT(U)

CALL INC-PUPIL(U)

CALL INC-PUPIL-TRANS(U)

CALL FINFT(U)

CALL OBJECT(U)

where the subroutine INC-PUPIL carries out multiplication by the operator **H** and INC-PUPIL-TRANS carries out multiplication by its transpose. If the transfer function is real and symmetric it will be identical to its transpose, and we can use the same subroutine twice, but in chap. 4 we shall meet a transfer function which has an imaginary part, in which case two different subroutines are required. The $\mathbf{w}_k$ are given by

$$\mathbf{B} = \mathbf{H}^T\mathbf{F}^T\mathbf{O}\mathbf{F}\mathbf{H} \qquad 2.3.16$$

and to compute them we merely reorder the above sequence of subroutines as specified by 2.3.16.

Chapter Three

Computer Simulations

Section 3.1

3.1.1 Introduction

In this chapter we shall look at some computer simulations of super-resolution. I have generated some fictitious "objects" inside the computer by creating arrays of data and then calculated the "images" of these objects using equn. 1.3.7. I have then applied the EVD and SVD algorithms to these images to see if they return the original object. This should test the internal consistency of my mathematical model and the EVD and SVD algorithms. However, even if the model and algorithm pass this test, it does not necessarily imply that they will work in the real world, when applied to images derived from a real optical system. We shall test that in chapter 4.

I have also added artificial noise to the computer-generated images by adding a random number to each sample value. I have then applied the EVD and SVD algorithms to the noisy images to see if the noise has the detrimental effect predicted in chapter 1.

Bertero and Pike[1] give an analysis of the effect of noise on super-resolution. In their analysis they represent the object, image and noise as continuous functions whereas in this chapter I shall represent them as discrete functions. One has to be careful in carrying over the results of Bertero and Pike's continuous analysis to the discrete case. This is the subject of the next two subsections. In subsection 3.1.2 I give a condensed version of Bertero and Pike's continuous analysis and in 3.1.3 I give my discrete analysis which parallels theirs.

3.1.2 Bertero and Pike's analysis of the noise

In section 1.5 we went through Bertero and Pike's account of EVD and SVD in great detail. However, I did not say a great deal about the effect of noise other than to say that SVD was superior to EVD in the presence of noise because the singular

values, $\lambda_k^{\frac{1}{2}}$, were greater than the eigenvalues, λ_k. Bertero and Pike go on to give a more detailed account of the effect of noise in both cases. I am now going to give a condensed version of that here. The analysis will be one-dimensional for simplicity. It can easily be generalised to two dimensions.

The purpose of the analysis is to calculate the mean square error, $<\beta_k^2>$, in our measurement of each image coefficient, g_k. They make the following assumptions. They assume that the noise is white noise and that it is not correlated with the signal. They further assume that the noise and signal are additive. Therefore, the observed image, $g(y)$, is related to the "true" (i.e. noise-free) image, $\hat{g}(y)$, as follows

$$g(y) \quad = \quad \hat{g}(y) + n(y) \qquad 3.1.1$$

where n(y) is the noise. Because n(y) represents white noise it has the following property

$$< n(y)n^*(y') > \quad = \quad \varepsilon^2 \delta(y - y') \qquad 3.1.2$$

Bertero and Pike go on to calculate the error due to the noise in the image coefficients, g_k. Let us first consider the EVD case. In section 1.5 we saw that the g_k are given by the scalar product of the image, $g(y)$, with the eigenfunctions, $u_k(y)$.

$$g_k \quad = \quad \int_{-x_0}^{x_0} g(y)u_k(y)\,dy \qquad 1.5.5(d)$$

Now, if we substitute 3.1.1 into equn. 1.5.5(d) we get

$$\begin{aligned} g_k \quad &= \quad \int_{-x_0}^{x_0} \hat{g}(y)u_k(y)\,dy \quad + \quad \int_{-x_0}^{x_0} n(y)u_k(y)\,dy \\ &= \quad \hat{g}_k + \beta_k \end{aligned} \qquad 3.1.3$$

where g_k is the observed image coefficient, $\hat{g}_k$ is the "true" image coefficient and β_k is the error due to the noise.

$$\beta_k \quad = \quad \int_{-x_0}^{x_0} n(y)u_k(y)\,dy \qquad 3.1.4$$

Finally, by combining 3.1.4 and 3.1.2 we get

$$< \beta_k^2 > \quad = \varepsilon^2 \qquad 3.1.5$$

Thus we have now obtained an estimate for the mean square error in each image coefficient, g_k, in the EVD case.

Now, let us consider the SVD case. In SVD g_k is given by the scalar product of $g(y)$ with the singular functions, $v_k(y)$, thus

$$g_k \quad = \quad \int_{-\infty}^{\infty} g(y) v_k(y)\, dy \qquad 1.5.13(d)$$

and now β_k is given by

$$\beta_k \quad = \quad \int_{-\infty}^{\infty} n(y) v_k(y)\, dy \qquad 3.1.6$$

But Bertero and Pike show that the value of $< \beta_k^2 >$ is still given by

$$< \beta_k^2 > \quad = \varepsilon^2 \qquad 3.1.7$$

i.e. the mean square error, $< \beta_k^2 >$, is the same whether we use SVD or EVD. The reason SVD is superior is that the error is amplified by $\lambda_k^{\frac{1}{2}}$ rather than by λ_k. The value of $< \beta_k^2 >$ is important because it allows us to decide which of our measured image coefficients are badly corrupted by noise and so must be excluded from the reconstruction process.

3.1.3 Discrete analysis

The results of the previous subsection were derived using a continuous analysis. In this subsection we shall derive the same results from a discrete analysis. The discrete analysis parallels the one given in the previous subsection.

First of all, let us consider the type of device which might be used to detect an image. Let us suppose it consists of a square array of small detector elements each of which itself is square of area, d. Let us suppose the elements are contiguous so that

there are no gaps between them. For the purposes of this chapter we shall assume that these detectors can measure the complex amplitude of the image. In practice this would only be possible by using interferometric techniques but we shall ignore that in this chapter.

The output from each detector is the integral of the signal falling upon it, integrated over the area of the detector. We assume that the signal does not vary significantly over this area. Therefore, we can obtain an estimate for the image sample value at the centre of the detector, g^k, thus

$$g^k = \frac{1}{d}\int_{det.} g(y)\,dy \qquad 3.1.8$$

where we have divided the output from the detector by the area, d, of the detector. In this chapter a superscript, eg. g^k, denotes a sample value whereas a subscript denotes either a scalar product, g_k, or a particular vector, eg. u_k.

As in equns. 3.1.1 and 3.1.3 we can use the additivity of the noise to obtain

$$\begin{aligned} g^k &= \frac{1}{d}\int_{det.} \hat{g}(y)\,dy + \frac{1}{d}\int_{det.} n(y)\,dy \\ &= \hat{g}^k + n^k \end{aligned} \qquad 3.1.9$$

where $\hat{g}^k$ is the true sample value and n^k is the error due to the noise.

$$n^k = \frac{1}{d}\int_{det.} n(y)\,dy \qquad 3.1.10$$

N.B. n^k is not an estimate of the value of n(y) at the detector. We can never measure n(y) itself- only the integral of n(y) over a certain non-zero area.

By combining equns., 3.1.10 and 3.1.2 we can show

$$< n^j n^k > = \frac{\varepsilon^2}{d}\delta_{jk} \qquad 3.1.11$$

Equn. 3.1.11 is the equivalent of 3.1.2. We can now use equn. 3.1.11 to calculate $< \beta_k^2 >$ for the discrete case.

We consider first the EVD case. Just as in Bertero and Pike's analysis, the image coefficients, g_k , are given by the scalar product of the image with the eigenfunctions, u_k .

$$g_k \quad = \quad d\sum g^j u_k^j \qquad 3.1.12$$

where the u_k have been normalised thus

$$d\sum u_k^j u_k^j \quad = \quad 1 \qquad 3.1.13$$

In 3.1.12 and 3.1.13 u_k^j denotes the jth component of the kth eigenfunction. and g^j denotes the jth sample value of the image. 3.1.12 is the discrete equivalent of equn.1.5.5(d) but the integral has been replaced by a summation. The factor, d, is the equivalent of the differential, dx, in equn. 1.5.5(d).

Just as in Bertero and Pike's case we can now invoke the additivity of the noise.

$$\begin{aligned} g_k \quad &= \quad d\sum \hat{g}^j u_k^j + d\sum n^j u_k^j \\ &= \hat{g}_k + \beta_k \end{aligned} \qquad 3.1.14$$

This is the equivalent of 3.1.3. Now

$$\beta_k \quad = d\sum n^j u_k^j \qquad 3.1.15$$

By combining equns. 3.1.15 and 3.1.11 we can show that

$$< \beta_k^2 > \quad = \quad \varepsilon^2 \qquad 3.1.16$$

which is the same result as was obtained by Bertero and Pike in equn. 3.1.5. Just as in Bertero and Pike's analysis we can also show that the value of $< \beta_k^2 >$ is the same in the SVD case as in the EVD case.

We have now shown that both the discrete and continuous analyses give the same result for $< \beta_k^2 >$.

Section 3.2 Numerical Examples

3.2.1 Introduction

In this section, we consider some numerical examples of EVD and SVD. We shall consider one particular imaging situation. The imaging system will be the same which we discussed in chapter 2, i.e. the square object/circular pupil system, except we shall use a 16x16 grid rather than a 64x64. As in that chapter, our a priori information is that the object lies within a square whose corners are at the points $(\pm 1, \pm 1)$. The pupil is a circle centered on the origin of the Fourier plane and of radius π frequency units. Both the object and the pupil lie on a square grid of 16x16 sample points. So the sample spacing in the object plane is 1/8 and in the Fourier plane is $\pi/8$. The detector area, d, is therefore 1/64.

The object consists of four delta functions of height 8units placed at the points $(\pm 0.4375, \pm 0.4375)$ Thus it is symmetric about both the horizontal and vertical axes and the leading diagonal. Therefore it will be orthogonal to all the eigenfunctions except those in the SSS symmetry set. See fig. 3.1.

The image within the geometrical image region is shown in fig. 3.2. The geometrical image region is identical to the object region and so is a square with corners $(\pm 1, \pm 1)$ and also lies on a 16x16 grid. It is the sample values in this region which are used by the EVD algorithm.

A similar argument to that given in subsection 2.2.4 leads to a value for N, the total number of points in the truncation cell, of 128. Thus the entire image lies on a 128x128 grid of sample points. It is the sample values in this region which are used by the SVD algorithm.

The noise consists of random numbers, n^k, added to each image sample value. The n^k are derived from the random numbers, r^k, generated by the NAG random number generator G05CAF. The r^k are uniformly distributed over the range [0,1]. The n^k are calculated from the r^k by

$$n^k = \frac{(r^k - 0.5)}{d^{\frac{1}{2}} t} \qquad 3.2.1$$

where t is a scaling factor introduced so that the variance of the noise distribution can be changed. Thus from equn 3.1.11

$$\varepsilon^2 \quad = \quad d < (n^k)^2 > \quad = \frac{d < (r^k - 0.5)^2 >}{dt^2} \quad = \quad \frac{< (r^k - 0.5)^2 >}{t^2} \qquad 3.2.2$$

and the value of $< (r^k - 0.5)^2 >$ turns out to be 1/12. Thus

$$\varepsilon \quad = \quad \frac{1/\sqrt{12}}{t} \quad = \quad \frac{0.288}{t} \qquad 3.2.3$$

In most of the following subsections the value of t was chosen to be 100. Therefore, throughout this section I assume that we know in advance the value of ε. As we saw from equn. 3.1.16 ε^2 is the expected value of $< \beta_k^2 >$ - the mean square error in the image coefficients due to the noise. We shall use this fact in sections 3.2.5. and 3.2.6 to decide which image coefficients are badly corrupted by noise and which aren't.

3.2.2 A noise-free eigenvalue decomposition

table 3.2.2

OBJECT	IMAGE	RECON	LAMBDA
0.218580E+00	0.206526E+00	0.218580E+00	0.944854E+00
-0.153398E+00	-0.275864E-01	-0.153398E+00	0.179836E+00
0.572087E-01	0.722903E-03	0.572087E-01	0.126362E-01
-0.424099E+00	-0.260556E-03	-0.424099E+00	0.614374E-03
0.765347E-01	0.721735E-06	0.765347E-01	0.943017E-05
-0.242735E+00	-0.432934E-07	-0.242694E+00	0.178387E-06
0.320511E+00	0.232751E-07	0.320673E+00	0.725820E-07
0.165846E+00	0.158165E-08	0.995860E+00	0.158823E-08

Table 3.2.2 illustrates the eigenvalue decomposition of the object described above. The first column shows the scalar products of the first eight eigenfunctions in the SSS symmetry set with the original object. Hereafter, these will be called the "object coefficients". The second column shows the scalar products of the same eigenfunctions with the image of that object. Hereafter these will be known as the "image coefficients". The third column shows the scalar products of the same eigenfunctions with the reconstructed object as produced by the EVD algorithm. These will be

known as the "reconstruction coefficients". The final column shows the eigenvalues corresponding to each eigenfunction.

The first thing to note is that reconstruction coefficient is equal to the appropriate image coefficient divided by the appropriate eigenvalue. This is as we would expect for the EVD algorithm. The second thing to note is that the first seven reconstruction coefficients are equal to the corresponding object coefficient. This shows we have succeeded in reconstructing at least the first seven of the eigenfunctions of the original object. The reconstruction obtained with these functions is shown in fig. 3.3.

This table illustrates the internal consistency of our mathematical model and EVD algorithm. If the algorithm had not returned exactly the same reconstruction coefficients as those of the original object there would clearly be something wrong with the algorithm.

Note that there is an error between the reconstruction and object coefficients and that this increases as we go down the table. In fact the error is roughly proportional to $1/\lambda_k$. This error is due to the round-off error introduced during the computation of the eigenfunctions. See section 2.3.7. This is the reason why we use no more than seven functions even in a "noise-free" situation like this. After the seventh function the round-off error becomes too large to allow accurate reconstruction.

3.2.3 A noise-free singular value decomposition

table 3.2.3

OBJECT	FOURIER	RECON	SQRT(LAMBDA)
0.218580E+00	0.212468E+00	0.218580E+00	0.972036E+00
-0.153398E+00	-0.650514E-01	-0.153398E+00	0.424071E+00
0.572087E-01	0.643089E-02	0.572087E-01	0.112411E+00
-0.424099E+00	-0.105120E-01	-0.424099E+00	0.247866E-01
0.765347E-01	0.235027E-03	0.765347E-01	0.307086E-02
-0.242735E+00	-0.102521E-03	-0.242735E+00	0.422359E-03
0.320511E+00	0.863490E-04	0.320511E+00	0.269410E-03
0.165846E+00	0.661047E-05	0.165873E+00	0.398526E-04

The above table demonstrates a singular value decomposition of the same object as in section 3.2.2. The decomposition was carried out in the Fourier plane using

the w-functions, as described in subsection 1.5.4. Again column 1 shows the object coefficients. Column 2 shows the scalar products of the w functions with the Fourier transform of the image. These will be known as "pupil coefficients". Col. 3 shows the scalar product of the reconstruction generated by the SVD algorithm with the u functions. Col. 4 shows the singular values (which are the square root of the eigenvalues shown in table 3.2.2)

The first thing to note is that the reconstruction coefficients are equal to the pupil coefficients divide by the singular values. The second thing to note is that all eight of the reconstruction coefficients are equal to the object coefficients. And, just as in the previous case, this illustrates the internal consistency of the SVD algorithm.

The error in the reconstruction coefficients still increases as we go down the table, but it does not increase so fast as in the previous case. Here it is proportional to $1/\lambda_k^{\frac{1}{2}}$ rather than λ_k. This is why more of the reconstruction coefficients correspond to the object coefficients.

3.2.4 The additivity of noise

table 3.2.4

SIGNAL	NOISE	SIG.+NOI
0.212468E+00	-0.882588E-03	0.211585E+00
-0.650514E-01	-0.272742E-02	-0.677788E-01
0.643089E-02	-0.493974E-02	0.149115E-02
-0.105120E-01	-0.217475E-02	-0.126867E-01
0.235027E-03	-0.459366E-03	-0.224339E-03
-0.102521E-03	-0.128429E-02	-0.138682E-02
0.863490E-04	-0.794539E-03	-0.708190E-03
0.661047E-05	-0.528153E-02	-0.527492E-02

We have now illustrated the internal consistency of both the EVD and SVD. We must now turn our attention to behaviour of both algorithms in the presence of noise. The noise was generated as described in subsection 3.2.1.

Table 3.2.4 illustrates the additive property of the scalar products of image and noise. Column 1 shows the coefficients of the noise-free signal. Column 2 shows the scalar products of the u functions with the noise alone before it is added to the

image. Column 3 shows the scalar products of the u functions with the noise and image once they've been added together. As can be seen, the numbers in col.3 are the sum of those in cols. 1 and 2.

The values in col.2 represent samples from the distribution of β_k. In this case the value of t was chosen to be 100. So from equn. 3.2.3. ε = 0.288E-2. From equn.3.1.16 we should expect the root mean square (rms) value of the coefficients in col.2 to be approximately equal to ε and it can be seen that they are all of that order of magnitude. This was was the value of ε chosen for the next two subsections.

3.2.5 A noisy eigenvalue reconstruction

table 3.2.5

OBJECT	IMAGE	RECON	LAMBDA
0.218580E+00	0.209853E+00	0.214769E+00	0.944854E+00
-0.153398E+00	-0.292015E-01	-0.158270E+00	0.179836E+00
0.572087E-01	0.343510E-02	0.272798E+00	0.126362E-01
-0.424099E+00	0.325056E-02	0.529089E+01	0.614374E-03
0.765347E-01	-0.172033E-02	-0.182428E+03	0.943017E-05
-0.242735E+00	-0.129704E-02	-0.727091E+04	0.178387E-06
0.320511E+00	0.205778E-02	0.283511E+05	0.725820E-07
0.165846E+00	-0.196039E-02	-0.123432E+07	0.158823E-08

Table 3.2.5 shows a reconstruction of the same object we have already met in section 3.2.2 and 3.2.3 but this time noise has been added to the image. The object coefficients are the same as before but this time the image coefficients are the sum of the previous image coefficients and the noise coefficients. The reconstruction coefficients are shown in col. 3.

Comparing the reconstruction coefficients with the object coefficients, we can see that they do not correspond as closely as they did in the noise-free case. The most obvious point is that the error increases dramatically as we go down the table. The error in the first coefficient is approximately 0.01 of the object coefficient but in the eighth coefficient it is approximately 1.0E7 times the object coefficient. Thus only about the first two reconstruction coefficients bear any resemblance to the object coefficients.

In an experimental situation we would not know what the object coefficients were, so how could we tell which of the reconstruction coefficients were valid? This is where our knowledge that $< \beta_k^2 >$ is equal to ε^2 is useful. In this case we know that ε =0.288E-2. Now, any image coefficients which are at approximately this level are bound to be seriously affected by noise, and we can see from col. 2 that all the coefficients lower than eigenvalue 2 are at about his level, therefore we must discard them. But coefficient 1 is 100 times larger than ε and coefficient 2 is 10 times larger. Therefore, these two are likely to be reasonably accurate. We can therefore include them in our summation. The reconstruction using these coefficients is shown in fig. 3.7.

3.2.6 A noisy singular value reconstruction

table 3.2.6

OBJECT	FOURIER	RECON	SQRT(LAMBDA)
0.218580E+00	0.211585E+00	0.217671E+00	0.972036E+00
-0.153398E+00	-0.677788E-01	-0.159829E+00	0.424071E+00
0.572087E-01	0.149115E-02	0.132652E-01	0.112411E+00
-0.424099E+00	-0.126867E-01	-0.511838E+00	0.247866E-01
0.765347E-01	-0.224339E-03	-0.730543E-01	0.307086E-02
-0.242735E+00	-0.138682E-02	-0.328350E+01	0.422359E-03
0.320511E+00	-0.708190E-03	-0.262867E+01	0.269410E-03
0.165846E+00	-0.527492E-02	-0.132361E+03	0.398526E-04

Table 3.2.6 shows a reconstruction of the same object as the one we have met in previous sections. But this time it is reconstructed using a singular value decomposition. The noise level was the same as in the eigenvalue reconstruction in section 3.2.5.

Just as we did in the previous section, let us compare the object coefficients with the reconstruction coefficients. Again we see that the error increases as we go down the table but it does not increase so fast as in the eigenvalue case. Here the error in the eighth coefficient is only about 1.0E+3 times the object coefficient as opposed to 1.0E7 times in the previous case. Here, eigenfunctions 1, 2 and 4 are usable as opposed to only 1 and 2 in the previous case. We can tell they are usable by looking

at the pupil coefficients in col.2. Any coefficients which are large compared with ε are usable. In this case ε is still 0.288E-2.

It may seem odd that reconstruction coefficient no.3 is not usable whereas no.4 is, despite the fact that no.4 has a smaller singular value. We can see the reason if we look at col.1. Object coefficient no.3 just happens to be very small compared to the others, whereas no.4 is large compared to the others. Therefore, no.4 is less severely affected by noise despite its smaller eigenvalue. We can exclude no.3 from the summation but include no.4. (The exclusion of no.3 doesn't actually make much difference to our reconstruction since its contribution is so small anyway.) The reconstruction using these coefficients is shown in fig. 3.5.

We can see by comparing the results in table 3.2.6 with those in 3.2.5 that at least in this case SVD is superior to EVD. This is essentially because the pupil coefficients in the case of SVD are higher than the corresponding image coefficients in the case of EVD. The noise coefficients are the same in both cases. The higher values in SVD means that they are less severely affected by the noise.

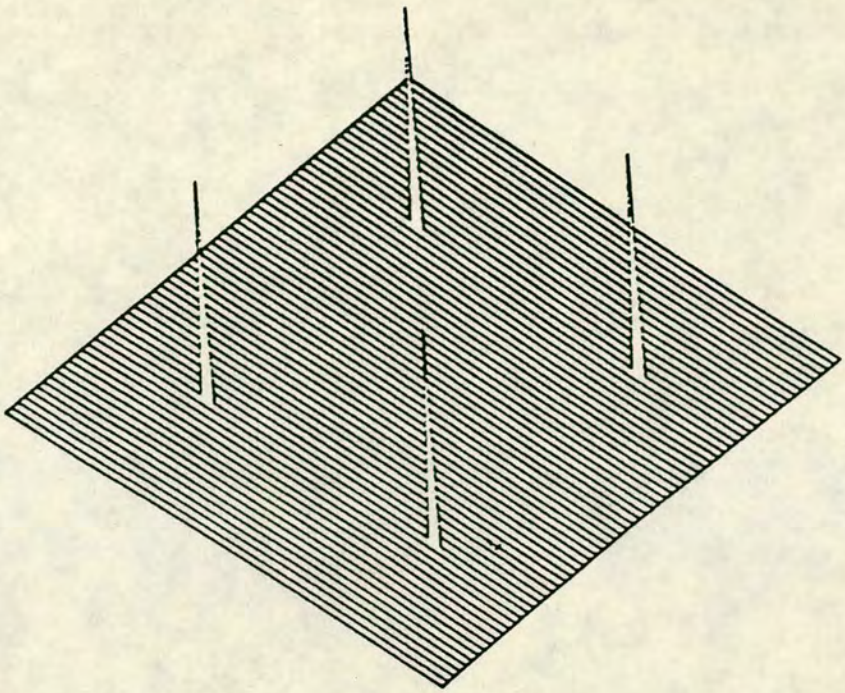


fig 3.1 The original object

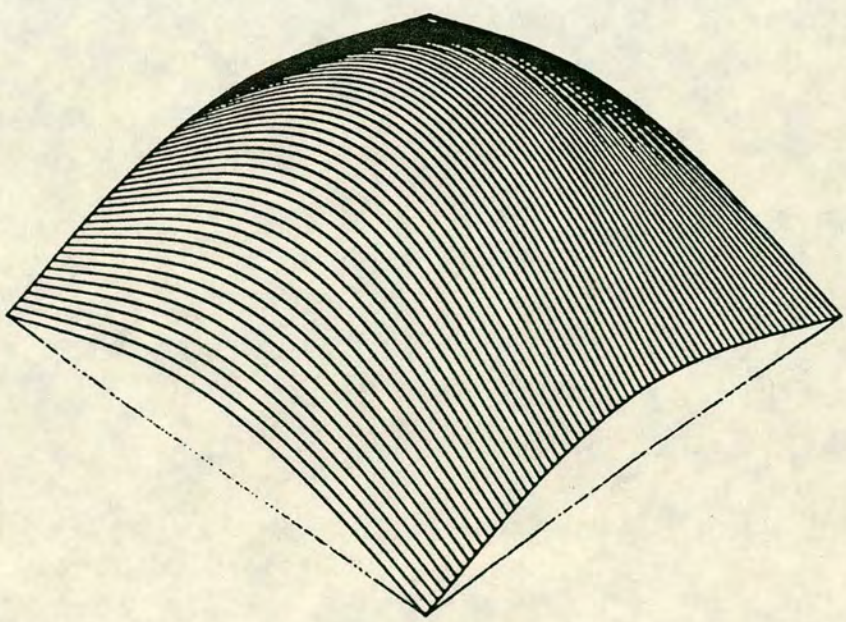


fig 3.2 The bandlimited image within the geometrical image region

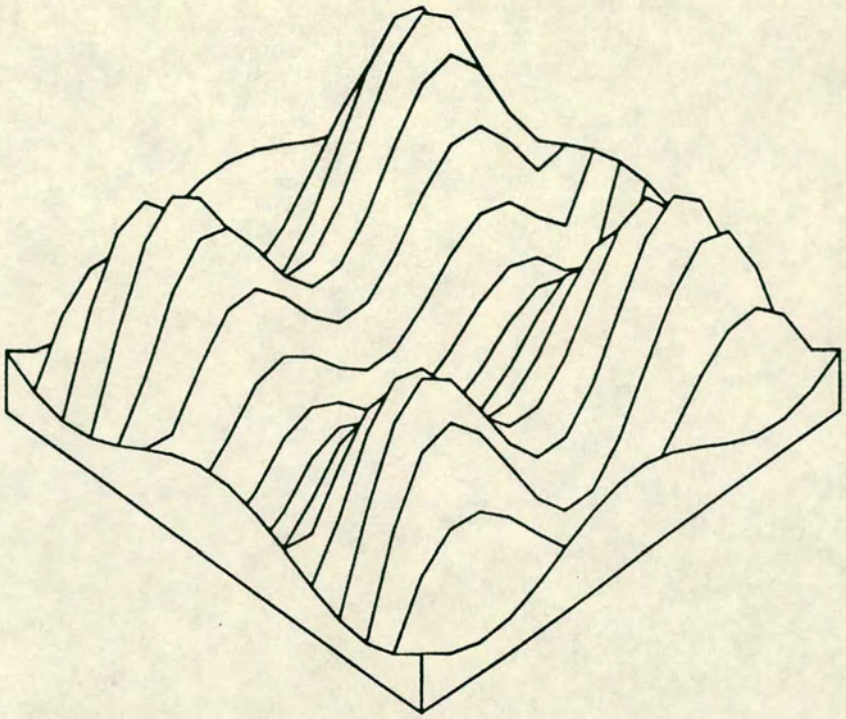


fig 3.3 The reconstruction with 7 singular functions

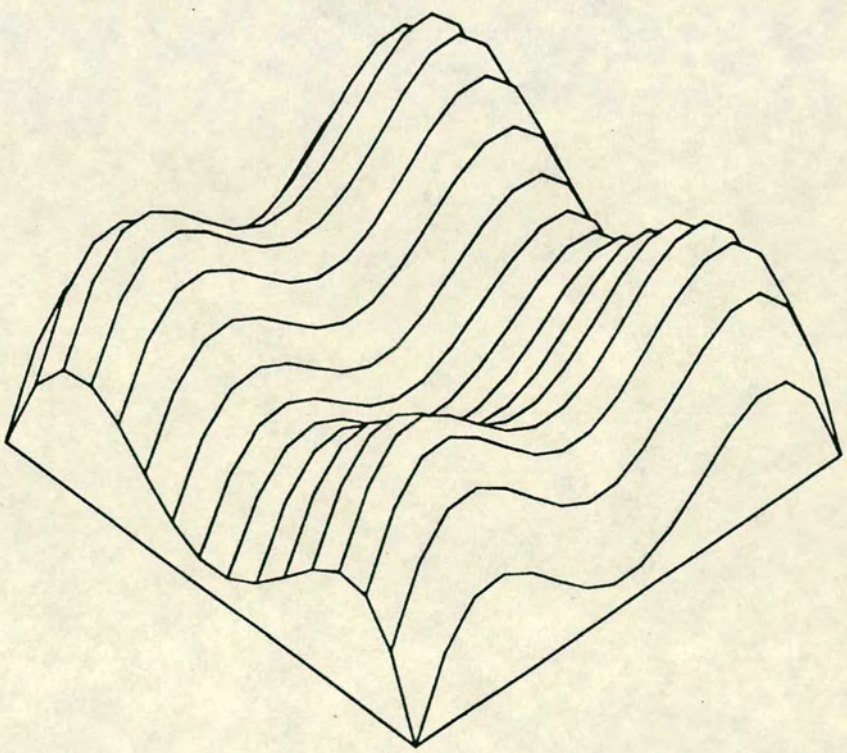


fig 3.4 The reconstruction with 5 singular functions

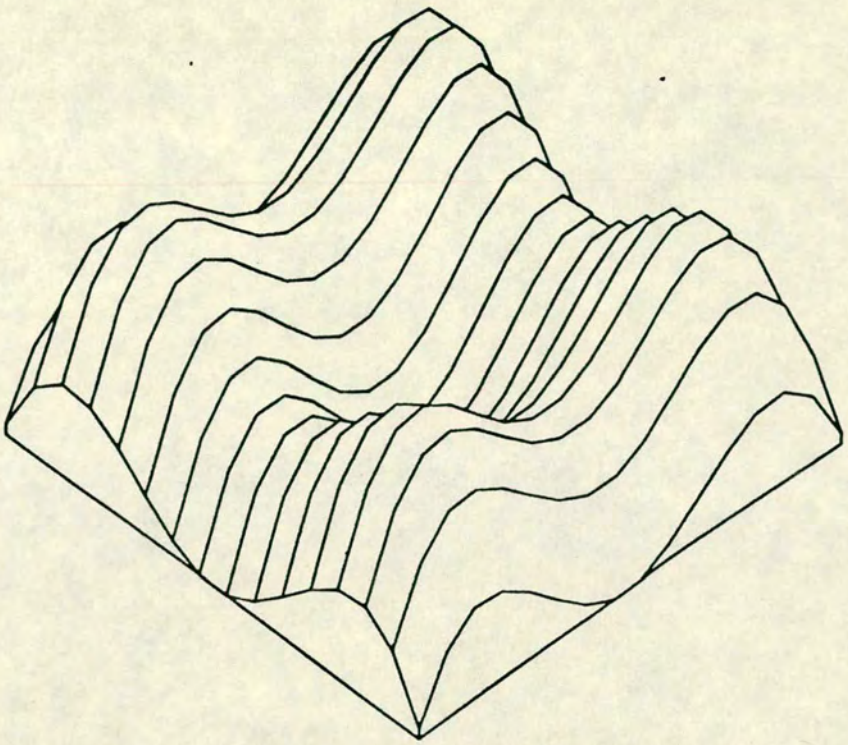


fig 3.5 The reconstruction with 4 singular functions

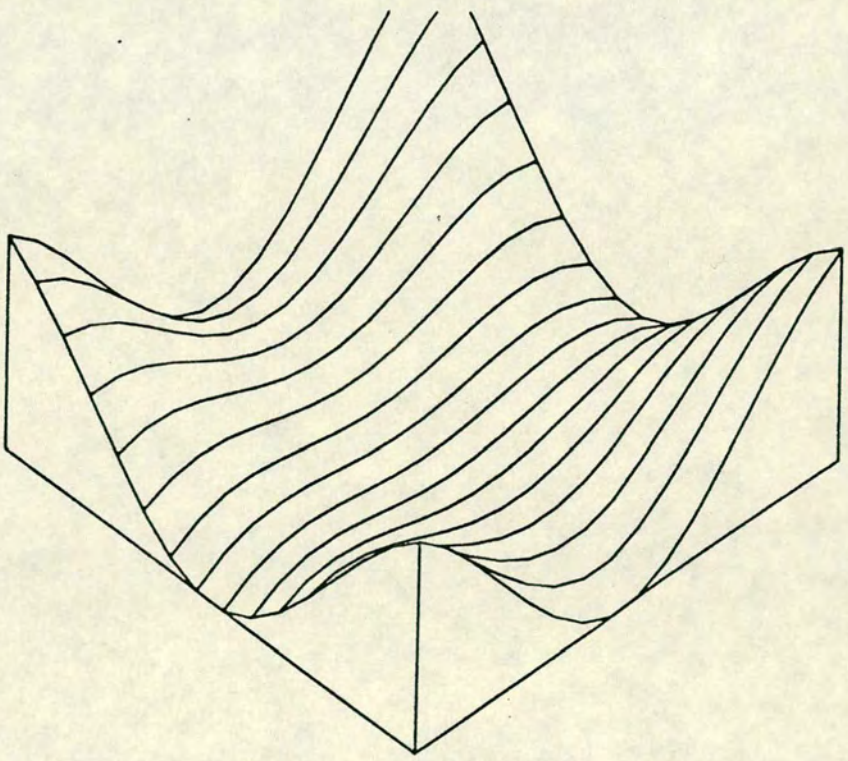


fig 3.6 The reconstruction with 3 singular functions

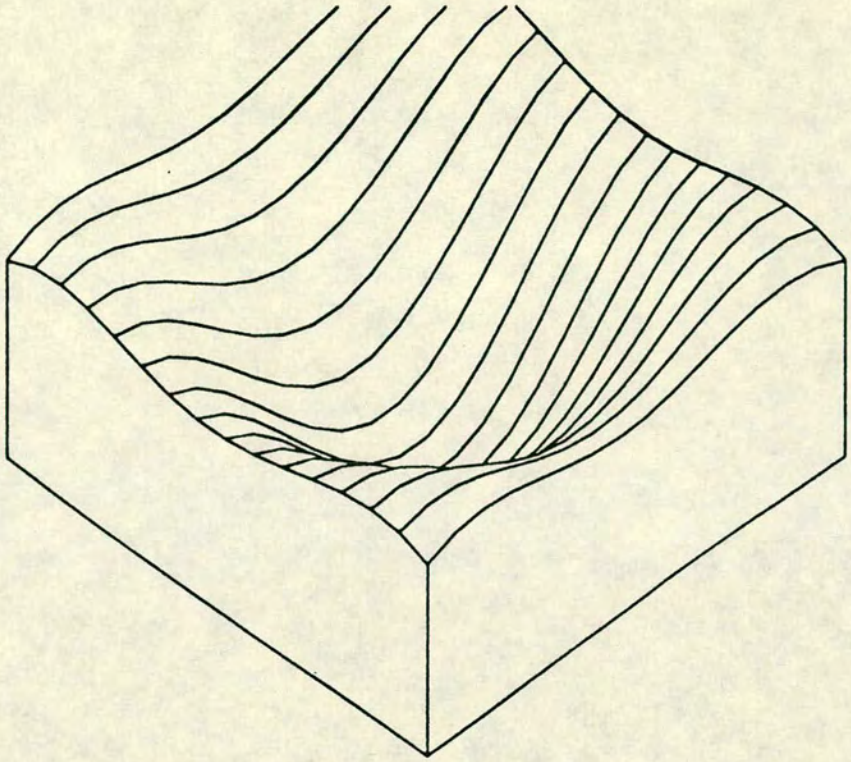


fig 3.7 The reconstruction with 2 singular functions

Chapter Four

Experimental Work

Section 4.1 Introduction

The computer simulations carried out in the previous Chapter tested the internal consistency of our mathematical model and the SVD algorithm. They did not, however, test consistency with the external, real world. To do this, we must apply the algorithm to images from a real optical system and compare the reconstructed objects with the original objects. The technique is the same as in the previous chapter but we are now dealing with images generated by an optical system rather than a computer subroutine.

In the first part of this chapter we discuss how the optical images were generated. We explain why incoherent rather than coherent illumination was used and discuss the form of the incoherent transfer function. We describe the optical bench used to generate the images and the equipment used to digitise them and store them in the computer. Finally, we discuss the distortion introduced by this equipment and the methods used to remove it.

In the second part of this chapter we give some examples of degraded images and reconstructed objects. We compare these with the original objects and discuss how far the algorithm has succeeded.

Section 4.2 The Optical System

4.2.1 Incoherent versus coherent illumination

The theories of incoherent and coherent imaging both follow the lines laid down in Chapter 1 for the Fourier theory of imaging. The main difference is that in the incoherent case the functions, f(x) and g(y), represent intensity distributions whereas, in the coherent case, they represent amplitude distributions. The intensity is the modulus squared of the amplitude. This difference is crucial when it comes to carrying

out experiments in the real world, for, while it is easy to measure an intensity distribution directly, it is impossible to measure an amplitude distribution without the aid of interferometry. Therefore incoherent images can be adequately measured using a camera whereas coherent images require much more elaborate equipment. It is for this reason that I chose to carry out the experiments with incoherent illumination.

4.2.2 The incoherent transfer function

For a given optical system the transfer function, $H_I(k)$, under incoherent illumination is the auto-correlation of the transfer function, $H_C(k)$, under coherent illumination. Fig. 1.4 shows the incoherent transfer function for a one-dimensional diffraction-limited system. As was explained in section 1.6.5 we have a choice as to how we treat an incoherent SVD. We can either use Frieden's method or Bertero and Pike's method. In the appendix I show that a numerical simulation indicates that the latter method is better. In this chapter I use both and compare the results.

Frieden's method requires us to deconvolve incoherent images before we can apply the SVD algorithm. We can do this, as described in Chapter 1, by dividing G(k), the Fourier transform of the incoherent image, by the transfer function, $H_I(k)$.

$$F(k) \quad = \quad \frac{G(k)}{H_I(k)}$$

where F(k) is the segment of the Fourier transform of the object within the pass-band.

In theory, this should allow us to estimate F(k) on the interval $[-2\Omega, 2\Omega]$. However, please note that as k approaches $\pm 2\Omega$, H(k) becomes $<< 1$, so that division by H(k) becomes unstable. In practice we must impose a cut-off point at a value of k somewhat less than 2Ω. So, in fact, we can only know F(k) on an interval shorter than $[-2\Omega, 2\Omega]$.

Of course, it would be a mistake to assume without evidence that our optical system is diffraction-limited. It is always necessary to measure the transfer function as accurately as possible. Even in cases where the system is not diffraction-limited it should still be possible to deconvolve the images provided the transfer function is

known. We must bear in mind that the function may be complex if the system is aberrated.

4.2.3 The Optical System

The ultimate purpose of work on super-resolution is that it should be applied to "useful" optical systems such as microscopes and telescopes. However, before we can go on to these more complex systems we must test our algorithm on a much simpler system, so that, if anything goes wrong, the cause can be traced more easily. Such a system is shown below.

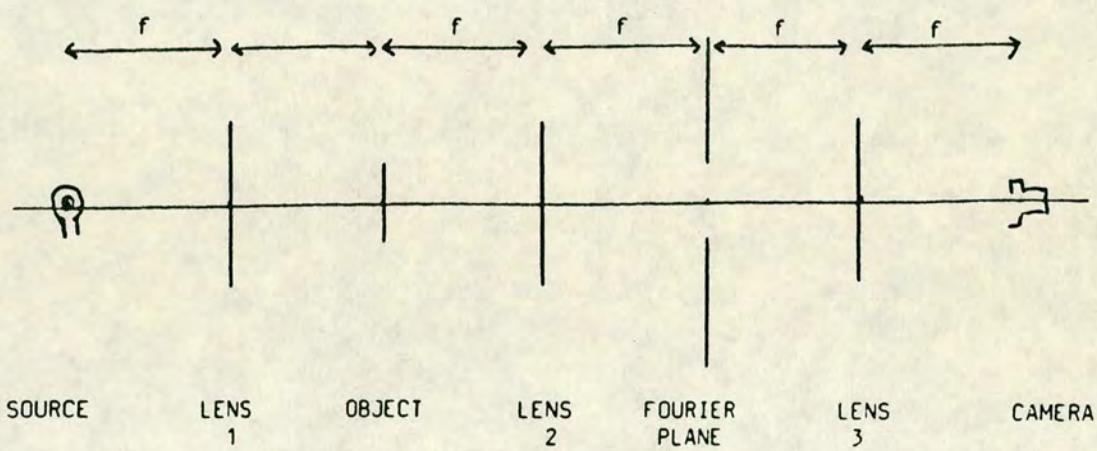


FIG 4.1 The 6f optical system

This arrangement is very common in the optics world and is known as a "6f" optical bench. Its virtue is that, while it is very simple, it can be used to model or simulate the behaviour of a large variety of more complex systems. In fact, in theory, it should be capable of simulating any LSI system.

The reason for this generality is that the bench acts as a physical analogue of the mathematical model of LSI imaging given in Chapter 1. There we saw that imaging could be broken down into three stages:

1) Fourier transformation of the object

2) Multiplication of the Fourier transform by the transfer function, H(k), to give G(k)

3)Inverse Fourier transformation of G(k) to give the image, g(y).

Each of these three stages is represented on the 6f optical bench:

1) Lens 2 Fourier transforms the object

2) A filter in the Fourier plane performs the multiplication by H(k) to give G(k)

3) Lens 3 Fourier transforms G(k) to give the image on the detector plane of the camera.

As we saw in Chapter 1, the transfer function, H(k) completely characterises a given optical system. Therefore, our ability to simulate any particular system depends on our ability to reproduce the appropriate form of H(k) as a filter. This is easy for diffraction-limited systems, since, in that case, the appropriate filter is an aperture in an opaque screen. However, it becomes much more difficult when H(k) is complex. In this experiment we shall restrict our attention to diffraction-limited systems.

It is the generality of the 6f bench that allows us to hope that if the SVD algorithm works for it then it will also work for other optical systems such as microscopes and telescopes.

4.2.4 Description of the Optical Equipment

All the lenses, mirrors, filters etc were held in adjustable saddles mounted on an optical bench. The lenses were 7.5 cm diameter achromatic doublets with 1m focal length. The overall length of the bench was thus 6m.

The incoherent source was a Hg arc lamp. This emits several spectral lines, but a colour filter was placed in front of it so that only one of these lines is transmitted. The source is thus a good approximation to mono-chromatic light. Neutral density filters of different transmissivities could be inserted to reduce the intensity of the light. The objects were 35 mm photographic transparencies made from lith film. Lith film has a very large contrast ratio; i.e. the transmissivity at any point is either 1 or 0. Thus, it behaves like an aperture cut into metal sheet. Thus, all the objects used were binary objects.

All the objects were designed to lie within a square of side 1mm. This is the a

priori information that we build into our algorithm. It has been assumed throughout this thesis that we know the spatial region within which the object lies. It is at this point, that this assumption is incorporated into the experiment.

The other piece of information necessary for the algorithm is the bandwidth of the system. This is determined by the width of the filter in the Fourier plane of the optical bench. We have already said that we are going to restrict our attention to diffraction-limited systems. This implies that the filter should be an aperture i.e. a binary function. In this case, a rectangular aperture was used with horizontal and vertical widths which were independently adjustable. Thus, we could have different bandwidths in the horizontal and vertical directions. It is the width of the aperture which determines the variable Ω in figure 1.4.

$$\Omega = \frac{1}{2}\frac{2\pi}{\lambda}\frac{h}{f} \qquad 4.2.1$$

where h is the width of the aperture. The value of h is approximately 1mm.

We can now give an approximate value of the space-bandwidth product, c.

$$c = X\Omega \qquad 4.2.2$$

where X is the half-width of the object. As we have seen, X = 0.5 mm. If we substitute values λ = 550 nm and f = 1000 mm we obtain, $c = 3.5$. We must double this value because we are working with incoherent light. Thus, $c = 7$, Therefore, we are working in the region $c < 10$ where super-resolution has been seen to be most useful.

We should note here that the exact position of the aperture in the Fourier plane does not matter. This is because positional information is lost when we take the auto-correlation of the pupil function to form H(k). This is in contrast to the coherent case where the position of the aperture relative to the optic axis would have had to have been accurately known.

4.2.5 The camera and framestore

The image is detected by an array camera. An array camera ought to give better response than the more common vidicon camera. The detector elements are CCDs which give out a signal directly proportional to the intensity of the light incident upon them. Their response is linear whereas that of a vidicon is not. Furthermore, a vidicon suffers from spatial distortion of the image near the edges whereas an array camera ought not to. However, the array camera does have the drawback that it has a threshold intensity below which it will not respond and a maximum intensity above which it will give out a constant signal. This is why the neutral density filters are required;- to reduce the intensity of the image below the camera's maximum.

The signal from the CCDs is read out in a horizontal raster scan and converted into a standard video signal. This signal is read by a framestore which acts as an interface device between the camera and the VAX computer. The framestore contains an analogue-to-digital converter which samples each line of the video signal and converts it into an 8-bit digital signal (i.e. on a scale of 256 grey levels). On this scale the camera's threshold intensity is 20-30 grey levels and its maximum intensity is 256 grey levels.

There are 236 sample points per horizontal line. N.B. these do not correspond to the positions of the CCD elements in the camera - their positions are determined by the speed of the A-D converter in the framestore. On the other hand, the vertical sample spacing is determined by the spacing of the lines in the detector array. The horizontal and vertical sample spacings are not the same; respectively they are

$$dH = 0.036mm \qquad dV = 0.027mm$$

The framestore receives commands from the VAX entered by a user sitting at a VAX terminal. The software for communicating with the framestore is contained in a FORTRAN program called "FRM" written by Dr N.E. Fancey. The program is run interactively and sends commands, which have been entered at the terminal, down

an output channel to the framestore. On the command "CTM" the framestore will display the continuously varying video signal on a TV monitor screen; the sample values are continuously updated as the video signal comes in from the camera. Once the user has achieved the desired image on the monitor screen, he can send the command "FGR". This causes the framestore to store the current set of 244 x 236 sample values. This operation is known as "grabbing a frame". The set of sample values is embedded in a slightly larger 256 x 256 array. The extra array elements are usually set to zero but can sometimes contain random values. I shall call this 256 x 256 array a "frame". On the command "RFS" this frame is sent down a serial channel to the VAX where it is stored as a 256 x 256 array of CHARACTER*1 variables in the program FRM.

As well as communication software, FRM also contains numerical routines such as Fast Fourier Transforms (FFTs) so that the frame can be processed once it has been received by the VAX. It is also possible to send a frame from the VAX to the framestore. Thus, we can create artificial frames on the VAX and send them to the framestore to be displayed on the monitor screen. FRM also allows us to write a frame out to a datafile on the VAX so that it can be stored permanently. The frame can then be read back in at any subsequent time and redisplayed on the monitor screen.

Together the camera, framestore, monitor and the VAX program FRM form a complete system for acquiring, storing, manipulating and displaying images.

4.2.6 Background Subtraction

The frames produced by the camera-framestore system are not ready to be put into the super-resolution algorithm. They contain certain unwanted features which must be removed. The first of these is the background illumination which has been deliberately introduced to raise the image above the camera's threshold level. This is uniform across the frame. The second set of unwanted features is a number of spurious electronic signals which are added to the frame as it passes through the

system. An example of such a signal is shown in fig. 4.2, which is a horizontal cross-section of a frame which has been exposed to uniform background illumination. Although the illumination is uniform, the camera's response is clearly not. A ripple of amplitude ~ 5 grey-levels and period ~ 4 pixels has been added by the camera or the framestore. By looking at other rows from the same frame we find that the ripple is constant from one row to the other.

If we look at a column from the same frame we see that here a signal has also been added but that this is not of such a high frequency. See fig. 4.3. Again we find by looking at other columns that the signal is constant from one column to another.

In fact, if we look at the 2D Fourier transform of the frame we see that most of the signal is concentrated on the horizontal and vertical axes. In other words, the unwanted signals can be separated into horizontal and vertical components. We use this fact in removing these signals from the frame.

Let us make the assumption that the unwanted signals are merely additive and do not distort the real signals in any other way. Then, if we can estimate the unwanted signals, we can simply subtract them off leaving only the real signal. In order to estimate the horizontal component, X(x), we take the top 40 rows and the bottom 40 rows and average the 80 values in each column. Likewise we estimate the vertical component, Y(y), by averaging the left-hand 40 columns and the right-hand 40 columns. By multiplying these two components together, X(x)Y(y), we can obtain a good estimate of the unwanted signal at each point in the frame. We can then subtract it off. The program which carries this process out is called BACSUB.

For this to work, we must ensure that the image does not fall into any of the rows or columns which are used to estimate the unwanted signal. This is easy to achieve in practice since all our images tend to fall within the central 40x40 pixels.

Figs. 4.4 and 4.5 show a cross-sections across a frame containing an image before and after the unwanted signals have been subtracted. You can see that the second cross-section is much smoother than the first as we would expect.

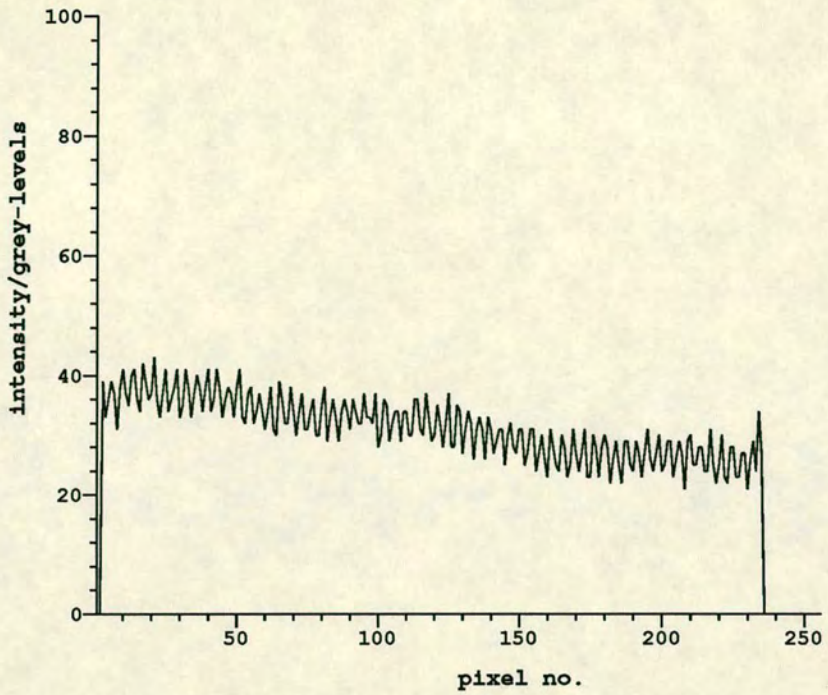


Fig 4.2

A cross-section across a frame exposed to uniform illumination

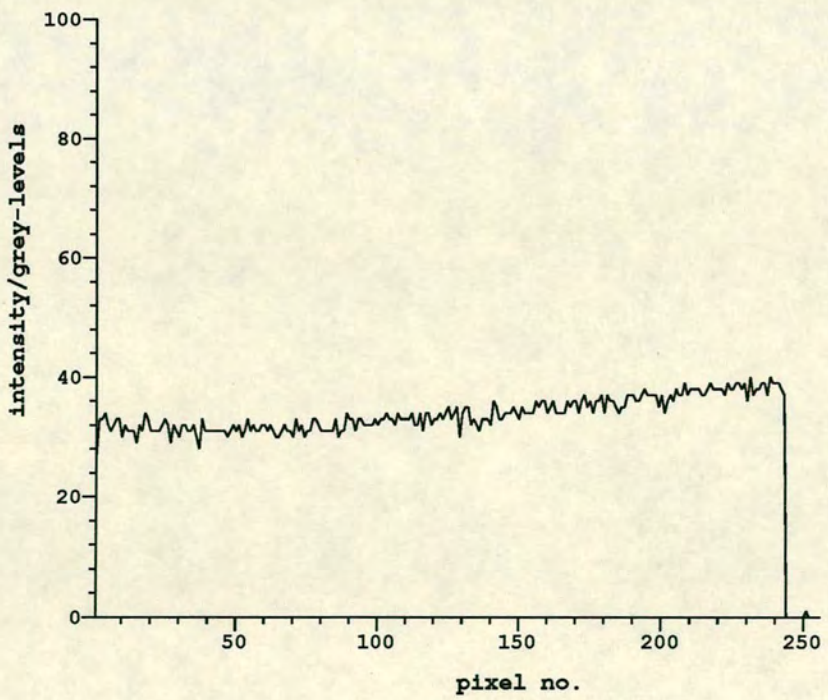


Fig 4.3

A vertical cross-section from the frame shown in Fig. 4.2

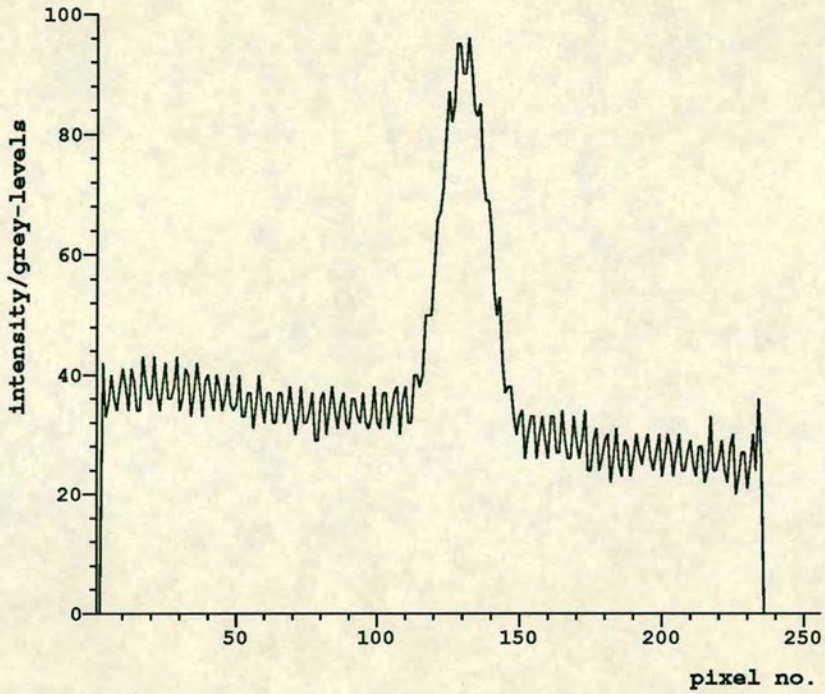


Fig 4.4

A frame containing an image before the unwanted signals have been subtracted

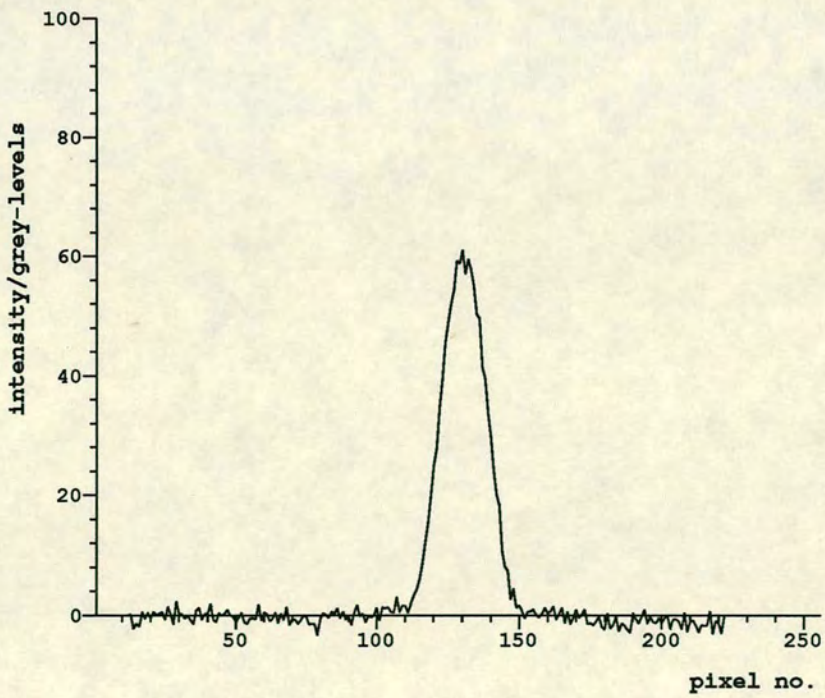


Fig 4.5

The same frame as in fig 4.4 after the unwanted signals have been subtracted

It is possible that this process may result in negative values at some pixels in the final frame. This is, of course, not allowed since the intensity must be always ≥ 0. But I tend to leave these negative values in and treat them as being due to noise.

The noise can be further reduced in the case of blurred images. To do this we make use of two facts about blurred images. The first is that in all cases the blurred image falls to zero within a 40x40 pixel region in the centre of the frame. Of course, in principle, a band-limited image extends out to infinity in all directions, but, in practice, it will tend to fall very quickly below the minimum intensity detectable by most detectors.

The second fact is that they are band-limited i.e. the Fourier transform of the image is zero outside a certain known area: usually a 30x30 square in the centre of the Fourier plane.

We can use these two facts in the following way. First we set to zero all the pixels outside the 40x40 region in the centre of the image frame, because we know that any data outside that region must be noise. Then we Fourier transform the frame using a 256x256 FFT. Now, we set to zero any data outside the central 30x30 region of the Fourier plane because again we know that any data here must be noise. Finally we inverse transform back to the image plane. This ought to have removed a substantial part of the noise.

The above procedure is not applicable to unblurred images because they are not band-limited.

Fig. 4.6 shows the result of applying the noise-reduction procedure to the same frame as shown in figs. 4.4 and 4.5.

We can test this procedure by taking two different frames of the same blurred image grabbed at different times. Then we compare them to see how similar they are after applying BACSUB and the noise-reduction procedure.

We can compare them in two ways. One way is to work out the complex

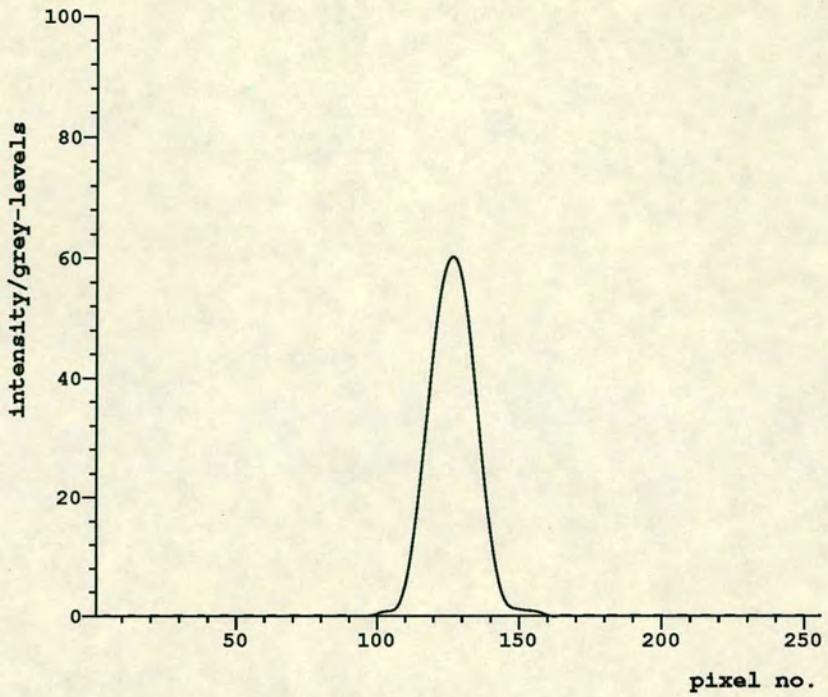


Fig. 4.6

The same frame as in fig. 4.4 after the noise-reduction procedure

correlation coefficient, ρ, defined by

$$\rho = \frac{\sum_i U_i^* W_i^*}{\sqrt{(\sum_i |U_i|^2)(\sum_j |W_j|^2)}}$$

where U_i and W_i represent the ith pixels of the two images respectively. ρ lies in the range $[-1.0, 1.0]$ with value 1.0 indicating very high similarity and 0.0 indicating no similarity.

The second method is to calculate, C, the square root of the sum of squares of the differences as follows

$$C = \sqrt{(\sum_i |U_i - W_i|^2)}$$

However, we must be careful in interpreting C because there may be a scaling factor difference between the two images due to fluctuations in the intensity in the arc lamp. To remove this difficulty we should first scale both images so that their norms are

the same i.e.

$$\sum_i |U_i|^2 \quad = \quad \sum_j |W_j|^2$$

It was found for a typical image, the values of ρ and C for two different frames after using BACSUB were

$$\rho \quad = \quad 0.9995587 \quad C \quad = \quad 81.5$$

Then, after the noise-reduction procedure was applied to both frames, the values became

$$\rho \quad = \quad 0.9999779 \quad C \quad = \quad 18.1$$

and furthermore, the largest difference at any pixel of the frame was 0.99. i.e. the intensity values are accurate to better than one grey-level. This is the best we could hope for from such a camera system. The above values were computed over a 60x60 region in the centre of the frame.

In conclusion, we can say that the random error on each pixel is less than one grey-level and so is less than the digitisation noise caused by sorting the data into grey-levels in the first place. Thus we can treat the noise on the data as being Gaussian with variance of ~ 0.5 grey-level.

4.2.7 Distortion in the camera

However, although the Gaussian random error may be small, there may still be some distortion introduced by the camera, i.e. a systematic error. There is some evidence for this. See figs 4.7 and 4.8. Fig. 4.7 shows a horizontal cross-section across two different frames. Fig. 4.8 shows a vertical cross-section across the same two frames. Both frames have been passed through the noise removal process described in the previous section. Both frames contain blurred images of the same object, the only difference between them is that the number of neutral density filters placed in the path of the light was different in each case. The images were then scaled to have the same norm each.

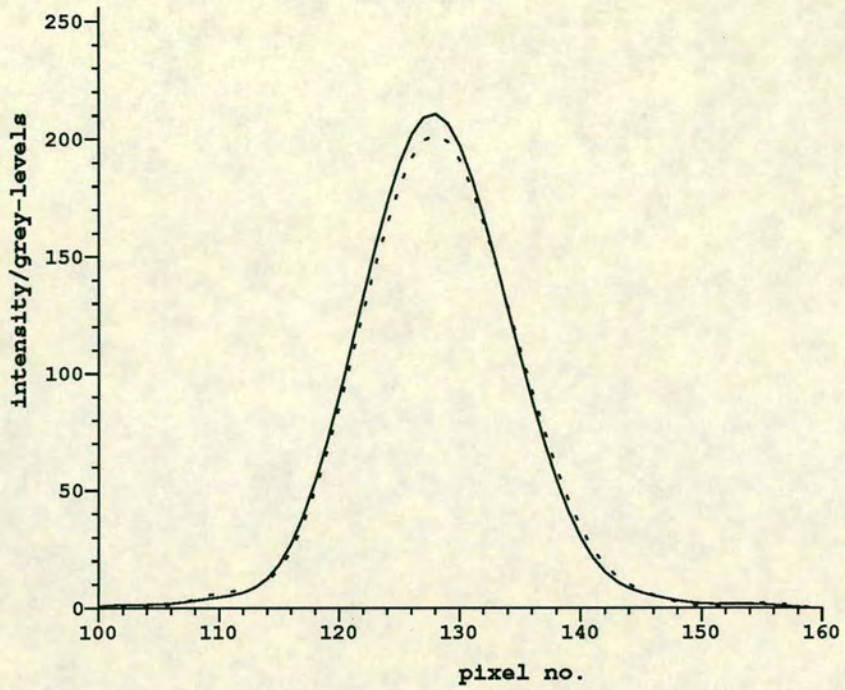


Fig 4.7

A horizontal cross-section across two frames after the noise-removal procedure

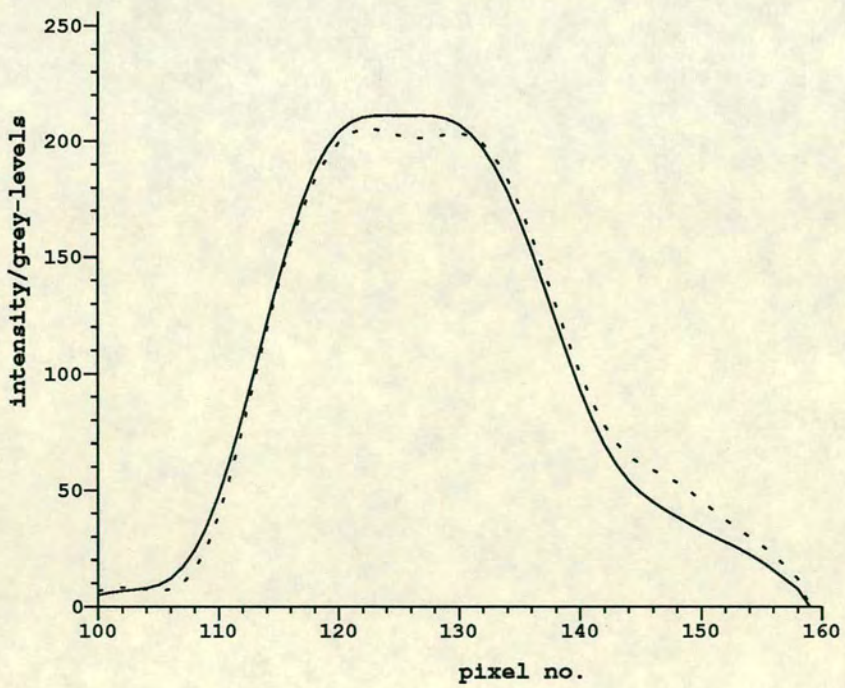


Fig 4.8

A vertical cross-section across the same frames as in fig. 4.7

Now, in the absence of a systematic error changing the number of neutral density filters should merely result in an identical scaling of the intensity at every pixel in the image. Thus, after the norms of the images have been set to the same value, the images should be identical, but from figs. 4.7 and 4.8 they clearly are not.

The dotted image was grabbed at the lower intensity. It seems to have spread vertically over a larger area than the other image. The explanation might be that the electronics in the system responds differently to the different rise times involved in the two images.

The distortion remains constant from one frame to another. So it cannot be treated as random noise. It is not removed by the noise-removal procedure.

There is some evidence that low intensity images are better than high intensity images. Let us take two pairs of images of the same object. Of each pair one image is blurred and the other unblurred. One pair is taken at high intensity (say ~ 200 grey-levels) and the other at low intensity (~ 100 grey-levels). We Fourier transform all four images. If we multiply the Fourier transform of the unblurred images by the OTF, H(k), and calculate the correlation coefficient with the Fourier transform of the corresponding blurred image of the same pair, we find that the low-intensity pair always gives a higher correlation coefficient. In the low-intensity cases $\rho > 0.99$ whereas in the high-intensity cases $0.9 < \rho < 0.99$.

I cannot think of any way of removing this distortion other than to grab images at those intensities which give the best results. This distortion is likely to have a much more adverse effect on the SVD algorithm than the random noise which we discussed earlier. The random noise introduced an error of at most one grey-level. From figs. 4.7-8 we can see that the distortion introduces errors of $\sim 4 - 5$ grey-levels.

4.2.8 Measuring the Transfer Function

As we have already said, it is always necessary to measure the transfer function H(k), to test for any possible aberrations. These may be caused by imperfections in the lenses or misalignment of the optical system. One can never assume in advance

that the optical system is perfectly diffraction-limited. In this section, we discuss the particular method that we used to measure the transfer function and quote the results. We discuss possible drawbacks to this method and suggest improvements which would increase the accuracy but are more time-consuming.

As we have seen in Chapter 1 the transfer function, H(k), is defined by

$$H(k) \quad = \quad \frac{G(k)}{F(k)} \qquad 4.2.3$$

where G(k) is the Fourier transform of the image g(y) and F(k) is the Fourier transform of the object, f(x). This equation suggests that the most obvious way to measure H(k) is to find some object, whose transmission function, f(x), is known, and measure its image, g(y). We can then compute F(k) and G(k) using an FFT and finally we can find H(k) from equation 4.2.3.

However, the problem is that we must find an object whose Fourier transform, F(k), does not have zeros within the pass-band of the system, otherwise equation 4.2.3 will become unstable. In theory, we can always find such an object: e.g. we can select a circular aperture whose diameter is small enough that the first zero of its Fourier transform will lie outside the pass-band. The radius of the first zero is inversely proportional to the radius of the object. So, the smaller the object the further out this zero will be. This is actually the method adopted in this thesis. But, it has a drawback:- the smaller the spatial extent of the image, the fewer pixels are used to detect it and so its shape is less accurately determined.

We could minimise this drawback to a certain extent by using a set of much larger objects which, because they are larger, are more accurately measured, and, even though they may have zeros within the passband, we can choose the objects so that their zeros do not lie at the same places. Thus, when we take together all the measurements from the whole set, we can derive a much more accurate set of values for the transfer function. However, this is a much more time-consuming process than the previous method. Unfortunately, I did not have time to carry out the more time-consuming procedure.

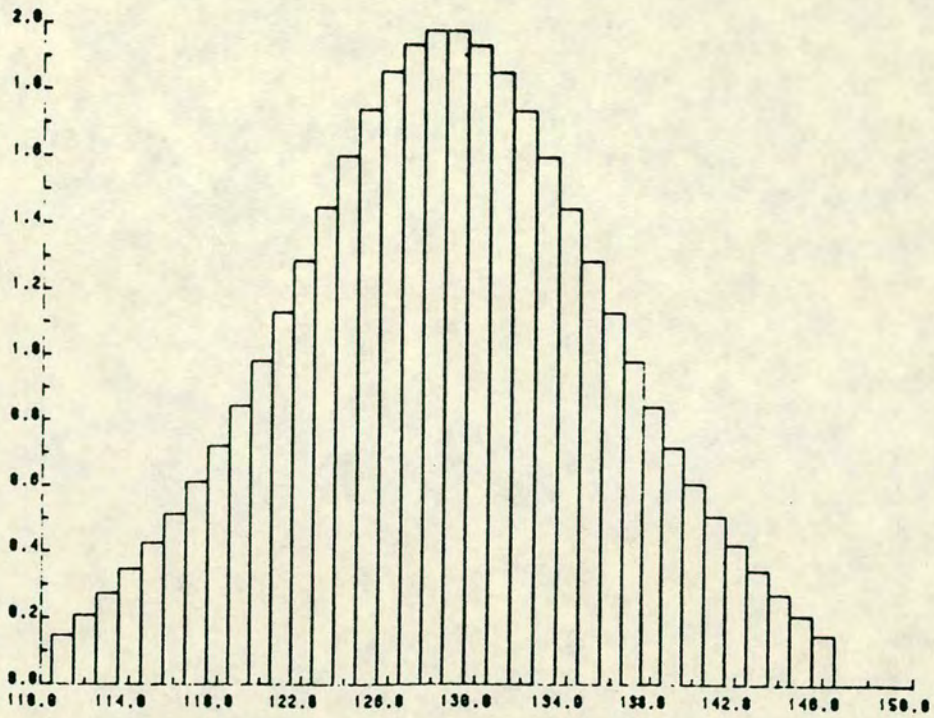


fig 4.9 The real part of the transfer function

Figure 4.9 shows the real part of H(k) as measured. The imaginary part is small compared to the real part but it is not negligible. Therefore, the system is aberrated. However, this presents no serious problem provided we can measure H(k) accurately enough. We should always be able to recover F(k) within the pass-band by using equation 4.2.3.

4.3 Comparison of original and reconstructed objects

4.3.1 Aquiring the images

In this section we take our experimental image and feed it into the SVD algorithm. We then compare the reconstructed object with the original object in the same way as we did in the previous chapter where we dealt with computer-generated "objects" and "images". Just as then, we are going to compare the singular coefficients of the original object with the singular coefficients of the reconstructed object. We will then discuss how far the SVD algorithm has succeeded and the sources of any

error.

Let us now look at a typical object and its band-limited image. Figures 4.10 and 4.11 show grey level plots of such an object: in this case a letter "A" (made on Lith film) and its band-limited image. Figure 4.10 was obtained by imaging the original object onto the detector array of the camera without the aperture in the Fourier plane. We thus obtain an image which is a reasonably close approximation to the original object. There will certainly be some degradation since the optical system is not perfect (as we saw when we measured the transfer function ; there is a certain amount of aberration) but this degradation is insignificant compared with the degradation we are about deliberately to introduce by inserting the aperture into the Fourier plane. It is the frame shown in Figure 4.10 which we use as our "original object".

While frame 4.10 was being grabbed certain neutral density filters were inserted into the optical system to reduce the intensity of the image below the maximum intensity of the camera. When the aperture is inserted into the Fourier plane it excludes a large percentage of the light and so the image is a great deal less intense. Therefore, some of the neutral density filters are removed to increase the intensity of the image. N.B. Exactly the same operation was carried out when the transfer function, H(k), was measured in the previous section. Exactly the same neutral density filters were inserted and removed in both cases. Thus, there is no necessity to scale our estimate of F(k), as there would have been if the neutral density filters had been different in the two cases.

Figure 4.11 shows the image we obtain once the aperture has been inserted into the Fourier plane and the neutral density filters have been removed. This image was Fourier transformed using a 256 FFT routine to obtain G(k), the Fourier transform within the passband.

In order to carry out SVD by Frieden's method, i.e. using the coherent singular functions, we must estimate F(k) within the pass-band by dividing G(k) by the transfer function. This operation is carried out on the VAX computer.

The files containing the original object, G(k), F(k) and H(k) are then transferred to EMAS. H(k) is used to compute the incoherent singular functions as described in section 2.3.9. F(k) is used as the input to the coherent SVD and G(k) is used as the input to the incoherent SVD.

Figs. 4.12-14 show the reconstructions of object "A" using coherent singular functions with singular values above 0.1, 0.05 and 0.01 respectively. Figs. 4.15-17 and show the reconstructions of this object using incoherent singular functions with singular values above 0.1, 0.05 and 0.01 respectively.

Figs. 4.18 and 4.19 show another object and its image, in this case a letter "E". The reconstructions for this object are shown in figs. 4.20-25.

Some of the reconstructions have clearly been badly affected by the distortion, particularly fig. 4.14. Others are apparently quite good eg. figs. 4.17 and 4.22 but we need a more objective criterion to assess the reconstruction quality. This is the subject of the next section.

4.3.2 Discussion of Experimental Results

Tables 4.1 and 4.2 show the object coefficients and reconstruction coefficients for the A object and the E object using the coherent singular functions. Tables 4.3 and 4.4 show the same coefficients but using the incoherent singular functions.

Clearly the object and reconstruction coefficients do not agree very well in either the incoherent or coherent cases. One might at first think that this was due to noise in the blurred images, but in section 4.2.6 the noise was measured and found to be ~ 1 greylevel. This would lead to an error of about 1 unit in the pupil coefficients (see chap. 3). Therefore we would expect that those singular functions where the singular value is about 0.9 would have an error of about 1.0, whereas we can see discrepancies of up to 50 -100 in some cases, even among the top-most singular functions, which should be the most accurately measured. Neither do the errors show the pattern expected due to random noise i.e. a steady increase as λ_k decreases. Many of the higher-order singular functions seem to be more accurately measured

than the lower-order functions.

The most likely explanation for this discrepancy is the distortion in the camera mentioned in section 4.2.7. It is difficult to predict the pattern of errors such distortion would produce, but it would certainly affect the low-order singular functions just as much as the high-order ones. This is what we see in tables 4.1-4.4.

It is even more difficult to predict whether the incoherent or the coherent singular functions will give better results. In order to test which method is better we should count the number of singular functions which are reasonably well restored, i.e. those whose reconstruction coefficients come within some limit of their object coefficients.

In tables 4.1-4.4 I have marked those functions which fall within 10 units of each other. I have chosen an absolute limit rather than a relative limit - I could, for instance, have chosen all functions which lie within 10% of each other - because a relative limit discriminates against those coefficients which happen to be small compared to the average. I would consider a reconstruction coefficient of 15 to be well restored when the original is 10 even though the relative error is 50%. This is because they are both small with respect to the average which is ~ 100.

We can present the number of well-restored functions as follows

Image	Coherent	Incoherent
A	20	29
E	9	30

This would suggest that the incoherent functions were better.

Another measure of goodness-of-fit is to calculate the sum of the squares of the differences over the first 30 singular functions.

The results are as follows

Image	Coherent	Incoherent
A	11807	4865
E	15088	5388

Again the incoherent functions are better.

Finally we can look at the reconstructed objects and find the correlation coefficient with the original in all four cases.

The results are as follows. Firstly for reconstructions with singular values > 0.1

Image	Coherent	Incoherent
A	0.845	0.875
E	0.813	0.891

Secondly for reconstructions with singular values > 0.05

Image	Coherent	Incoherent
A	0.859	0.896
E	0.827	0.895

Thirdly for reconstructions with singular values > 0.01

Image	Coherent	Incoherent
A	0.723	0.862
E	0.702	0.862

In every case the incoherent functions are better. For comparison the correlation coefficients of the blurred images with the original objects are 0.821 for A and 0.812 for E.

These results appear inconsistent with the visual appearance of the reconstructions, eg. fig. 4.22 appears very similar to the original object, yet it has a low correlation coefficient. This is probably explained by the spurious high intensities around the edges of 4.22.

In summary, the discrepancies between the reconstructed objects and the original objects are probably caused by distortion in the camera-framestore system and not by noise - Gaussian, Poisson or otherwise. The incoherent singular functions give a better reconstruction than the coherent ones.

Table 4.1

singular value	object coefficient	reconstruction coefficient	difference	
0.99986	779.81653	780.00189	0.18536	t
0.99969	-201.56616	-255.83763	54.27147	
0.99569	128.19354	148.01115	19.81761	
0.99498	-147.38097	-158.86098	11.48001	
-0.99480	-7.13563	-7.76843	0.63280	t
0.99083	8.95982	18.61282	9.65300	t
0.95200	8.59160	-13.64705	22.23865	
-0.94735	-15.88266	-20.54638	4.66372	t
0.93528	-129.07303	-154.95367	25.88065	
0.93512	-193.04417	-165.86646	27.17772	
0.93138	130.99170	147.56567	16.57397	
0.89052	-55.29791	-64.94203	9.64412	t
0.75069	-113.72530	-107.45815	6.26715	t
0.74703	22.56014	27.05872	4.49858	t
0.70221	17.32973	30.46861	13.13888	
0.68257	39.80496	46.89472	7.08976	t
-0.68245	108.01588	116.46363	8.44775	t
0.67973	-68.95851	-88.45247	19.49396	
-0.64990	28.31269	33.65222	5.33953	t
0.51247	-2.97465	-6.33977	3.36512	t
0.39783	21.55294	75.38470	53.83176	
-0.39589	8.79401	-2.45541	11.24942	
0.37213	53.58462	60.90078	7.31616	t
0.31308	19.26572	29.90482	10.63910	
0.31302	205.96979	207.55907	1.58928	t
0.31177	-86.55023	-102.91745	16.36722	
0.29809	5.20459	31.65096	26.44637	
-0.27158	-30.90416	-24.22013	6.68403	t
0.23506	0.16486	-0.22334	0.38820	t
0.13904	105.14939	136.32819	31.17879	
0.13836	-35.03479	-43.21393	8.17914	t
0.13006	-80.65472	-120.96232	40.30760	
0.12457	-28.02561	-2.87018	25.15543	
0.09492	38.60055	47.92625	9.32570	t
0.09300	17.42200	-23.70356	41.12556	
-0.09299	-162.40649	-168.40131	5.99481	t
0.09262	56.94916	76.90064	19.95148	
-0.08855	9.77504	14.78006	5.00502	t
0.06983	-5.66318	-30.20978	24.54660	
0.04354	43.52828	-56.20618	99.73446	
-0.03701	16.04651	12.81151	3.23500	t
0.03669	-30.44319	124.74892	155.19211	
-0.03651	6.11629	-28.16105	34.27734	
0.03432	-3.31703	154.85272	158.16975	
-0.02505	3.06062	-25.67078	28.73140	
0.02089	40.00661	648.95477	608.94818	
0.02089	-172.19637	-86.95854	85.23782	
0.02081	-4.40799	100.47966	104.88765	
0.01989	54.74364	-21.91137	76.65501	
0.01569	6.33920	63.56225	57.22305	
0.01293	-21.97884	82.81652	104.79536	
0.01149	5.88445	31.52985	25.64540	
0.00831	-16.30542	196.52303	212.82845	
0.00386	-57.37170	-659.15649	601.78479	
-0.00386	129.98891	345.02130	215.03239	
0.00384	50.28517	252.79474	202.50957	
-0.00367	-85.91872	187.38342	273.30212	
-0.00341	-8.49281	-103.10224	94.60943	
0.00291	-10.18778	-565.87921	555.69141	
0.00290	-2.40659	-544.05347	541.64685	
-0.00154	39.70573	29.50434	10.20139	
0.00077	-1.55668	1832.59863	1834.15527	
0.00054	-13.04620	579.06042	592.10663	
-0.00014	1.03839	-3535.09277	3536.13110	

Table 4.2

singular value	object coefficient	reconstruction coefficient	difference	
0.91591	688.97897	683.82426	5.15471	T
0.83719	-211.35939	-223.65143	12.29205	
0.76344	-113.24873	-126.96664	13.71791	
0.72419	153.82450	144.43225	9.39226	T
-0.69782	17.23751	9.18777	8.04974	T
0.60363	-44.09956	-55.53886	11.43930	
0.59831	-99.56517	-104.67788	5.11271	T
0.57051	292.19067	291.14688	1.04378	T
0.52147	103.95495	84.69216	19.26279	
-0.49871	16.53857	19.96868	3.43011	T
0.47940	-142.05598	-145.98076	3.92478	T
0.45109	203.25656	193.44888	9.80768	T
0.39959	27.99695	30.30255	2.30560	T
0.38953	-48.63171	-66.28180	17.65009	
0.37938	46.00238	53.44022	7.43783	T
0.37268	-0.39796	-5.50711	5.10915	T
-0.35605	-113.82977	-149.07194	35.24217	
-0.31622	-3.13682	0.63416	3.77098	T
0.30799	-91.42124	-118.60323	27.18199	
0.30166	-91.10935	-93.13954	2.03019	T
0.29861	-24.49926	-17.89663	6.60263	T
-0.25446	-26.32876	-48.50944	22.18069	
0.25144	24.95846	32.14020	7.18175	T
0.24735	-8.97458	-5.19058	3.78400	T
0.23631	-37.17774	-41.70507	4.52733	T
0.23420	44.20923	27.59458	16.61465	
0.22609	239.32028	231.97563	7.34464	T
0.20388	-10.05577	-23.73695	13.68117	
0.19557	94.37234	104.47675	10.10441	
-0.19521	-12.26164	-4.78915	7.47249	T
0.18790	-129.63440	-133.96302	4.32862	T
0.16158	112.96081	107.82046	5.14035	T
-0.16134	33.74148	55.26915	21.52766	
0.14610	44.46536	59.58387	15.11851	
0.14588	67.18609	59.14262	8.04347	T
-0.13354	-153.82375	-187.30636	33.48260	
0.12947	22.92333	23.78272	0.85939	T
0.12829	65.37055	92.70123	27.33069	
0.11551	-11.66191	-5.95092	5.71098	T
0.10245	-25.71956	-32.31510	6.59554	T
-0.09960	-32.66031	-34.58870	1.92839	T
-0.09544	-112.78065	-138.82756	26.04691	
0.08147	-81.93421	-95.92028	13.98607	
0.07692	-38.01633	-10.36642	27.64990	
0.07647	-9.71508	-7.62012	2.09496	T
0.07031	101.52450	25.11232	76.41219	
0.06325	33.82405	55.65409	21.83005	
0.06082	-60.14788	-94.37816	34.23027	
-0.06051	-6.99508	-14.09455	7.09947	T
0.05025	122.45219	108.51716	13.93503	
0.04812	29.98696	20.82451	9.16244	T
0.04026	-31.58057	7.04413	38.62471	
-0.03736	1.06128	34.37221	33.31093	
0.03357	28.05584	-128.48014	156.53598	
0.03186	43.18091	77.78068	34.59978	
-0.03068	-44.79057	197.83885	242.62943	
0.02654	113.56280	132.65615	19.09335	
0.02533	-3.17964	-15.26692	12.08729	
-0.02193	-123.85000	2.13367	125.98367	
0.01967	-8.82993	32.78002	41.60995	
0.01757	55.99099	158.87251	102.88152	
-0.01390	-75.38834	-177.02639	101.63805	
0.01106	-21.86123	-81.59066	59.72943	
-0.00858	27.31357	212.55330	185.23972	

Table 4.3

singular value	object coefficient	reconstruction coefficient	difference	
0.99986	722.99280	703.18970	19.80310	
0.99969	46.99235	20.75820	26.23415	
0.99569	372.72870	356.66623	16.06247	
0.99498	88.05326	115.48068	27.42742	
-0.99480	-42.88035	-59.08782	16.20747	
0.99083	-21.17209	3.41468	24.58677	
0.95200	-94.44568	-135.46123	41.01555	
-0.94735	10.71413	-1.61696	12.33109	
0.93528	-192.93297	-199.63560	6.70264	t
0.93512	-8.36020	6.65501	15.01521	
0.93138	-96.54527	-103.53133	6.98605	t
0.89052	31.54552	47.33224	15.78672	
0.75069	33.21903	49.70102	16.48199	
0.74703	-77.91582	-68.37629	9.53953	t
0.70221	-3.65382	-19.14701	15.49319	
0.68257	-109.14474	-124.44189	15.29715	
-0.68245	24.82502	36.00320	11.17818	
0.67973	-6.37569	-23.43113	17.05544	
-0.64990	-7.31295	-1.15906	6.15389	t
0.51247	56.54103	46.45458	10.08645	
0.39783	18.15454	9.48509	8.66945	t
-0.39589	5.31281	18.32521	13.01240	
0.37213	-7.09121	4.36377	11.45498	
0.31308	67.74558	56.63664	11.10894	
0.31302	2.97838	10.61656	7.63818	t
0.31177	34.23706	21.06571	13.17135	
0.29809	-14.09099	-11.79528	2.29571	t
-0.27158	-1.86140	-10.13145	8.27005	t
0.23506	3.50591	20.58120	17.07529	
0.13904	-271.08072	-185.99567	85.08505	
0.13836	130.60468	111.41199	19.19269	
0.13006	66.86873	40.08825	26.78048	
0.12457	1.35568	-18.83525	20.19093	
0.09492	-81.52227	-55.56000	25.96227	
0.09300	162.87785	86.41583	76.46202	
-0.09299	-14.94933	10.44726	25.39659	
0.09262	34.74912	56.13332	21.38420	
-0.08855	5.92755	-12.69229	18.61984	
0.06983	-52.38124	-37.17914	15.20210	
0.04354	-21.31795	-20.10336	1.21459	t
-0.03701	1.00875	41.32473	40.31598	
0.03669	47.44357	274.06281	226.61923	
-0.03651	-7.43154	-63.15173	55.72019	
0.03432	-16.96745	111.18098	128.14842	
-0.02505	3.01162	62.42525	59.41363	
0.02089	33.32594	459.18063	425.85471	
0.02089	-3.93611	43.52869	47.46480	
0.02081	12.54368	255.30440	242.76071	
0.01989	0.29871	-78.93842	79.23714	
0.01569	-8.18370	56.06810	64.25180	
0.01293	57.73847	-1.99606	59.73453	
0.01149	10.97672	122.56929	111.59257	
0.00831	3.90790	86.77235	82.86445	
0.00386	-189.45328	-528.37061	338.91733	
-0.00386	-1.83534	192.89001	194.72536	
0.00384	-59.37821	427.07205	486.45026	
-0.00367	2.11336	116.82750	114.71414	
-0.00341	-1.83559	82.93154	84.76713	
0.00291	-4.04875	-222.44473	218.39598	
0.00290	37.69435	-49.55513	87.24948	
-0.00154	0.52180	200.50966	199.98785	
0.00077	-6.23947	830.73322	836.97266	
0.00054	-12.72417	-580.76599	568.04181	
-0.00014	-0.05809	2394.00342	2394.06152	

Table 4.4

singular value	object coefficient	reconstruction coefficient	differenc	
0.91591	696.69073	694.25989	2.43084	T
0.83719	8.98840	-12.78349	21.77190	
0.76344	45.10680	64.12693	19.02014	
0.72419	-70.96959	-70.69867	0.27092	T
-0.69782	-29.33692	-35.75967	6.42275	T
0.60363	56.37465	48.46464	7.91002	T
0.59831	84.20093	81.74516	2.45577	T
0.57051	420.93695	419.38819	1.54875	T
0.52147	-2.97905	-18.39647	15.41742	
-0.49871	-22.26146	-19.18263	3.07883	T
0.47940	-41.59974	-50.88753	9.28779	T
0.45109	-38.96907	-42.11910	3.15002	T
0.39959	-53.16683	-49.46192	3.70491	T
0.38953	126.13566	127.99366	1.85800	T
0.37938	16.53555	52.22409	35.68854	
0.37268	59.48275	61.03444	1.55170	T
-0.35605	-27.20501	-31.20924	4.00423	T
-0.31622	2.49240	-8.40300	10.89540	
0.30799	71.58434	73.13982	1.55548	T
0.30166	363.75601	343.14040	20.61560	
0.29861	-27.59023	-31.11356	3.52333	T
-0.25446	-26.78551	-38.71896	11.93345	
0.25144	-138.59131	-125.01392	13.57740	
0.24735	89.17847	117.23170	28.05324	
0.23631	8.93736	28.36616	19.42880	
0.23420	-15.95153	8.20048	24.15201	
0.22609	-2.24467	-5.09934	2.85468	T
0.20388	-82.99679	-92.68975	9.69295	T
0.19557	-11.01461	-14.68339	3.66878	T
-0.19521	-12.68403	-18.44211	5.75808	T
0.18790	205.36934	209.52057	4.15123	T
0.16158	20.05301	29.91611	9.86310	T
-0.16134	3.55371	-5.03720	8.59091	T
0.14610	201.61341	199.91630	1.69711	T
0.14588	-18.31161	0.38573	18.69734	
-0.13354	-7.59198	-12.76598	5.17400	T
0.12947	-3.00233	-12.08058	9.07825	T
0.12829	-148.04353	-167.10541	19.06187	
0.11551	45.19638	31.90587	13.29051	
0.10245	1.04481	9.57470	8.52989	T
-0.09960	-14.13178	-26.44605	12.31427	
-0.09544	-7.87917	-5.49709	2.38208	T
0.08147	39.11351	59.67347	20.55995	
0.07692	-86.07177	-147.91639	61.84462	
0.07647	-71.84078	-77.46396	5.62317	T
0.07031	8.92401	14.10471	5.18070	T
0.06325	-13.04124	25.38438	38.42561	
0.06082	-0.99050	-14.29072	13.30022	
-0.06051	-1.08919	-19.10330	18.01411	
0.05025	-1.81442	-3.07227	1.25785	T
0.04812	-49.66442	-62.36002	12.69560	
0.04026	21.81344	105.46999	83.65656	
-0.03736	-1.51811	39.55442	41.07254	
0.03357	187.33403	-76.68665	264.02075	
0.03186	-10.66552	-23.66090	12.99538	
-0.03068	2.17652	68.35998	66.18346	
0.02654	15.90195	-2.67885	18.58080	
0.02533	-17.09328	-47.51629	30.42300	
-0.02193	8.24348	-35.14616	43.38965	
0.01967	1.16183	67.61196	66.45013	
0.01757	-39.84493	37.09917	76.94409	
-0.01390	-0.59836	-98.54047	97.94211	
0.01106	25.43482	133.53030	108.09547	
-0.00858	0.65907	186.46541	185.80634	

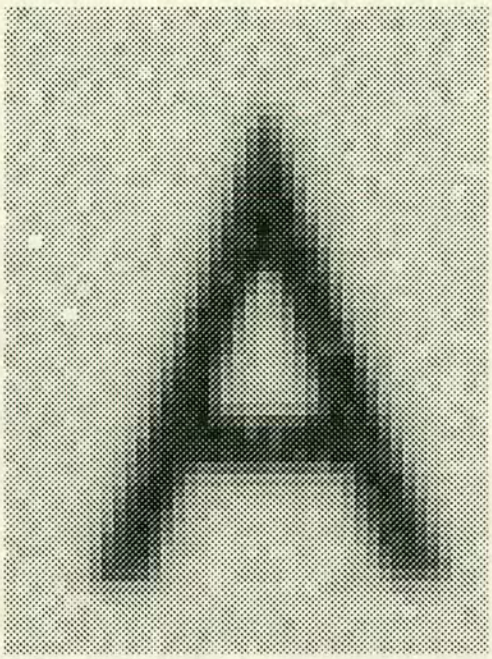


Fig 4.10

The original A object

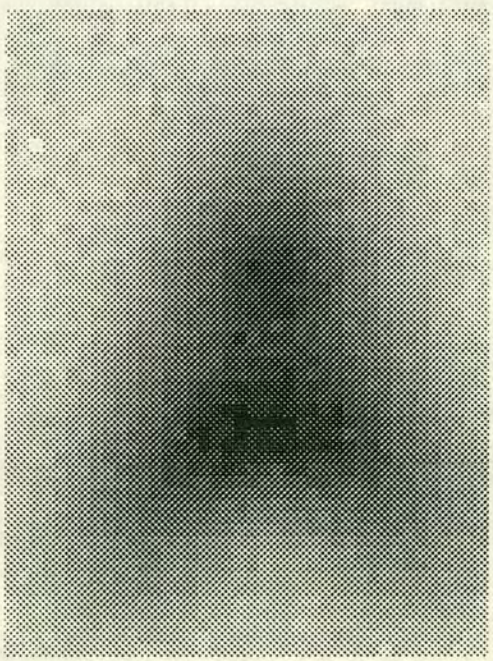
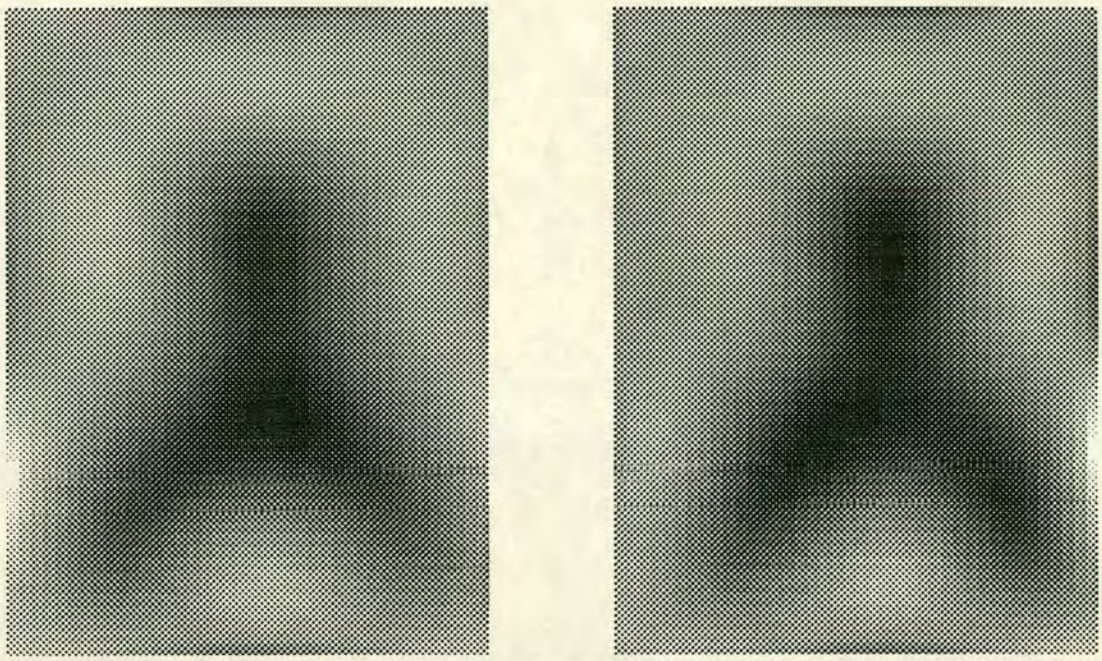


Fig 4.11

The blurred image of the A object



Figs. 4.12 and 4.13

Reconstructions of A using coherent singular functions, $\lambda_k \geq 0.1$ and $\lambda_k \geq 0.05$

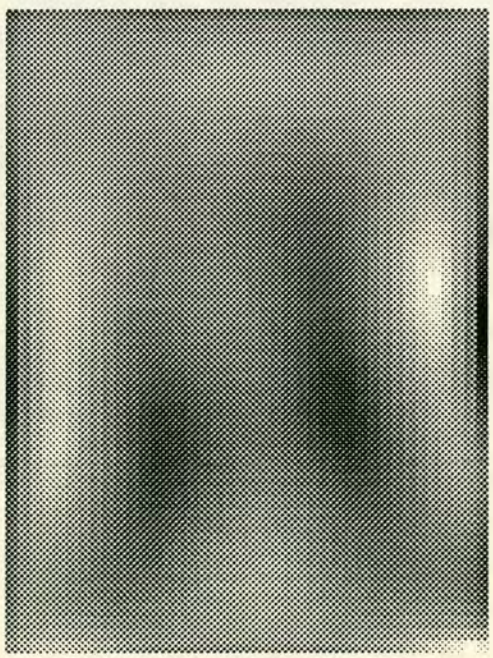
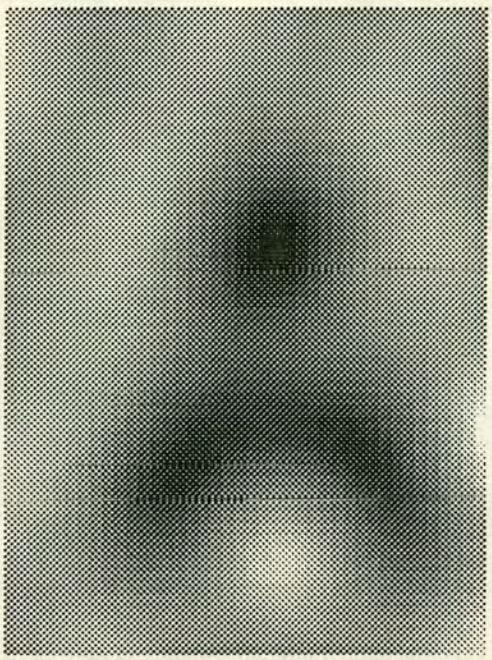
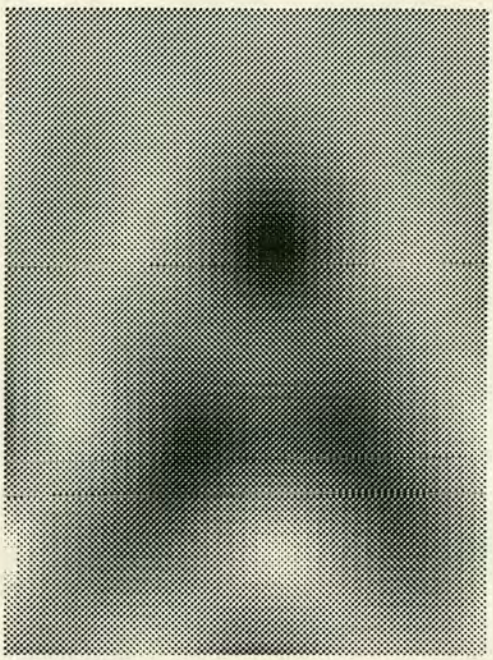


Fig 4.14

A reconstruction of A using coherent singular functions, $\lambda_k \geq 0.01$





Figs. 4.15 and 4.16

Reconstructions of A using incoherent singular functions, $\lambda_k \geq 0.1$ and $\lambda_k \geq 0.05$

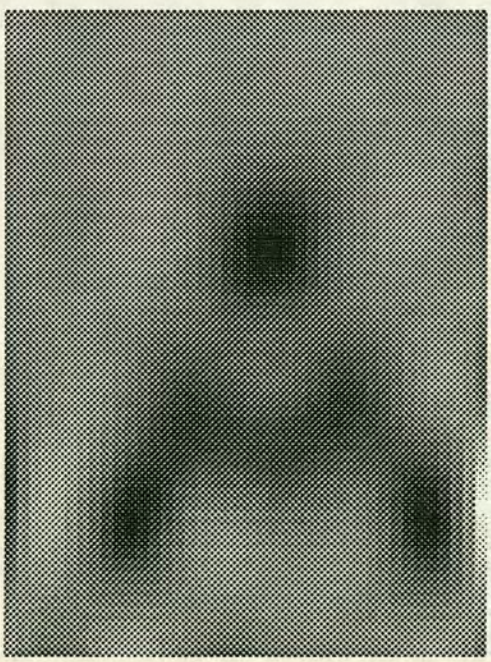


Fig 4.17

A reconstruction of A using incoherent singular functions, $\lambda_k \geq 0.01$

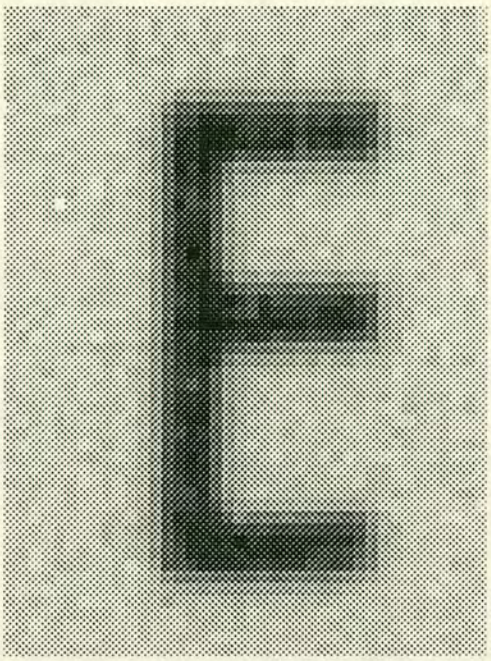


Fig 4.18

The original E object

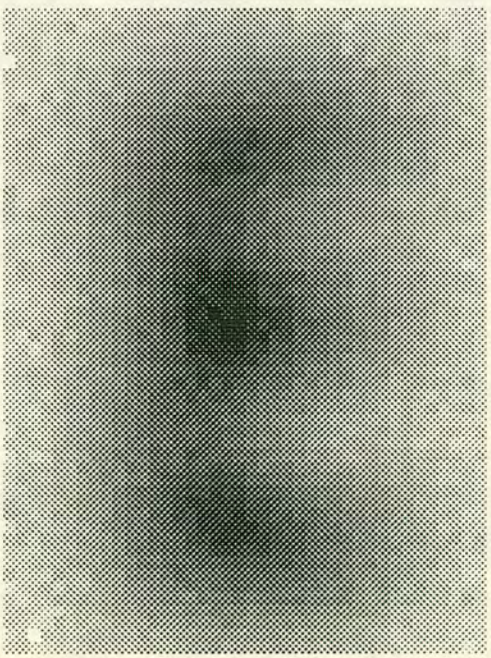
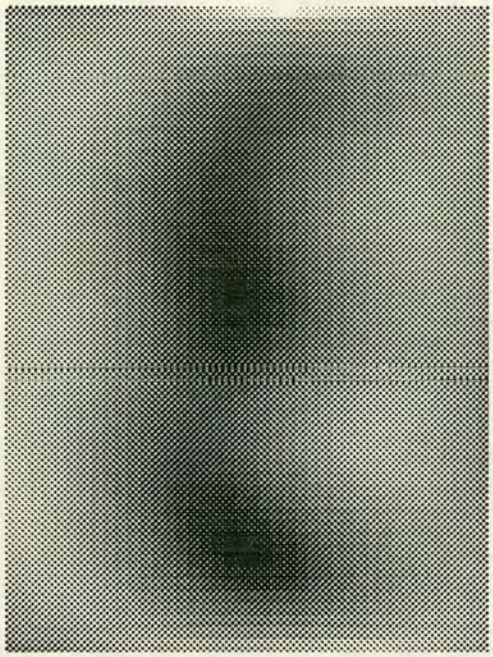
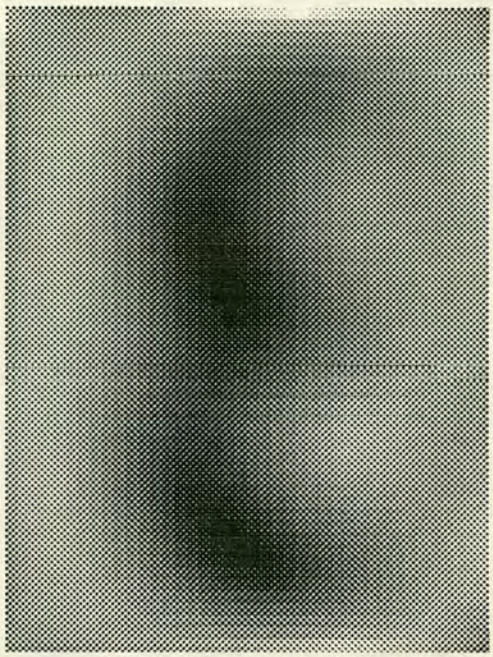


Fig 4.19

The blurred image of the E object





Figs. 4.20 and 4.21

Reconstructions of E using coherent singular functions, $\lambda_k \geq 0.1$ and $\lambda_k \geq 0.05$

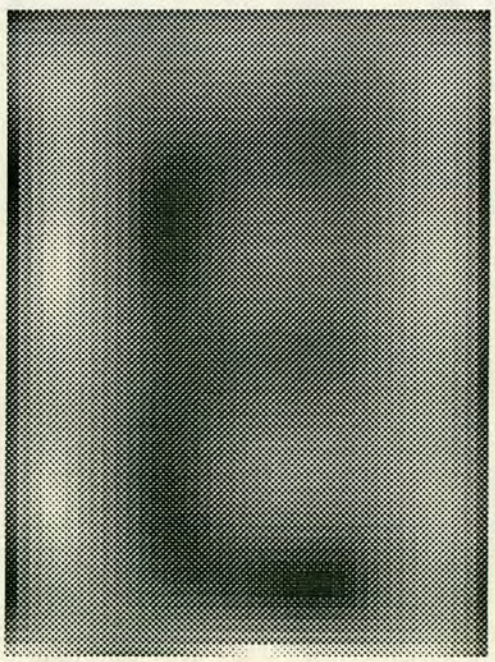
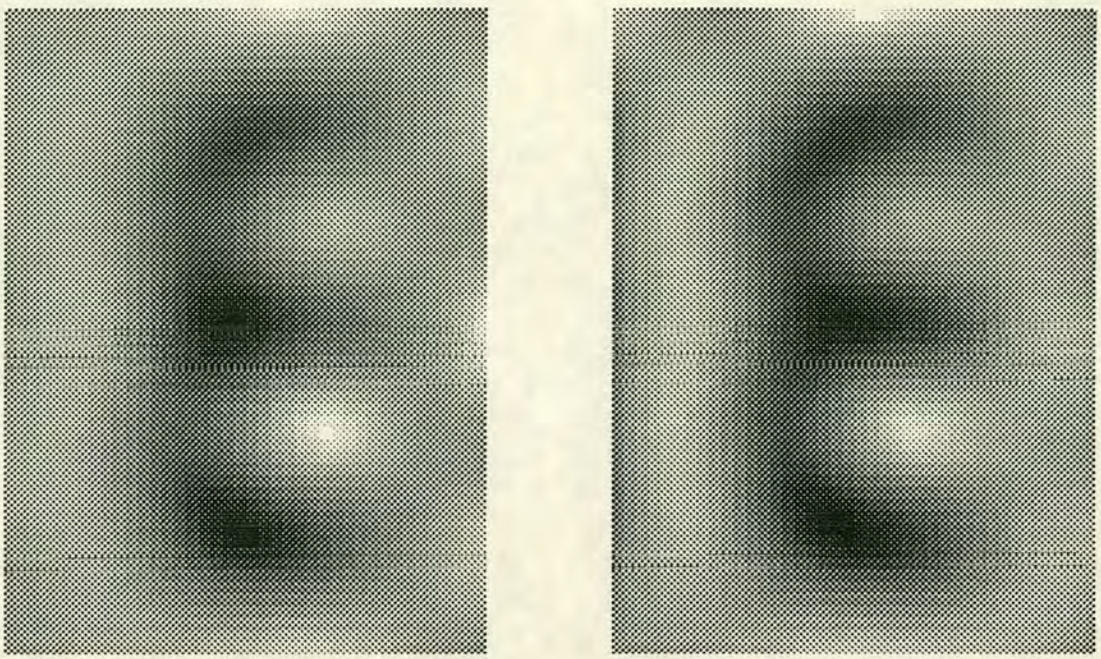


Fig 4.22

A reconstruction of E using coherent singular functions, $\lambda_k \geq 0.01$



Figs. 4.23 and 4.24

Reconstructions of E using incoherent singular functions, $\lambda_k \geq 0.1$ and $\lambda_k \geq 0.05$

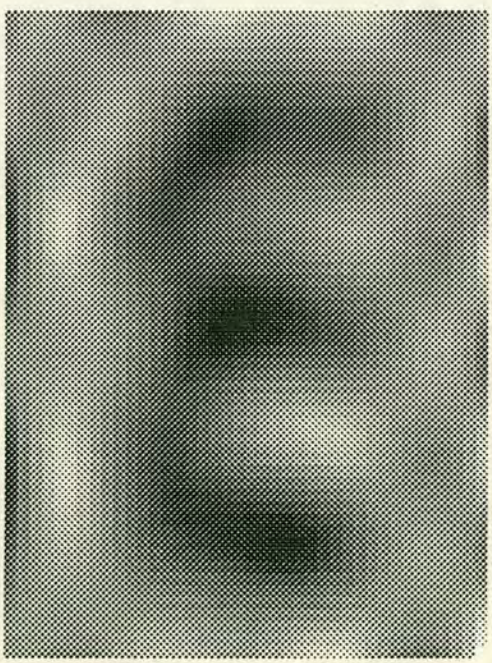


Fig 4.25

A reconstruction of E using incoherent singular functions, $\lambda_k \geq 0.01$

Chapter Five

Other Algorithms for Super-Resolution

5.1 Introduction

In this chapter we shall consider some possible developments of the SVD algorithm to include other types of a priori information. We shall also look at other algorithms different to SVD which carry out super-resolution. The first of these is the Gerchberg algorithm which uses the same a priori information as SVD. Then we shall look at the Bayesian techniques: maximum entropy and conditional autoregression. The Bayesian techniques use different a priori information from SVD.

The main theme of the chapter will be the effect of different a priori information. The different techniques use different types of a priori information. We shall see how this is built into the different algorithms and how it affects the final reconstructed objects.

5.2 Developments of SVD

5.2.1 Introduction

SVD uses only one form of a priori information viz. the finite size of the object. We have seen that SVD suffers from two problems: sensitivity to noise and restriction to low space-bandwidth products. It might be possible to overcome these problems if we could assume additional a priori information eg. non-negativity, smoothness etc. One way of doing this has been shown by Sasaki and Yamagami [36] and another by Maeda and Murata [40]. Both methods are iterative.

5.2.2 Sasaki and Yamagami

The method of Sasaki and Yamagami is as follows. They assume that the first p singular vectors can be estimated accurately but that the rest are badly corrupted by noise. Thus the coefficients, $f_k, k = 0, 1, 2...p$ are known to within an error of ε/λ_k where ε is the standard deviation of the noise (assumed Gaussian) but the remaining

$f_k, k = p+1, \ldots$. are unknown. We must use some form of additional a priori information to estimate these. There is a number of different forms of information we could use but Sasaki and Yamagami consider only non-negativity.

They adopt an iterative procedure in each loop of which the following steps occur.

(1) We form an estimated object, $\hat{\mathbf{f}}$ from the first q singular vectors, where $q > p$

$$\hat{\mathbf{f}} = \sum_{k=0}^{q} f_k \mathbf{u_k} \qquad 5.2.1$$

As I have said above, the values of f_k for $p < k < q$ will be corrupted by noise at the start of the iteration procedure but we hope that as the iteration proceeds this noise will be filtered out.

(2) All the negative pixel values in $\hat{\mathbf{f}}$ are set to zero to give a new object vector, $\mathbf{z}$.

(3) $\mathbf{z}$ is decomposed into its singular vectors

$$z_k = \mathbf{z}.\mathbf{u_k} \quad k = 0, q \qquad 5.2.2$$

(4) The first p values of z_k are compared to the first p values of f_k. If

$$|f_k - z_k| < \frac{\varepsilon}{\lambda_k} \quad k = 0, p \qquad 5.2.3$$

then the iteration stops. If not we set the values of $f_k, k > p$ equal to the corresponding values of z_k

$$f_k = z_k \quad k = p+1, q$$

The iteration then returns to step 1.

Sasaki and Yamagami show that this iteration must eventually converge, thus producing a solution vector, $\mathbf{z}$, whose top p singular components have the known values, $f_k \quad k = 1, p$ and which also satisfies the non-negativity constraint. Sasaki

and Yamagami give a 1D numerical example of this procedure and show that it does in fact yield accurate estimates of the object components, f_k, for $k > p$. In other words, it has increased the resolution beyond that achievable by pure SVD without the additional non-negativity constraint.

However, I would make the following comment on this method. Sasaki and Yamagami do not state how to choose q- which is the total number of singular vectors we can expect to restore. In their example they set q=N - the number of pixels in the object. N is also the number of singular vectors necessary to completely describe the object. In their example N=10. But they cannot surely be claiming to restore the entire set of singular vectors in every case. There must be a limit even though this limit may be greater than that in the case of pure SVD and this limit must depend on the noise level, ε. Sasaki and Yamagami give no consideration to this problem.

5.2.3 Maeda and Murata

Maeda and Murata adopt another approach. First of all they make a slight modification to the SVD process itself. Instead of merely dividing each image coefficient, g_k, by λ_k, they multiply by

$$\frac{\lambda_k}{\lambda_k^2 + \alpha/\lambda_k^2} \qquad 5.2.4$$

where α is called the "regularising parameter". Its effect is to damp out the higher-order singular vectors. This process is effectively equivalent to truncating the summation as we have always done up till now.

Maeda and Murata use the following notation

$$\hat{\mathbf{f}} \quad = \quad \mathbf{H}^{+}\,\mathbf{g} \qquad 5.2.5$$

to represent the formation of the reconstructed object, $\hat{\mathbf{f}}$, from the original image, $\mathbf{g}$. They call the operator, $\mathbf{H}^{+}$, the "pseudo-inverse" of $\mathbf{H}$. Thus $\mathbf{H}^{+}$ represents the process of first decomposing g into singular vectors, multiplying by 5.2.4 and then reconstructing $\hat{\mathbf{f}}$.

Maeda and Murata then go on to show that if we have an initial estimate, $\hat{\mathbf{f}}$, for the original object we can calculate a better estimate, $\hat{\mathbf{f}}_1$, given by

$$\hat{\mathbf{f}}_1 = \hat{\mathbf{f}} + \Delta\hat{\mathbf{f}} \qquad 5.2.6$$

where $\Delta\hat{\mathbf{f}}$ is a correction vector given by

$$\Delta\hat{\mathbf{f}} = \mathbf{H}^{+}(\mathbf{g} - \mathbf{H}\hat{\mathbf{f}}) \qquad 5.2.7$$

This step is used in the iteration procedure. We can now describe that procedure. Each loop consists of the following steps.

(1) We take an initial estimate, $\hat{\mathbf{f}}_k$, and then impose our a priori information on it. For instance, we could set all the negative components to zero, as in the case of Sasaki and Yamagami, or we could smooth it, if we have a priori information about smoothness.

(2) We calculate the correction vector, $\Delta\hat{\mathbf{f}}_k$, given in equn 5.2.7.

(3) We calculate a new estimate

$$\hat{\mathbf{f}}_{k+1} = \hat{\mathbf{f}}_k + \Delta\hat{\mathbf{f}}_k \qquad 5.2.8$$

and return to step (1)

The iteration terminates when

$$\|\mathbf{g} - \mathbf{H}\hat{\mathbf{f}}_k\| - \|\mathbf{g} - \mathbf{H}\hat{\mathbf{f}}_{k+1}\| < \epsilon_1 \qquad 5.2.9$$

and

$$\|\Delta\hat{\mathbf{f}}_k\| < \epsilon_2 \qquad 5.2.10$$

where ϵ_1 and ϵ_2 are predetermined values.

At each iteration the value of α in equn. 5.2.4 is incremented.This is because as the number of iterations increases the value of $\Delta\hat{\mathbf{f}}_k$ is increasingly affected by noise.

By increasing the value of α the higher-order singular vectors are damped down. Since it is the higher-order vectors which are affected most by the noise this helps to repress the effect of the noise on $\Delta\hat{\mathbf{f}}_k$. Maeda and Murata suggest multiplying α by a constant, c (> 1.0), at each iteration. Thus

$$\alpha_k \quad = \quad c^k \alpha_0 \qquad 5.2.11$$

Maeda and Murata show that this procedure will converge in most circumstances.

They give 1D and separable 2D numerical examples showing that the iterative procedure gives better results than pure SVD without the additional a priori information.

I would make the same comment on Maeda and Murata as I did on Sasaki and Yamagami. They do not specify how many extra singular vectors we could expect to estimate in the presence of noise. In the case of Maeda and Murata that is equivalent to specifying α_0, which determines the effective number of singular vectors in the reconstruction. Maeda and Murata do not tell us how to choose α_0. Presumably we are to carry out a number of numerical simulations for different values of α_0 and choose the value which works best. Maeda and Murata do not consider this problem.

5.2.4 The Gerchberg algorithm

The Gerchberg algorithm [39] is a different way of carrying out super-resolution. It uses exactly the same a priori information as pure SVD i.e. the finite spatial extent of the object. Therefore, one would expect it to give similar if not identical results to pure SVD.

The method is effectively restricted to coherent images. It is iterative. Before the iteration begins the band-limited image is Fourier transformed. This gives us the Fourier transform, G(k), of the image within the pass-band. This is assumed to be known reasonably accurately, with an error of ε on each sample point. This function, G(k), is stored permanently throughout the whole process.

The raw image, g(y), provides our first estimate of the original object, $\hat{\mathbf{f}}$. Each loop of the iteration consists of the following steps

(1) We take our estimated object, $\hat{\mathbf{f}}$, and impose the known spatial limit on it, i.e. we set to zero any values of $\hat{\mathbf{f}}$ outside the known spatial limit.

(2) $\hat{\mathbf{f}}$ is Fourier transformed to obtain an estimated Fourier transform, $\hat{F}(k)$, of the object.

(3) The part of $\hat{F}(k)$ within the pass-band is replaced by G(k). G(k) is known to be the correct value of F(k) within the pass-band.

(4) $\hat{F}(k)$ is inverse Fourier transformed to give a new $\hat{\mathbf{f}}$ and we return to step (1)

As the number of iterations increases so does the degree of super-resolution but, as with SVD, so does the amplification of the noise. Therefore the iteration has to be truncated at a point where the noise has not begun to swamp the signal.

The reason that this method is inapplicable to incoherent images can be seen from step (3). At this step it is essential to be able to equate G(k) with F(k) within the pass-band. But as we have seen with incoherent images this is not possible. We would have to divide G(k) by H(k) within the pass-band. But this is an ill-conditioned process and results in the amplification of noise. For this reason it is inappropriate to use the Gerchberg algorithm on incoherent images. Thus SVD is a more appropriate algorithm for incoherent images since we do not need to divide G(k) by H(k).

In the case of coherent images SVD and the Gerchberg algorithm should yield very similar results but, as far as I know, this has not been tested in practice. I would expect SVD to be much faster computationally since it involves only the computation of scalar products whereas Gerchberg involves computing two FFTs per iteration, and there may be as many as 50-100 iterations per image. Thus SVD has the advantage for real-time applications or applications where a large number of different images need to be processed quickly.

5.3 Bayesian Techniques

5.3.1 Outline of Bayesian arguments

There is a class of image processing algorithms which use statistical or Bayesian arguments. This class includes the Maximum Entropy method (MaxEnt) and the conditional autoregression method (CAR). MaxEnt and CAR were not designed specifically as super-resolution algorithms. Their originators, Gull and Skilling and Ripley, scarcely ever use that term. However, it is obvious that the methods have similar aims to the super-resolution techniques. They all seek to improve image quality by using a priori information. It is in the type of a priori information that the main differences can be seen. The arguments of all these methods follow the following lines.

Let the observed image be represented by a vector, $\mathbf{g}$. Let the original object, which we are trying to reconstruct, be represented by a vector, $\mathbf{f}$. Then as we saw in chap.1

$$\mathbf{g} = \mathbf{Hf} + \mathbf{n} \qquad 5.3.1$$

where $\mathbf{H}$ is a blurring matrix and $\mathbf{n}$ is additive Gaussian noise.

As we saw in chap.1 the problem is to find $\mathbf{f}$ given $\mathbf{g}$. This is an inverse problem where there is a large class of possible solutions, $\mathbf{f}$, each of which could have given rise to $\mathbf{g}$. As we saw there, in order to make this problem easier we can use a priori information. All the Bayesian techniques assign to each member, $\mathbf{f}$, of the solution class a probability, $P(\mathbf{f}|\mathbf{g})$. We seek that member $\mathbf{f}$ which has the largest probability $P(\mathbf{f}|\mathbf{g})$. This probability depends (a) on how closely $\mathbf{f}$ corresponds to our a priori information and (b) on how well $\mathbf{f}$ could have given rise to $\mathbf{g}$. $P(\mathbf{f}|\mathbf{g})$ is given by Bayes' Theorem

$$P(\mathbf{f}|\mathbf{g}) \propto P(\mathbf{f})\,P(\mathbf{g}|\mathbf{f}) \qquad 5.3.2$$

The probability, $P(\mathbf{f})$, incorporates our a priori information: if $\mathbf{f}$ corresponds to our a priori information then $P(\mathbf{f})$ is high, if not then it is low. $P(\mathbf{g}|\mathbf{f})$ measures

how well **f** could have given rise to **g**. In other words, it measures how well **f** fits the data, **g**.

It is usual to represent the two components on the RHS of equn.5.3.2 as exponential functions. Thus

$$P(\mathbf{f}) \quad \propto \quad \exp(-\frac{1}{2\kappa_s} U(\mathbf{f})) \qquad 5.3.3$$

where U(**f**) can be regarded as an "energy". Examples of **f** which correspond to our a priori information have low energy, those which do not have high energy. κ_s is a scaling parameter which we shall consider later.

And, in the case of additive Gaussian noise

$$P(\mathbf{g}|\mathbf{f}) \quad \propto \quad \exp(-\frac{1}{2\kappa_n} \|\mathbf{g} - \mathbf{Hf}\|^2) \qquad 5.3.4$$

where κ_n is the variance of the noise. Similar forms are possible in the cases of Poisson and level-dependent Gaussian noise.

If we now substitute these into 5.3.2 and take logs of both sides we obtain

$$L \quad = \quad -2ln(P(\mathbf{f}|\mathbf{g})) = \quad const + \frac{1}{\kappa_s} U(\mathbf{f}) + \frac{1}{\kappa_n} \|\mathbf{g} - \mathbf{Hf}\|^2 \qquad 5.3.5$$

It is this functional which we must minimise to obtain the most likely member of the solution class. Notice 5.3.5 has the typical Lagrangian form

$$L \quad = \quad U + \lambda C \qquad 5.3.6$$

for minimising a function, U, subject to a constraint, C. In the case of Gaussian noise

$$C \quad = \quad \|\mathbf{g} - \mathbf{Hf}\|^2 \qquad 5.3.7$$

and its value determines how well our solution, **f**, fits the given data, **g**. The value of U determines how well **f** corresponds to our a priori information.

The parameter λ is a Lagrange multiplier. In 5.3.5 $\lambda = \kappa_s/\kappa_n$. The value of λ determines the trade-off between how closely our solution will fit the a priori

information and how closely it will fit the data. If we choose a small value of λ the solution will fit the a priori information very closely. If, on the other hand, we choose a large value the solution will fit the data closely. The value of λ is determined both by κ_n (which is a measure of the noise in the data) and κ_s which is part of the a priori information. Usually κ_n is quite easy to determine since it is merely the variance of the noise. But it is often not easy to know the best value of κ_s. This leaves an uncertainty about the best value of λ to use.

The simplest solution is to set some limit for the value of C, say C_{aim}, and increase λ until $C \leq C_{aim}$. For instance, in the case of additive Gaussian noise we could set $C_{aim} = N\kappa_n$ where N is the number of pixels. This is because we would expect on average a square error of κ_n on each pixel.

If we were to increase λ further than this and so force C to be substantially less than C_{aim} then we would be forcing the solution, $\mathbf{f}$, to fit the data much more closely than the noise would warrant. In other words, we would be forcing the solution to fit the noise. This would cause spurious features to appear in the solution. Therefore, we should choose that value of λ which causes C to be just less than C_{aim}.

All Bayesian techniques follow this form. It is in their choice of a priori information and hence the energy functional, U($\mathbf{f}$), that they differ.

5.3.2 Maximum Entropy

The maximum entropy method (MaxEnt) was first suggested by Jaynes [27] as a solution to various problems in statistical mechanics. It was first applied to images by Frieden [28,29] who used it only on numerical examples. Its main proponents now are Gull and Skilling [26] who have developed it into a sophisticated algorithm and applied it to many large images from radio astronomy, medical imaging etc.

The maximum entropy method assumes that we have a priori information that the object is everywhere positive. We can assume this if the illumination is incoherent and the vectors $\mathbf{g}$ and $\mathbf{f}$ represent intensity distributions. Thus the maximum entropy

method is restricted to incoherent illumination.

The key step in the argument is to make the intensity distribution, $\mathbf{f}$, equivalent to a probability distribution, $\mathbf{p}$. We can do this as follows

$$\begin{aligned} p_i &= \frac{f_i}{\sum f_i} \\ &= \frac{f_i}{\alpha} \end{aligned} \qquad 5.3.8$$

where $\alpha = \sum_i f_i$. Thus all the values of p_i lie between 0 and 1 and the sum, $\Sigma p_i = 1$, as is required of a probability distribution. The values, p_i, are interpreted as " the probability that the next photon will come from pixel i."

We can now assign to any vector, $\mathbf{f}$, an "entropy", S, given by

$$S = -\sum_i p_i \log(p_i) \qquad 5.3.9$$

This quantity was first put forward by Gibbs to represent our uncertainty about the state of a thermodynamic system, e.g. molecules of gas in a box. Shannon [24] also used it to represent our uncertainty (or lack of information) about a message source in communication theory. Here it represents our uncertainty about which part of the image the next photon will come from.

Gull and Skilling argue that we must find the object which has the maximum possible entropy and yet which is consistent with the data. If we were to accept an object which had less than the maximum entropy we would be assuming information about the object which was not justified by the data.

Gull and Skilling show that the probability, $P(\mathbf{f})$, of any object, $\mathbf{f}$, is given by

$$P(\mathbf{f}) \propto \exp(S(\mathbf{f})/2\kappa_s) \qquad 5.3.10$$

where κ_s is a constant which needs to be determined for each image. This has the same form as equn. 5.3.3, therefore it is the entropy functional, S($\mathbf{f}$), which is used as the energy, $U(\mathbf{f})$, in equn. 5.3.5. This gives us

$$L = \frac{1}{\kappa_s}\sum_i p_i \log(p_i) + \frac{1}{\kappa_n}\|\mathbf{g} - \mathbf{Hf}\|^2 \qquad 5.3.11$$

where κ_n is the variance of the additive Gaussian noise. In other words, we must find the object, $\mathbf{f}$, which has the maximum entropy subject to the constraint $\|\mathbf{g} - \mathbf{Hf}\|^2 < C_{aim}$.

Ripley [41] has criticised this idea on the grounds that the entropy functional depends only on the marginal distribution of grey-levels and not on their spatial distribution, i.e. if we permute the pixels in the object, $\mathbf{f}$, the entropy would remain unaltered. Gull and Skilling partially meet this criticism by incorporating a prior model, m, into the entropy as follows

$$S = -\sum_i p_i \log(p_i|m) \qquad 5.3.12$$

m could for example include information about spatial correlation between pixels. m could be a Markov random field (MRF). (See section 5.3.4) Shannon [24] gives examples of how to calculate the entropy of an MRF.

There are various computational approaches to minimising 5.3.11. Gull and Skilling use a modified version of the conjugate gradient technique [26]. This is an iterative hill-climbing technique. During each iteration it searches for the point along a certain search vector at which 5.3.11 is minimised. At the next iteration it uses the previous minimum as a starting point and searches along a vector conjugate to that of the previous iteration. This method is somewhat laborious to program.

Molina and Ripley [42] take a different approach, which was first suggested by Gull and Daniel [43]. They argue that in order to minimise L it should be stationary with respect to both $\mathbf{p}$ and α, thus

$$\frac{\partial L}{\partial p_i} = 0 \quad \forall i \qquad 5.3.13$$

$$\text{and} \quad \frac{\partial L}{\partial \alpha} = 0 \qquad 5.3.14$$

Equn. 5.3.13 gives us

$$p_i \propto \exp(\lambda\alpha\frac{\partial C}{\partial f_i}) \qquad 5.3.15$$

where $\lambda = \kappa_s/\kappa_n$, α is defined in equn. 5.3.8 and

$$\frac{\partial C}{\partial \mathbf{f}} = \mathbf{H}^T(\mathbf{g} - \mathbf{Hf}) \qquad 5.3.16$$

Equn. 5.3.14 gives us

$$\frac{\partial L}{\partial \alpha} = \frac{\partial C}{\partial \alpha} \propto (\alpha \mathbf{Hp} - \mathbf{g})^T \mathbf{Hp}$$

$$\Rightarrow \alpha = \mathbf{g}^T \mathbf{Hp} / \|\mathbf{Hp}\|^2 \qquad 5.3.17$$

We can find the solutions to these equations iteratively. If we start with arbitrary values of $\mathbf{p}$ and α we can calculate new values using 5.3.15 and 5.3.17 and repeat until they converge. In practice the scheme will only converge when the starting values are reasonably close to the final result. A smoothed version of the original image is usually good enough. It will then converge in 20-30 iterations.

5.3.3 Conditional Autoregression Method

The conditional autoregression (CAR) technique was suggested by Ripley [41] This method again assumes that the object, $\mathbf{f}$, and image, $\mathbf{g}$, are positive and so is also restricted to incoherent imaging. It further assumes that the object has a certain smoothness, i.e. that there are no abrupt variations between neighbouring pixels. We can incorporate this idea by assuming that the object, $\mathbf{f}$, can be modelled by a Markov random field (MRF).

In a MRF we regard the values, f_i, of all the pixels as being random variables such that $P(f_i|f_j, j \neq i)$ depends only on the set of f_j which are neighbours of pixel i. This is clearly a good method of modelling smoothness because we can assign a high probability, $P(f_i|f_j)$, to values of f_i which are close to the given values of the neighbouring pixels: this avoids abrupt variations between pixels.

Ripley adopts a type of MRF known as a conditional autoregression (CAR). In a CAR the pixel values, f_i, are assumed to follow a Gaussian distribution with variance κ_s and an expected value given by

$$E(f_i|f_j, j \neq i) = \sum_j C_{ij} f_j$$

where C is given by

$$\begin{aligned} C_{ij} &= 1/N \qquad \text{if i and j are neighbours} \\ &= 0 \qquad \text{if not} \end{aligned} \tag{5.3.18}$$

and N is the number of nearest neighbours (usually four or eight)

Let us now consider the probability, $P(\mathbf{f})$, of the whole object vector rather than the probabilities, $P(f_i)$, of the individual object pixels. Ripley [44] shows that $P(\mathbf{f})$ follows a multivariate normal distribution given by

$$P(\mathbf{f}) \quad \propto \quad \exp(-\frac{1}{2\kappa_s}\mathbf{f}^T(\mathbf{I}-\mathbf{C})\mathbf{f}) \tag{5.3.19}$$

and this clearly falls into the pattern of equn. 5.3.3 with the energy, U(**f**), defined by

$$U(\mathbf{f}) \quad = \quad \mathbf{f}^T(\mathbf{I}-\mathbf{C})\mathbf{f} \tag{5.3.20}$$

The value of U(f) tends to be low for smooth objects- i.e. objects which correspond to our a priori information- and high for rough objects.

We substitute this functional for U(**f**) in equn.5.3.5 and obtain

$$L \quad = \quad const. + \frac{1}{\kappa_n}\|\mathbf{g}-\mathbf{H}\mathbf{f}\|^2 + \frac{1}{\kappa_s}\mathbf{f}^T(\mathbf{I}-\mathbf{C})\mathbf{f} \tag{5.3.21}$$

then proceed as for MaxEnt i.e. we differentiate equn. 5.3.21 and set it equal to zero. Ripley obtains the following expression

$$(\mathbf{H}^T\mathbf{H} + \lambda\mathbf{I} - \lambda\mathbf{C})\mathbf{f} \quad = \quad \mathbf{H}^T\mathbf{g} \tag{5.3.22}$$

where $\lambda = \kappa_n/\kappa_s$. Equn. 5.3.22 can be re-arranged to give the following iterative scheme

$$\mathbf{f}_{k+1} \quad = \quad (1-\alpha)\mathbf{C}\mathbf{f}_k + \alpha[\mathbf{H}^T(\mathbf{g}-\mathbf{H}\mathbf{f}_k) + \mathbf{f}_k] \tag{5.3.23}$$

where $\alpha = \kappa_s/(\kappa_n + \kappa_s)$. This scheme converges within 20-30 iterations provided we start from a suitable starting point.

5.3.4 Examples of MaxEnt and CAR

Figs. 5.1-8 show some examples of MaxEnt and CAR reconstructions. These reconstructions were generated by programs written by me with assistance from Prof. B.D.Ripley. The programs were based on the algorithms described in the previous two sections. The raw data for the reconstructions was provided by the Instituto de Astrofisica de Andalucia. The raw data were astronomical images of galaxies recorded by CCD arrays at the end of a telescope. The images are $\sim 256 * 256$ pixels and the grey-levels range from 0-20000. The blurring in the images is not caused by the band-limited nature of the telescope but by the motion of the Earth's atmosphere. The blurring has been well studied and can be modelled by the following point spread function

$$h(r) \quad = \quad \frac{\beta}{\pi r_0^2 [1 + (r/r_0)^2]^\beta}$$

where r is the radial distance from the origin and β and r_0 are constants which can be determined from the images. In the case of these images the above function leads to an effective bandwidth of ~ 120, in units of $\pi/128$ radians. The object in the image has a spacewidth of ~ 256 pixels. Thus these images have a one-dimensional space-bandwidth product of

$$c \quad \sim \quad \frac{\pi}{128} \frac{256}{2} \frac{120}{2} \quad = \quad 60\pi$$

The noise in the original image is signal-dependent and is given by

$$var(n_i) \quad = \quad 200 + 0.5 g_i$$

where n_i and g_i are the noise and signal values at the ith pixel.

Figs. 5.1-8 are half-tone dot representations of the images. They contain effectively only 50 grey-levels and tend to obscure some of the differences between the different reconstructions. Much more detail can be seen on a 256 grey-level representation on a video screen, but these are sufficient to show the broad features I want to talk about.

Fig. 5.1 shows the blurred image. Figs. 5.2-4 show MaxEnt reconstructions for different values of λ. The noise is seen as speckle most easily visible around the edges of the galaxy, although it is of course present throughout the whole image. In fig. 5.2 the noise has been reduced but at the expense of smoothing the image, i.e. reducing the resolution. As λ increases the restoration is made to fit the data more closely and so more structure appears. However, beyond a certain point the algorithm is merely being made to fit noise and so the new structure which appears is spurious. This is the case in fig. 5.4 where most of the apparent fine detail is probably not there in the original object. Figs. 5.5-8 represent a similar set of reconstructions from the CAR technique.

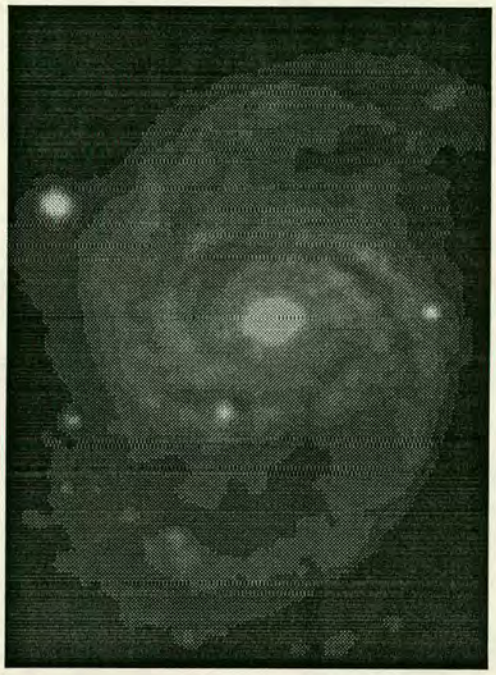
Thus, to summarise, in MaxEnt and CAR we are again seeing a trade-off between super-resolution and noise. By trying to increase the super-resolution we also amplify the noise - just as we saw in the case of SVD. Figs. 5.3 and 5.7 probably represent the ideal balance between super-resolution and noise.

5.4 Comparisons and Conclusions

In this chapter we have looked at some developments of SVD and some other techniques of achieving super-resolution. One of the main themes of the chapter has been the way the different techniques use different types of a priori information and the effect this has on the reconstructed image.

Both CAR and MaxEnt assume positive data, i.e. incoherent illumination, whereas SVD can cope with both coherent and incoherent illumination. CAR makes the further assumption of image smoothness. Neither MaxEnt nor SVD assume this.

SVD assumes that the object is space-limited. This is also a tacit assumption made by CAR and MaxEnt. Clearly it is impractical for them to cope with infinite objects. The vectors representing the object and image must be truncated somewhere and, although it is in principle possible for the object to extend outside the edge of the vector, this tends to produce peculiar edge effects. Therefore, both CAR and MaxEnt tend to be happier when the object is well contained within the vectors $\mathbf{f}$ and $\mathbf{g}$.

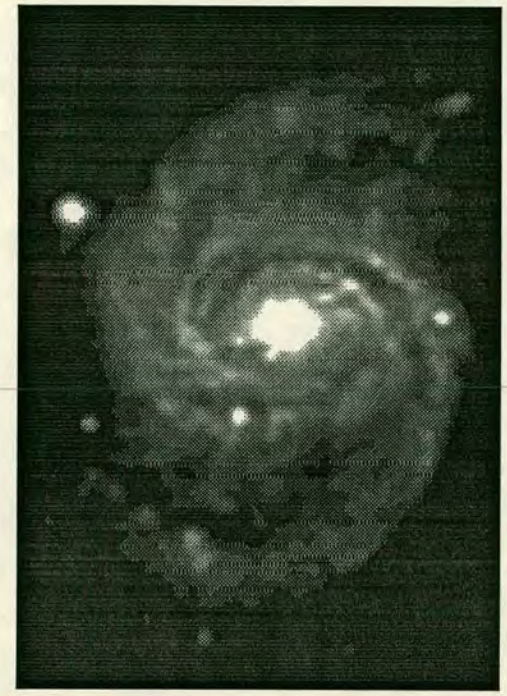


Figs. 5.1 and 5.2

The blurred image and reconstruction with $\lambda = 10^{-7}$

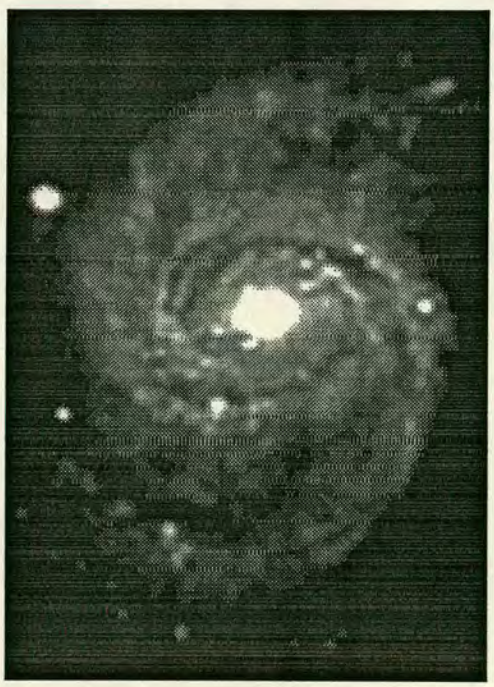
Figs. 5.3 and 5.4

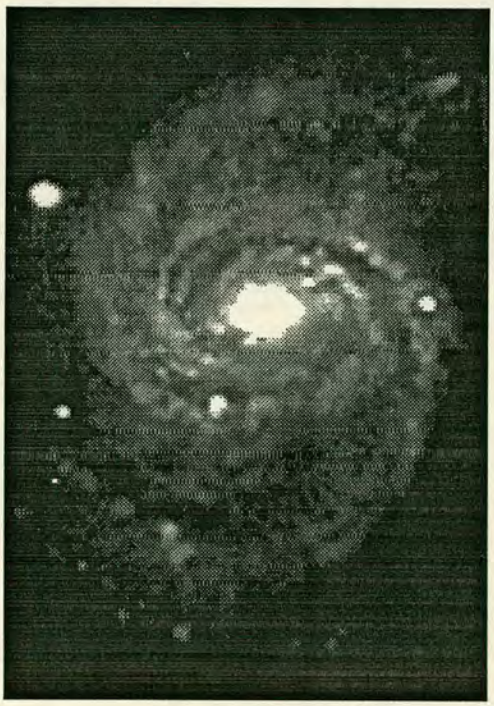
Reconstructions with $\lambda = 10^{-6}$ and $\lambda = 10^{-5}$



Figs. 5.5 and 5.6

The blurred image and reconstruction with $\lambda = 0.05$





Figs. 5.7 and 5.8

Reconstructions with $\lambda = 0.1$ and $\lambda = 0.4$

Thus SVD makes the least number of assumptions of all three methods. This gives it an advantage in that it can cope with coherent as well as incoherent light whereas the others cannot, but as we have seen, it has two disadvantages: it is not very robust with respect to noise and it cannot cope with large space-bandwidth products. But, as we saw in section 5.3.4, both MaxEnt and CAR can cope with noisy images and large space-bandwidth products. This is because the additional assumptions built into the other methods cut down the class of possible solutions and make them more robust.

We saw how Sasaki and Yamagami and Maeda and Murata incorporated further assumptions into SVD. They claim that this makes SVD more robust with repect to noise as we would expect from the above argument. They do not mention whether it allows SVD to cope with larger space-bandwidth products, but I expect that it probably will. No direct comparision of the methods of Sasaki et al. with MaxEnt has been done. It would be interesting to see if they all yield the same reconstructed object from the same blurred image, when they all make the same a priori assumptions.

In summary, pure SVD is flexible in that it can be applied to both coherent and incoherent illumination. However, it is not robust. We can make it more robust by incorporating other a priori information, at the expense of reducing its flexibility.

SVD is a much faster process than MaxEnt and CAR because it involves only the computation of a few scalar products whereas the other methods are iterative. In both MaxEnt and CAR each iteration involves calculating several Fourier transforms and so can take several CPU minutes. SVD thus has the advantage of speed.

Chapter Six

Conclusions and Future Work

Section 5.1 Summary of thesis

In this thesis I have done three main things: a) I have presented an algorithm for the computation of the singular functions of a two-dimensional imaging system where object and pupil differ in shape. b) I have presented numerical examples of SVD and EVD with and without additive noise. c)I have presented results of an experimental test of SVD using incoherent illumination.

5.1.1 Computing the Singular Functions

In chapter two I described my algorithm for computing the singular functions of a two-dimensional imaging system where the object and pupil differ in shape. All previous workers have computed the singular functions only for either one-dimensional systems or for two-dimensional systems where object and pupil have the same shape.

The essence of the algorithm is to express the imaging operator, A, as a sequence of subroutines. The most important subroutine is called the "finite Fourier transform" subroutine and it calculates a finite region in the centre of the Fourier transform of a finite object. The singular functions of this operator were found using the power method followed by Hotelling's deflation.

5.1.2 Numerical Examples

In chapter three I presented some numerical examples of SVD and EVD. These were computer simulations based on the system with square object and circular pupil.

Firstly I presented examples of EVD and SVD without noise. I showed that they both returned values of the reconstruction coefficients which were equal to those of the original object. This demonstrated the internal consistency of the algorithms. Secondly I added random numbers to the images and then carried out the reconstructions again. SVD was seen to be superior to EVD in that more of the singular coefficients of the object could be estimated.

5.1.3 Experimental Work

In chapter four I presented results of an experimental test of SVD. Images were generated using an optical bench. The images were deliberately bandlimited by placing an aperture in the Fourier plane. Incoherent illumination was used. The object and pupil were both rectangular, therefore separable singular functions were used. SVD was carried out using both coherent and incoherent singular functions. The reconstruction coefficients were compared to the object coefficients for both techniques. Three different objective tests were applied to the reconstructed objects to see how well they had been restored. The incoherent method was found to be better. Discrepancies between the object and reconstruction coefficients were thought to be due to distortion in the camera/framestore system.

Section 5.2 Suggestions for future work

5.2.1 Other shapes of object and pupil

So far I have calculated the singular functions only for the case of the square object and circular pupil. This is a relatively simple case since both regions have quarter-turn symmetry and, as we saw in chapter two, this greatly simplifies the algorithm. The more general case where the object and pupil have no particular symmetry would be much harder to compute. We would not be able to simplify the finite Fourier transform routine as we did in equn. 2.3.6 by substituting sines or cosines for the complex exponentials. Furthermore, it would be difficult to separate degenerate or near-degenerate eigenfunctions as we did in chapter two by sorting the functions into different symmetry sets. This would mean that the power method would take a very long time to converge. Further work needs to be done on solving these problems.

5.2.2 Comparison of SVD with other techniques

In chap. 5 we looked at some other techniques for achieving super-resolution: the Gerchberg algorithm, MaxEnt and CAR. It would be useful to compare the re-

constructions given by these techniques with those given by SVD when starting from the same blurred image.

Both SVD and Gerchberg assume the same a priori information, viz. that the object is space-limited. We should then expect that the two algorithms should give identical reconstructions when given the same starting data. We could test this by taking a reconstruction produced by the Gerchberg algorithm and finding the scalar product of it with each of the singular functions, u_k. These scalar products should be identical to the reconstruction coefficients given by SVD.

In chap. 5 we saw that MaxEnt and CAR could cope with much larger space-bandwidth products than pure SVD but it is possible that when other forms of a priori information are built into SVD that its performance would then be comparable. My SVD algorithm could be developed along the lines suggested by Sasaki and Yamagami and Maeda and Murata and then applied to the same images as we saw in chap. 5. We could then compare the results with MaxEnt and CAR.

5.2.4 Experimental Work

The experiment described in chapter four used incoherent illumination. A further experiment might use coherent illumination. This would be much more difficult to carry out since we would need to measure the complex amplitude of the image rather than merely its intensity. This requires the use of interferometry. The most likely application for this would be scanning laser microscopy. A design for such a system has been put forward by Hunter [37].

SVD is well-suited for processing coherent images. This is one of its advantages over techniques such as Maximum Entropy which require the object and image to be non-negative, and are thus suitable only for incoherent images.

Appendix A

Incoherent vs. Coherent Singular Functions in the Presence of Noise

In incoherent imaging the transfer function is no longer a simple top-hat function but has the form shown in fig. 1.4. We are now faced with a choice. We can either follow the path suggested by Frieden in which we divide the Fourier transform, G(k), of the image by the transfer function, H(k), and then use the coherent singular functions to carry out the SVD. Or we can follow the path suggested in Bertero and Pike's second paper, in which we carry out the SVD using incoherent singular functions.

It is not obvious which of these two methods will be better in the presence of noise. The first has the problem that H(k) has values which are close to zero and so division becomes unstable. The second has the problem that the singular values are smaller than in the first method and so division by them is less accurate.

Let us consider a noise distribution, n(k), such that

$$< n(k)n^*(k') > \quad = \quad \varepsilon^2 \delta(k - k') \qquad A.1$$

We can calculate the contribution made by this noise to the two techniques.

First, let us consider Frieden's technique. Let β_k be the scalar product of the kth singular function with the noise

$$\beta_k \quad = \quad \int_{-2\Omega}^{2\Omega} \psi_k^*(x) \frac{n(x)}{H(x)} \, dx \qquad A.2$$

Then the time-averaged value of β_k is given by

$$< |\beta_k|^2 > \quad = \quad \varepsilon^2 < \int_{-2\Omega}^{2\Omega} \frac{\psi_k^*(x)\psi_k(x)}{H^*(x)H(x)} \, dx > \qquad A.3$$

We can calculate the value of A.3 numerically. But we have to avoid the problem of dividing by zero at the points $k = \pm 2\Omega$. To do this 32 equally spaced sample points were placed on the interval $[-2\Omega, 2\Omega]$ with the left-hand sample point at

$k = -31\Omega/16$. A.3 was approximated by a discrete scalar product over these points. There is thus no need to calculate the value of the integrand of A.3 at $k = \pm 2\Omega$. The contribution of the noise to the final reconstructed object is given by

$$\frac{\sqrt{< |\beta_k|^2 >}}{\lambda_k} \qquad A.4$$

where λ_k is the singular value. The quantity in equn A.4 is tabulated in the right-hand side of table A.1. for each of the first nine singular functions with the space-bandwidth product, $c = 2\pi$. ε has been set equal to unity.

In the second technique the coefficient, β_k, of the kth singular function with the noise is given by

$$\beta_k \quad = \quad \int_{-2\Omega}^{2\Omega} \phi_k(x) n(x)\, dx \qquad A.5$$

where ϕ_k is the kth incoherent singular function. The time-averaged value of β_k is given by

$$< |\beta_k|^2 > \quad = \quad \varepsilon^2 \qquad A.6$$

and the contribution to the final reconstructed image is given by

$$\frac{\varepsilon^2}{\alpha_k} \qquad A.7$$

where α_k is the incoherent singular value - different from λ_k in equn. A.4. These values are tabulated on the LHS of table A.1, again for $c = 2\pi$. As can be seen the incoherent values are smaller so the incoherent method is superior to the coherent method for this particular value of c.

The values shown in table A.1 were calculated for one particular value of c, 2π. I cannot find a general proof which shows that the incoherent fucntions are better for all values of c. However, I can say that table A.1 is typical for values of c in the range where SVD is useful.

Table A.1

Function No.	Coherent	Incoherent
1	1.5	1.2
2	3.1	1.6
3	7.3	2.4
4	16.0	4.2
5	36.3	9.6
6	101.6	27.9
7	346.2	101.8
8	1378.5	448.2
9	6255.0	2290.4

Bibliography

[1] Bertero and Pike, Opt. Acta, vol.29, 6, p727-746, (1982)

[2] Bertero and Pike, Opt. Acta, vol.29, 12, p1599-1611, (1982)

[3] Bertero and Pike et al, Opt. Acta, vol.31, 2, p181-201, (1984)

[4] Bertero and Pike et al, Opt. Acta, vol.31, 8, p923-946, (1984)

[5] Slepian and Pollak, Bell Syst. Tech. J., vol.40, p43, (1961)

[6] Slepian D., Bell Syst. Tech. J., vol.43, p3009, (1964)

[7] Slepian D., J. Math. and Phys., vol.44, p99, (1965)

[8] Barnes C.W., J. Opt. Soc. Am., vol.56, p575, (1966)

[9] Harris J.L., J. Opt. Soc. Am., vol.54, p931, (1964)

[10] Frieden B.R., J. Opt. Soc. Am., vol.57, p1013, (1966)

[11] Frieden B.R., Progress in Optics, vol.IX, p311, (1971)

[12] Rushforth and Harris, J. Opt. Soc. Am., vol.58, p539, (1968)

[13] McCutchen C.W., J. Opt. Soc. Am., vol.57, p1190, (1967)

[14] Pask C., J. Opt. Soc. Am., vol.66, p68, (1976)

[15] Toraldo di Francia, J. Opt. Soc. Am., vol.45, p497, (1955)

[16] Toraldo di Francia, J. Opt. Soc. Am., vol.59, p799, (1969)

[17] Cathey, Frieden et. al., J. Opt. Soc. Am. A, vol.1, p241, (1984)

[18] Flammer C. *Spheroidal Wave Functions*, Stanford University Press (1957)

[19] Saleh B., J. Opt. Soc. Am., vol.67, p71, (1977)

[20] Cox and Shepherd, J. Opt. Soc. Am. A, vol.3, p1152, (1986)

[21] Fellget and Linfoot, Proc. R. Soc. London A, vol.247, p369, (1955)

[22] Lukosz W., J. Opt. Soc. Am., vol.56, p1463, (1966)

[23] Lukosz W., J. Opt. Soc. Am., vol.57, p932, (1967)

[24] Shannon C.E. *The Mathematical Theory of Communication*, Illinois Uni. Press, (1948)

[25] Gull and Skilling, IEE Proc., vol.131 F, p646, (1984)

[26] Gull and Skilling *Maximum Entropy and Bayesian Methods in Inverse Problems*, Eds. Gandy and Smith, p83, D. Reidel, Dordrecht, (1985)

[27] Jaynes E.T., *Papers on probability, statistics and statistical physics*, Ed. Rosencrantz, D. Reidel, Dordrecht, (1983)

[28] Frieden B.R., J. Opt. Soc. Am., vol.62, p511, (1972)

[29] Frieden B.R., J. Opt. Soc. Am., vol.62, p1202, (1972)

[30] Fedotowsky and Boivin, Quarterly of Applied Maths, XXX, 3, p235, (1972)

[31] Goodman J.W. *Introduction to Fourier Otics*, McGraw-Hill, N.Y., (1968)

[32] Newland D.E. *Random Signals and Spectral Analysis*, Longman, London, (1984)

[33] Brigham E.O. *The Fast Fourier Transform*, Prentice-hall, (1974)

[34] Ralston A. *A First Course in Numerical Analysis*, McGraw-Hill, N.Y., (1965)

[35] Bershad, J. Opt. Soc. Am., vol.59, p157, (1969)

[36] Sasaki and Yamagami, Opt. Letters, vol.10, p433, (1985)

[37] Hunter I., *Scanning Imaging Technology*, Eds. Wilson and Balk, Proc SPIE 809, p138, (1987)

[38] Walker J.G., Opt. Acta, vol.30, 9, p1197-1202, (1983)

[39] Gerchberg R.W., Opt. Acta, vol. 21, p709, (1974)

[40] Maeda and Murata, J. Opt. Soc. Am. A, vol.1, p28, (1984)

[41] Ripley B.D. *Statistical Inference for Spatial Processes*, Cambridge University Press, (1988)

[42] Molina and Ripley, *forthcoming publication in* J. Appl. Stats.

[43] Gull and Daniel, Nature, vol.272, p686, (1978)

[44] Ripley B.D. *Spatial Statistics* Wiley, New York, (1981)