Analysis of Image Sequences using Redundant Representations

Thesis by

Subrata Rakshit

In Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

California Institute of Technology

Pasadena, California

1994

(Defended Jan. 1994)

Acknowledgements

I would like to thank my research advisor, Charles H. Anderson, for his support and guidance over the years. His insights when I was stuck, his optimism when I was diffident and his caution when I was reckless went a long way towards ensuring the completion of this thesis. I would like to thank my advisor at Caltech, Demetri Psaltis, for getting me started in graduate school. His support and confidence in me during the early years in graduate school made all subsequent developments possible by keeping me in graduate school. I would also like to thank my 'third advisor', David Van Essen, for his altruism in supporting an engineering student on detached duty in the department of Anatomy and Neurobiology, Washington University at St Louis.

Success in graduate school depends to a large extent on the formative years leading up to it. For that I thank my parents for their nurturing, encouragement and strength, no less when I was 20,000 mi. away than when I was home. I would like to thank all my teachers at L.F.S., La Martiniere for Boys, I.I.T. Bombay and Caltech for their motivation and support.

For their companionship, humor and spirited discussions I wish to thank all members of the Psaltis group at Caltech and the Van Essen group at Wash. Univ, especially Mark Neifeld, Hsin-Yu Li, Yong Ciao, Jack Gallant, Bruno Olshausen, Jim

Lewis, Shashi Kumar and Tom Coogan. My thanks also to the secretarial staff in EE Dept and the administrative staff in the Graduate Office at Caltech and the staff at Dept of Anatomy, Wash. Univ for their help in navigating the red tape involved in a student on F-1 visa going on detached duty.

This research was supported in part by the EE Dept, Caltech and ONR Grant N00014-89-J-1192.

Abstract

This thesis focuses primarily on two techniques often used for low level analysis of motion in image sequences, the computation of optical flow fields and the determination of the spatio-temporal frequencies present. The complexity, cost and accuracy of image sequence processing is shown to be related to the manner in which the information is represented. The optical flow problem is formulated in terms of the basis functions underlying the discrete representation of images. Besides giving good results for a wide variety of inputs, this formalism highlights the benefits of a redundant representation of the input image in terms of reducing the overall cost of analysis. The spatio-temporal filter banks are analyzed using information theory. This approach makes explicit use of the input *prior* and provides an objective way of comparing the cost effectiveness of filter banks of different sizes. The output is used to generate a probability distribution over selected parameters in order to provide higher visual modules with a richer input. The formalism developed here provides a means of measuring the redundancy in filter bank outputs. This redundancy is shown to provide robustness to noise within the system. Likewise, a redundant representation of the image can also be used for error correction in case of corruption during storage or transmission. An algorithm for using the Burt Laplacian pyramid for such error correction is also developed and demonstrated.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Effects of Redundancy

With the rapid expansion of the field of image processing, it has become necessary to differentiate between the requirements of various tasks. This is partly due to the ever increasing sizes of the image sequences and partly due to the increased sophistication of analysis that is demanded. In order to meet all these requirements with available resources it is important to optimize each type of operation to utilize resources as efficiently as possible. For archival and transmission systems, this would mean using the most compact representation in order to save storage costs and bandwidth. For computational tasks, it is the amount of computation that must be minimized. This thesis examines the cost for the two different types of computation most frequently carried out by low level image processing or visual systems.

An unfortunate side effect of compactly representing an image by sampling at the critical (Nyquist) rate is that the information is encoded globally. The information about the highest frequency features in any part of the image is encoded in pixels distributed over a wider area about that point. This poses problems even for static (single frame) analysis since feature matching routines must use bigger kernels. The problem is worse for motion since the movement of a small feature, over a relatively small distance, can cause global changes in the resulting new representation. This property is best seen for wavelet representations where a small shift in the input can completely change the representation. Thus, motion analysis in such a context acquires a huge computational cost. This problem was pointed out in [Simoncelli et al., 1992] where the authors stressed the need for having representations that were shiftable. One conclusion of that work was that since critically sampled representations like the wavelets typically violates the Nyquist criterion for individual subbands, the information moves from one band to another under translation. The steerable pyramids they designed to overcome this problem turned out to be less compact than the wavelet but were considered computationally more desirable. In this thesis, this issue is studied further in the context of calculating optical flow from an image sequence and estimating the cost of obtaining results of a given accuracy as a function of the input frequency, since the (digital) input frequency depends on the sampling rate. How far can computation costs be reduced by using more redundant representations and what are the consequences of the redundancy in natural images ? These questions are answered with the help of an analysis based on the explicit use of basis functions.

The use of basis functions is shown to be the key to interpolation and the estimation of derivatives in images. The estimation of derivatives in a discrete representation can be justified analytically by performing the derivative operation on the underlying continuous basis. The extent to which information is encoded locally, reflected in the compactness of the basis functions, determines the cost of estimating derivatives with a given precision. The use of different basis functions allows the interpolation and calculation of derivatives with varying precision. These are essential capabilities for a variety of low level tasks. Since it is the low level operations that need to be performed directly on the image, the image representation can be expected to have its biggest effect on the cost and accuracy of such systems. The higher levels of analyses based on identified features, tokens, edges and flow fields should remain unaffected.

An alternative to the computation of optical flow is the use of a bank of spatiotemporal filters, the underlying idea being that a moving pattern would have its energy in the fourier domain concentrated about a line through the origin. The cost of building a filter bank to cover all scales, speeds and orientation is huge. An attempt to reduce this cost by requiring only a few filters was made by the introduction of steerable kernels that allowed one to use a finite set of filters to compute the outputs of filters distributed over a continuum of orientation and/or scale. Given two families of kernels, one of which is spanned by n filters and the other by m filters, how does one compare the cost effectiveness of the two families? Since the filter outputs usually undergo a nonlinear operation and are used for making decisions about the image content, they can be regarded as detectors of analog features. The analysis of their cost effectiveness is best done in terms of information delivered. To this end, an information theoretic analysis can be done to quantify the effectiveness of a set of filters. The output can be characterized by its expected information content, redundancy and cost. When continuous variables are being estimated, it is often desirable to cast the output in terms of a probability distribution. A probability measure is developed on the output to this end and the ability to incorporate top down constrains and pool information across multiple parameters is demonstrated. The issue of hyperacuity can also be addressed in this framework. It is well known that overlapping broadly tuned filters are used by biological systems to make fine judgements. How fine can this get and what kind of tuning curve is best suited for such applications? These are relevant questions for engineering applications as well since they could reduce the number of filters required to estimate input image parameters with a certain precision. Finally, there ought to be a way of incorporating prior knowledge about the inputs into the filter bank design.

The effect of redundancy in the output of an analysis system is less straightforward but no less important. Redundancy in the output of filters (feature detectors) is shunned since it is taken to imply that the system is not optimally designed and computational costs can be reduced by making the filters more independent. While that is true to a certain extent, it is not true that making the filters completely independent would necessarily make the system optimal. If the information content of the filter bank is maximized at a fixed computational cost, the output still contains a significant amount of redundancy. In this context, the redundancy can be considered to be benign, if not desirable. On the other hand, the redundancy does make a big difference in the presence of noise *within* the system, as would be the case for computers if there were no error correcting facilities built into the disk read/write mechanisms. The omnipresence of such systems in modern computers enables engineers to design algorithms on the assumption that results can be stored, transmitted and retrieved without errors. Biological systems, on the other hand, are noisy and hence the evolution of biological visual systems may be expected to be influenced by the need to have redundant encoding of their outputs. While detailed case studies of biological systems will not be a part of this thesis, it is a topic of further research and serves to highlight the importance of understanding the role of redundancy in the output.

Since redundancy in the output can be used for error correction in the presence of noise, it is natural to ask if the same cannot be done for the input. Given the need for a redundant representation of the input as outlined earlier, is it possible to use such a representation for error correction ? It will be shown how that promise can actually be realized for one such redundant representation, the Burt laplacian pyramid. This multiresolution representation of an image can be considered to be one huge codeword and noise added to this representation can be detected and removed. It is worth emphasizing that the decoding scheme for this system as well as the one developed for decoding the redundant outputs of filters have the property of degrading gracefully in the presence of excessive noise. This follows from the fact that similar inputs produce similar codewords. Such a property is extremely desirable for applications where it is better to make the best guess than to give up completely.

1.2 Thesis Overview

In Chapter 2 the standard problem of calculating optical flow from two successive frames is addressed making explicit use of the underlying basis functions associated with the discrete representation of images and flow fields. The basis functions provide an accurate way of evaluating derivatives of images and performing interpolation, both of which are essential for the motion estimation process. A given precision can be obtained by trading off basis function complexity with the sampling rate. Computational costs are shown to have a minimum at a sampling rate above the Nyquist limit, indicating the desirability of overcomplete representations for motion estimation. These results are utilized to construct a multiresolution motion estimation procedure that was tested using artificial and real images with non-rigid motion and large displacements. The major conclusion is that linear basis functions are adequate for motion estimation using Gaussian pyramid representations of natural images, except possibly at the highest resolution. The use of Laplacian pyramids or the presence of high frequency features moving independent of any low frequency features at the highest resolution requires the use of larger basis functions. In such cases, a less expensive alternative is to use a costly interpolating function to expand the image thereby lowering its spatial frequencies and then use linear basis to analyze this expanded image. This chapter shows why the redundant encoding of the input to an image sequence analyzer is desirable. Brief descriptions of pyramid formation will be given in Chapter 3 and Chapter 4. For a comprehensive description of Gaussian and Laplacian pyramids, see [Burt, 1983], [Burt and Adelson, 1983].

In Chapter 3 the use of spatiotemporal filters to characterize an image sequence by its frequency contents is studied and the effect of redundancy in the output is examined. The approach here is to design filters such that the filter banks, regarded as detectors of analog quantities, are maximally informative about their input. An information theoretic analysis shows that for a fixed cost the optimum filters are not necessarily the most narrowly tuned. For the task of estimating a single value for each input parameter (like orientation) there is an incremental gain in information at huge increase in cost for filter banks using a set of filters tuned narrower than a certain filter width. This calls into question the advisability of building filter banks with a large number of very narrowly tuned filters for increasing the estimation accuracy for simple inputs. Hyperacuity in orientaion can be achieved by a collection of moderately tuned filters so long as the input has only one dominant orientation. The effects of filter tuning widths, number of filters and bits of precision in the output are explored. The output of a filter bank of moderately tuned filters is shown to encode information in a redundant manner, and the right decoding process not only extracts the most information from the output but is also robust to noise in the system. The analysis shows the relationship between computation cost and information extracted for different filter banks, and the relationship between estimation accuracy and input complexity. The role of the *prior* in the filter design shows how the design process may take advantage of prior information about the input in order to maximize the expected information content of the outputs. An efficient pyramid based implementation for orientation, spatial frequency and temporal frequency estimation is given and the results discussed. This chapter shows how the maximally informative output is one that encodes the information in a redundant format and how this output can be decoded to both provide estimates of maximally probable input parameters as well as probability distributions covering the entire input range.

In Chapter 4 the error correcting properties of one particular redundant representation is explored. The Burt pyramid was developed as a means of decomposing images into multiscale representations. Unlike the wavelet transform, which is a complete orthonormal transformation, the pyramid transforms are overcomplete. In addition, the Burt pyramid has an exact reconstruction rule. The consequences of the redundancy and exact reconstruction are examined and a way to characterize the pyramids is given. The issue of error detection is addressed next, followed by the development of an error correction algorithm. Several examples are given to show the performance for different amounts and types of noise.

Chapter 2

Computation of Optical Flow Using Basis Functions

2.1 Introduction

The problem of estimating optical flow from a sequence of images has been studied extensively. Beginning with the work of Horn [Horn and Schunck, 1981], [Horn, 1986], one approach has been to impose a constant brightness assumption to the image sequence and then use spatial and temporal derivatives of the gray-scale images to solve for the displacement. Since this left the problem underconstrained, further constraints were imposed by way of regularization of the optical flow field. Many variations on the original scheme for solving the resultant equations have been proposed [Haralick and Lee, 1983], [Tretiak and Pastor, 1984], [Nagel, 1987]. Motion estimation algorithms using a multiresolution framework [Bergen et al., 1992], [Weber and Malik, 1992] have also been based on the computation of optical flow. All these techniques require the calculation of spatial derivatives and a first-order approximation. This leads to large errors in the presence of high spatial frequencies and large displacements. Some methods, in an attempt to overcome the aperture problem, invoke second derivatives and are thus even more sensitive to errors in estimation of derivatives [Tretiak and Pastor, 1984], [Nagel, 1987].

The development here is based on the continuous basis functions that underlie the discrete representation of images. The optical flow field is regarded as a continuum represented by its own basis functions and their coefficients. The constant brightness assumption then leads to a system of coupled differential equations where the derivative operation has to be performed only on the continuous basis functions. A multiresolution approach allows a coarse to fine estimation with an image reregistration at each stage which keeps the first-order approximation valid even for large displacements ($>$ 30 pixels). By chosing the appropriate basis function, one can handle high spatial frequencies and irregular flow fields.

In this chapter, we will first derive the standard equations for the optical flow field and then introduce the basis functions. An examination of the resulting equations and similarities with previous work will be made. Next, we take a look at the various candidate basis functions and evaluate their strengths and weaknesses. The relationship between image spectrum and cost of analysis is used to derive an optimum sampling rate for image processing. Finally, results for synthetic and real image sequences will be presented.

2.2 Gradient Based Optical Flow Field

Given two images separated by a time interval Δt, the constant brightness assumption implies that all changes in the image are due to motion of image components and not due to altered lighting conditions or occlusion. Hence, there exists a warping $\Delta x(x, y), \Delta y(x, y)$ that will map the first image onto the second. In the absence of a perfect solution, the problem can be formulated as an optimization problem in Δx

and Δy.

$$E = \int [I(x,y,t+\Delta t) - I(x+\Delta x, y+\Delta y, t)]^2 dxdy \tag{2.1}$$

In order to do the minimization of E w.r.t. $\Delta x, \Delta y$ by gradient descent, it is necessary to express E as an explicit function of $\Delta x, \Delta y$. This can be done by making the first-order approximation

$$I(x+\Delta x, y+\Delta y) = I(x,y) + \Delta x.I_x(x,y) + \Delta y.I_y(x,y).$$

This gives

$$E = \int\int [\Delta I(x,y) - \Delta x.I_x(x,y) + \Delta y.I_y(x,y)]^2 dxdy, \tag{2.2}$$

where

$$\Delta I(x,y) = I(x,y,t+\Delta t) - I(x,y,t).$$

In order to regularize the flow field, a cost term $-\log \mathcal{P}$ can be added where $\mathcal{P}$ is the probability of the flow field given by

$$\mathcal{P} = \int\int exp(-((\Delta x_{ave}(x,y) - \Delta x(x,y))^2 + (\Delta y_{ave}(x,y) - \Delta y(x,y))^2)/\sigma^2)dxdy$$

$\Delta x_{ave}(x,y), \Delta y_{ave}(x,y)$ are local averages.

In going from Eq(1), which would have required a random search or blockwise quadratic fit [Anandan, 1989],to Eq(2) which can be solved by gradient descent, we have committed ourselves to calculating derivatives on images and limiting the analysis to displacements that are small enough for the first-order Taylor series to be accurate. The calculation of derivatives is a problem not only because of its noise sensitivity but also due to the fact that the images stored in computers are discretized versions of continuous functions. By performing the derivative operation on the underlying continuous basis functions, derivatives of images can be calculated in an analytic fashion. The problem of large displacements can be partly solved by using a multiresolution approach, [Burt et al., 1992], [Simoncelli, 1993]. Since displacements need to be small with respect to the "local" frequency, if the motion estimates for the lowest frequency components in an image are used to re-register the image before estimating the motion of features at the next level of resolution, then large displacements can also be estimated accurately. The requirement of re-registration leads to the problem of interpolation and this requires the use of a continuous basis function as well.

2.3 Basis Function Formulation

The discrete image $I(m,n)$ and the continuous image $I(x,y)$ are related by the analysis and synthesis functions

$$I(m,n) = \iint I(x,y)\phi_{mn}(x,y)dxdy \quad \text{Analysis} \tag{2.3}$$

$$I(x,y) = \sum_m \sum_n I(m,n)\Phi_{mn}(x,y) \quad \text{Synthesis} \tag{2.4}$$

$$\sum_m \sum_n \phi_{mn}(x',y')\Phi_{mn}(x,y) = \delta(x-x', y-y'). \tag{2.5}$$

Equality in eqns(2.3,4) holds if the functions are a complete set and eqn(2.5) assures perfect reconstruction. Our chief interest here is in eqn(2.4) which relates the discrete $I(m,n)$ to the continuous $I(x,y)$, for detailed discussion of analysis and synthesis functions and filter banks see [Simoncelli and Adelson, 1991], [Oppenhiem and Schafer, 1975]. For a choice of Φ_{mn} that does not satisfy eqn(2.5), eqn(2.4) can still be a good approximation. As will be seen, considerations of computational cost and time will restrict choice of Φ_{mn} so as to rule out the use of sinc and other nonlocal functions.

Differentiating eqn(2.4) w.r.t x and y yields

$$I_x(x,y) = \sum_m \sum_n I(m,n)(\Phi_{mn}(x,y))_x \tag{2.6a}$$

$$I_y(x,y) = \sum_m \sum_n I(m,n)(\Phi_{mn}(x,y))_y. \tag{2.6b}$$

The derivatives of the basis function Φ_{mn} thus give the appropriate coupling coefficients for using the image pixels $I(m,n)$ to estimate the derivative of the image. It should be noted that eqn(2.6) can also be used to estimate the derivatives off grid points, *i.e.*, interpolate the derivatives.

The desired displacement field can also be expressed in a similar manner. Since the displacement field may be smoother (or more irregular) than the image, one may assume that it has a different basis function, Ψ_{mn}. If this distinction is not required, one can always set $\Psi = \Phi$. The displacement fields Δx and Δy are

$$\Delta x(x,y) = \sum_m \sum_n P(m,n)\Psi_{mn}(x,y) \tag{2.7a}$$

$$\Delta y(x,y) = \sum_m \sum_n Q(m,n)\Psi_{mn}(x,y). \tag{2.7b}$$

The estimation of displacement requires the estimation of $P(m,n)$ and $Q(m,n)$.

2.3.1 Displacement Field Estimation

The added assumption shall now be made that the basis has the property

$$\Phi_{mn}(x,y) = \Phi_m(x)\Phi_n(y) \tag{2.8a}$$

$$\Phi_m(x) = \Phi_0(x-m). \tag{2.8b}$$

This leads to significant computational simplification without imposing undue constraints since all widely used basis functions have the above property. Eqn(2.2) can now be reformulated in terms of the basis functions and their coefficients. For convenience, let us denote $I(x,y,t)$ as $I(x,y)$ and $I(x,y,t+\Delta t)$ as $I_2(x,y)$.

$$\begin{aligned}
I(x,y) &= \sum_{mn} I(m,n)\Phi_m(x)\Phi_n(y) \\
\Delta I(m,n) &= I(m,n) - I_2(m,n) \\
I_x(x,y) &= \sum_{mn} I(m,n)\Phi'_m(x)\Phi_n(y) \\
I_y(x,y) &= \sum_{mn} I(m,n)\Phi_m(x)\Phi'_n(y) \\
\Delta x(x,y) &= \sum_{mn} P(m,n)\Psi_m(x)\Psi_n(y) \\
\Delta y(x,y) &= \sum_{mn} Q(m,n)\Psi_m(x)\Psi_n(y)
\end{aligned}$$

Eqn(2.2) can now be used to express E in terms of the unknowns $P(m,n), Q(m,n)$. Differentiating w.r.t $P(M,N), Q(M,N)$ gives us the required updating rules for iteratively minimizing E and finding the desired $P(m,n), Q(m,n)$. The regularization terms arising from $-log\mathcal{P}$ are shown only in the update equations.

$$\begin{aligned}
E = \iint \Big[& \sum_{mn} \Delta I(m,n)\Phi_m(x)\Phi_n(y) \\
& - \sum_{mn} P(m,n)\Psi_m(x)\Psi_n(y) \sum_{pq} I(p,q)\Phi'_p(x)\Phi_q(y) \\
& - \sum_{mn} Q(m,n)\Psi_m(x)\Psi_n(y) \sum_{pq} I(p,q)\Phi_p(x)\Phi'_q(y) \Big]^2 \, dxdy \qquad (2.9a)
\end{aligned}$$

$$\begin{aligned}
\frac{\partial E}{\partial P_{MN}} = & \sum_{mn}\sum_{pq} \left\langle \Phi_m\Phi_n\Phi'_p\Phi_q\Psi_M\Psi_N \right\rangle \Delta I(m,n) I(p,q) \\
& - \sum_{mn} P_{mn} \sum_{pq}\sum_{st} \left\langle \Psi_m\Psi_n\Phi'_p\Phi_q\Phi'_s\Phi_t\Psi_M\Psi_N \right\rangle I(p,q)I(s,t)
\end{aligned}$$

$$
\begin{aligned}
& -\sum_{mn} Q_{mn} \sum_{pq}\sum_{st} \left\langle \Psi_m \Psi_n \Phi_p \Phi'_q \Phi'_s \Phi_t \Psi_M \Psi_N \right\rangle I(p,q) I(s,t) && (2.9b) \\
\frac{\partial E}{\partial Q_{MN}} &= \sum_{mn}\sum_{pq} \left\langle \Phi_m \Phi_n \Phi_p \Phi'_q \Psi_M \Psi_N \right\rangle \Delta I(m,n) I(p,q) \\
& -\sum_{mn} P_{mn} \sum_{pq}\sum_{st} \left\langle \Psi_m \Psi_n \Phi'_p \Phi_q \Phi_s \Phi'_t \Psi_M \Psi_N \right\rangle I(p,q) I(s,t) \\
& -\sum_{mn} Q_{mn} \sum_{pq}\sum_{st} \left\langle \Psi_m \Psi_n \Phi_p \Phi'_q \Phi_s \Phi'_t \Psi_M \Psi_N \right\rangle I(p,q) I(s,t) && (2.9c) \\
P^{new}_{MN} &= P^{old}_{MN} - \epsilon \frac{\partial E}{\partial P_{MN}} + \lambda(\overline{P_{MN}} - P_{MN})^2 \\
Q^{new}_{MN} &= Q^{old}_{MN} - \epsilon \frac{\partial E}{\partial Q_{MN}} + \lambda(\overline{Q_{MN}} - Q_{MN})^2
\end{aligned}
$$

where $< . >$ denotes the integral over the entire image. Note that the integration has to be done over the continuous variables x, y and involves only the continuous basis functions. The discrete quantities like $I(m,n)$ are only involved in discrete operations like summation, addition and subtraction. The basis functions provide an analytic method for deriving coupling constants (or convolution kernels) that yield solutions of any desired accuracy. The question of how much computation has to be done, and when, is addressed in the next section. The issue of accuracy is discussed subsequently.

2.3.2 Computational Issues

In eqns(2.9), the unknowns are $P(m,n), Q(m,n)$. Of the rest, the images will vary from frame to frame while the basis functions will always remain the same. Thus quantities that depend only on the basis functions can be precomputed (to keep open the option of different basis functions, one may compute more than one set). Since this is a once in a lifetime calculation and involves only file I/O, this constitutes a memory cost rather than a computation cost. After the images have been input, the quantities that depend only on the basis functions and $I(m,n), \Delta I(m,n)$ can be computed. These are computations that do not involve iteration and will be called overhead costs. The final summations that include $P(m,n), Q(m,n)$ will have to be iterated as $P(m,n), Q(m,n)$ get updated and this will be called cost of solving eqn(2.9). The integrals and summations in eqn(2.9) extend over the entire image area. However, if the basis functions are localized, then the summations and integrals need to be evaluated over local regions since the rest of the coupling coefficients will be zero. Hence the computation costs are dependent on the width of the basis functions.

Let the functions Φ, Ψ be such that

$$\Phi_0(x) = \quad 0 \qquad \|x\| > k/2 \tag{2.10}$$

$$\Psi_0(x) = \quad 0 \qquad \|x\| > h/2. \tag{2.11}$$

Due to the choice of separable basis functions, the inner products of eqn(2.9) can be expressed as a product of an x integration and a y integration. For brevity let us label them as

$$\begin{aligned}
\left\langle \Phi_m \Phi_n \Phi'_p \Phi_q \Psi_M \Psi_N \right\rangle &= \left\langle \Phi_m \Phi'_p \Psi_M \right\rangle . \left\langle \Phi_n \Phi_q \Psi_N \right\rangle &&\equiv A1x_{mpM}.A2x_{nqN} \\
\left\langle \Phi_m \Phi_n \Phi_p \Phi'_q \Psi_M \Psi_N \right\rangle &= \left\langle \Phi_m \Phi_p \Psi_M \right\rangle . \left\langle \Phi_n \Phi'_q \Psi_N \right\rangle &&\equiv A1y_{mpM}.A2y_{nqN} \\
\left\langle \Psi_m \Psi_n \Phi'_p \Phi_q \Phi'_s \Phi_t \Psi_M \Psi_N \right\rangle &= \left\langle \Psi_m \Phi'_p \Phi'_s \Psi_M \right\rangle . \left\langle \Psi_n \Phi_q \Phi_t \Psi_N \right\rangle &&\equiv B1x_{mpsM}.B2x_{nqtN} \\
\left\langle \Psi_m \Psi_n \Phi_p \Phi'_q \Phi'_s \Phi_t \Psi_M \Psi_N \right\rangle &= \left\langle \Psi_m \Phi_p \Phi'_s \Psi_M \right\rangle . \left\langle \Psi_n \Phi'_q \Phi_t \Psi_N \right\rangle &&\equiv C1x_{mpsM}.C2x_{nqtN} \\
\left\langle \Psi_m \Psi_n \Phi'_p \Phi_q \Phi_s \Phi'_t \Psi_M \Psi_N \right\rangle &= \left\langle \Psi_m \Phi'_p \Phi_s \Psi_M \right\rangle . \left\langle \Psi_n \Phi_q \Phi'_t \Psi_N \right\rangle &&\equiv B1y_{mpsM}.B2y_{nqtN} \\
\left\langle \Psi_m \Psi_n \Phi_p \Phi'_q \Phi_s \Phi'_t \Psi_M \Psi_N \right\rangle &= \left\langle \Psi_m \Phi_p \Phi_s \Psi_M \right\rangle . \left\langle \Psi_n \Phi'_q \Phi'_t \Psi_N \right\rangle &&\equiv C1y_{mpsM}.C2y_{nqtN}.
\end{aligned}$$

Using eqn(2.8b) yields

$$A1x_{mpM} = A1x_{m-M,\, p-M,\, 0} \tag{2.12}$$

Since the functions have widths $= k, h$ it follows $(l = h + k - 1)$

$$A1x_{mpM} = 0 \qquad \text{if } \|M - p\| \geq l \text{ or } \|M - m\| \geq l.$$

Thus l^2 integrals each for $A1x, A2x, A1y, A2y$ need to be calculated. Using similar arguments, one finds that for $B1x$ etc. one needs to precompute and store l^2h integrals each. These memory costs are small compared to the memory cost for the images.

The bulk of the overhead computation costs consist of performing the summations of the above matrices with the image (first frame). If the image size is $D \times D$ and $C = 2h - 1$, $(h = k)$ then a summation of the form

$$\sum_{p,q,s,t} B1x_{mpsM}\, B2x_{nqtN}\, I(M-p, N-q)\, I(M-s, N-t)$$

will produce a 4 dimensional array of size D^2C^2 requiring C^4 multiplication per element for a total cost of D^2C^6. There are 4 such summations (Bx, By, Cx, Cy) and 2 much smaller summations (Ax, Ay). The indices M, N assume D values each and m, n, p, q, s, t assume C different values. However, using the fact that we have a separable kernels like $B1x.B2x$, the computation per element reduces to $2C^2$ bringing the total cost down to $2D^2C^4$. Since there are 4 such summations, the overhead computation requires $8D^2C^4$ multiplications.

$$\begin{aligned}
B(m,n,M,N) &= \sum_{p,q,s,t} B1x_{mps}\, B2x_{nqt}\, I(M-p, N-q)\, I(M-s, N-t) \\
&= \sum_{p,s} B1x_{mps} \sum_{qt} B2x_{nqt}\, I(M-p, N-q)\, I(M-s, N-t) \\
T(n, M-p, M-s, N) &= \sum_{q,t} B2x_{nqt}\, I(M-p, N-q)\, I(M-s, N-t) \\
B(m,n,M,N) &= \sum_{p,s} B1x_{mps}\, T(n, M-p, M-s, N) \\
\text{where} \qquad m,n &\in [1 \ldots C]
\end{aligned}$$

$$\begin{aligned} q,t,p,s &\in [-C/2\ldots C/2] \\ M,\ N,\ M-p &\in [1\ldots D] \\ M-s &\in [M-p-C/2\ldots M-p+C/2]. \end{aligned}$$

Once the above type of summations have been done, solution of eqn(2.9) requires performing the summations over m, n for each $P(M,N), Q(M,N)$ at each iteration. Thus the computational cost per iteration (*Loop*) is D^2C^2. For the smallest value of h (h = 3) this cost dominates the overhead costs. For larger values of h, (h = 7, 11), $4D^2C^6$ (the unseparable basis overhead cost) may dominate $Loop \times D^2C^2$, where *Loop* is the number of iterations required for convergence. Hence, having a separable basis and keeping the overhead costs down to $8D^2C^4$ means that when wider basis functions are used the computational cost goes up as C^2 initially and then as C^4. On the other hand, with a nonseparable basis, the cost goes up as C^6 since the overhead costs begin to dominate. Note that arrays like $B(m,n,M,N)$ cannot be separable since the image $I(p,q)$ is not generally separable. Hence the final summation over m, n cannot be reduced to $2D^2C$ multiplications/iteration.

2.4 Choice of Basis Function

The choice of basis functions is guided by two considerations - accuracy of representation and computational cost of solving eqn(2.9). While considering the issue of accuracy, one must not only keep in mind the accuracy with which a set of basis functions can represent the image as in eqn(2.4) but also the accuracy with which it can represent the derivatives as in eqn(2.6). The two requirements are not identical since the process of taking derivatives accentuates the high frequencies and attenuates the low frequencies.

2.4.1 Sampling Rate

To a good approximation, the digitized images may be considered to be formed from continuous real images by sampling with an array of delta functions. So long as the image is bandlimited and the sampling meets the Nyquist criterion, a perfect reconstruction is possible using sinc basis functions. The disadvantage with the sinc function is that the amplitude only decays as $1/x$ leading to global coupling and too high a cost for solving eqn(2.9). As a result, one is forced to consider more compact functions, like (windowed) truncated sinc functions and functions underlying linear and cubic spline interpolations. In [Anderson and Rakshit, 1992] we have given an analysis of the interpolation accuracy of the above basis functions as a function of frequency. It was observed that as the sampling frequency approached the Nyquist limit (and the digital frequency approached π) it became necessary to use wider functions to obtain accurate interpolations. This is of greater concern in the present context since we are also interested in calculating derivatives of images, which are strongly dependent on the higher frequency components.

The Nyquist sampling rate is the minimum rate that assures an exact reconstruction and no aliasing. It represents the result of optimizing memory requirement for no aliasing. If, after the sampling process, subsequent processing requires interpolation, reconstruction or shift invariant analysis, then our criterion for chosing a sampling rate should also take into account the computational complexity of these processes. As Simoncelli *et al.* have shown ([Simoncelli et al., 1992]), a critically sampled representation like wavelets cannot be shift invariant since the Nyquist criterion is violated. In [Anderson and Rakshit, 1992] we went beyond existence and addressed the issue of optimal sampling rate to minimize the computational cost. That argument shall be repeated here in the context of image motion estimation.

If a sinusoidal signal $f(x)$ with frequency f_a is sampled at f_s, the samples are

$$\begin{aligned} f(n) &= \cos(2\pi f_a n/f_s) = \cos(n\pi - n\pi(f_s - 2f_a)/f_s) \\ &= \cos(n\pi)\cos(\pi(f_s - 2f_a)n/f_s). \end{aligned}$$

Thus $f(n)$ has values of alternating sign with an envelope of frequency $f_e = (f_s - 2f_a)$. In order to estimate f_a from $f(n)$ one needs to know f_s and f_e. To know the latter one must have samples of $f(n)$ over a width of $1/f_e$ and only basis functions of width $1/f_e$ or more will be able to pool information from region that wide while performing the interpolation. A basis function of that width, sampled at the same frequency as the image (f_s), will have $f_s/(f_s - 2f_a)$ coefficients. An image of size $L \times L$ when sampled at f_s will have $L^2 f_s^2$ pixels and as many $P(M, N)$ and $Q(M, N)$. Hence the cost will scale as

$$\begin{aligned} \text{Total number of Coefficients} &= L^2 f_s^2 \\ \text{Cost per Coeff per iteration} &= [f_s/(f_s - 2f_a)]^2 \\ \text{Total cost per iteration} \quad C_t &= L^2 f_s^2 [f_s/(f_s - 2f_a)]^2 \\ \text{Minimizing } C_t \text{ wrt } f_s \quad f_{s,opt} &= 4f_a. \end{aligned}$$

Thus while the Nyquist limit is necessary, it is not the optimal from a computational point of view. Oversampling by a factor of 2 above the Nyquist limit in each dimension trades off computational cost for excess memory requirements by allowing the use of more compact basis functions on larger images. In most cases, a linear basis is implicitly used (Φ_{lin} as defined subsequently) and for these algorithms the error could be decreased by oversampling. For a given sampled image $I(m, n)$, if there is significant energy above $\pi/2$, it will pay to first upsample the image by interpolating with a big filter (e.g., 20 tap FIR) and then do the motion analysis with a compact basis. The usual practise of prefiltering images by Gaussian lowpass filters tries to achieve a similar effect by discarding the high frequency information together with high frequency noise. This is unwarranted for our algorithm since the coupled system of eqn(2.9) ignores the random local motions generated by noise and locks into the motion of the true image and the higher levels of a pyramid representation tend to be immune from high frequency noise.

2.4.2 Derivatives

The question of calculating derivatives has only recently been given significant attention. In [Simoncelli, 1993], Simoncelli addresses the problem of designing prefilter-derivative filter design by minimizing a weighted least square error in the frequency domain for constant phase separable lowpass filters. He then used a bi-cubic spline to interpolate the image and to estimate derivatives at fractional pixel locations. The consistent way to calculate derivatives and interpolate the image and derivatives is to use eqn(2.4) and eqn(2.6). For a choice of Φ that is separable and localized, Φ' is also separable and localized. It is easy to see that the first-order difference, the most popular choice for derivatives, amounts to using eqn(2.6) with the basis function underlying the linear interpolation. Hence, the use of first-order difference for derivatives is certainly consistent with using linear interpolation. The linear interpolation has the advantage that its associated basis function has width = 2 making this choice of basis function computationally the least expensive. If computational resources are available, is it possible to make better estimates of derivatives in images, or does the discrete nature of images impose a limit ?

Consider the process of calculating derivatives of sinusoids $f(x) = \sin(2\pi f_0 x)$

$$f'(x) = \lim_{h \to 0} [f(x+h) - f(x)]/h. \tag{2.13}$$

For a discretized image, the smallest h is considered to be $1/f_s$ leading to

$$\begin{aligned} f'(n) &= f(n+1) - f(n) \\ &= \sin(2\pi f_0 x)[\cos(2\pi f_0/f_s) - 1]/(1/f_s) \\ &\quad + \cos(2\pi f_0 x)\sin(2\pi f_0/f_s)/(1/f_s). \end{aligned}$$

Two points are worth noting:

- The error is frequency dependent. Various attempts have been made to compensate for this error [Weber and Malik, 1992], [Anderson and Rakshit, 1992] by using a multiplicative correction factor that best approximates the error in a given frequency range.
- Using $0.5*[f(n+1) - f(n-1)]$ will *increase* the error by increasing the deviation from $h \to 0$. (The error being considered here is the systematic error, not error due to noise in the input.) The high frequency performance cannot be improved by a smoother filter.

The systematic way to improve the estimate of derivatives is to use eqn(2.6) with better choice of Φ. Just as interpolation can be improved to any desired accuracy by using bigger (windowed) truncated sincs, so too for estimation of derivatives. The derivative of the sinc also has a $1/x$ envelope and hence performance for derivatives improves no slower than for interpolation.

$$\mathrm{sinc}'(x) = \frac{1}{x}\left[\cos(\pi x) - \frac{\sin(\pi x)}{\pi x}\right] \tag{2.14a}$$

$$f'(n) = \sum_m \mathrm{sinc}'(n-m) f(m). \tag{2.14b}$$

At the first glance, eqn(2.14) would seem to be against the spirit of eqn(2.13) - calculation of *local* difference. The discrepancy is due to the fact that a sampled representation encodes information about a continuous signal at off grid points in a *distributed* way. Recall that during reconstruction (or interpolation), for $x = n$ we have $f(x = n) = f(n)$ but for $x \neq n$ the sinc interpolation has to pool in information stored in many pixels. As was seen earlier, when the sampling frequency approaches the Nyquist limit and creates digital frequencies close to π, the distributed nature of the representation becomes increasingly important for computational purposes. Since the calculation of derivative requires information about the function at two points close together, it always requires information about at least one off grid point. Hence, unlike sinc, the sinc$'$ has nonzero values at other grid points even for $x = m$. If eqn(2.14b) is used to generate a 7-tap filter to estimate derivatives at grid points, one gets $\{0.33, -0.5, 1.0, 0.0, -1.0, 0.5, -0.33\}$. The process of convolution will flip the filter around and give the leading term $f(n+1) - f(n-1)$. Note that the higher order terms alternate in sign - this is essential to capturing the high frequencies. Derivatives off grid points can be directly computed by eqn(2.14b) and will be of equal accuracy as on grid points. In [Simoncelli, 1993], the 5-tap filter had the form $\{-0.09, -0.31, 0.0, 0.31, 0.09\}$ which does not have the alternating behavior. This was due to the fact that the filter was optimized to match the derivative of a low-pass prefilter, not that of a true derivative response. An added source of error for that approach was the necessity of using an interpolation on derivatives calculated at grid points to estimate derivatives off grid points.

If the goal was just the calculation of derivatives, large filters (basis functions with wide support) could have been used for accurately estimating image gradients. However, the goal here is to pick a set of basis functions and solve the coupled system of equations, eqn(2.9). Since the computational cost grows as h^2, where $2h$ is the width of the basis functions, the choice will be restricted to linear, cubic, 5-tap and 7-tap sinc (henceforth called sinc5, sinc7) bases. The cubic basis is also a 5-tap basis but has better low frequency response than a sinc5. The low frequency dominated power spectrum of natural images, the coarse to fine approach using pyramids and the inherent coupling in eqn(2.9) lead to accurate results for real images even for linear basis. However, it will be shown how the wider cubic, sinc7 outperform the linear basis in the presence of high frequencies and large displacement. The choice of basis functions will be

$$\begin{aligned}
\Phi_{\text{lin}}(x) &= \begin{cases} 1+x & -1 < x \leq 0 \\ 1-x & 0 < x \leq 1 \\ 0 & \text{otherwise} \end{cases} \\
\Phi_{\text{cub}}(x) &= \begin{cases} 2x^3 - 3x^2 + 1 + s[X^3 + 2X^2 + X] & 0 \leq x \leq 1 \quad X = x - 1 \\ s[X^3 - 2X^2 + 1] & 1 < x \leq 2 \quad X = x - 1 \\ 0 & 2 < x \\ \Phi_{\text{cub}}(-x) & x < 0 \end{cases} \\
\Phi_{\text{sinc5}}(x) &= \text{sinc}(x).W_5(x) \\
\Phi_{\text{sinc7}}(x) &= \text{sinc}(x).W_7(x) \\
W_5(x) &= \begin{cases} \frac{1}{e^{(1.75-x)/0.150} + 1} & 2 \geq x \geq 0 \\ 0 & x > 2 \\ W(-x) & x < 0 \end{cases} \\
W_7(x) &= \begin{cases} \frac{1}{e^{(2.625-x)/0.175} + 1} & 3 \geq x \geq 0 \\ 0 & x > 3 \\ W(-x) & x < 0 \end{cases} .
\end{aligned}$$

The window is essential to remove ringing from the interpolation of low frequency and dc signals. While ringing is just a nuisance for interpolation, it can seriously corrupt the calculation of derivatives. However, the more conventional windows like the Hann window lead to such excessive smoothing that the high frequency response of the bigger basis functions show only marginal improvement. The above type of window is a good compromise that leaves one with two free parameters to adjust the trade off between suppressing ringing and high frequency response. Φ_{lin} does not have a derivative defined at $x = 0$. However, we can define it to be 0 and the derivative, being piecewise smooth, is integrable. For simplicity, we will use $\Psi = \Phi$, *i.e.,* the same basis for the image and displacement field.

2.4.3 Equivalent Formulations

The formulation of the optical flow problem as given by eqn(2.9) can be related to previous work under suitable choice of Ψ, Φ and approximations. This is easier to see if we rewrite eqn(2.9) without expanding $I(m,n), I_x(m,n), I_y(m,n)$ (consider these the new 9a,9b,9c)

$$\begin{aligned} E &= \iint \left[\Delta I - \sum P(m,n)\Psi_{mn} I_x - \sum Q(m,n)\Psi_{mn} I_y\right]^2 dxdx \\ \frac{\partial E}{\partial P_{MN}} &= \iint \Psi_{MN} I_x \left[\Delta I - \sum P(m,n)\Psi_{mn} I_x - \sum Q(m,n)\Psi_{mn} I_y\right] dxdx \\ \frac{\partial E}{\partial Q_{MN}} &= \iint \Psi_{MN} I_y \left[\Delta I - \sum P(m,n)\Psi_{mn} I_x - \sum Q(m,n)\Psi_{mn} I_y\right] dxdx. \end{aligned}$$

If the Ψ are chosen to be constant over a rectangular region and nonoverlapping, then one is working under the assumption of constant displacement inside each block and the equation reduces to the familiar ($\Delta x \equiv P, \Delta y \equiv Q$)

$$\begin{bmatrix} < I_x^2 > & < I_x I_y > \\ < I_x I_y > & < I_y^2 > \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = \begin{bmatrix} < \Delta I . I_x > \\ < \Delta I . I_y > \end{bmatrix} \qquad (2.15)$$

where $< . >$ denotes an averaging process over the extent of the block. It may be noted that the various $\Delta x(m,n), \Delta y(m,n)$ decouple only if the Ψ are nonoverlapping. The use of overlapping windows to get a more dense estimate of $\Delta x, y$ from the above amounts to the approximation of ignoring the coupling.

For any Φ and Ψ the equations for P, Q for neighboring pixels are coupled due to the nonzero diagonal terms in the matrices $B1x.B2x$ as defined in Sec 2.3.2. However, for the choice of $\Phi = \Psi = \Phi_{lin}$, the ratio of the diagonal terms to the off diagonal terms in the product $B1x.B2x$ is about 10:1 . If these off diagonal terms are ignored, then one gets an independent set of equations for each pixel. This amounts to removing the spatial averaging process in the above matrix formulation, eq(2.15), and the determinant becomes singular. There is only one equation for every set of two variables and the system of equations reduces to that of Horn [Horn and Schunck, 1981]. Thus the choice of basis function automatically results in a choice of averaging kernel for eqn(2.15) via the coefficients of matrices like $B1x$ etc.

The Ψ_{mn} is usually introduced in an ad hoc manner in many formulations as a *window function* in the integral (9b). Its purpose is usually to give more weight to the center than to the periphery during the process of averaging the derivatives and the difference image. In our formulation, it occurs naturally and serves a similar purpose. Besides being a weighting function, it also determines the coupling between neighboring $\Delta x, \Delta y$ and serves as the appropriate basis function if the displacement field needs to be interpolated to off grid points. By explicitly choosing Ψ as a basis for the displacement field, one can be aware of the amount of detail the result will have.

The regularization term in Eq(9) was derived from a prior distribution for P, Q. The coupled equations in (9) can have a large number of local minima. The term arising from $-\log \mathcal{P}$ can be viewed as a term that drives the solution towards a particular minima. Thus even when Ψ is chosen much wider than Φ and there are only as many unknowns as equations, it is still advantageous to include the regularization term. Thus in our formulation, the regularization term has a justification independent of the aperture problem.

2.4.4 Multiresolution

For reasons of both accuracy and computational efficiency, motion algorithms are best implemented in a multiresolution format. In this work, a 2D version of the Gaussian pyramid was used. The set of original images is designated level **g0**. From this **g1** is constructed by forming the next level Gaussian images for each frame *independently*. Thus our pyramids were pyramids in space only and involved no time averaging and subsampling. This approach was adopted since it was observed that most image sequences are acquired with low sampling rates in time and hence not conducive to averaging and subsampling in time. Moreover, most of the computational cost is incurred while working on **g0** and the subsampling in time to reduce the size of higher Gaussian levels does not lead to significant savings. On the other hand, an accurate motion estimation by the low resolution (higher **n**levels) can significantly reduce the amount of computation needed at the high resolution levels leading to big savings in computation.

The first-order approximation underlying eqn(2.2) limits displacement estimates to a fraction of the wavelength of the local frequency. For very low frequency features like the overall outline of an object, this can be a large displacement. However, finer features within the object would dominate the calculation of derivatives in the original images making large displacement estimation error prone. A multiresolution framework overcomes this problem by allowing one to first work on an image from which all high frequency has been removed. In practise, these blurred images are subsampled so that their spectrum once again has some high frequencies but the process of subsampling reduces the frame to frame displacements as well. Hence, *small* displacements need to be estimated in the high pyramid levels. Once this is done, this information can be used to reregister the next level. In this way multiresolution increases the estimation accuracy for large displacements.

The question often arises as to the choice of representations : the low-pass Gaussians or the band-pass Laplacians. The Gaussian pyramids were used for two reasons - robustness of derivative calculation and use of prior estimates in the pyramid. As will be shown in the next section, the accuracy of derivative estimation is much better for low frequencies than for mid to high frequencies. Hence any approach that requires the use of derivatives will be better off using Gaussian pyramids. The argument in favor of the band-pass Laplacians is that they constitute the new (independent) information at each level and only their motion should need to be calculated at each level. While that argument is true, it can be implemented by using a Gaussian pyramid as well. At each level, we have used the motion estimate from the previous level (scaled appropriately by a factor of 2) to reregister the Gaussians. The motion estimation at each level was done for this reregistered pair, thereby incorporating the knowledge gained so far. All calculation done at a level was done to accommodate only the new information injected at that level. The Gaussian pyramid has an added advantage - all the image information that was used at the higher pyramid levels is still present in the images. This means that the motion estimate at level **g(n-1)** will be consistent with **gn**. If the motion estimation is done with truly independent bands-pass images, then either an added cost term must be incorporated in the equations to combine information across bands or final results from the different bands must be combined using some weighted means approach. It may be noted that our implementation does not require iteration across levels, *i.e.,* calculations at each level are done just once.

2.5 Results

In [Anderson and Rakshit, 1992] we showed the performance of various basis functions for interpolation. Here we shall concentrate on the linear, cubic and sinc7 basis functions. The performance of these functions for interpolation and derivative estimation is tested for sinusoids over a wide frequency range. Next we shall look at the accuracy of motion estimation for sinusoidal inputs to see how this relates to the ability of the underlying basis to represent the sampled input. Motion estimation for an artificially generated sequence from a single natural image demonstrates how the spectrum of natural images allows the use of the linear basis in most cases. Finally, we shall look at a pseudo motion sequence generated by looking at a stack of brain cross sections where rigidity, constant brightness and small displacement assumptions are all violated. We show the ability of our system to handle rotations, dilations and shears under such circumstances as the change between frames is modelled as a set of local affine motions. For this reason, our algorithm was designed to examine a small window (32×32) at a time and fit the final flow field, in a least square error sense, to a superposition of translations, rotation, dilation and shears. We did this in order to get a compact description of the flow field. If the concern was determination of motion boundaries, then we could have used a different regularization like Multiwindow Least-squares [Bartolini et al., 1993].

2.5.1 Synthetic Images

The first set of results we present are for synthetic patterns and motion, namely travelling sinusoidal gratings. The motivation for this exercise is to characterize the performance of our method for various well defined inputs. Before trying to estimate motion in real images, we would like to get a feel for the types of inputs that lead to accurate results and those which cause large errors. A set of well designed inputs will also serve to illustrate the pros and cons of using large basis functions.

The coupled system of eqn(2.9) was based on eqn(2.4) and eqn(2.6), our ability to use a set of basis functions to reconstruct the original image and its derivatives. It is thus natural to expect that our motion estimation accuracy will be dependent on the

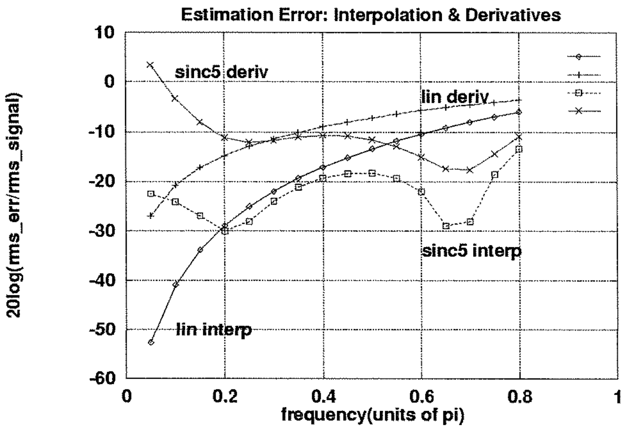

Figure 2.1: Linear and sinc5 basis performance: interpolation and derivative estimation for sinusoidal inputs

extent that those equations are a good approximation for a given image. Given the noise in real images and other sources of error in motion estimation like occlusion, it does not matter a great deal if the systematic errors are reduced below some point, say -20 dB. Thus a basis with -25dB error for pure sinusoids at a certain frequency f_0 will do as well for real image sequence (bandlimited around f_0) as a basis with -45dB error for pure sinusoids. On the other hand if the errors increase above -10dB at any frequency, there will be large errors in motion estimation if the input has significant energy near that frequency. Hence, before looking at motion estimation, we first look at the results of interpolation and derivative estimation for sinusoids. The problems were posed as follows : *Given a set of points* $f(n) = \sin(\pi k n)$, *estimate the function and its derivatives at nine points between consecutive points and estimate the derivatives at the given points as well.* This was done for various values of k, $0 < k < 1$ and with three different basis functions, $\Phi_{lin}\Phi_{sinc5}$ and Φ_{cub}. The results are shown in Fig. 2.1 for linear and sinc5 bases and in Fig. 2.2 for cubic bases of $s = 0.7, 1.0$.

The interpolation errors for the linear and cubic bases get arbitrarily small as the

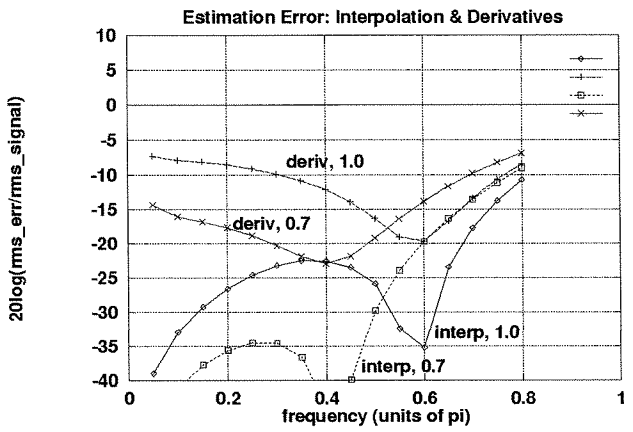

Figure 2.2: Cubic spline basis performance: interpolation and derivative estimation for sinusoidal inputs

input frequency approaches 0 (dc), as is to be expected since these interpolations would give perfect results for dc or ramped inputs. The error for interpolation with sinc5 does not go to 0 for dc, but it is still low, less than -20dB. The functionally relevant difference between interpolating with Φ_{lin} and Φ_{sinc5}, Φ_{cub} is in the mid to high frequency regions. The error using sinc5 remains below -15dB till $f = 0.8\pi$ while the error from linear basis rises above -15dB at $f = 0.45\pi$. At the highest frequencies (near π) both basis functions become inadequate. Wider basis functions would push the -15dB mark closer to π ([Anderson and Rakshit, 1992]). For the choice of $s = -0.7$, the cubic basis gives a low frequency response as good as a linear basis while outperforming the sinc5 in mid frequency regions.

The derivative estimation errors are, in general, bigger than the interpolation errors. It must be noted that the absolute value (rms amplitude) of the derivatives decreased with frequency. For the lowest frequencies, the ratio of error to signal becomes very large for sinc5 even though the power in the error does not grow large since the signal (derivative) goes to 0 at dc. The cubic basis with $s = -0.7$ does much better than sinc5 in the low to mid frequency region. Since both these functions,

Φ_{cub} and Φ_{sinc5}, have the same width $(= 4)$ we shall not use the Φ_{sinc5} for further analysis. Hence our choice of functions will be Φ_{lin}, Φ_{cub} and Φ_{sinc7} for bases of width $2, 4, 6$ respectively. Referring to Sec 3.2, eqn(2.10), these functions have $h, k = 2, 4, 6$ respectively yielding $C = 3, 7, 11$.

The three basis functions were used to estimate motion for a range of sinusoidal gratings. The inputs varied in frequency and frame-to-frame displacement. Since the first-order approximation is sensitive to the smoothness of the image, the displacements were characterized as fractions of the input wavelength. Thus we gave sinusoidal inputs at frequencies ranging from 0.05π to 0.8π and with frame-to-frame displacements of $\lambda/10$, $\lambda/6$ and $\lambda/4$. The input was analytically calculated for each of five successive frames and the displacement estimated between four consecutive pairs. The errors for the four frames were used to calculate an rms error which was then expressed, in dB, as a fraction of the true displacement. Four frames were used in order to remove artifacts arising due to sampling of the sine wave at a particular phase. The results are shown in Table 2.1. The first two columns gives the frequency (units of π) and displacement (pix/frame), the subsequent columns give motion estimation error in dB. For each frequency, the displacements correspond to $\lambda/10, \lambda/6$ and $\lambda/4$. The wavelengths corresponding to the frequencies shown are $25, 15, 10, 5, 3.5, 3.0, 2.5$. Two differences between the linear and bigger bases functions are of future importance.

One difference is in the $f > 0.5\pi$ frequencies where the linear system does very poorly, falling below -12dB for all displacements. The cubic and sinc7 basis gives good results for small displacements even at high frequencies but the performance declines for large displacements. At $\lambda/4$ displacement/frame the sinc7 at high frequencies does only 4dB better than the linear basis. These results are to be expected since by using bigger basis functions one can better approximate the high frequency behavior of the input but not improve upon the first-order Taylor approximation. Hence the Φ_{sinc7} gives better results than Φ_{lin} only if the Φ_{lin} fails due to the high frequency in the input

Table 2.1: Motion estimation fractional error in dB for sinusoidal inputs

freq (f/π),	disp = $\lambda/10$	Linear Basis	Cubic Basis	Sinc7 Basis
0.80	0.250	-10.2	-17.8	-22.6
0.67	0.300	-9.7	-17.8	-22.7
0.57	0.350	-11.5	-20.2	-24.9
0.40	0.500	-27.8	-55.2	-51.5
0.20	1.000	-51.2	-57.6	-60.0
0.13	1.500	-62.6	-42.7	-39.4
0.08	2.500	-51.8	-39.0	-35.3

freq (f/π),	disp = $\lambda/6$	Linear Basis	Cubic Basis	Sinc7 Basis
0.80	0.417	-6.0	-10.7	-14.5
0.67	0.500	-6.7	-11.8	-14.7
0.57	0.580	-8.6	-13.6	-18.6
0.40	0.833	-30.1	-50.0	-44.8
0.20	1.667	-49.0	-37.8	-38.8
0.13	2.500	-64.0	-52.7	-43.7
0.08	4.167	-57.3	-42.9	-37.9

freq (f/π),	disp = $\lambda/4$	Linear Basis	Cubic Basis	Sinc7 Basis
0.80	0.625	-4.5	-7.2	-8.5
0.67	0.750	-4.9	-7.4	-9.0
0.57	0.825	-5.5	-9.0	-10.5
0.40	1.250	-21.0	-31.4	-38.8
0.20	2.500	-51.1	-45.6	-41.4
0.13	3.750	-50.2	-53.4	-41.2
0.08	6.250	-45.5	-46.9	-37.7

but not due to large displacements. Motion estimation in a multiresolution framework using three level Burt pyramids to deal with large displacements works only if there are low frequency features undergoing large displacements coherently with the high frequency details so that the displacement estimates from the higher pyramid levels can be used to reregister the lower levels leaving only small residual displacements to be calculated from the high frequency features. For inputs that are purely high frequency sinusoids, the higher pyramid levels are devoid of signal and hence useless. The pyramid scheme used, the 5-tap Burt pyramid, produces aliasing for high input frequencies. Since for these inputs the higher pyramids do not help, motion estimates were done without using multilevel pyramids for $k = 0.57\pi, 0.67\pi, 0.8\pi$. These results are shown in Table 2.1. On the other hand, for low frequency inputs, the pyramid structure did help. The actual $\lambda/4$ displacements were in fact much larger for the lower frequencies (and longer wavelengths) but these displacements were calculated very accurately, better than -20dB for Φ_{lin} and -30dB for Φ_{sinc7} at frequencies below 0.5π.

The second difference is in the very low frequencies. Here the linear basis, with its near perfect ability to represent low frequencies, does better than cubic or sinc7 bases. In real images, this feature plays a key role since real images have a $1/f$ power spectrum and hence biased towards the low frequencies. This is fortunate since it implies that Φ_{lin} can be used for the majority of natural images and the computationally expensive wider functions are needed only for special inputs.

Table 2.1 was for inputs with a translation along x direction only. Due to the coupled nature of eqns(2.9), the presense of motion along both x and y axis leads to increased error. This increase is mainly for the motion component corresponding to the higher frequency. Results for one such input are shown in Table 2.2. The input was a plaid composed of two sinusoidal gratings as shown in cols 2, 3. The error for the plaid motion estimation is to be contrasted with the error for the corresponding

Table 2.2: Motion estimation fractional error in dB for plaid input. Corresponding result for pure sinusoid in brackets

Direction	freq/π	disp (pix/fr)	Linear Basis	Cubic Basis	Sinc7 Basis
along x	0.2	1.0	-52.6 (-51.2)	-63.0 (-57.6)	-52.0 (-60.0)
along y	0.4	0.5	-17.6 (-27.8)	-32.4 (-55.2)	-41.4 (-51.5)

single gratings shown alongside in brackets.

2.5.2 Real Image, Synthetic Motion

In this section, we look at the motion estimation performance for pseudo motion sequences generated from a single natural image. The test images were chosen to demonstrate the issues discussed earlier: large displacements and high frequencies.

Our first test consisted of estimating motion in sequences of moving face (the Lena image) where the four sequences had frame to frame displacements varying from 1 to 4 pixels per frame. The sequences will be labelled as LENASEQ1.G0 ... LENASEQ4.G0 respectively. Each sequence had 5 frames of size 160×160 and motion was estimated for a 32×32 window at the center. A three level Gaussian pyramid was used with the algorithm first estimating motion at level **g2** and finally **g0**. The 4 frame-to-frame estimated translations were used to calculate an rms error in dB as in Sec 2.5.1, table 2.1. The object of this test was to see how well the multiresolution format could cope with increasing displacements. The result, shown in Table 2.3, shows that large displacements like 4 pix/fr can be estimated as accurately as 1 pix/fr. where the error is measured as a fraction of the actual displacement. The absolute value of the error does increase. The Lena image, like most natural images, is dominated by low frequency. For such images, the most suitable basis function is one with very good low frequency response irrespective of its high frequency response.

The ability to handle high frequencies becomes important when dealing with textured surfaces. One such texture, denim, was used to test the functions Φ_{lin}, Φ_{cub}, Φ_{sinc7}. In order to test high frequency performance, we picked the denim texture image which has only mid to high frequency components. The simulated motion direction was left to right, at right angles to the dominant orientation of the texture. The result is shown in Table 2.4, col 1. As can be seen from this motion sequence (labelled DENIMSEQ) even the denim texture was not of sufficiently high frequency to cause the Φ_{lin} to give a poor result. Referring to Table 2.1, the error stays small up to $f = 0.5\pi$, *i.e.,* for features bigger than 4 pixels wide. The vertical stripes in the denim are about that wide. This result shows, along with Table 2.3, why various techniques based on linear basis assumptions have given good results for natural images where most of the energy is in the lower frequencies. Often the high frequencies present move *coherently* with the surrounding low frequency features. Under such circumstances, the motion estimation from the higher Gaussian levels captures the motion in the image. Thus it is only the presence of high spatial frequencies in the absence of low frequencies or motion of high frequency features independent of the motion of coarser features that will require the use of Φ_{cub} or Φ_{sinc7}.

A second texture was created from the denim texture by subsampling by factor of 2 to further bias the image towards the high frequencies. Two tests were carried out on this new pattern: a) the pattern was moved by 0.5 pix/frame (DENIM2SEQ) b) the pattern was expanded by a factor of 2 and then displaced by 1 pix/frame. The motivation for (b) was to demonstrate that even for the occasional input that has very high frequencies in it, the optimum strategy is to use Φ_{lin} on an expanded version of the input. For our test, this expansion was done using a 41-tap FIR (truncated sinc with Blackman Window) to interpolate in x and y directions successively. The separability of this filter and the D^2C^2 factor in computational cost and a D^2C^4 factor in overhead computation cost makes this approach cheaper for even though D increases by a factor of 2, C decreases by 2.3 (Φ_{cub}) or 3.7 (Φ_{sinc7}). The results for the

Table 2.3: Motion estimation fractional error in dB for Lena sequences. Frame to frame displacements were 1, 2, 3, 4 pix/fr for the four sequences.

Basis Function	LENASEQ1	LENASEQ2	LENASEQ3	LENASEQ4
Linear	-13.3	-14.9	-16.1	-16.7
Cubic	-10.9	-12.8	-12.8	-12.2
Sinc7	-10.3	-10.4	-9.5	-9.1

Table 2.4: Motion estimation fractional error in dB for denim sequences. Displacements were 1 pix/fr for DENIMSEQ, DENIM2SEQ and 2 pix/fr for DENIMSEQ3

Basis Function	DENIMSEQ	DENIM2SEQ	DENIM3SEQ
Linear	-32.0	-15.6	-27.1
Cubic	-33.2	-22.8	-32.2
Sinc7	-30.9	-21.7	-29.5

subsampled denim (DENIM2SEQ) and the magnified version of the subsampled denim (DENIM3SEQ) are shown in Table 2.4 cols 2,3. The three textures are displayed in Fig 2.3. The top frame shows the original denim texture, the middle frame is the result of demagnification by 2.0 and the bottom frame shows the result of magnifying the demagnified texture back to original scale. It no longer has the very high frequencies that are present in the original. It must be pointed out that expansion by 2 not only reduces the high frequencies but also doubles the frame-to-frame displacements in pixels. Thus, this approach is best suited for inputs with only high frequencies but small displacements.

2.5.3 Real Image Sequence, NonRigid Motion

The optical flow estimation by eqn(2.9) gives a pixelwise estimation of v_x, v_y. For most 32×32 windows, the flow field does not require 1024 free parameters to describe it. Since motion usually segments images into large coherently moving parts, it is often possible to represent the flow field with much fewer than 1024 parameters.

Here, this approach has been taken to an extreme by postulating that the motion observed within each window can be modelled by just 6 parameters : x translation, y translation, dilation, rotation, shear along x and shear along y. For irregular flow fields, we used 10×10 windows for more accuracy.

To study how well our algorithm was able to handle really large displacements and magnification changes, we picked a sequence of monkey cortex cross sections as our data, BRAIN [Coogan et al., 1993]. When viewed as a sequence, this set gives the appearance of an irregularly shaped object undergoing complex elastic transformation. Any local region can therefore be modelled as undergoing nonrigid motion and the appearance/disappearance of various features can be regarded as occlusion. The task for our algorithm was to use every 5^{th} frame as a reference and describe the other frames as warped versions of the nearest reference frame. Each frame was 640×480 in size and was analyzed as 64×48 blocks of 10×10 each. For each block the optical flow field was estimated and then the 6 parameters calculated. In the absence of ground truth, we checked our results by using them to regenerate the intermediate frames from the reference frames. By looking at the difference between the original sequence and the regenerated sequence one can see how well the system has been able to capture the large displacements and transformations.

Since this is an unconventional data set and an unconventional performance measure, we also show the following. In order to see how much the warping algorithm has accomplished, we subtracted the reference frames from the others to see how much difference there would have been between frames in the absence of any warpings. Thus for any sequence DUMMY, we have DIFF-DUMMY as the sequence formed by subtracting the nearest 5^{th} frame from each frame and ERR-DUMMY as the sequence formed by subtracting the regenerated version of DUMMY from the original. By looking at the pixel intensity distribution in these two sequences we get a clearer picture of how well our algorithm has captured the frame-to-frame motion. A similar test was carried out by Namazi and Lipp in [Namazi and Lipp, 1992] but for a simpler data set, namely uniform translation. In order to make a comparison, we artificially generated a sequence LENASEQ which consisted of 160×160 images with the frame-to-frame displacements being translations. However, unlike [Namazi and Lipp, 1992], our sequence had a motion boundary with an inset of 90×90 moving independently of the outer border. In spite of that, our NMSE was also 0.1. [Normalized Mean Square Error is defined as (residual error/initial error)].

Sixteen frames from the sequence LENASEQ and BRAIN are shown in Fig 2.4 and Fig 2.5 respectively. Since these sequences were of natural images with most of the power in the low frequencies we used Φ_{lin}.The pixel intensity distributions for DIFF-LENASEQ and ERR-LENASEQ are shown in Table 2.5. We show the minimum, maximum, the standard deviation for 10 frames. The last four columns show how many pixels in that frame had a magnitude greater that $4, 8, 16, 32$. The LENASEQ images had 25600 pixels/frame, ranging from 0 to 255. Table 2.6 and Table 2.7 gives the corresponding results for the BRAIN sequence where each frame had 307200 pixels. The reconstructed sequences and their differences from the originals are shown in Fig 2.6 and Fig 2.7 for LENASEQ and in Fig 2.8 and Fig 2.9 for BRAIN. The images shown in the figures are of the level **g2** for BRAIN related sequences, *i.e.,* Mag $\times 0.25$. The contrasts in all the image sequences were scaled to match the 8 bit dynamic range of the display.

The sequences in Fig 2.4 and Fig 2.5 are displayed such that the reference frames appear in the center and the four frames that are warped relative to it appear to its left and right on the same row. The same convention is followed for the DIFF and ERR sequences accounting for the null image in the middle of each row in Fig 2.7 and Fig 2.9. These null images also account for every fifth row in Table 2.5 and Table 2.6 and Table 2.7 being 0.

In Fig 2.7 it can be seen that most of the errors are close to the motion boundary

Table 2.5: DIFF-LENASEQ and ERR-LENASEQ pixel intensity distribution. Only results corresponding to the first 10 frames shown in Fig.2.4 are displayed.

DIFF-LENASEQ

fr no	max	min	mean	sd	pix > 4	pix > 8	pix > 16	pix > 32
0	0.0	0.0	0.000	0.0	0	0	0	0
1	158.0	-134.0	-0.300	16.4	11388	6882	3618	1511
2	230.0	-185.0	-0.581	24.0	13273	8690	5152	2593
3	185.0	-184.0	0.532	24.2	13531	8917	5254	2650
4	134.0	-113.0	0.259	16.5	11683	7080	3708	1555
5	0.0	0.0	0.000	0.0	0	0	0	0
6	113.0	-134.0	-0.240	16.5	11771	7151	3741	1563
7	184.0	-185.0	-0.491	24.3	13719	9069	5338	2686
8	185.0	-184.0	0.602	24.5	13975	9282	5449	2732
9	134.0	-113.0	0.285	16.7	12050	7356	3846	1593
10	0.0	0.0	0.000	0.0	0	0	0	0

ERR-LENASEQ

fr no	max	min	mean	sd	pix > 4	pix > 8	pix > 16	pix > 32
0	0.0	0.0	0.000	0.0	0	0	0	0
1	72.0	-98.0	0.523	5.4	5554	1974	481	61
2	107.0	-135.0	0.561	7.7	7452	3112	1038	220
3	116.0	-98.0	0.431	7.4	7349	3100	946	219
4	75.0	-70.0	0.480	5.3	5700	1979	431	53
5	0.0	0.0	0.000	0.0	0	0	0	0
6	69.0	-64.0	0.535	5.4	5742	2035	488	68
7	73.0	-101.0	0.566	7.5	7643	3258	1021	232
8	116.0	-78.0	0.424	7.3	7659	3218	985	193
9	90.0	-80.0	0.450	5.4	5963	2057	456	45
10	0.0	0.0	0.000	0.0	0	0	0	0

Table 2.6: DIFF-BRAIN sequence's pixel intensity distribution. This sequence was created from BRAIN sequence by subtracting the nearest fifth frame from each frame. Some of the frame to frame differences are due to shape changes and some due to intensity changes. See Fig.2.5 for the actual images in the sequence.

fr no	max	min	mean	sd	pix > 4	pix > 8	pix > 16	pix > 32
0	0.0	0.0	0.000	0.0	0	0	0	0
1	152.0	-95.0	-0.194	9.4	65234	43349	21586	6586
2	126.0	-116.0	-0.541	15.7	76284	58997	36792	18108
3	156.0	-136.0	2.166	17.1	76046	57911	37375	20285
4	132.0	-110.0	0.514	10.5	68764	46838	23950	7515
5	0.0	0.0	0.000	0.0	0	0	0	0
6	212.0	-103.0	0.346	11.1	61398	40446	20868	7813
7	239.0	-229.0	0.248	17.4	75411	56662	36490	18983
8	148.0	-166.0	1.047	15.8	78501	59206	38176	18867
9	92.0	-139.0	0.104	8.9	68442	43269	19875	5277
10	0.0	0.0	0.000	0.0	0	0	0	0
11	168.0	-125.0	-0.167	11.7	74062	49767	26535	10917
12	179.0	-125.0	-0.171	16.1	83267	61930	37929	18835
13	104.0	-196.0	-2.244	17.3	95726	80139	48236	21594
14	92.0	-156.0	-0.918	10.7	69948	46733	26010	9548
15	0.0	0.0	0.000	0.0	0	0	0	0

Table 2.7: ERR-BRAIN sequence's pixel intensity distribution. See Fig.2.9 for the corresponding images.

fr no	max	min	mean	sd	pix > 4	pix > 8	pix > 16	pix > 32
0	0.0	0.0	0.000	0.0	0	0	0	0
1	62.0	-138.0	0.091	4.1	40389	15821	3362	384
2	83.0	-118.0	0.136	4.9	48729	22246	5177	691
3	91.0	-83.0	-0.141	5.3	48465	22887	6309	1151
4	76.0	-111.0	0.096	4.9	49854	22724	5673	607
5	0.0	0.0	0.000	0.0	0	0	0	0
6	77.0	-112.0	0.010	4.5	42558	18236	4638	631
7	230.0	-98.0	0.006	6.0	52838	26495	8538	1609
8	136.0	-100.0	-0.106	6.5	55249	28010	9435	2182
9	93.0	-79.0	0.095	4.6	49707	20555	4643	494
10	0.0	0.0	0.000	0.0	0	0	0	0
11	130.0	-89.0	0.108	5.4	50790	22744	6503	1206
12	130.0	-149.0	0.188	6.5	57010	28554	9353	2126
13	105.0	-108.0	1.154	8.2	79393	54489	18478	3310
14	87.0	-65.0	0.379	5.6	53089	26751	9109	1053
15	0.0	0.0	0.000	0.0	0	0	0	0

or at high frequency regions like the feathers. The errors in Fig 2.9 are due to large changes in shape and violation of constant brightness. Some of the gray matter stripes appear as faint stripes and progressively darken through the sequence. The motion algorithm tries to fit the shape of the stripe from the reference frame to the other frames but is unable to compensate for the change in intensity. This causes the large error for frame 13. The BRAIN sequence was analyzed using a 6-level pyramid in order to deal with the large image size and big frame to frame displacements required. At the level **g0**, displacements at some points were of the order of 50 pixels/frame which is close to 10% of the image dimension. Since every 5^{th} frame was used as reference, frames $1, 4, 6, 9$ etc were just one frame removed from their nearest reference frame while frames $2, 3, 7, 8$ etc were two frames removed. This means that these frames needed to be warped more and were most likely to have larger intensity changes. This is reflected in the tables where it can be seen that frames closest to the reference frames tend to have much smaller errors than those further away.

2.6 Summary

In this chapter it was shown how the use of basis functions could lead to a formulation of the optical flow estimation problem that was more analytic and insightful. The use of the basis functions is the easiest way to apply operations defined on continuous functions to discrete representations. This approach indicates the way for calculating better derivatives in images which have high frequencies. In doing so we have tried to illustrate our view that one must use the underlying basis functions to determine the coefficients or rules that one uses to deal with any discrete representation.

The issue of varying the sampling rate to trade off memory versus computational cost/accuracy was also emphasized. Too often, the sampling rate is chosen keeping in mind only the input frequency and the desire to minimize the storage requirement.

When significant image processing is anticipated, it is important to factor in the role that the sampling frequency plays on computational cost. In Sec 2.4.1 a computational cost optimization was done for the sampling rate to find $f_{opt} = 2f_{nyquist}$. It was implicitly assumed that the input spectrum had significant energy up to the maximum frequency, other assumptions can give slightly different results. However, the bias towards oversampling is clear. This can be an important factor in chosing between overcomplete representations like (oriented) pyramids and wavelets. Our choice, in this chapter, has been the 4/3 overcomplete Gaussian pyramids.

The proposed formulation of the optical flow problem allows the user to choose a desired balance between speed and accuracy by selecting an appropriate basis. The estimation results for different frequencies enables one to make a choice of basis given an image sequence and desired accuracy. Finally, it was shown how the algorithm can handle a difficult image sequence like BRAIN where there are large translations, nonrigid deformations and intensity level changes.

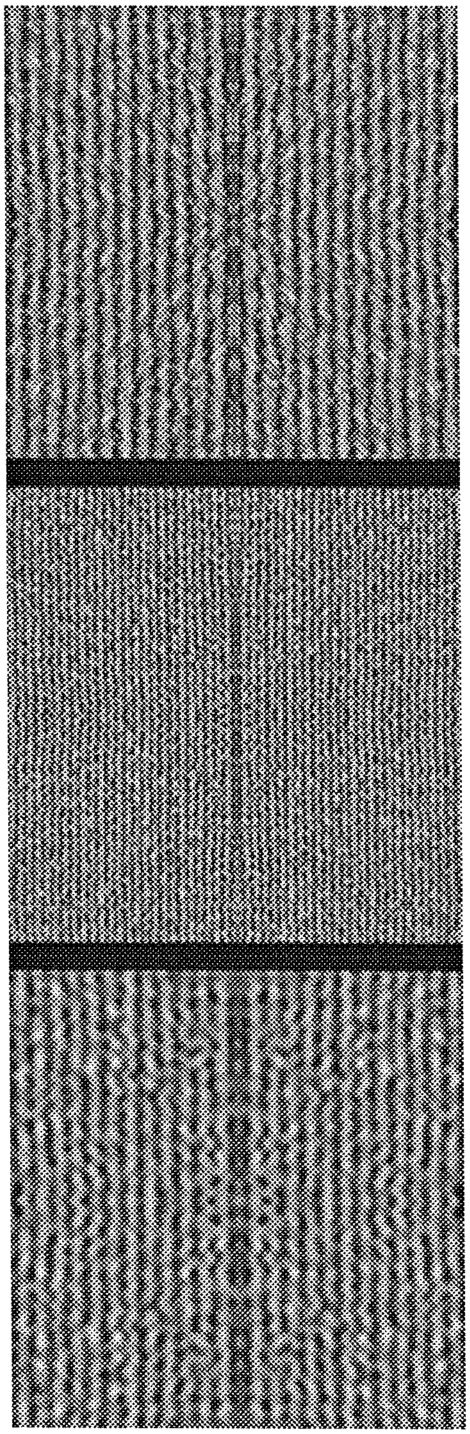

Figure 2.3: Sample images from DENIMSEQ, DENIM2SEQ, DENIM3SEQ are shown top, middle and bottom respectively. The DENIM2SEQ is a subsampled version of DENIMSEQ and DENIM3SEQ is an expanded version of DENIM2SEQ

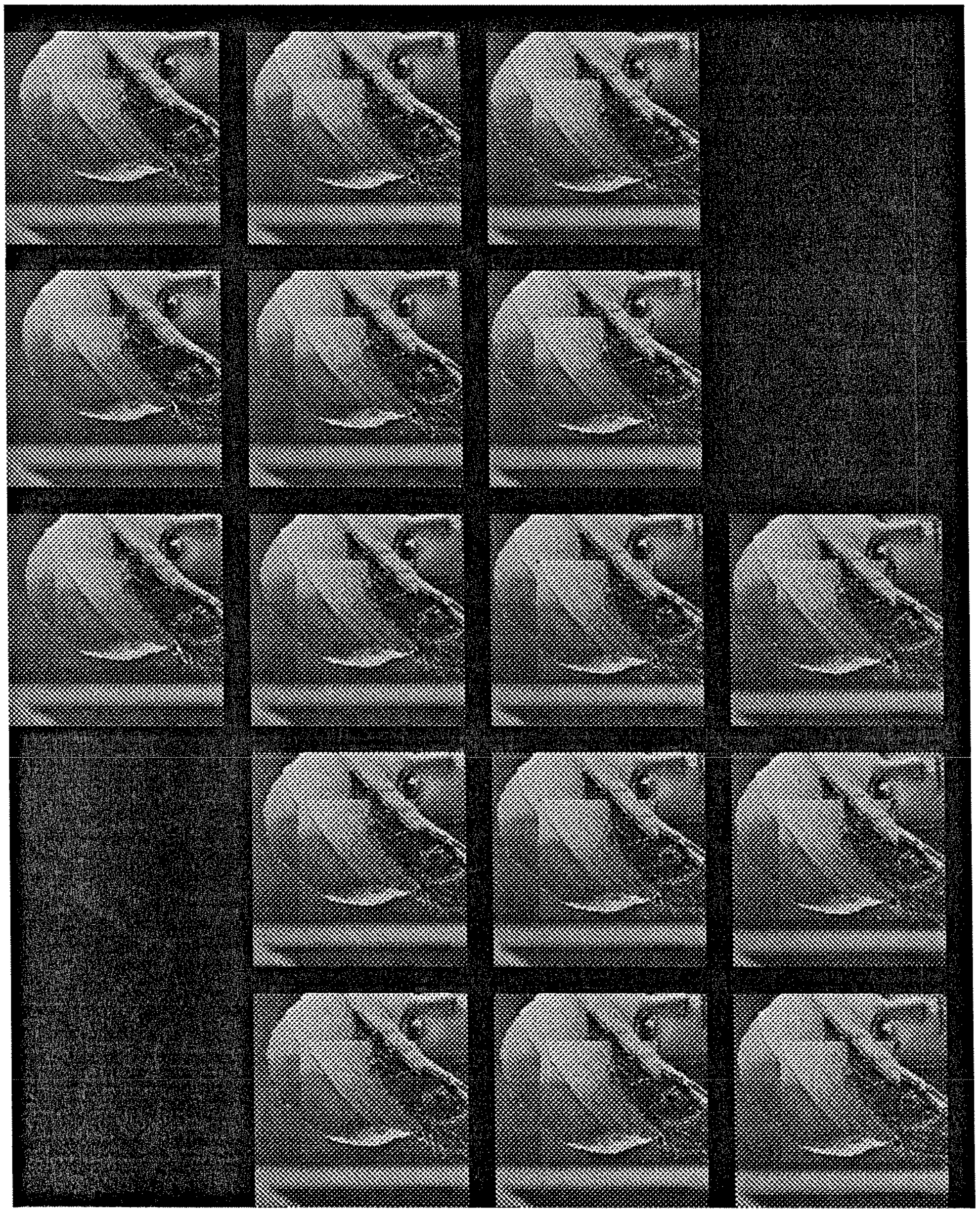

Figure 2.4: 16 frames of the original LENASEQ in 4 rows of 5 cols. The border moves 1 pix/fr up for the first 11 frames and then 1 pix/fr to the left. Inset moves 1 pix/fr to the left for the first 11 frames and then 1 pix/fr up.

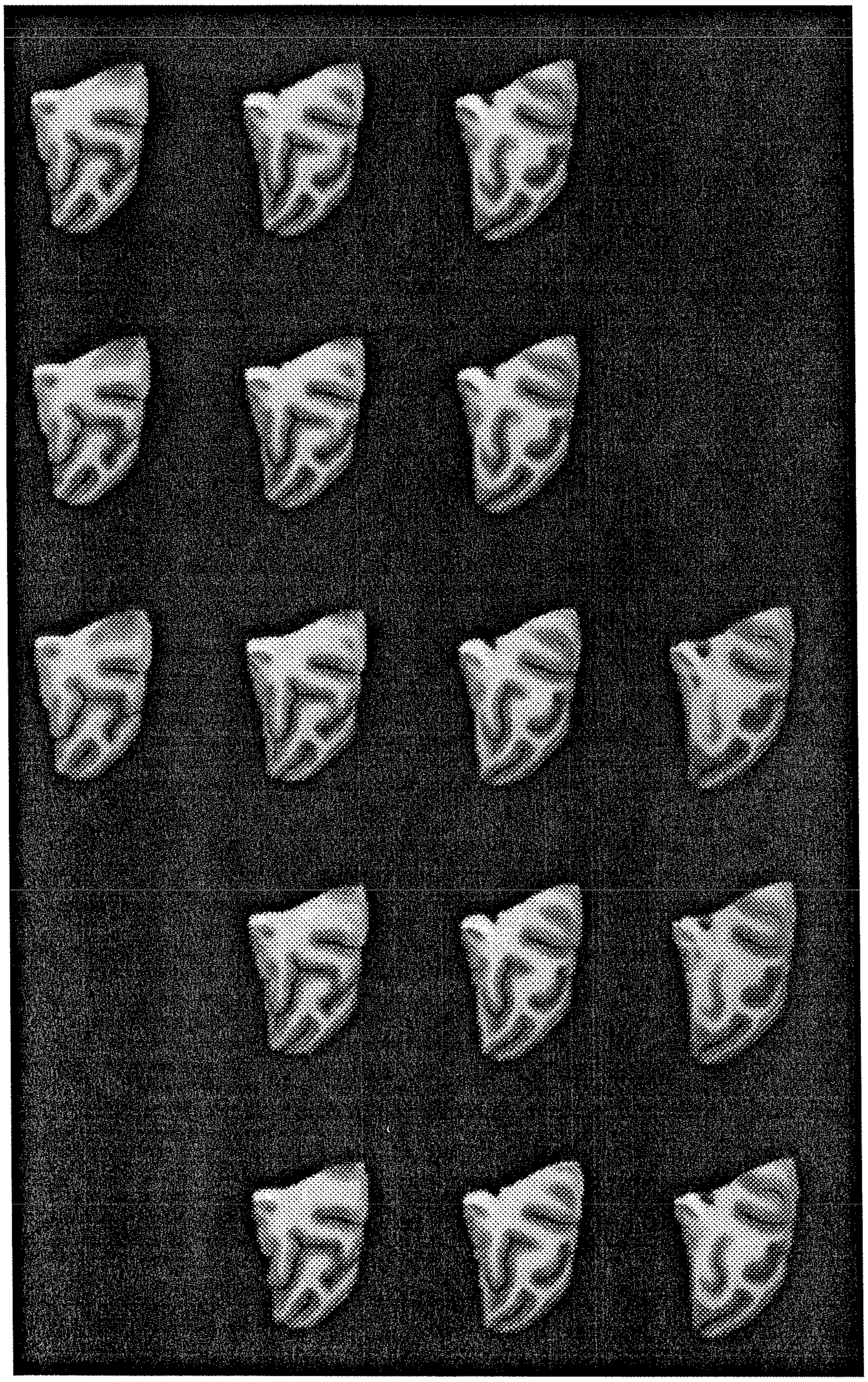

Figure 2.5: The BRAIN sequence. 16 consecutive cross sections arranged in 4 rows of 5 cols. The four images in the center column were used as reference while calculating the flow-fields for the images in each row. Note the appearance and disappearance of some gray matter stripes.

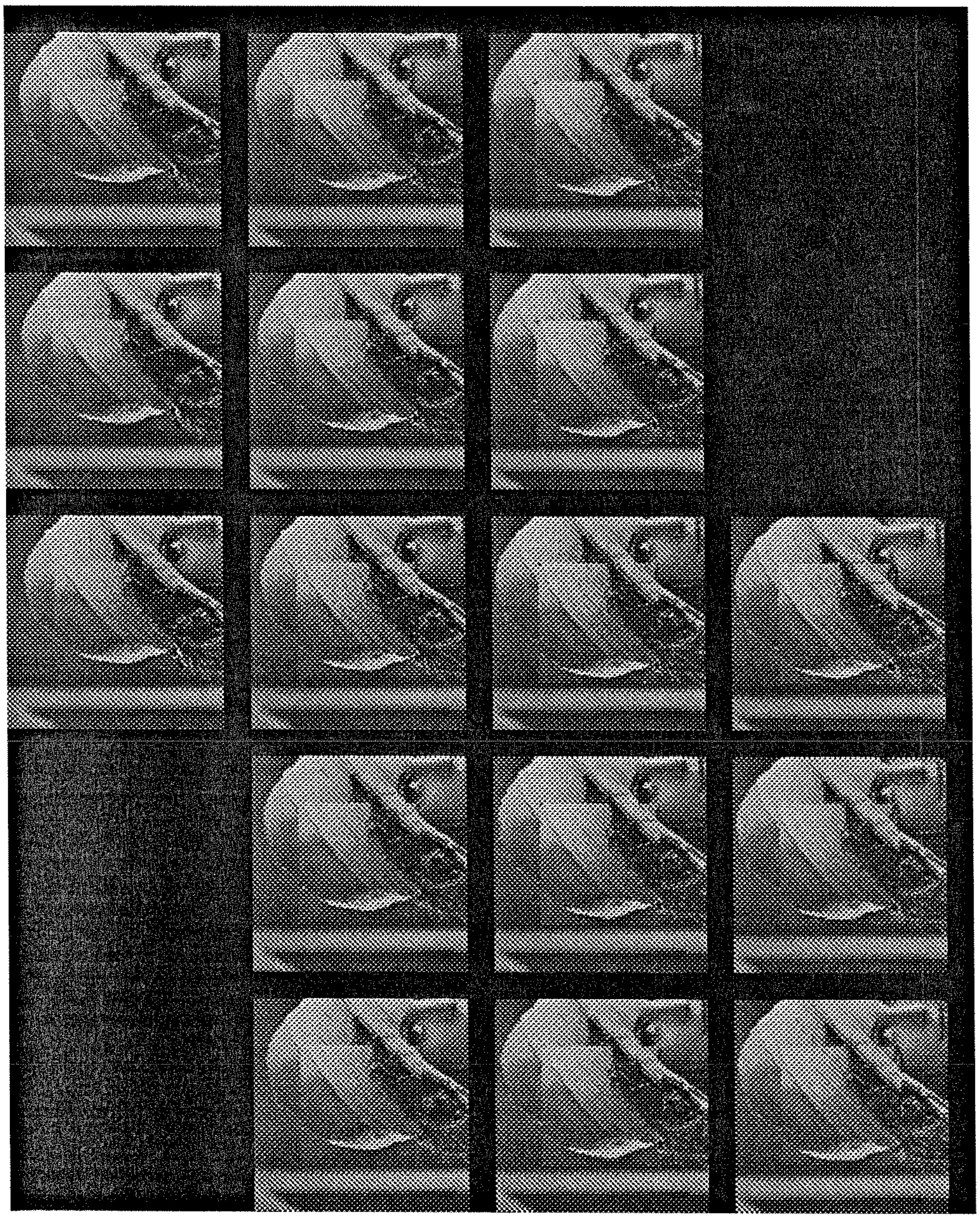

Figure 2.6: The LENASEQ reconstructed using center column images and estimated flow fields. The flow field was estimated in 10×10 blocks for these 160×160 images.

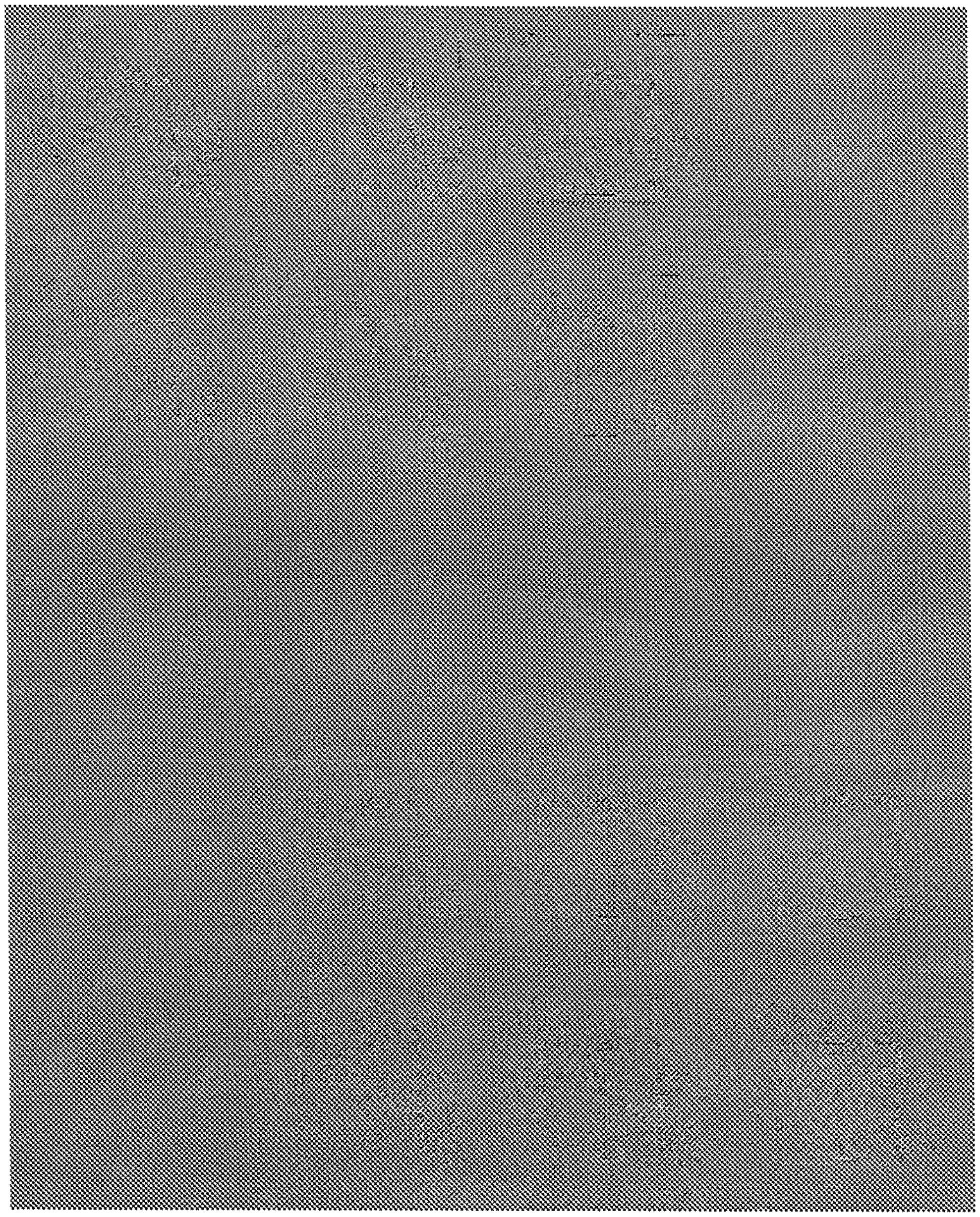

Figure 2.7: The Err-Lenaseq images: the difference between Fig.2.4 and Fig.2.6. Note that most errors are at motion boundaries (flow-field estimation error) and at hard edges of the image (interpolation errors during reconstruction).

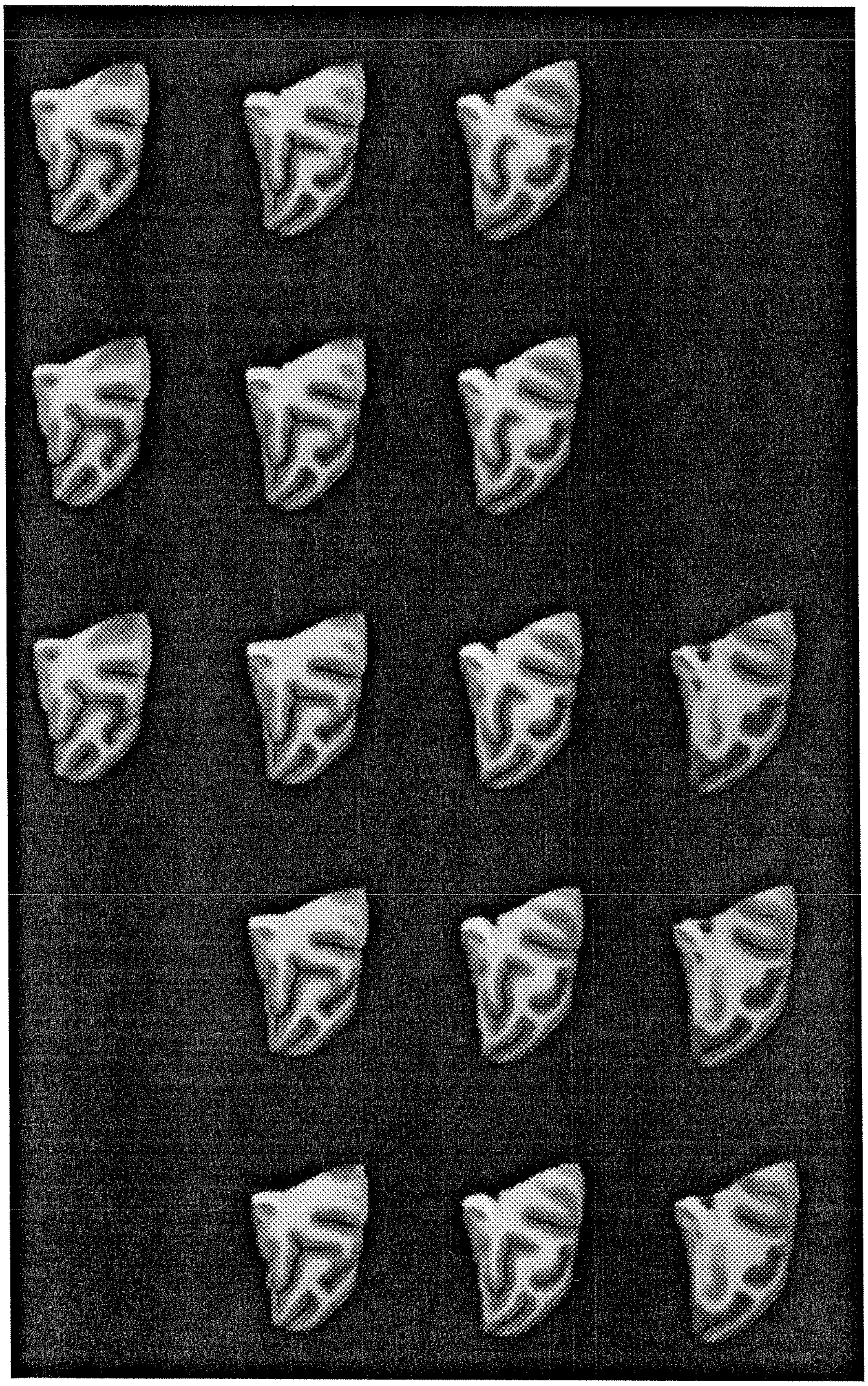

Figure 2.8: The BRAIN reconstructed using center column images. The algorithm tries to match the shapes of all the stripes but has no way of adjusting the intensity. Hence the big horizontal stripe in the bottom row has the right shape but wrong intensity.

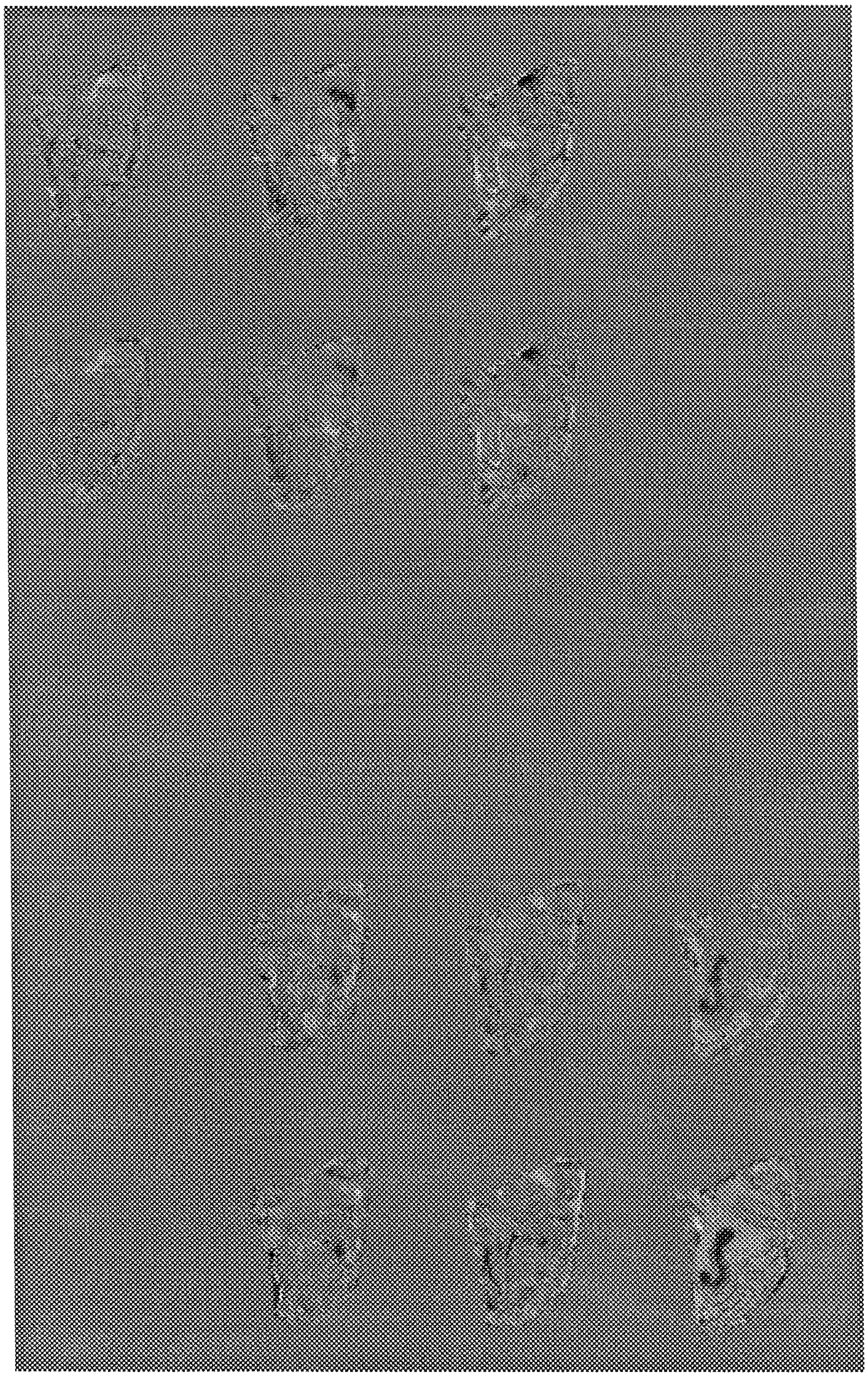

Figure 2.9: The ERR-BRAIN images. : the difference between Fig.2.5 and Fig.2.8. The violation of the constant brightness assumption leads to large differences between the two sequences even though the algorithm recreates the right shapes.

Chapter 3

Image Analysis Using Maximally Informative Filters

3.1 Introduction

The objective of analyzing image sequences with spatio-temporal filters is to determine the spatial and temporal frequencies present in a region of the image and estimate the velocity. This relatively straightforward idea has been well explored and the main problems faced have been the design of narrowly tuned filters and the overall computation cost. The earliest work in this field was driven by the idea that one had to have very narrowly tuned filters since the input frequency could only be localized to the extent that the individual filters were localized in frequency This idea was best articulated by Fleet in [Fleet and Jepson, 1989] : "the filter should be tunable to a narrow range of orientation or normal velocity, with its amplitude spectrum concentrated about a line through the origin in frequency space." Even if this can be achieved, it immediately leads to the next problem - computation cost. As noted in [Fleet and Jepson, 1989], if each of the three parameters (frequencies in x,y,t) are discretized into n samples, then there are n^3 filters requiring m^3 multiplications each. For very narrowly tuned filters, the filter size m can be large. While Fleet has developed efficient ways to implement filter banks of narrowly tuned filters, it will be shown here that it is not necessary to have n filters to discretize the input parameter into n samples or design very narrowly tuned filters. Previous work on steerable filters had already shown that a finite number of filters could be used to define filters tuned over a continuum [Perona, 1994] and that these ideas could be incorporated into a multiresolution framework [Freeman and Adelson, 1991], [Greenspan et al., 1994]. In the presence of quantization, noise and approximation errors, how accurately can different finite bases estimate a continuous input parameter like orientation? Since the constraint of steerability does not define a unique family of kernels, additional constraints can be imposed. One such constrain could be a constant cost, *i.e.,* given a limit on the computation (and hence on the number of filters) what kind of kernel generates the best family of filters? The answers to these questions require a quantitative measure or figure of merit defined on the output of the filter bank. The approach adopted here is to estimate the information content of the filter bank output.

It will be shown that the combined output of a set of broadly tuned spatio-temporal filters can localize parameters for a single input more narrowly than the tuning widths of the filters. A single input implies an input with a single dominant orientation, spatial frequency and speed. In that sense, the present work is similar to that of Heeger [Heeger, 1987] and Ogata [Oagata and Sato, 1992]. Heeger used a least square fit to the output of the entire filter bank, a collection of Gabor filters, to estimate a single set of input parameters. The trade off between the amount of information extracted (as measured by the accuracy of estimation) versus the computation cost (as the number of filters increased with narrower tuning of the filters) was not addressed. The standard approach to this issue has been to arrange the amplitude spectra of the filters so as to avoid an overlap [Heeger, 1987]. The issue is not quite so simple, however. Firstly, a filter that has its support partially overlapping another does provide some extra information. Secondly, overlaps lead to redundancy and that is not necessarily bad. Redundancy in the filter outputs could enable robust coding and decoding of the information in the presence of noise in the system. While this may not be an important issue for computers, it may have influenced the evolution of biological systems. The nature and role of this redundancy could therefore be important for understanding biological systems. An information theoretic approach allows one to calculate the amount of information and the amount of redundancy that a filter bank composed of a certain number of filters, of a certain type, will have for a given *prior*. The most common *prior* is the assumption of a single dominant input pattern. The increase in cost to estimate multiple values for each parameter at each point for complex inputs turns out to be exponential. So long as the dominant pattern has at least a $2:1$ contrast ratio over the others, its parameters can be estimated with the single input *prior*. A family of curves can be generated for this *prior* showing the information output for various filter widths and various filter bank size. These can be used to design the optimal filter bank for a fixed computation cost. It will be shown that such a system can tolerate system noise large enough to reduce the signal quality of the filter outputs to $16dB$, an essential capability for systems like the brain that must transmit signals as spike trains.

Any encoding algorithm is useful only if there exists a viable decoding algorithm. A robust decoding algorithm will be presented that gives the input parameters for any given filter bank output. The creation of the codebook and the robustness of the decoded output to noise will be illustrated with reference to a particular implementation. The issues of accuracy and multiple inputs will also be addressed in the context of codebook creation and information available. It will be shown that there is actually more information in the filter bank output than implicitly assumed by Ogata in [Oagata and Sato, 1992] where each filter was used independently to constrain the local velocity. As a result even 10 filters could only localize the orientation to ± 90degs. It will be shown that 8 filters can be used to determine orientation to ± 12degs in the presence of a single dominant input. The issue of handling multiple inputs will be shown to be a matter of trading off the accuracy in measuring each input versus the number of different inputs that must be simultaneously measured. This multiple input detection can be done with any filter bank, not just by one with narrowly tuned filters as is usually suggested. This decision regarding the trade off between accuracy and multiple inputs can be made *after* the filtering process by using a different codebook for decoding. The decoding algorithm will be found to be invariant to input contrast. Coupled with a Laplacian pyramid representation, the system can thus be optimal with respect to information/cost and invariant to input bias and contrast.

3.2 Information Content of Filter Output

The role of filters in motion detection is different from that in noise removal or blurring. For noise removal or blurring, the aim is to produce an image with an altered spectrum that is in some way more desirable than the original. In that context it is convenient to analyze filters and filtering operations in terms of their frequency response and their effect on the input spectrum. For motion detection, however, the aim is extraction of specific information, namely orientation, spatial frequency and temporal frequency. The filters operate on images not to produce new images but to produce outputs that will help one decide on the nature of the input. The information content of a filter output is of primary interest in this context while the frequency response is relevant only so far as it influences the output's information content. The design of filters for motion detection should thus deal directly with the information content of the filter outputs and an information theoretic approach to the problem is the most instructive. Due to the rapid increase in cost for trying to estimate parameters for multiple patterns at each point, it should be attempted only when necessary. In the presence of a significant contrast difference, the task may be decomposed serially: the dominant pattern can be analyzed first and the result incorporated into the *prior* for the next pattern. Most of the development here will be for the single dominant input only and the extension to multiple inputs will be discussed in a separate section.

3.2.1 The Single Filter, Single Input

Suppose there is a filter F that acts on an input x to give $F(x)$. The information content of any analog measurement is dependent on the noise in the output. Let us assume that given the noise in $F(x)$, the dynamic range of the output can be divided into N distinct significant levels. The quantization of the output could also result from the fact that the input is quantized (say to 256 levels) and hence the output is to be regarded as significant only up to a certain precision. Let the output levels be denoted as $f_1, f_2, ...f_N$. If for a particular input the output is $F(x) = f_k$, then what is the information content of this output ? Information theory tells us that the bits of information, I, is given by (all log are base 2)

$$I = -\log[\mathcal{P}(F(x) = f_k)]. \tag{3.1}$$

Having quantized the output to reflect its precision, let us now use the filter response $F(x)$ to correspondingly divide the input range into partitions Δ_i such that each partition represents the range of inputs that will give the output f_i. If the quantization is coarse enough

$$\mathcal{P}(x \in \Delta_i \mid f_j) = \delta_{ij}. \tag{3.2}$$

The quantization of the output and the corresponding partitioning of the input space will not form a part of the actual algorithm. They are constructs that are useful in the analysis of the problem. The information content of $F(x) = f_k$ can now be expressed as

$$I = -\log[\mathcal{P}(x \in \Delta_k)]. \tag{3.3}$$

This is the information for those cases where $x \in \Delta_k$, something that happens with a probability $p_k = \mathcal{P}(x \in \Delta_k)$. The expected information output from the filter can now be expressed in terms of the probabilities $\{ p_i \}$ as

$$I = \sum_{i}^{N} -p_i \log(p_i). \tag{3.4}$$

The same result can be derived more formally by calculating the *conditional entropy* $H(X \mid F)$ (see [McEliece, 1977]) and calculating the *mutual information* between f_n and x_i where let X denote the set $\{x_i\}$, the set of exemplars from each partition.

$$\begin{aligned}
H(X) &= \sum_{i}^{N} -p_i \log(p_i) \qquad \text{(Entropy of X)} \\
H(X \mid F) &= \sum_{i,j}^{N} -\mathcal{P}(x \in \Delta_i,\ f_j) \log[\mathcal{P}(x \in \Delta_i,\ f_j)] \\
&= \sum_{i=j}^{N} -1.0 \log[1.0] + \sum_{i \neq j}^{N} -0.0 \log[0.0] \\
&= 0 \qquad \text{(Conditional Entropy of } X \text{ given } F \text{)} \\
I(X;F) &= H(X) - H(X \mid F) \qquad \text{(Mutual Information, by def)} \\
&= H(X).
\end{aligned}$$

The information content of the filter depends on the way the various distinct output levels divide the input space. It can be shown that the maximum for I can be achieved if and only if $p_i = 1/N$ for all i [McEliece, 1977] and the maximum is

$$I = \log[N] \tag{3.5}$$

where N can only be so large as to still keep eqn(3.2) valid. If N is too large for eqn(3.2) to hold, $H(X)$ and $H(X \mid F)$ both increase. In such a case the estimation of $H(X \mid F)$ and the interpretation of the output get considerably more complex.

In terms of the filter output, the optimum filter is one that has an equal probability of giving an output over its dynamic range, $\{f_1, f_2, ...f_N\}$. In terms of the filter response and its input, the optimum filter is one that assigns an equally probable partition of the input to each of its possible output states. This means that the *prior* for the input is important in designing a filter. For a continuous input x with a *prior* probability density $\rho(x)$, an optimal filter must satisfy

$$\int_{\hat{x}_i}^{\hat{x}_{i+1}} \rho(x)dx = 1/N \qquad \text{for all } i \tag{3.6}$$

where the output has N levels and it makes the transition from f_i to f_{i+1} at $\hat{x}_{i+1}$, the boundary between Δ_i and Δ_{i+1} in the absence of noise.

The role of the filter width can now be described as follows. If the filter is tuned too narrowly, then it very rarely gives an informative output. For those inputs that do fall within its support, the output is very informative since it says where the input was with great precision. On the other hand, a filter that is too widely tuned always gives a response that is not very informative since it does not localize the input narrowly. In this context, a filter can be looked upon as a quantizer of the input space and the optimal filter is an entropy constrained quantizer. The response curve of the optimal filter is determined by the input *prior* and independent of how many quantized output levels, N, are chosen for subsequent analysis. What does depend on N is the amount of information that the optimal filter can provide.

3.2.2 Filter Bank, Single Input

It may be argued that while a single filter should not be narrowly tuned, a filter bank having many narrowly tuned filters could surpass the performance of one broadly tuned filter. However, each individual narrowly tuned filter's output will still give very little information for most of the values of the input. The filter bank output will be more informative because there will always be at least one filter output that will be very informative for any input. Clearly, going from a single filter to a filter bank does nothing to increase the efficiency of each filter and the extra information is gained at the expense of additional computation. What the bank of narrowly tuned filters provides is a way to get more information if the computational resources are available. If the resources are available, could a bank of broadly tuned filters be just as informative? This clearly depends on how correlated the outputs of the filters are and on how much new information is being added with every additional filter. Since the first optimally tuned filter will be more informative than the first very narrowly tuned filter (filter bank of size 1), a certain amount of correlation in the output of each filter added to the filter bank could still leave the bank of very narrowly tuned filters less informative upto a certain filter bank size.

To analyze filter bank outputs, the output of all the filters will be analyzed both collectively and separately. Let there be M filters $F_1(x), F_2(x), ...F_M(x)$ and let the output of each filter be quantized to N levels as before. The output of the filter bank can be regarded as an M dimensional vector $\{f_i^1, f_j^2, ...f_k^M\}$. This filter bank will have M^N possible outputs and the input space can once again be divided into partitions corresponding to the different output states. It may be noted that many of these partitions will have measure 0 if no single input could possibly produce these output states. The easiest way to generate all the nonzero sections is as follows. For each filter $F_k(x)$, let $\mathcal{S}_k$ be the set of all transition points $\hat{x}_i^k$ as defined for eqn(3.6). Let $\mathcal{S}$ be the union of all these sets. The ordered sequence of all elements of $\mathcal{S}$ forms the ordered sequence of transition points for the filter bank output and each region between two consecutive $\hat{x}$ forms a distinct partition of the input. Within each partition, no transition points for any of the filters are traversed and hence the filter bank output stays fixed. While going from any partition to the next, at least one such point is traversed and so at least one of the filters changes its output state. If this partitioning of the input range for the filter bank output produces $\tilde{N}$ (nonzero) partitions, then the information content of the filter bank output, I^b, is given by

$$I^b = \sum_i^{\tilde{N}} -p_i^b \log p_i^b \tag{3.7}$$

where $p_i^b = \mathcal{P}(x \in \Delta_i^b)$ and Δ_i^b are the partitions of the input for the filter bank as defined by $\mathcal{S}$.

One can also partition the input using any one of the $\mathcal{S}_k$ to calculate the information content of an individual filter output, I_k, as

$$I_k = \sum_i^{N} -p_i^k \log p_i^k. \tag{3.8}$$

The information in the output can be regarded as the number of bits it would take to specify it. If the output of all the filters was regarded as a concatenation of independent filter outputs, then one would require $I_1 + ... + I_M$ bits to describe it. However, eqn(3.7) tells us how many independent bits are really required to specify the collective output. The excess of the sum over I^b thus gives the redundancy of the filter bank output, R^b.

$$R^b = \sum_k^M I_k \quad - I^b. \tag{3.9}$$

Thus given the filter response $F(x)$, the number of filters M and their placement over the input range, it is possible to calculate the total information content and the redundancy in the filter bank output. Note that the different placements of the filters over the input range could make the I_k unequal even if all the filters have similar tuning widths.

The relevant figure of merit for a filter bank depends on the amount of post-filtering noise in the system. If this is almost zero, as in a computer, then R^b is irrelevant and I^b should be the figure of merit. (If computers did not use error correcting codes and had high bit error rate, then it would be necessary to encode filter outputs.) On the other hand, if one uses a front end to do the filtering and transmits the filter outputs back to a system via a noisy channel for further analysis, as in [Burt et al., 1992] or the brain, then the redundancy is required to protect the information. The figure of merit must then include R^b. As will be seen later, the choice of filters that maximize I^b always has an $R^b > 0$. Since increases in R^b beyond a certain point will not make a meaningful difference, the figure of merit can be taken to be I^b with the additional constraint that $R^b \geq R_{min}$ where R_{min} is to be determined by the noise level. The optimization problem at a fixed cost now becomes one of chosing a tuning curve for individual filters that will maximize the figure of merit at a constant M. This is a more objective way to choose the sigma for Gabor filters than to say that the filter spectra should not significantly overlap. The more significant contribution of eqn(3.7) is to provide a relationship between information gained and cost incurred since the cost varies at least linearly with M, the number of filters. (In practice, larger M imply narrower filter responses in the frequency domain and thus more expensive filters.)

3.2.3 Outputs of Gaussian Filter Banks

In order to see the variation of I^b and R^b with M, it is necessary to pick a certain input *prior* and a family of filter responses. In this section, the behavior of filter banks composed of Gaussian filters acting on a single input with a flat *prior* will be examined. The analysis of orientation using oriented Gabor filters or oriented pyramids [Greenspan et al., 1994] would correspond to such a situation. It must be noted that for a flat prior, the optimum filter would be one that has a linear response and hence evenly spaced $\hat{x}_i$. However, a filter with a linear response to input orientation cannot be easily synthesized and so the best possible Gaussian filter must be found.

The information content of a filter bank depends on three parameters of the filter bank - the number and placement of filters, the filter response and the accuracy attributed to the outputs. As will be seen, the first two are relevant to the design process since a filter bank having an optimum combination of filter responses and number of filters will continue to be optimum when the number of output levels is changed. Since a family of Gaussian tuned filters was considered, the different responses were characterized by the sigma of the response functions

$$F_i(x) = exp(-(x - x_i)^2/\sigma^2). \tag{3.10}$$

The input range was $0 \leq x \leq 1$ and the filter centers were evenly spaced in this range. For all subsequent analysis in this section, the σ values must be seen in the context of this range.

Fig. 3.1 shows I^b as a function of M, the number of filters in the filter bank, for six different values of $\sigma = 0.01, 0.05, 0.10, 0.20, 0.40, 0.60$. I^b was calculated for $M = 1, 2, 4, 8, 16, 32, 64$ and for $N = 16$, a 4 bit output for each filter. For the lowest values of M, the narrowest filters are least informative but even the most informative

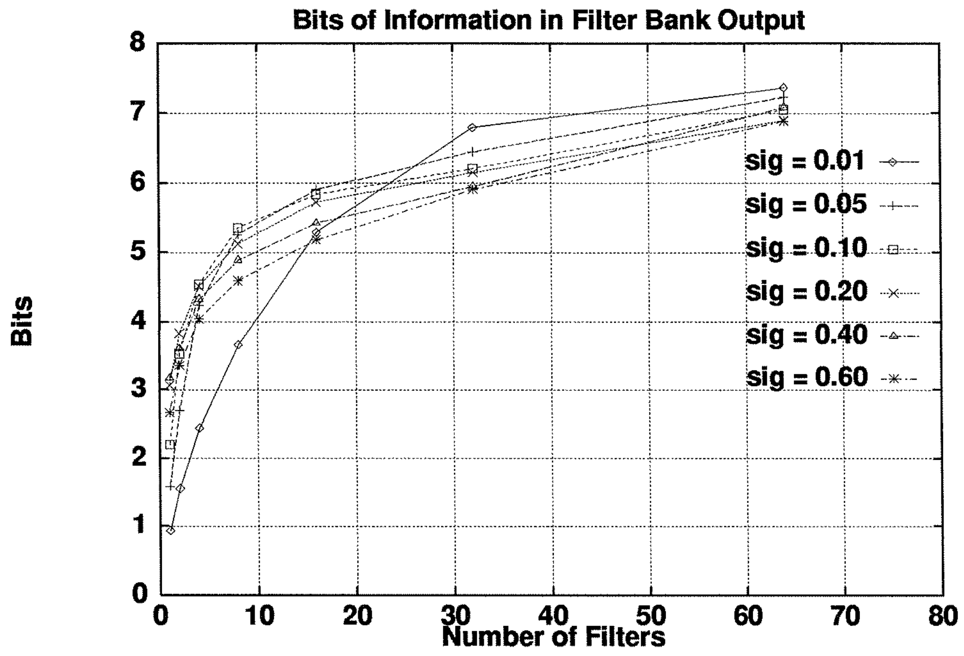

Figure 3.1: Increase in information in filter bank output with number of Gaussian filters for different sigma

single filter ($\sigma = 0.4$) has $I = 3.2$, not the theoretical maximum of 4 given the 4 bit output. This is so because a Gaussian filter can never really achieve the linear response that would be necessary to satisfy eqn(3.6). As the number of filters is increased, the I^b for the broader filters begin to saturate faster due to their increased overlap. Eventually even the filter bank with the narrowest filters begins to saturate giving less than an extra bit of information as the number of filters is doubled from 32 to 64. Beyond a filter bank of $M = 8$, one is clearly in a region of diminishing returns. At $M = 8$, the optimum $\sigma \approx 0.1 \approx 1/M$. Fig. 3.2 shows the redundancy R^b for filter banks in the range $M = 1..16$. For $M = 8$, $\sigma = 0.1$, the filter bank output can be seen to have $R^b = 10$ and $I^b = 5.2$.

The graphs for Fig. 3.1 and Fig. 3.2 were calculated for $N = 16$. This was an arbitrary choice and it is certainly feasible to have finer quantizations of the output.

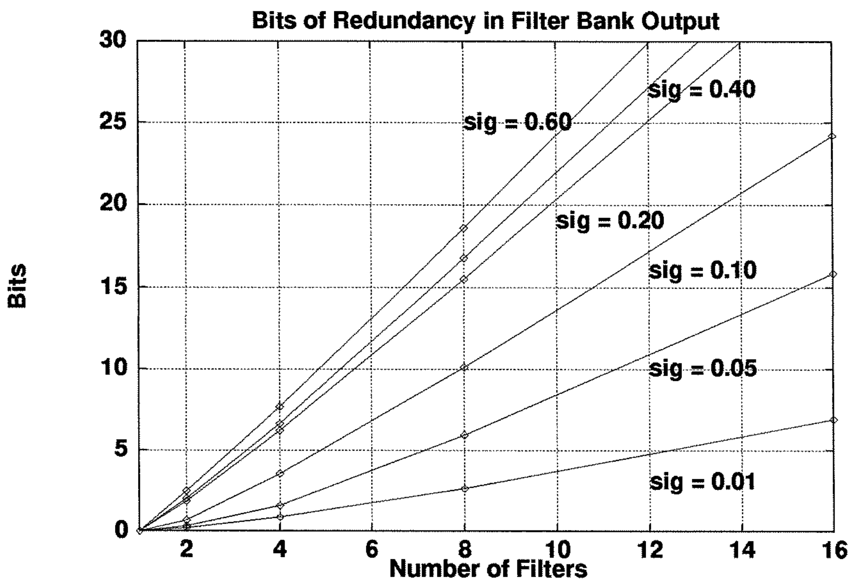

Figure 3.2: The increase in redundancy in the filter bank output with increasing number of filters and tuning widths

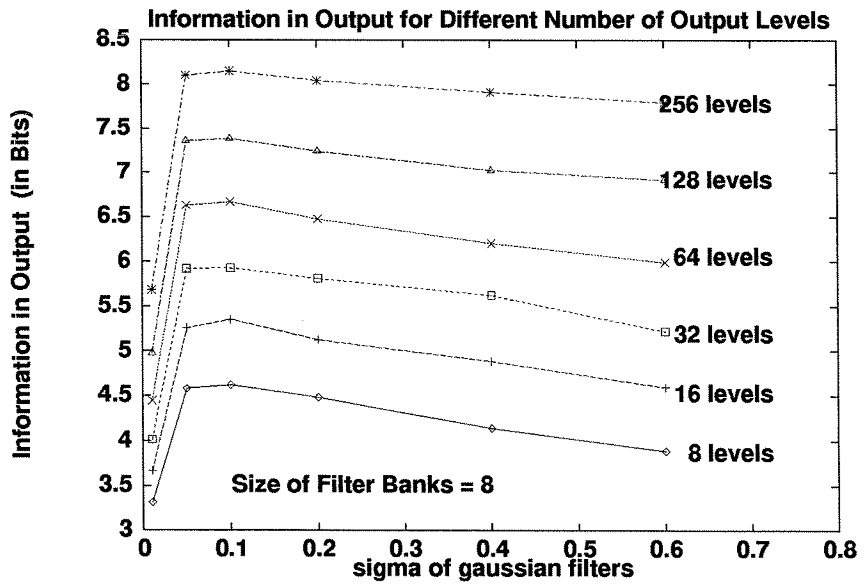

Figure 3.3: Changes in the information versus sigma curves with changes in the number of output levels

The variation of I^b with σ at a fixed $M = 8$ is shown for various N in Fig. 3.3. The values of N were chosen to correspond to outputs of $3, 4, ..., 8$ bits of precision. Two features worth noting are the constant shape of the curves for changing N and the systematic increase in I^b by approximately 1 bit for every additional bit in the outputs. An *a priori* estimation of the maximum N that creates Δ_i to satisfy eqn(3.2) is not required for the estimation of σ_{opt}. One can evaluate I^b with any arbitrary N to generate Fig. 3.1 in order to estimate $\sigma_{optimal}$. It will be shown how the performance of the filter bank can be used to estimate N in the absence of any knowledge about noise and quantization effects.

3.3 Decoding Filter Bank Outputs

3.3.1 Decoding Algorithms

What would have come as a surprise to many in the previous section is that Fig. 3.1 indicates that a large filter bank ($M = 64$) composed of broadly tuned filters, $\sigma = 0.6$, is almost as informative as one composed of narrowly tuned filters, $\sigma = 0.01$. One would intuitively feel that narrowly tuned filters ought to provide a more accurate estimate of the input parameter. This bias is the result of assuming that the output should be analyzed with a winner-take-all strategy. For narrowly tuned filters this is an appropriate decoding scheme since a set of narrowly tuned filters encodes information *sparsely*. At any given time, for a single input, only a single filter output should be high. Broadly tuned filters, by contrast, encode the information *densely*. A single input significantly activates multiple outputs. Since the information is in the pattern of activation, the decoding process must be suitably modified. If a winner-take-all strategy on the output of broadly tuned filters fails to estimate the input accurately, it is due to a failure of the decoding mechanism, not due to the encoding of less information by the filters themselves. The virtue of the analysis in the previous section is that it determines the amount of information being encoded independent of the presence of a good decoding algorithm. The issues can thus be studied separately.

The output of a filter bank with M filters can be regarded as a vector $\vec{O} = \{f_1...f_M\}$. The required decoding mechanism should try to differentiate between vectors based on all the components. It would also be desirable that the measure of similarity be independent of the vector magnitude since that would give contrast invariance in the case of frequency and orientation discrimination. The cosine of the angle between two vectors is one such measure. Denoting the vectors corresponding to two different filter bank outputs as $\vec{A}$ and $\vec{B}$

$$\cos(\theta) = \frac{\sum_k A_k.B_k}{\sqrt{\|\vec{A}\|.\|\vec{B}\|}}.$$

This measure normalizes for the output magnitude and pools together all the filter outputs. It is a measure of similarity, not difference, and the numerator is linear in A_k, B_k. Thus a large difference in the output of a single filter cannot dominate the measure. Similarly, if two vectors both have the largest value for one of the components, that alone will not give a large $\cos(\theta)$. That will depend on the pattern of activity across all the components. Note that a winner-take-all scheme would have classified two such vectors as similar irrespective of the other components.

The output vectors corresponding to known selected inputs are the codewords for this system. Given a codebook composed of the output vectors for known inputs, the decoding scheme consists of calculating $\cos(\theta)$ between the given output vector and each of the codewords. The codeword with the largest $\cos(\theta)$ would be judged closest to the output and the input parameter that produced that codeword would be the estimate of the input. The calculation of $\cos(\theta)$ does not require that the components of the vectors be quantized, an analog or floating point number will do just as well as an integer. In that sense, it is not really necessary to explicitly form the quantized levels $\{f_1, ...f_N\}$ and their corresponding transition points $\hat{x}_i$ in the input. However, the existence of N implies that there is only a certain amount of information in the output vectors, $= I$. This determines the size of the codebook since there can only be 2^I distinguishable codewords. The issue of resolution can now be addressed. If the input range is Δ_{tot} and the accuracy of the system is Δ_r then in order to have a codeword for every resulting section of the input it is necessary that

$$\frac{\Delta_{tot}}{\Delta_r} = 2^I. \tag{3.11}$$

In practice N may not be known accurately and the estimate of I could be off. One solution then is to try out bigger and bigger codebook sizes (smaller sections Δ_r) until a certain error rate is reached.

Since the decoding algorithm does make errors, it is important to know how and why. The codewords are output vectors, $\vec{O}_i$, corresponding to inputs x_i that are chosen to be exemplars of their partition, Δ_i. If too big a codebook is constructed, it will have vectors corresponding to very nearby inputs. Since the filter outputs vary smoothly with the input, similar inputs will produce similar output vectors. A certain amount of noise in the system would then lead to

$$\mathcal{P}(x \in \Delta_{i\pm 1} \mid \vec{O}_i) \approx \mathcal{P}(x \in \Delta_i \mid \vec{O}_i).$$

Fortunately, the converse is also true. Since all codewords close to a given codeword correspond to similar inputs, most decoding errors will be small. This will be seen in the numerical examples.

3.3.2 Multiple Inputs

It is easy to generalize to multiple inputs. If the *prior* for single input is flat and the multiple inputs are independent, then the *prior* for the joint distribution will also be flat. However, the information content of the filter bank output will change since $\tilde{N}$, the number of partitions of measure > 0, will increase. Thus the Δ_i will have to be recalculated and I revaluated using eqn(3.7). The possibility of multiple (D) inputs can be thought of as extending the 1 dimensional input space into a D dimensional space where the coordinates of each point in this space represents the values of the D separate inputs. The optimal filter bank, by the principle of entropy constrained quantization, is one that assigns to each possible output state a volume element in the D dimensional space such that the input has an equal probability of occurring in any one of the volume elements. Let the information content of the output in the presence of D possible inputs be I_d. If each of the inputs are to be estimated up to Δ_{rd} then

$$\left(\frac{\Delta_{tot}}{\Delta_{rd}}\right)^D = 2^{I_d}$$

$$\Rightarrow \frac{\Delta_{tot}}{\Delta_{rd}} = 2^{I_d/D}. \tag{3.12}$$

The accuracy decreases rapidly in the presence of multiple inputs. To achieve comparable accuracy for multiple inputs, I_d must be drastically increased. This can be done by increasing N (reducing the noise) or by increasing M (more computation). It may not be necessary to build even more sharply tuned filters while increasing M. In fact Fig. 3.1 shows that narrower tuning would provide only marginal gains even over a decade of sigma ($\sigma = 0.1, 0.01$) though this difference will increase with D. A convenient feature of the angle based decoding is that when the input space changes from single to multiple inputs, the filter bank and the decoding algorithm can stay unchanged. It is only necessary to pick a new set of exemplars from the input (separated from each other by Δ_{rd}) and recalculating the codebook. Thus for a given filter bank one could have a set of precomputed codebooks that could be swapped into a LUT as required. This will enable systems to operate under a constant resource constraint (fixed M) while switching back and forth between a high accuracy single input mode and a low accuracy multiple input mode. An example with plaids will illustrate this point later.

3.3.3 Multiple Parameters and Probability Measures

Filter banks are often used to measure multiple independent parameters for each input. For velocity estimation they are orientation, speed and spatial scale (or k_x, k_y, ω). The filter banks are composed of a number of filters each of which has a certain response characteristic for each parameter. Let the parameters be $\{a, b, c\}$. If the filters are separable $F(a, b, c) = F^a(a).F^b(b).F^c(c)$. Depending on the desired allocation of resources to the different parameters, one could factorize the total number of filters, M, into $M = M_a.M_b.M_c$. The three types of filters can then be independently optimized to match the *priors* for the corresponding parameters. The final filter bank would be composed of M filters $\{F(a, b, c)\} = \{F_i^a\} \times \{F_j^b\} \times \{F_k^c\}$. As in the case of multiple inputs, the principle of entropy constrained quantization of the joint input space would apply for the optimum filter bank for any given M_a, M_b, M_c.

The decoding process for such a system can be made very flexible. The filter bank output can once again be considered an M dimensional vector and the codebook composed of vectors corresponding to inputs of the type (a_l, b_m, c_n) where a_l is an exemplar from the partition Δ_l^a of a. If the goal is to measure a, b and c then the decoding process is simply that of finding the codeword closest to the output vector. Suppose, however, for a particular input it is only required to estimate a. The optimum decoder should pool together information from *all* codewords corresponding to a particular a_l in the absence of any knowledge of b, c. In the presence of knowledge, the pooling must be restricted to a subset and a weighted mean used. These kinds of operations require the definition of a conditional probability measure for each codeword for any given output vector.

Let $I(A, B, C)$ denote an input with parameters $a = A$, $b = B$, $c = C$ and corresponding to the output $\vec{O}$. The codebook consists of outputs for a set of chosen inputs, $I(a_m, b_n, c_p)$, and will be labelled as $\vec{O}(m, n, p)$. Given an output $\vec{O}$ what is the probability that the input belonged to a partition of the input space $\Delta_{m,n,p}$ represented by $I(a_m, b_n, c_p)$? Using Bayes Rule for the conditional gives

$$\mathcal{P}[I \in \Delta_{m,n,p} \mid \vec{O}] = \frac{\mathcal{P}[\vec{O} \mid I \in \Delta_{m,n,p}].\mathcal{P}[I \in \Delta_{m,n,p}]}{\mathcal{P}[\vec{O}]}. \tag{3.13}$$

For an optimally partitioned input space, $\mathcal{P}[I(A, B, C) \in \Delta_{m,n,p}]$ is constant for all m, n, p. The denominator is a normalization factor, also independent of m, n, p. Thus for the purpose of searching for the best (m, n, p)

$$\mathcal{P}[I(A, B, C) \in \Delta_{m,n,p} \mid \vec{O}] \propto \mathcal{P}[\vec{O} \mid I(A, B, C) \in \Delta_{m,n,p}].$$

Denoting the cosine of the angle between two vectors as $\mathcal{C}(\vec{O}_1, \vec{O}_2)$, the conditional probability on the RHS can be defined as $\mathcal{C}(\vec{O}, \vec{O}(m, n, p))$.

$$\mathcal{P}[I(A, B, C) \in \Delta_{m,n,p} \mid \vec{O}] \propto \mathcal{C}(\vec{O}, \vec{O}(m, n, p)). \tag{3.14}$$

The cosine is an admissible probability measure of conditional probability since with the vector components all positive, $\mathcal{C}(\vec{O}_1, \vec{O}_2) \geq 0$. Moreover, it is finite (≤ 1) and uniquely defined for any given $\vec{O}_1, \vec{O}_2$ and attains its maximum for $\vec{O}_1 = \vec{O}_2$. A normalization has to be done to ensure

$$\sum_i \mathcal{P}(\vec{O}_i \mid \vec{O}_j) = 1, \quad \text{where } i \text{ sums over all possibilities.}$$

Any power of the cosine would also satisfy all the requirements of being a measure of probability and the choice of a power will reflect on the *a priori* confidence that $I(A, B, C) \in \Delta_{m,n,p}$ will produce a vector closer to $\vec{O}(m, n, p)$ than to any other codeword. This can be incorporated into the definition of $\mathcal{C}$ as $\mathcal{C}(\vec{O}_1, \vec{O}_2) = \cos^\gamma(\theta)$. For the idealized case where eqn(3.2) holds, $\gamma \to \infty$. The estimates for any one parameter can now be expressed as (Δ^a_m is a partition in the input range of a)

$$\mathcal{P}[I(A, B, C) \in \Delta^a_m \mid \vec{O}] \propto \sum_{n,p} \mathcal{C}(\vec{O}, \vec{O}(m, n, p)). \tag{3.15}$$

The sum can be restricted to certain regions of the input space to incorporate any information about b or c. The proper normalization assures that the sum of the conditionals over m add up to 1.

3.4 Results using Pyramid Implementation

The Laplacian pyramid representation is an overcomplete subband decomposition that has been used by Anderson to design a bank of orientation tuned filters. The kernels underlying these filters, which create an 8/3 overcomplete oriented pyramid, were shown to be steerable by Greenspan in [Greenspan et al., 1994]. The oriented pyramid used 4 directionally tuned filters at each level (subband) of a single image frame. This has been generalized in the present context to 60 filters operating at each level of an image sequence, with the filters spanning orientation, spatial frequency magnitude and temporal frequency. The purpose here is to show that an image sequence can be analyzed for orientation, scale and temporal frequency using a set of filters and the mean values of these parameters can be estimated with greater precision than the inter-filter spacings in the frequency domain. Though each subband has energy concentrated around the midfrequency, the filters will span spatial frequency magnitude from $\pi/4$ to $3\pi/4$ so that fine estimates for spatial scale can be done independently within each subband. The pyramids built on image sequences are a sequence of pyramids built on individual frames, involving no averaging or subtracting across frames. Hence temporal frequencies in any subband are not restricted and the filters will span it from $-3\pi/4$ to $3\pi/4$. The main difference from prior work will be that the ensuing bank of 60 filters will be used to discriminate up to 1296 different spatio-temporal input frequencies.

If the spatio-temporal filters are the last stage of an analysis system, it makes sense to calculate a unique velocity. It is more likely, however, that they are a lower stage of a hierarchical system. In such a case, it is preferable that they do not make a hard decision but instead provide the next level with a probability measure over the range of each parameter. The results of Sec 3.3.3 make it possible to compute probabilities for various parameter values at a point or to compute the relative probability of finding certain parameter values across the image. Both algorithms were implemented and their results will be shown.

3.4.1 Pyramid Filters

Spatio-temporal filtering at a particular orientation, scale and temporal frequency was done by modulating the bandpass Laplacian image sequence with a quadrature pair of traveling sinusoids, lowpass filtering the results and computing their sum square over a fixed window size. The modulation has the effect of demodulating the input image spectrum so as to bring the desired part of the spectrum over the origin in the frequency space. The lowpass filter then rejects the rest of the spectrum and the summing and squaring measures the power by adding together the quadrature phase components. The information being extracted here is the amplitude, which is consistent with the fact that the amplitude response of the filters was used to calculate the information content of the bank. If one wished to use phase, then the optimization would have to be done based on the equipartitioning of the input phase space by the distinct filter outputs and the outputs of the quadrature pairs would be combined to provide phase information.

Using the notation G_n = Gaussian image, L_n = Laplacian image, both at level n

$$G^0_{n+1} = W * G_n; \quad L_n = G_n - G^0_{n+1}; \quad G_{n+1} = \text{subsampled}\, G^0_{n+1}.$$

The W is a Gaussian shaped lowpass filter. Once the L_n have been formed, they are modulated and filtered to produce oriented Laplacians

$$O^c(\vec{r}, t, \vec{k}, w) = LPF[\cos(\vec{k}.\vec{r} - wt).L_n(\vec{r}, t)] \tag{3.16}$$

$$O^s(\vec{r}, t, \vec{k}, w) = LPF[\sin(\vec{k}.\vec{r} - wt).L_n(\vec{r}, t)] \tag{3.17}$$

$$O(\vec{r}, t, \vec{k}, w) = \left\langle O^c(\vec{r}, t, \vec{k}, w)^2 + O^s(\vec{r}, t, \vec{k}, w)^2 \right\rangle, \tag{3.18}$$

where $\vec{k} = k[\hat{\imath}\cos(\theta) + \hat{\jmath}\sin(\theta)]$ determines the orientation and spatial frequency magnitude of the filter $O(r, \vec{k}, w)$. A bank of filters is created by setting

$$\begin{aligned} k &= k_i \quad i = 1..M_k \\ \theta &= \theta_j \quad j = 1..M_o \\ w &= w_p \quad p = 1..M_w \ . \end{aligned}$$

This creates a bank of $M = M_k.M_o.M_p$ filters spanning scale, orientation and temporal frequency as in Sec3.3.3. Since Fig. 3.1 indicates diminishing returns above $M = 8$ for a flat *prior*, filters were tuned 45 degrees apart in order to span the circle in 8 steps. Since scale could also be measured across subbands, M_k was set to 3. M_w was also set to 3 since the accuracy of speed estimation depends on both k and w. The stationary state, $w = 0$, for two opposite directions being equivalent, the filters were implemented with 4 orientations and 5 different w with w going from positive to negative. This gave $3 \times 4 \times 5 = 60$ filters that were centered around the points in frequency space defined by

$$\begin{aligned} k &= \pi/4,\ \pi/2,\ 3\pi/4 \\ \theta &= 0,\ \pi/4,\ \pi/2,\ 3\pi/4 \\ w &= -2\pi/3,\ -\pi/3,\ 0,\ \pi/3,\ 2\pi/3. \end{aligned}$$

For each point $\vec{r}, t$, there are 60 $O(\vec{r}, t, \vec{k}, w)$ forming a 60 element vector $\vec{O}(\vec{r}, t)$. This creates an overly redundant representation of the input since the elements of an $\vec{O}$ would be highly correlated with $\vec{O}$ of neighboring pixels. The vectors can hence be subsampled in $\vec{r}$ depending on the window used in eqn(3.18). The present work used a $8 \times 8 \times 4$ window for eqn(3.18) and $\vec{O}$ was evaluated only on a $4 \times 4 \times 2$ sampling grid. Given this subsampling, the memory requirement of the post-filtering representation with respect to the input to the filters (L_n) is about a factor of 2 ($60/(4 \times 4 \times 2) = 1.875$). Given the 4/3 increase in going from an image to a nonoriented Laplacian pyramid, the overall memory requirement for all the $\vec{O}$ is 8/3 that for the original image sequence. Note that there are two different issues of redundancy: redundancy across the elements of a vector $\vec{O}$ and redundancy of a given vector element across vectors defined at neighboring pixels. The former was dealt with in Sec 3.2.3 and shown in Fig 3.2 while the latter has been the motivation for the subsampling just mentioned.

As has been discussed in [Greenspan et al., 1994], the *LPF* used in eqn(3.16,17) determines the tuning of the filter. Pyramid filters constructed using 3×3 tap, 5×5 tap and 7×7 tap filters will be designated PF3, PF5, PF7 respectively. The tuning curves for these filters were determined numerically by looking at the response to oriented sine waves of a fixed amplitude and they are compared to the optimal tuning curve of $\sigma \approx 0.1$. In order to make the comparison, the tuning curve over 360 degs was mapped onto the $[0, 1]$ interval and all curves were centered on 0.5. The result, shown in Fig 3.4, shows that PF5 filters are almost optimally tuned for orientation discrimination. These orientation tuning curves for the pyramids were obtained for $k = \pi/2, w = \pi/3$. The steerability of the PF3, PF5 and PF7 was shown only for $k = \pi/2$ in [Greenspan et al., 1994] and does not hold quite so well for the wider range of frequencies being considered here. This can be seen in Fig.3.5 where the response of the entire filter bank is plotted against frequency. Here, the curves represent an average over the range of k, w. The fluctuations for PF5 are clearly noticeable. Since the decoding algorithm has been designed to be contrast invariant, such small fluctuations in the magnitude of the vector $\vec{O}$ will not affect the final output since the dot products $\mathcal{C}$ can be evaluated. If, however, some of the inputs were to give very little signal, then accuracy and robustness to noise would be affected.

3.4.2 The Exemplars and Their Outputs

Having selected the shape and distribution of the filters, the next task is to select a set of exemplars from the input space and create the codebook. The codebook size was so chosen that while doing a single parameter estimation using eqn(3.15) not more than 1/3 of the exemplars would be misclassified for any of the parameters. Even though each codeword can be uniquely identified as itself, the summation in eqn(3.15) can lead to errors if there are too many similar codewords. A codebook size

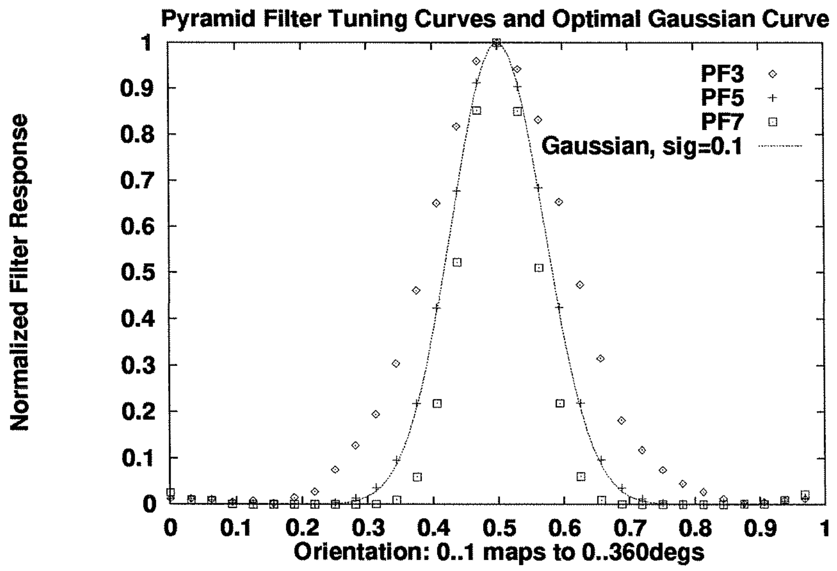

Figure 3.4: The orientation tuning curve of filters PF3, PF5, PF7 and the optimum Gaussian curve. PF5 is clearly the best of the three.

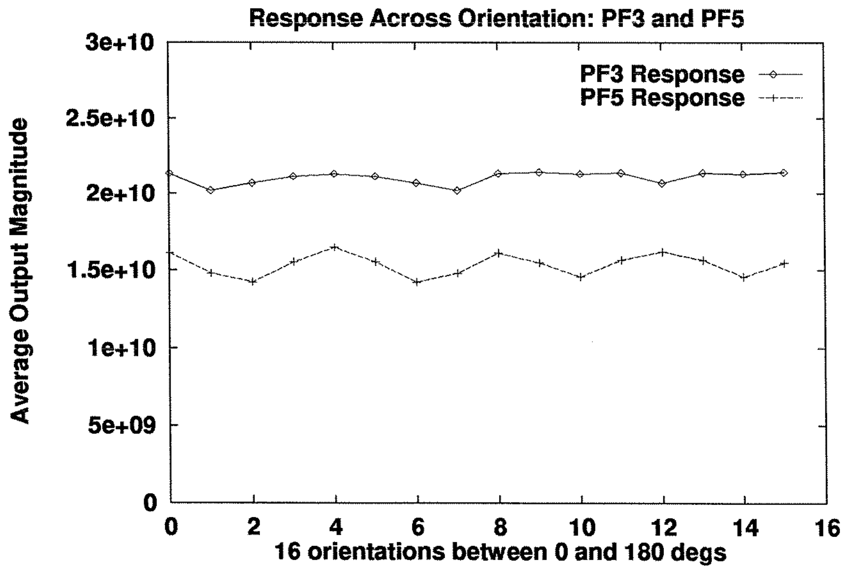

Figure 3.5: The response to different orientations, averaged over scale and speed, for PF3 and PF5. The more informative PF5 does not have the flatter response.

of 1296 was found to have such an error rate. The distribution of the exemplars was

$$\begin{aligned} k &= 4\pi/16, 5\pi/16, ..., 12\pi/16 \quad \text{(9 exemplars, } k_0..., k_8) \\ \theta &= 0, \pi/16, ..., 15\pi/16 \quad \text{(16 exemplars, } o_0...o_{15}) \\ w &= -4\pi/6, -3\pi/6, ..., 3\pi/16, 4\pi/6 \quad \text{(9 exemplars, } w_0...w_8). \end{aligned}$$

Thus 60 filters are being used to divide the input space into 1296 partitions.

It was argued in Sec3.3.1 that the system would be most likely to make small errors since neighboring code words would represent similar inputs. One would also wish that the 1296 codewords be evenly distributed in the 60 dimensional space. To see how far this was true in the present case, the cosine was calculated between every pair of codewords and a certain threshold (of the cosine value) was used to define a neighborhood around each codeword. The average, over all the codewords, of the number of codewords in the neighborhood was calculated. The mean difference (absolute) between the value of an input represented by a codeword in the neighborhood with that represented by the codeword itself was also calculated for each of the three parameters. The results, plotted as a function of the threshold on $\cos(\theta)$ used to define the neighborhood, is shown in Fig 3.6. The curve for the population of nhds expresses the nhd population as a fraction of 1296, the total. The figures for orientation and temporal frequency are fractions of 2π and for k it is π.

The redundancy in the filter outputs, as indicated by Fig. 3.2, makes the system very robust to noise added to the outputs. Since the codebook size was chosen big enough to force a certain error rate, what needs to be seen is how big these errors are and how the error rate is affected by noise in the system. This is shown in Fig. 3.7 where the solid lines indicate the error rates for the noiseless case and the dotted lines indicate the error rates for $16dB$ noise. The noise was injected into the system as a random addition to each filter output, up to $\pm 30\%$ of the output. As can be seen,

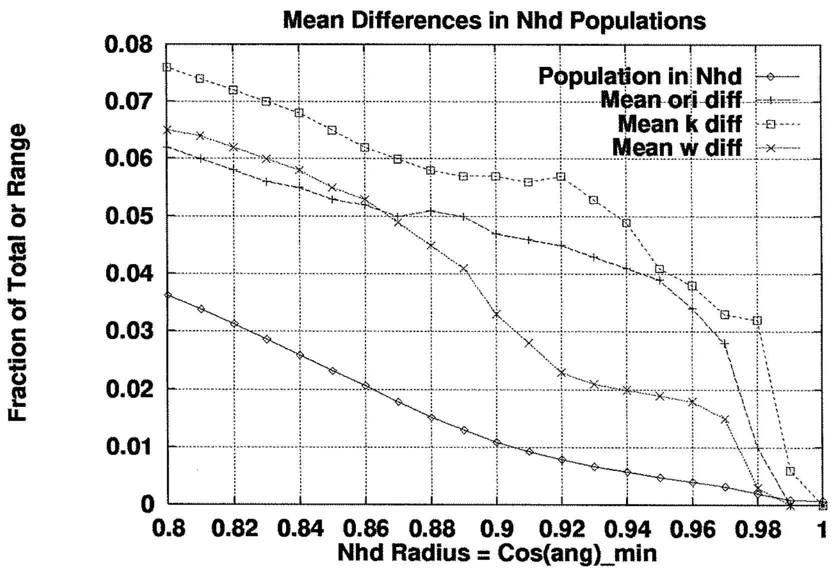

Figure 3.6: The average population of codewords around each codeword and the variation in the input parameters represented by them as a function of neighborhood radius

all the errors are misclassifications to the nearest partition. This means that with 8 filters separated by 45deg each, the input orientation can be determined correct to an accuracy of 12deg most of the time, and up to 23 all the time. This degree of robustness to noise in the outputs arises due to the fact that the information is encoded as a relative pattern of activity across 60 outputs and this pattern is not significantly altered by random fluctuation of the individual outputs. However, any decoding measure that depends on a *single* output (e.g., a winner-take-all) would be severely degraded by such noise. The presence of such robust encoding and decoding schemes may explain why biological systems with their noisy spike train based signals can be so robust. Gallant [Gallant et al., 1993] has shown how a cosine measure defined on a 90 dimensional output vector of neuron responses to a 90 stimuli input set can lead to robust classification of cells.

The three parameters were estimated separately using eqn(3.15). Orientation and

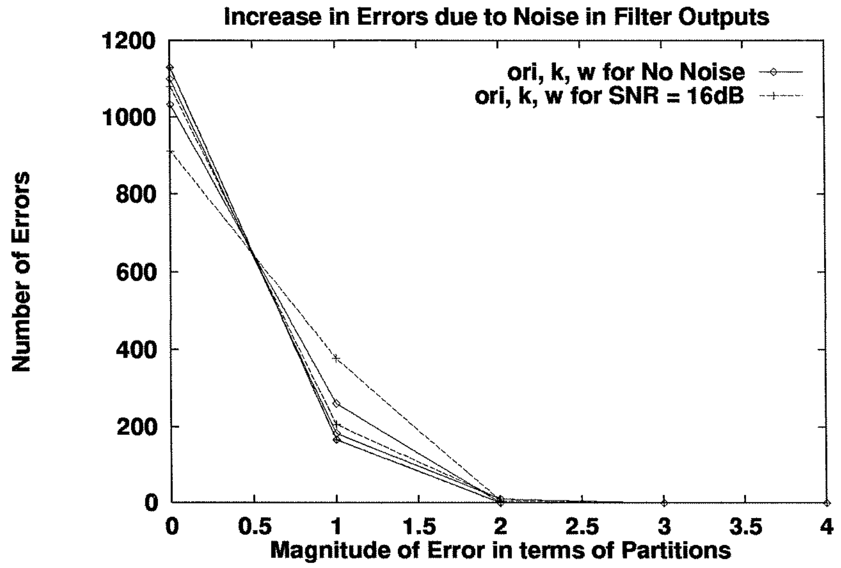

Figure 3.7: The system is robust to additive noise in the filter outputs. There were 1296 outputs: 16 partitions for orientation, 9 for scale and 9 for temporal frequency. The addition of noise leads to no significant increase in the error rate. The 3 curves for each case show the errors for o, k and w separately.

temporal frequency were estimated first and the summation for estimating k was restricted in the orientation space. This was necessary since the low frequency filters for various orientations in the frequency domain are clustered around the origin while the high frequency filters are far apart on the periphery. As a result, a summation across orientation biases the estimate of k towards low frequency. It is being assumed that separate estimates of o, k and w can be combined to give $\vec{v}$ at each point. Given eqn(3.15), a set constrains can be imposed on the range of summation if the presence of certain o, k, w or $\vec{v}$ is to be determined. The output then is a probability map showing the likelihood of the presence of the selected parameters.

3.4.3 Nonsinusoidal Inputs and Noise

The exemplars chosen while forming the codebook were all single frequency travelling sinusoids. Such a codebook can only be used to determine the parameters of a single input. In the case of multiple inputs, the result will closely match the dominant input. This can be seen by using the codebook to analyze outputs of travelling square waves. These inputs still have a unique orientation and speed but there is more than one spatial frequency harmonic present. The system's performance to square waves is shown in Fig 3.8 where the error for k was measured in terms of identifying the fundamental frequency. Since the filters incorporate a low-pass filter, high-frequency (random) noise in the inputs have little effect on the error rate. This was demonstrated by adding white noise to the square wave input. The result is shown in Fig 3.8. As was to be expected, the estimates for the spatial scale suffers an increase in error as compared to the noiseless sinusoids. However, most of the errors are just one partition away which is $\pi/16$ for k. The estimation accuracy for orientation, which could only have be affected by the noise, remains unchanged.

The LPF in eqn(3.16-17) had a $5 \times 5 \times 5$ support and there was a summation over $8 \times 8 \times 4$ in eqn(3.18) for $\vec{O}$. While such a system could work well for inputs whose characteristics were constant over space, it could have problems dealing with inputs whose parameters varied within each $8 \times 8 \times 4$ window. The system was tested using an image sequence of a rotating radial pattern of which one frame, 32×32, is shown in Fig 3.9. The image shown is the first bandpass level, L_0, and has the low frequencies attenuated. The center of rotation is at the bottom right corner, the radial ridges are 24degs apart and the motion is a rotation of 2degs/frame anticlockwise. The estimates of o, k and w were made using eqn(3.15) to generate a probability distribution. The $\pm w$ were used to define two orientations and the result gave the probabilities of finding each of 32 orientations, 9 spatial frequency magnitudes and 5 temporal frequencies at

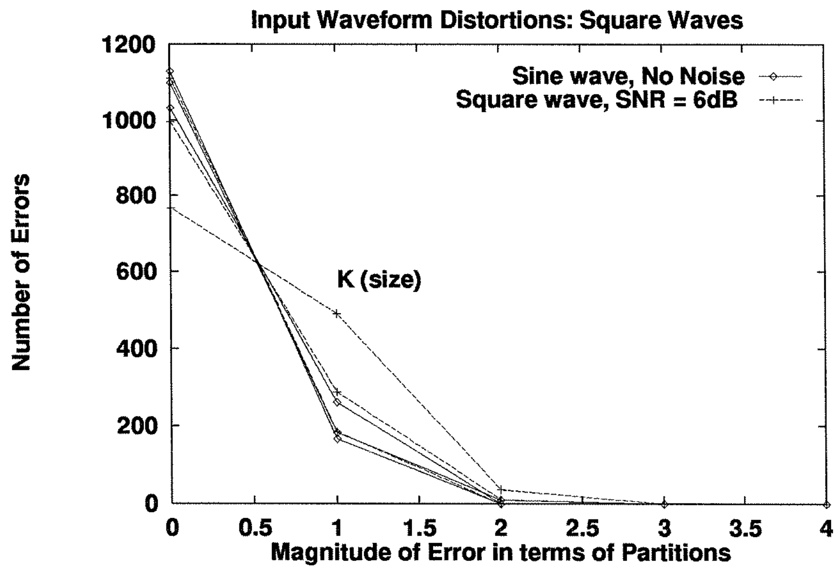

Figure 3.8: A noisy square wave input tests the systems ability to handle nonsinusoidal inputs with noise. The presence of harmonics in the square wave led to scale estimation errors. Note that most errors are 1 partition while the spacing between the filters in scale is 3.

a point. The estimates for the point $(7, 7, 5)$ are shown in Fig 3.10 and the 8×8 region about that point is shown bordered in Fig 3.9. The orientation was measured from the x-axis in an anticlockwise manner and the pattern was also rotating anticlockwise. The selected point was on the diagonal corresponding to 135 degs, or o_{12}. At a radius of 9.9, the spatial frequency of the pattern was $8\pi/16 = k_4$. The angular velocity being constant, the temporal frequency was a constant $\pi/6 = w_1$ over the whole image even though the spatial frequency was not. The results in Fig. 3.10 reflect these fact. The probability of w is peaked sharply at w_1. The spatial frequency varied within the area examined and the curve for k is broadly tuned with the maximum at k_5 with k_4 a close second. The orientation estimate is peaked at o_{12} but the presence of two other ridges widened the curve to give comparable probabilities to o_{11} and o_{13}. If forced to make a decision at this point, the system would correctly identify the w and o but be off by $\pi/16$ in its estimation of k.

A different way of analyzing the filter outputs is to query for the presence of certain parameter values in the image. The final output is a probability estimate for each point in the image. Two such maps are shown, Fig 3.11 shows the map for o_{12} and Fig 3.12 shows the map for k_3. The map for orientation is diffused at the top left since a 8×8 window cannot be confident about the orientation of wide ridges. The bottom right is partly aliased and contains multiple orientations. Hence the system was most confident about the presence of a 135deg orientation in the middle. The map for k highlights an arc since the spatial frequency in the radial pattern was constant at a given radius. The fall off for the higher frequencies towards the center of the pattern is sharper than the fall off for the lower frequencies towards the periphery. The spatial frequency falls of as $1/R$ along each ridge and so the fall off towards the lower frequencies is slower.

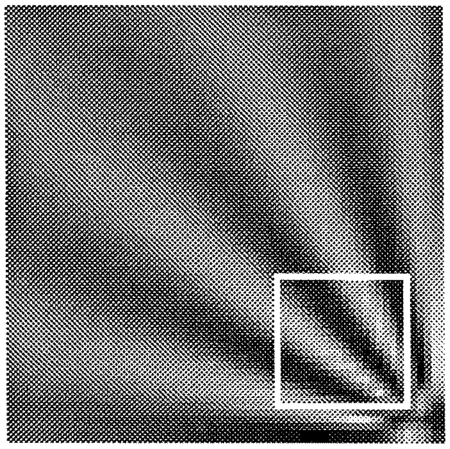

Figure 3.9: The first frame of the rotating pinwheel sequence, 32x32x10. The spokes are 24degs and motion was 2deg/frame. The box shows the area analyzed in Fig.3.10

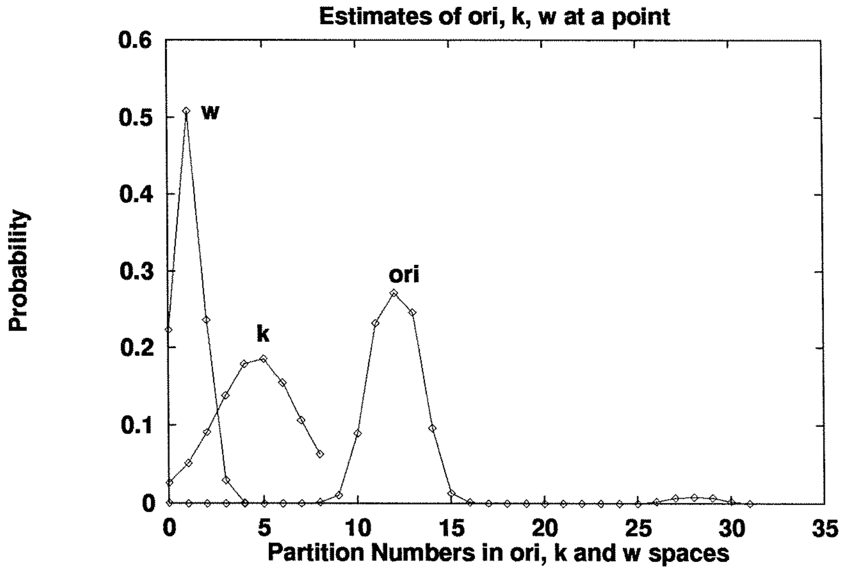

Figure 3.10: The probabilities estimated for orientation, spatial freq and temporal freq within the box in Fig.3.9. The true temporal freq was w1, spatial freq was k_5 and orientation was o_{12}.

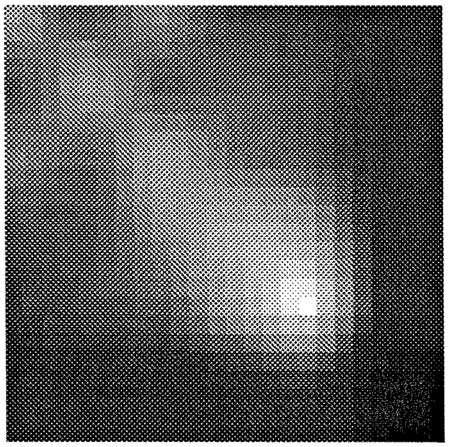

Figure 3.11: The probability map for orientation at 45 deg. There was aliasing near the bottom right. The spatial freq was too small at the top left to be determined by a 12x12 window. Hence the middle region of the diagonal is brightest.

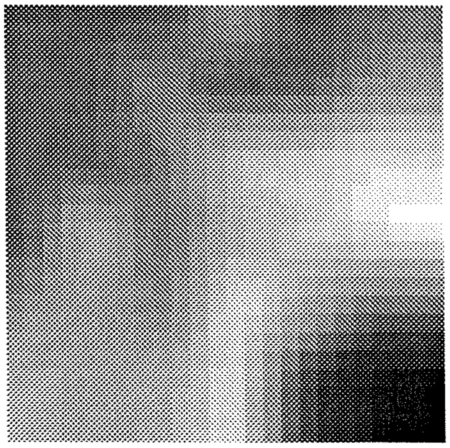

Figure 3.12: The probability map for spatial frequency magnitude $7\pi/16$. The filter bank had filters at three spatial frequency magnitudes.

3.4.4 Multiple Inputs

In this section, performance issues related to multiple inputs are briefly discussed. As stated before, to simultaneously estimate multiple inputs in the same region in the image with high accuracy involves a big increase in cost. The effort here is therefore to show how the system performs at constant cost. Given the bank of filters with 8 sets of filters tuned 45 degs apart, what can be done for complex inputs? The example with square waves showed how the dominant scale could be detected. In this section moving plaids will be used to test the accuracy for orientation estimation.

The three plaids that will be used are shown in Fig 3.13. The top left pattern consists of a horizontal grating and a diagonal grating, both of equal contrast. The top right plaid consists of a horizontal grating and a weaker diagonal grating, the contrast ratio being 2 : 1. The bottom plaid consists of a horizontal grating and a vertical grating, both of equal contrast. According to the discussion for multiple inputs, the accuracy for two inputs should go as the square root of the accuracy for single inputs. Since for single inputs the system could partition the input into 32 sections, for two inputs it should be able to partition each input into about 5 sections which is an accuracy of 65 degs. Hence for two orientations separated by 45 degs, the system should not be able to resolve the two inputs. Once one orientation becomes dominant, the system should be able to estimate that to within ± 12 degs. For two inputs separated by more than 65 degs, like the third plaid, the system should be able to resolve the two orientations and estimate them accurately. The results for the three plaids are shown in Fig 3.14. For the top left plaid, the system combines the two orientations at o_{16} and o_{20} to produce a single sharp peak at o_{18}. For the top right plaid, the system ignores diagonal at o_{20} and estimates the orientation between o_{16} and o_{17}. For the bottom plaid with its two orthogonal orientations, the system resolves the two orientations and estimates them accurate to 12 degs - with peaks at

o_{17} instead of o_{16} and o_{23} instead of o_{24}.

3.5 Summary

Filters used for detection and estimation of parameters in an image sequence should be regarded as information gathering entities. Their design should be governed by the principle of maximizing the information content of their output. The incremental cost incurred to get each incremental amount of information is an important quantity to consider while designing filter banks. When used collectively as a bank, filters not only extract information but encode it in a certain way depending on the design of the filter bank. An understanding of this process is necessary in order to use the correct decoding process at the output. An incorrect method can severely reduce the information gained. Thus even for noise free systems, the proper design of filter banks and output analyzer is necessary to extract maximum information for a given cost. The present approach can be easily extended to the design of noisy systems as well. In this context, the issue of redundancy in the output and its role in robustness to noise in the system is highlighted. Another issue emphasized here is the importance of the *prior* for the inputs. A nonuniform *prior* would require nonuniform shapes and placements of filters, even if this results in a departure from complete coverage or steerability.

The present work gives a way of evaluating the information content of filter banks independent of the decoding method. This is important since the overall performance of the bank depends on the encoding and decoding. An evaluation of the overall performance alone cannot determine the merits of the two processes separately. By first evaluating the comparative information content of the various filter bank outputs it becomes possible to decouple the filter design problem from the output analyzer design problem. Once the filter design is done, the decoder can be designed by

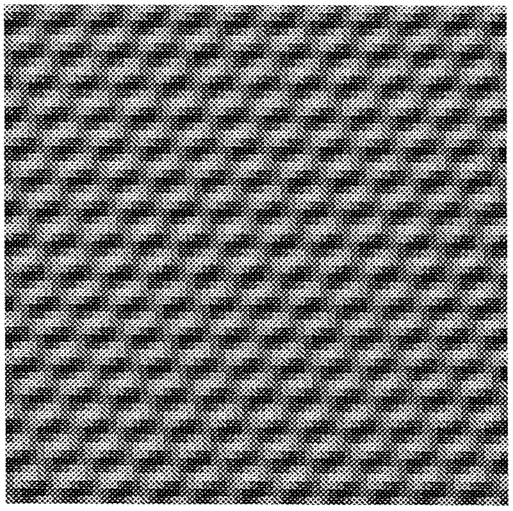
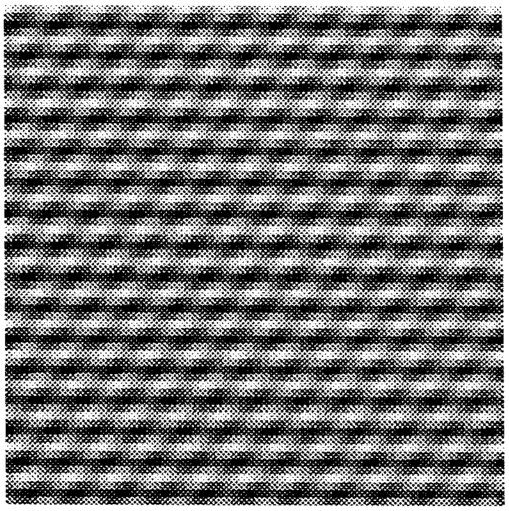
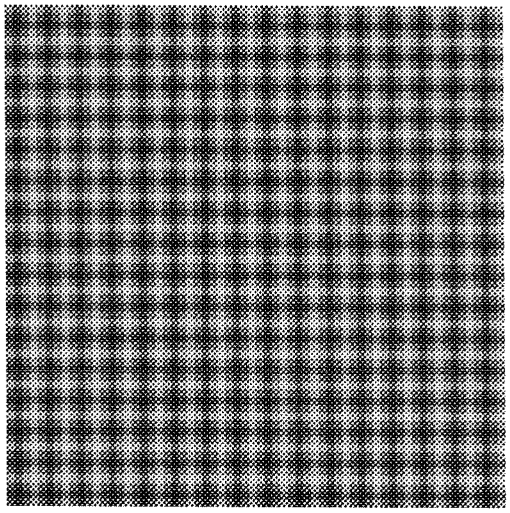

Figure 3.13: The top left plaid has orientations o_{16} and o_{20} at equal contrast. The top right plaid has orientations o_{16} and o_{20} at 2 : 1 contrast. The bottom plaid has orientations o_{16} and o_{24} at equal contrast.

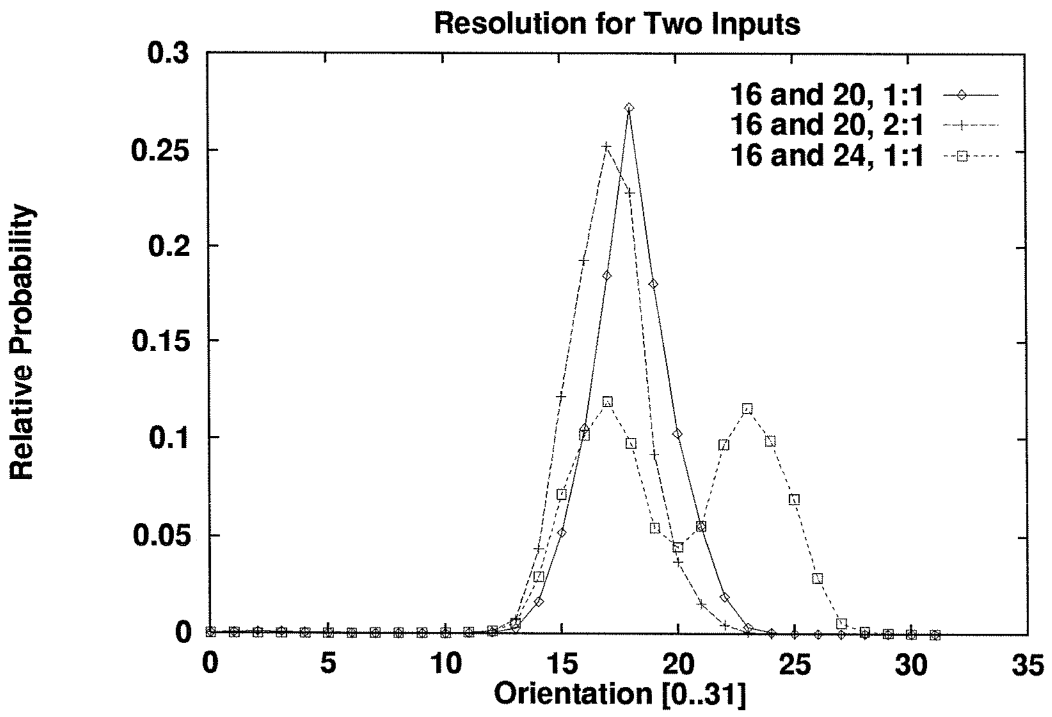

Figure 3.14: The orientations estimated for the three plaids. The presense of a dominant pattern or a large separation of the input parameters is required for an accurate estimation at a fixed cost.

checking the system performance for known inputs. The issue of handling multiple inputs is shown to be linked to the issue of accuracy. A constant cost system is proposed that could trade off accuracy for multiple inputs with minor modifications at the output.

Spatio-temporal filters are often the front end of more complex systems. Their outputs can be made more compatible with the requirements of higher levels of processing if they can be made to give probabilistic outputs. A way of doing that is demonstrated here by developing a probability measure on the filter outputs. A flexible system that can pool information across multiple parameters and handle constraints is ideal for incorporating top down instructions while doing a low level analysis. It has been shown that all this can be done with a representation that is 8/3 bigger than the input sequence. Once this representation is created, all queries regarding parameters at a point or presence of parameters within a range can be answered without additional filtering operations. The system is thus cost effective and versatile.

Chapter 4

The Burt Pyramid as an Error Correcting Code for Images

4.1 Introduction

The study of multiresolution representations of images led to the development of the pyramid [Burt, 1983], [Burt and Adelson, 1983]and wavelet representations [Mallat, 1989], [Daubechies, 1990]. The chief difference between them was the fact that the wavelet representation was a just complete, orthogonormal representation while the pyramids were overcomplete and nonorthogonal. While the overcompleteness was initially regarded as a disadvantage of the pyramid representations, it has been shown that it leads to more compact kernels for interpolation [Anderson and Rakshit, 1992] while Simoncelli *et al.* [Simoncelli et al., 1992] have shown that the critically sampled wavelet representations have no simple interpolation rules. The overcompleteness of the pyramids also make them a redundant representation, i.e., there are more possible pyramids than there are images. This fact bestows on the pyramid representations the properties of an error correcting code. It is worth emphasizing that the error recting properties of the pyramids studied in this paper are a property of the pyramid representation itself and not a result of making any assumptions about the original images. The use of pyramids as error correcting codes is thus not limited to low-pass or bandlimited images. This feature distinguishes the present work from the prior work on using pyramids for image compression and noise filtering. While the energy in the natural images may be mostly in the low frequencies, the important information is usually in the high frequencies. For example, since in military and medical applications the end user is often looking for differences in small detail or texture in the images, the preservation of high frequencies is important.

The use of the pyramid representation for image compression was studied by Burt and Adelson [Burt and Adelson, 1983]. Their technique was based on the fact that when decomposed into its bandpass components, the values were clustered around 0 for the lower levels of the Laplacian pyramid. Image compression was obtained by quantizing the Laplacian values and degradation of the reconstructed image was made imperceptable by the proper choice of the number and distribution of quantization levels. While the quantization errors in the compression stage were minimized, no effort was made to remove the errors at the decompression stage. Thus their technique could not be generalised to error correction.

A useful technique for noise removal is coring. It is based on the assumption that the noise is mostly high frequency and low amplitude. In the context of pyramids, it amounts to thresholding the lowest Laplacian band (L_0) and reconstructing the image. While removing the noise, it also attenuates the low amplitude high frequency signal. If the noise has a high amplitude and frequency or has a low frequency then coring is ineffective. It will be argued later that noise introduced during digital transmission or analog storage is usually of this type.

4.2 The Burt Laplacian Pyramid

The Burt Laplacian pyramid is described in detail in [Burt, 1983], [Burt and Adelson, 1983]. Here we review the rules for the formation and reconstruction of the pyramid and image respectively. The Burt pyramid has an exact reconstruction rule, unlike the FSD pyramid [Anderson, 1990]. Since the exact reconstruction property is essential for error correction we shall restrict all further discussions to the Burt pyramid.

4.2.1 Pyramid Formation and Reconstruction

The rules for the formation of the pyramid are recursive, with the original image being defined as G_0. The rules require two types of operations on the images, **Reduce** and **Expand**. The **Reduce** operation is a low-pass filtering followed by subsampling by a factor of 2 along each dimension. The **Expand** operation is the opposite, it consists of enlarging the image by inserting a zero between neighboring pixels, multiplying by 4 and low-pass filtering.

$$\begin{array}{lrcl} \text{Construction:} & G_{n+1} & = & \textbf{Reduce}\ G_n, \\ & L_n & = & G_n - \textbf{Expand}\ G_{n+1}\ . \\ \text{Reconstruction:} & G_n & = & \textbf{Expand}\ G_{n+1}\ +\ L_n. \end{array}$$

A Gaussian pyramid is formed (G_n) along with the Laplacian one (L_n). The reconstruction rule, however, requires only the top level of the Gaussian pyramid and the entire Laplacian pyramid. Hence, for purposes of storage, transmission and error correction, the Gaussian pyramid (except for the top level) can be discarded. A Laplacian pyramid corresponding to an image refers to all the bandpass components L_n and the top level Gaussian G_N.

4.2.2 Redundancy

The pyramid representation of an image is a redundant representation. The subsampling by a factor of 2 along each dimension means that each successive pyramid level will have only 1/4 as many pixels as the present level. Since the lowest level, L_0, has as many pixels as the original image, the total number of pixels in the pyramid will be less than $4/3N$. The limit of $4/3N$ is never attained since the pyramid is built upto a finite level only, usually till the top Gaussian image is $\approx 8 \times 8$. On the other hand, the dynamic range of the Laplacian bands needs to be twice that of the image. To see why this is necessary, consider the case where images are restricted to 0..255. An isolated 255 in a region of 0s will require a $\approx$ 255 pixel in the Laplacian while an isolated 0 in a region of 255 will require a ≈ -255 pixel. The doubling of the dynamic range requires only an additional bit per pixel and since most images have byte sized pixels or larger this does not lead to a significant increase in redundancy. The factor of 4/3 is thus a good approximation of the redundancy in pyramid representation in 2D.

The consequence of this redundancy is that there are many more possible pyramids than there are images. Suppose we consider images with a dynamic range of B, i.e., image pixels can have any one of B values. Since no restrictions are being placed on the nature of images, any array of pixels will be considered as a possible image. An image with N pixels will be referred to as an image of size N, and its corresponding pyramid will be a pyramid of size M where $M = N4/3$. For $N = 256 \times 256$ and $B = 256$ the total number of images and pyramids are given by

$$\begin{aligned} \text{Total number of Images} &= 2^{524288} \\ \text{Total number of Pyramids} &= 2^{786432}. \end{aligned}$$

The convention in this paper will be image size $= N$, corresponding pyramid size $=$

M and number of gray levels in image $= B$.

The huge excess of pyramids over images indicates that there exists the possibility of using pyramids as a 2D error correcting code for images. In order to realize that goal it is necessary to understand the nature of the mapping between images and pyramids and to characterize the pyramids.

4.3 Properties of Pyramids

The choice of Burt pyramids was motivated by the presence of an exact reconstruction rule, a pyramid constructed on an image always gives back the same image upon reconstruction. It will now be shown that this exact reconstruction gives rise to a special class of pyramids, called Stable pyramids, that are related one-to-one with images. Further, the entire space of pyramids will be characterized with the help of these and two other classes of pyramids called the Null and Constrained pyramids.

4.3.1 Definitions

For the sake of brevity, we shall define two operators **R** and **C** as the reconstruction and construction operators respectively. **R** operates on pyramids to yield images while **C** operates on images to yield pyramids. The exact reconstruction rule can now be stated as $\mathbf{RC}(\mathrm{I}) \equiv \mathrm{I}$ where I is any image.

Definition 1 (Stable pyramid) *A pyramid P is stable if $\mathbf{CR}(P) = P$.*

Definition 2 (Null pyramid) *A pyramid P is a null pyramid if $\mathbf{R}(P) = \mathbf{0}$, i.e., a zero image.*

Remark 1 *The $\mathbf{0}$ pyramid is both null and stable.*

Definition 3 (Constrained pyramid) *A pyramid P is constrained if* $\mathbf{R}(P)$ *does not involve an overflow at any level.*

Remark 2 *The dynamic range of the image and all Gaussian levels is half that of the Laplacian bands. It is thus possible that during the addition involved in the reconstruction algorithm there may be an overflow.*

4.3.2 Linearity

The operators **C** and **R** will be treated as linear operators with respect to images and pyramids. While this is true for **C** it is not strictly true for **R** due to the possibility of overflow. Even when restricted to constrained pyramids, it is possible that the addition of two constrained pyramids will produce an unconstrained pyramid. (Note: Here addition and subtraction of images and pyramids implies a pixel-wise addition or subtraction.) In most cases, however, the deviations from linearity produced by overflows is small enough to be ignored. This will be illustrated by the construction of null pyramids.

Since there are many more pyramids than images, **R** has to be a many-to-one mapping. This fact can be used to create null pyramids as follows. A 512x512 random image was generated and its L_1 calculated. Random sections of this was taken to form a pyramid P_1. This pyramid was reconstructed to produce an image I_1 = $\mathbf{R}(P_1)$. This is shown in the left pyramid in Fig. 4.1. (Each pyramid is shown as G_0 on top, L_0next, followed by G_n flush to left and L_n next to it on the right at each row.) A second pyramid is now constructed as $P_2 = \mathbf{C}(I_1)$. The difference of these two pyramids, P_3, should be a null pyramid by linearity since

$$\mathbf{R}(P_3) = \mathbf{R}(P_1) - \mathbf{R}(P_2) = I_1 - I_1 = \mathbf{0}.$$

The middle pyramid of Fig. 4.1 shows the formation of P_2 while the rightmost pyramid shows the reconstruction of P_3. Since P_1 was not stable (chances of a random pyramid being stable are almost zero) P_3 has nonzero Laplacian bands and even nonzero intermediate Gaussian bands. However the final G_0 was zero. Moreover, when this null pyramid P_3 was added to a pyramid P_4 constructed on yet another random image it resulted in a pyramid P_5 that reconstructed to the same image as P_4. Thus the following is also true :

$$\mathbf{R}(P_5) = \mathbf{R}(P_4 + P_{null}) = \mathbf{R}(P_4) + \mathbf{0} = \mathbf{R}(P_4).$$

Figure 4.2 shows the above result. The Gaussian pyramid to the left corresponds to the reconstruction of P_4, the middle to that of P_5 and the rightmost shows the difference between the two. Once again, it is the difference of G_0s that is zero while the intermediate levels have nonzero energy. There are a few isolated nonzero pixels in G_0 but they are too small to be seen (± 6, full range 0..255).

4.3.3 Stable Pyramids

Stable pyramids are a special set of pyramids that constitute a small fraction of all pyramids. They are the codewords for our error correction code. They are characterized by the following properties :

Theorem 4.1 *A pyramid P_0 is stable if and only if there is an image I_0 such that $P_0 = \mathbf{C}(I_0)$.*

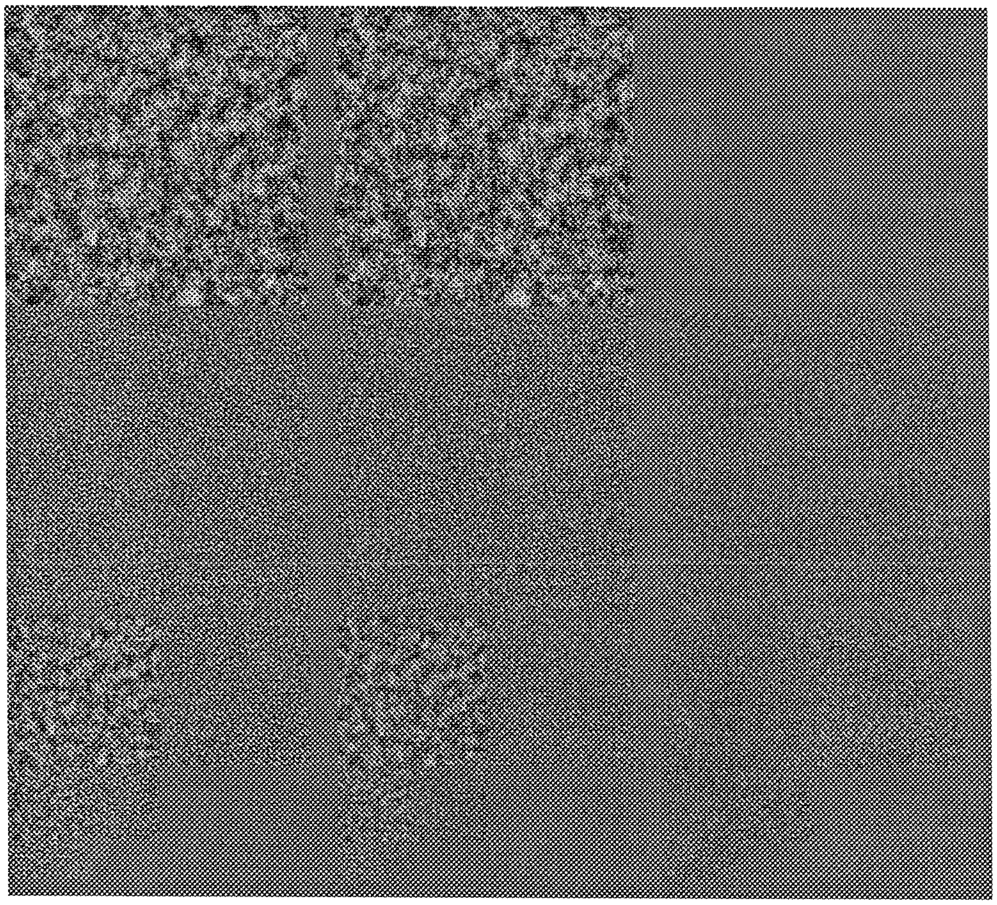

Figure 4.1: Null pyramid created by subtraction of two pyramids

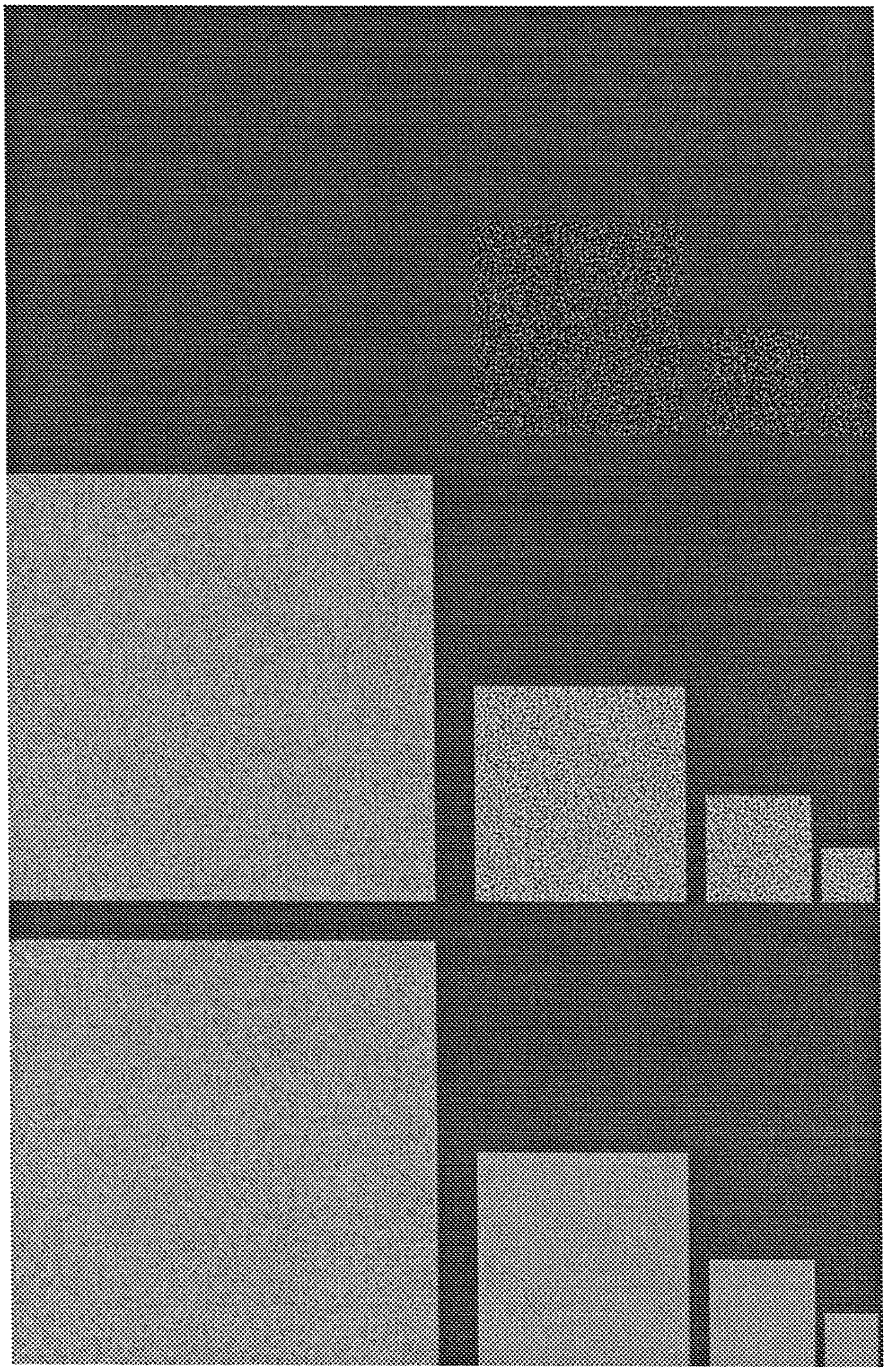

Figure 4.2: Effect of adding a null pyramid

Proof 1

$$\begin{aligned} \textit{If given} \quad P_0 &= \mathbf{C}(I_0) \\ \Rightarrow \mathbf{R}(P_0) &= \mathbf{RC}(I_0) = I_0 \quad \textit{Perfect Reconstruction} \\ \Rightarrow \mathbf{CR}(P_0) &= \mathbf{C}(I_0) \\ \Rightarrow \mathbf{CR}(P_0) &= P_0 \qquad \textit{Hence stable by definition} \end{aligned}$$

$$\begin{aligned} \textit{If } P_0 \textit{ is stable} \quad \mathbf{CR}(P_0) &= P_0 \\ \Rightarrow \mathbf{C}(I_0) &= P_0 \qquad \textit{where } I_0 = \mathbf{R}(P_0) \end{aligned}$$

□

Corollary 1 *There are B^N stable pyramids since stable pyramids are in 1-to-1 corresp to images.*

Theorem 4.2 *All stable pyramids are constrained.*

Proof 2 *By Theorem 4.1 stable pyramids are those that are produced by the process $\mathbf{C}(I)$. By the construction rules, the subtraction during formation of L_n cannot lead to an overflow because the L_n were given twice the range to explicitly avoid this problem. Moreover, if G_n is ϵ $\{0,B\}$ then G_{n+1} being a (subsampled) low-pass filtered version of G_n must also be ϵ $\{0,B\}$. Hence during reconstruction the equality $G_n = L_n + G_{n+1}$ will hold exactly and never involve an overflow.* □

While all stable pyramids are constrained, not all constrained pyramids are stable. It is possible to count the number of constrained pyramids by counting the different ways that the M pixels in the pyramid may be selected.

- The coeff of G_N can each have one of B possible values, that being the range of the Gaussians ($\{0,B\}$).

- The Laplacians have a range of {-B,B}, but the pixels have to be chosen to avoid any overflow during $G_n = L_n + \mathbf{Expand}G_{n+1}$. At each pixel, for any value that the $\mathbf{Expand}G_{n+1}$ pixel ϵ {0,B} assumes, there are B possible values ϵ {-B,B} that the L_n pixel can assume and still keep the G_n pixel ϵ {0,B}.

Since each of the M pixels of the pyramid may be picked in B possible ways, there are B^M different constrained pyramids. Since $M = N.4/3$, constrained pyramids far outnumber stable pyramids.

As has been noted earlier, addition (or subtraction) of two constrained pyramids can give rise to an unconstrained pyramid. However, any such unconstrained pyramid can be rendered constrained in a deterministic manner by adopting the following rules:

- If during reconstruction a G_n pixel should exceed B, reduce the corresponding L_n pixel such that the G_n pixel equals B.

- If a G_n pixel should go below 0, increase the L_n pixel such that it equals 0.

With the above convention, the unconstrained pyramids are transformed into constrained pyramids such that the two have identical reconstructions assuming that overflows are treated as saturation. In particular, an unconstrained null pyramid produced by subtraction of two pyramids as in Sect. 4.3.2 will converted into a constrained null pyramid.

4.3.4 Null Pyramids

We have seen in the previous section that the stable pyramids are a small subset of the set of constrained pyramids. We will now count the number of constrained null pyramids and then go on to show that all constrained pyramids can be uniquely characterized in terms of their stable and null components.

The problem of constructing constrained null pyramids is identical to that of constructing constrained pyramids, except that we have the additional constraint that G_0 must be 0. Thus we can chose all pyramid coefficients from G_N to L_1 as before, but then have no choice in selecting L_0 since it must be set equal to $-\mathbf{Expand}G_1$. Since G_1 is ϵ {0,B}, the required pixels for L_0 will be ϵ {-B,0} which is within the allowed range for L_n. Since there are N pixels in L_0 and M total pixels in the pyramid, we can chose $M - N$ pixels - each in B ways. Hence there are B^{M-N} null pyramids.

Theorem 4.3 *Every pyramid can be expressed as the sum of a stable and null pyramid and this decomposition is unique.*

Proof 3 *Given any pyramid P, construct $P_s = \mathbf{CR}(P)$ and $P_n = P - P_s$. P_s is stable by Theorem 4.1 as it is formed as $\mathbf{C}(I)$ where $I = \mathbf{R}P$. P_n is a null pyramid as $\mathbf{R}(P_n) = \mathbf{R}(P) - \mathbf{R}(P_s) = I - I = \mathbf{0}$. The uniqueness of this decomposition follows from* **Corr 5**. *Since any given P reconstructs to a unique I and the stable pyramids are related 1-to-1 with the images, P_s is unique. This forces $P_n = P - P_s$ to be unique since no other pyramid would satisfy $P = P_s + P_n$.* □

The presence of so many null pyramids raises several interesting questions regarding the choice of pyramids. Any image will give rise to a unique stable pyramid but there are a large number of unstable pyramids that will reconstruct to the image and these pyramids can be generated by adding null pyramids to the given stable pyramid. Depending on the application, one of these pyramids may be better suited. For the compression scheme given in [Burt and Adelson, 1983], the "best" pyramid would be one where the L_0 pixels are most clustered around 0. For other applications, it may be desirable to try to equalize the dynamic ranges or energies of various bands.

4.4 Error Detection and Correction

We have seen that the process of pyramid construction, **C**, maps images onto a special set of pyramids, the stable pyramids. If during the process of transmission or storage/retrieval these pyramids get corrupted by noise then they will no longer be stable. Strictly speaking, there is a non-zero probability of being transformed into another stable pyramid but for any significant amount of noise this probability is close to zero. As will be shown shortly, stability is a global property of a pyramid. This means that it is not possible to alter isolated pixels in a stable pyramid and still keep it stable. The pixels in a stable pyramid are not just related to the neighboring pixels in their band but to pixels in the bands above and below them.

4.4.1 Pyramid Instability

Why are some pyramids stable and others unstable ? Theorem 4.3 provides one answer : The null pyramid associated with most pyramids is not the **0** pyramid. However, knowing the null pyramid associated with an unstable pyramid does not help us determine the stable pyramid that is "closest" to it, Theorem 4.3 only gives us a stable pyramid that has the same reconstruction as the given pyramid. Since a noise corrupted pyramid would produce a noisy image, this is not the stable pyramid we will want while decoding. Since there are too many stable pyramids to use a LUT, it is necessary to gain a qualitative understanding of pyramid stability and instability in order to develop a decoding algorithm.

Since pyramids are a multiresolution decomposition, each band encodes a different bandpass section of the image subsampled at the appropriate rate. This makes the Laplacian bands have a band-pass spectrum as can be seen in Fig. 4.1 where the band-pass components of the pyramid in the middle look distinctly different from the Gaussians. It turns out, however, that the band-pass spectrum of the Laplacians is necessary but not sufficient for pyramid stability. If the various bands of a stable pyramid are taken in isolation they are no longer stable. This is shown in Fig. 4.3. Each of the four top Gaussian images were created by replacing any one band of a **0** pyramid with a band from a stable pyramid, L_0 for the left most, L_1 for the next and so on, and reconstructing. When pyramids were constructed on these images, the energy was spread over all the bands. This demonstrates the fact that stability is not a local property or a property of individual levels of a pyramid. This feature makes the pyramid code a robust error correcting code for images.

Those working with complete orthonormal representations like the wavelets often overlook the consequences of redundancy in the pyramid representation. It was stated in [Daubechies, 1990] that the input that would lead to $L_n = 0$ and a $G_N = \delta_{m,n}$ could be generated by reconstructing this pyramid. Such is not the case, however, since this pyramid is unstable. In fact, since it is an unstable pyramid, there is no possible input that can generate it.

4.4.2 Error Detection

We can now address the issue of error detection, namely when and how can we detect errors ? The obvious way to check for errors is to check for stability of the received (or retrieved) pyramid. This test will fail only if the noise added to the stable pyramid transforms it into another stable pyramid. The probability of this happening *decreases* with increase in the magnitude of noise added. Let the magnitude or weight of a pyramid (or image) be defined as $\sum ||\text{pixel}||$. If we consider sets of pyramids of increasing weight, the fraction of pyramids that are stable decreases. One way to see this is to recall that the ratio of stable pyramids to all pyramids goes as $B^{-(2M-N)}$ and for all constrained pyramids it goes as $B^{-(M-N)}$. We can generate sets of pyramids of increasing maximum weight by starting with $M = 1$,$B = 1$ and letting M and B

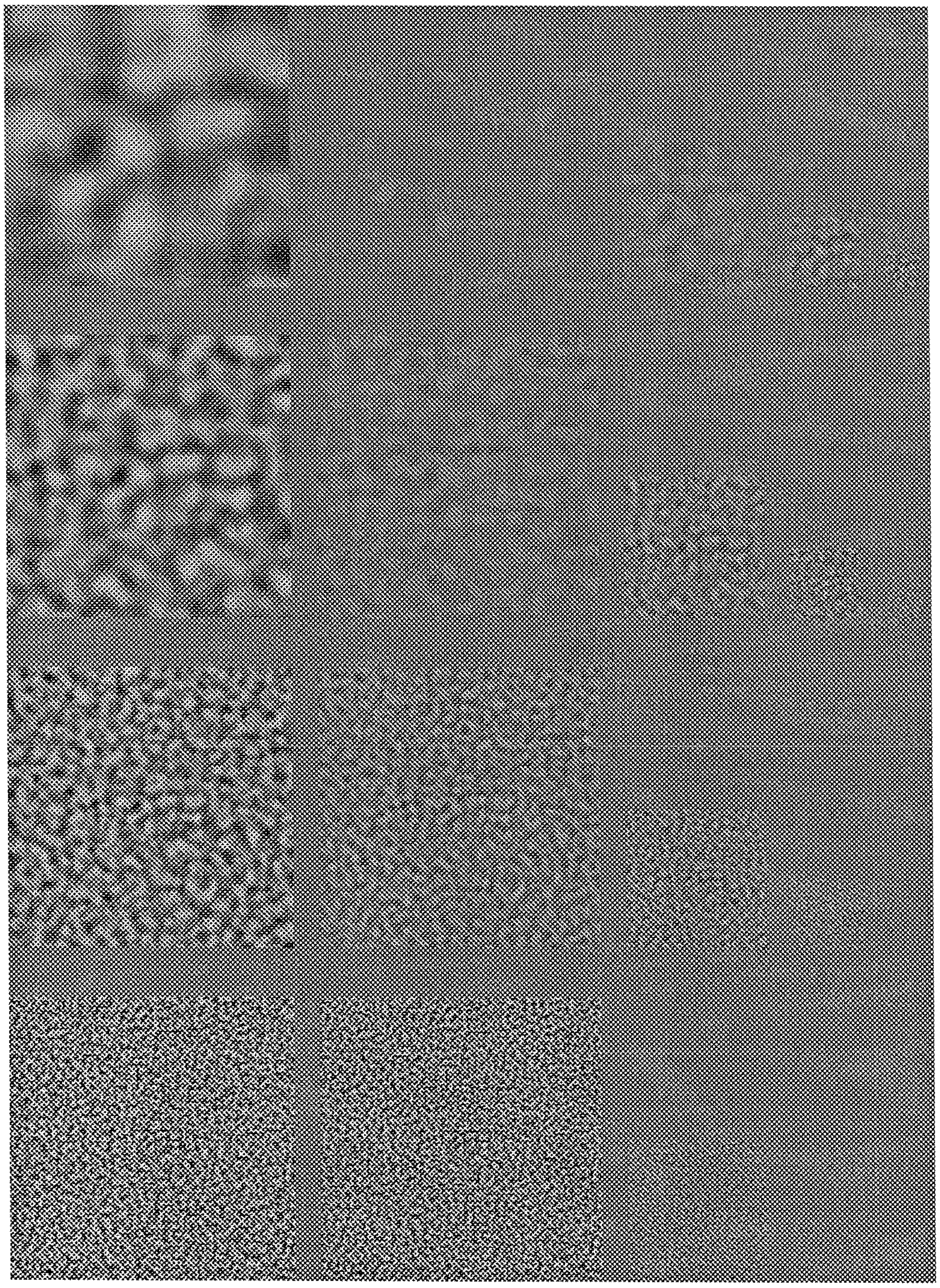

Figure 4.3: Instability of single band pyramids

increase, and the ratio will decrease rapidly. The implication for error detection is that while small errors may go undetected, the probability of significant errors being undetected is virtually zero.

We have thus far talked in terms of the error introduced into the pyramid. What we are really interested in is the residual error that may be left in the image. In general, small perturbations introduced into a pyramid can have a large effect on the image if it is done at a high level of the pyramid. Fortunately, as seen in Fig. 4.3, energy at any level cannot be stable unless it is accompanied by suitably distributed energy in other (esp lower) bands. Thus all the low weight stable pyramids are ones with energy mostly in L_0 and a little in L_1 and these are the possible undetectable error patterns. Consequently, they will produce an error in the image of weight approximately equal to themselves. Hence the small undetectable errors in the pyramids will translate only into small errors in the images. In practice the errors due to the limitation of the error correction algorithm far outweigh the errors undetected by the error detection step.

4.4.3 Error Correction

Error correction for the pyramid code is an iterative algorithm based on the fact that the noise pixels are unmatched to pixels in their neighborhood and to pixels in adjacent bands. Under the operation **CR** they will have to redistribute their energy both locally and across bands. This will happen just once since the first application of **CR** will produce a stable pyramid, though not the original stable pyramid. Let P^s be the original and P^n the noise pyramid added to it to produce P^t, the transmitted pyramid. By Theorem 4.3

$$\begin{array}{rcll}
P^s &=& P^s + \mathbf{0} & P^s \text{ is stable, its null part is } \mathbf{0} \\
P^n &=& P^n_s + P^n_n & P^n\text{'s stable and null parts} \\
\Rightarrow P^s + P^n &=& \{P^s + P^n_s\} + P^n_n & \\
\Rightarrow P^t &=& P^{sn} + P^n_n & P^{sn} \text{ is a stable pyramid} \\
\Rightarrow \mathbf{CR}(P^t) &=& P^{sn} & \neq P^s \quad \text{the desired stable pyr.}
\end{array}$$

The noise pixels may be seen as sources from which energy diffuses out to the neighboring pixels and to pixels in adjacent bands until they are at equilibrium with the other pixels. This means that if a pixel changes its value due to **CR** it could be a noisy pixel or a purely signal pixel picking up the redistributed noise energy. Hence we cannot identify noise contaminated pixels purely on account of their undergoing a change, nor can we use the direction of change since the noise is bipolar. However, since the excess or deficiency of energy from a noisy pixel gets distributed over many neighboring pixel, the magnitude of the change will be larger for the noisy pixel than for its neighbors. Based on this we can iteratively correct the errors by the following procedure: (the superscript is the iteration index)

$$\begin{array}{rcll}
P^l &=& \mathbf{CR}(P^{l-1}) & \\
\Delta P^l &=& \|P^l - P^{l-1}\| & \\
P^l_{i,j,k} &=& P^{l-1}_{i,j,k} & \text{if}\Delta P^l_{i,j,k} < th \\
&=& P^l_{i,j,k} & \text{if}\Delta P^l_{i,j,k} \geq th,
\end{array}$$

where $P_{i,j,k}$ is the pixel at (i,j) in level k of the pyramid P. The key step in this procedure is the resetting of the pixels in P^l to that of P^{l-1} when they do not change by more than th. This means that the noise energy that had diffused out is being removed. As a result the noisy pixels, even though they are now less noisy, are once again at inequilibrium with their neighborhood. Thus at the next iteration they will diffuse out some more energy and the process iteratively removes most of the noise as the value of th is gradually reduced.

In principle one could correct all errors by the above technique by chosing a small enough decrement for th. The trade-off is the decoding accuracy versus computational cost, which translates to an accuracy versus speed trade-off. In practise, the severest restrictions on th come from isolated noise pixels in L_0. The L_0 band encodes the top half of the frequency spectrum and has only one adjacent band (L_1). Hence high frequency noise in L_0 loses very little of its energy due to **CR**. Performance and speed can both be improved by using a slightly different approach to correcting all isolated errors from L_0 and then using the general procedure outlined above. The details are given in App A.

4.5 Results

The algorithm was tested on different images and for varying types and amounts of noise. In each case, a pyramid was constructed on the image and then the pyramid was corrupted by noise. The noise added to the L_0 band is reproduced identically in the image (G_0) and thus for any given type of noise, this unblurred version of the noise is indicative of the noise that the image would have picked up had it been transmitted without pyramid encoding. Two types of noise were studied, keeping in mind two potential applications - digital transmission and analog storage.

For the case of digital transmission, a pyramid would be built on an image and then the pyramid would be transmitted over a noisy channel. Since the pixels are byte size or larger, an error in a transmitted bit could cause a pixel to be in error by 1 or any 2^n. Hence the noise was modelled by picking pixels at random with a certain probability and then adding a certain amount of noise, the amount being a random number between -128 and 128. The noise added to the pyramid was thus sparse and high frequency but since it was added to all levels of the pyramid the reconstructed image (of the noisy pyramid) shows noise at all frequencies. The results for this kind of noise are shown in Fig. 4.4 and Fig. 4.5. The top pictures show the original image, the middle pictures show the reconstruction of the noisy pyramids and the bottom pictures show the reconstruction of the decoded pyramids. The F18 image was 320x240 and 0.2 % of its pyramid's pixels (about 200) were corrupted by noise. While the noise has been almost completely removed, all high frequency details of the original image have been preserved. Figure. 4.5 shows the deterioration of performance with increasing noise, here 0.5 % of the pyramids pixels were altered. While a complete removal of noise was not achieved in this case, a significant noise reduction was achieved without compromising the sharpness of the image or introducing artifacts.

Images are often stored as 2D images in analog media like films and holograms. In particular, 3D holographic media like photorefractive crystals hold the promise of dense storage. These media are noisy and conventional error correcting codes developed for 1D digital data streams are not suitable. These encoding schemes would essentially unraster 2D images, encode the resulting 1D data stream and then raster it back into a 2D image. This would render even the smooth natural images into high frequence random dot images greatly increasing their bandwidth, thereby increasing the errors made during storage and retrieval by any optical or analog system. Encoding images as pyramids would retain their 2 dimensional structure and never introduce any high frequencies. The errors or noise would tend to be blobs rather than dots, produced as a result of local defects in the media or defects in the recording system. This kind of noise was modelled by selecting a few random pixels as before, but then introducing a blob of noise centered around the chosen pixel. The size (spread) of the blob was made proportional to the peak value. The results with

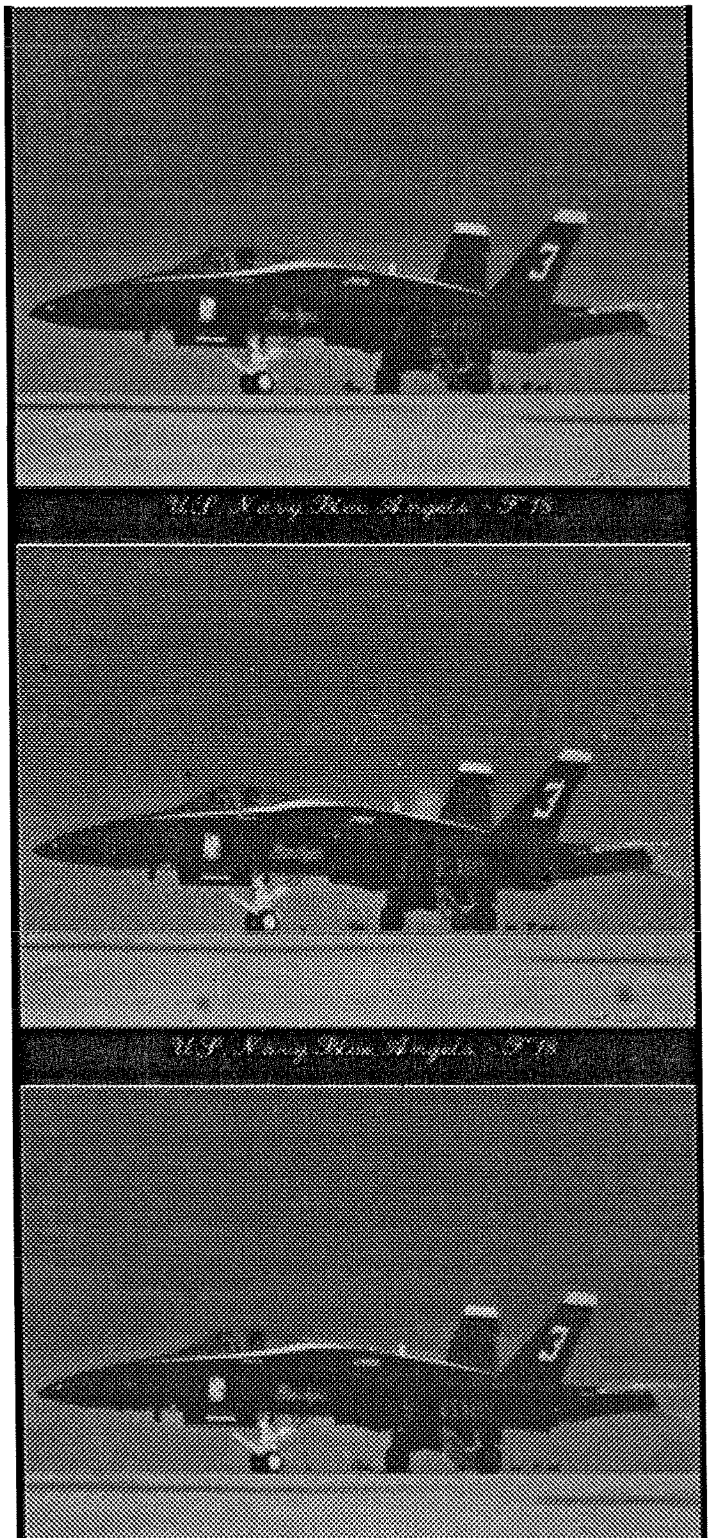
U.S. Navy Blue Angels - F18
U.S. Navy Blue Angels - F18
U.S. Navy Blue Angels - F18

Figure 4.4: Sparse high freq noise in pyramid

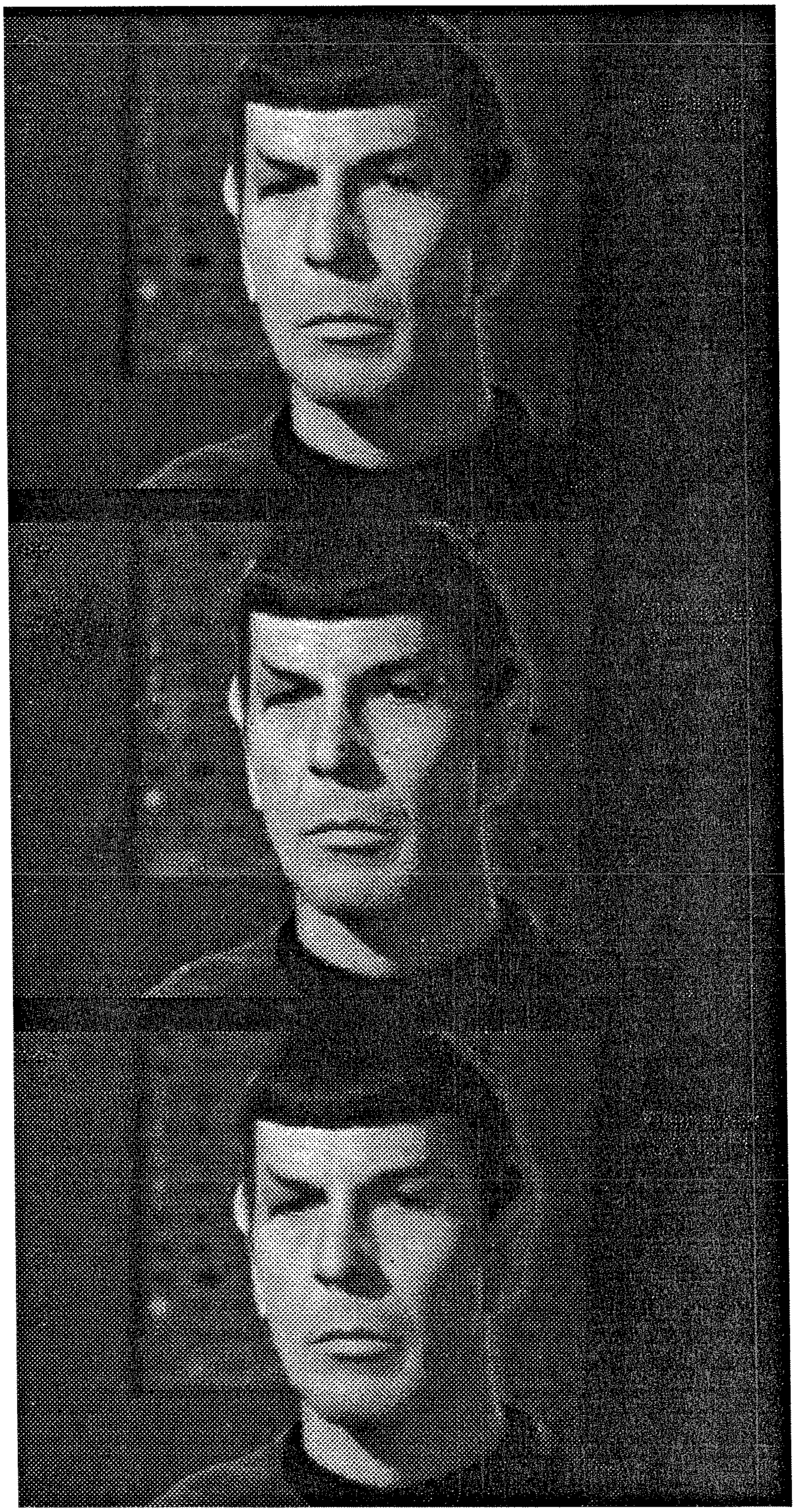

Figure 4.5: Dense high freq noise in pyramid

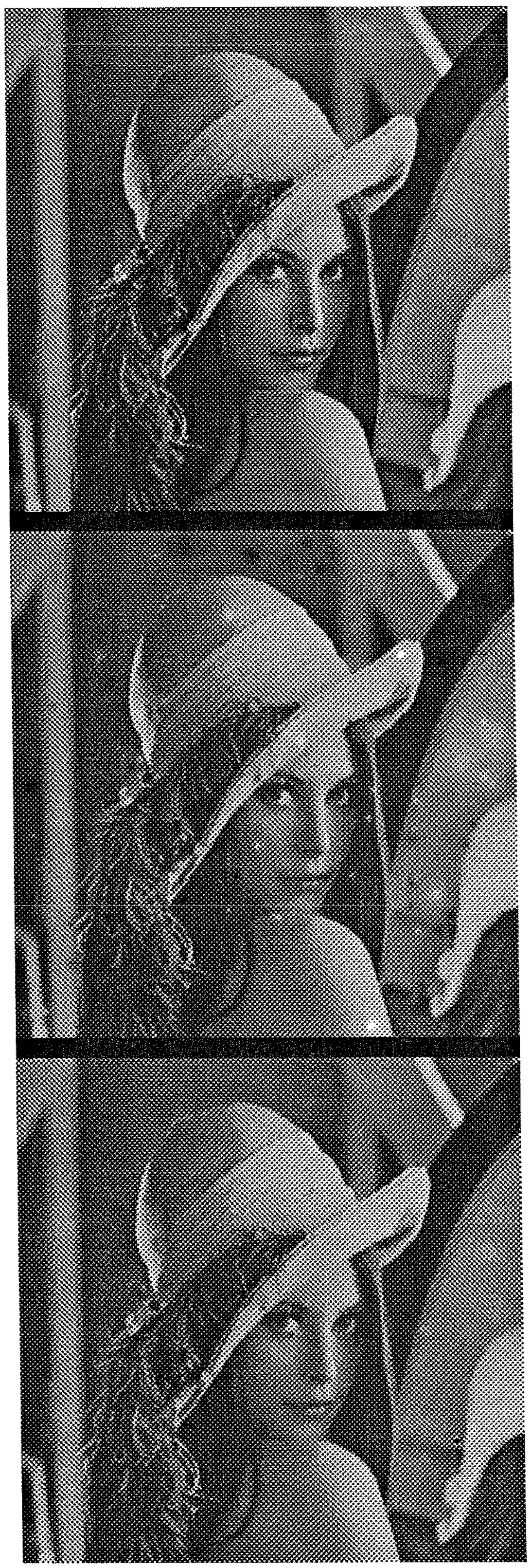

Figure 4.6: Sparse low freq noise in pyramid

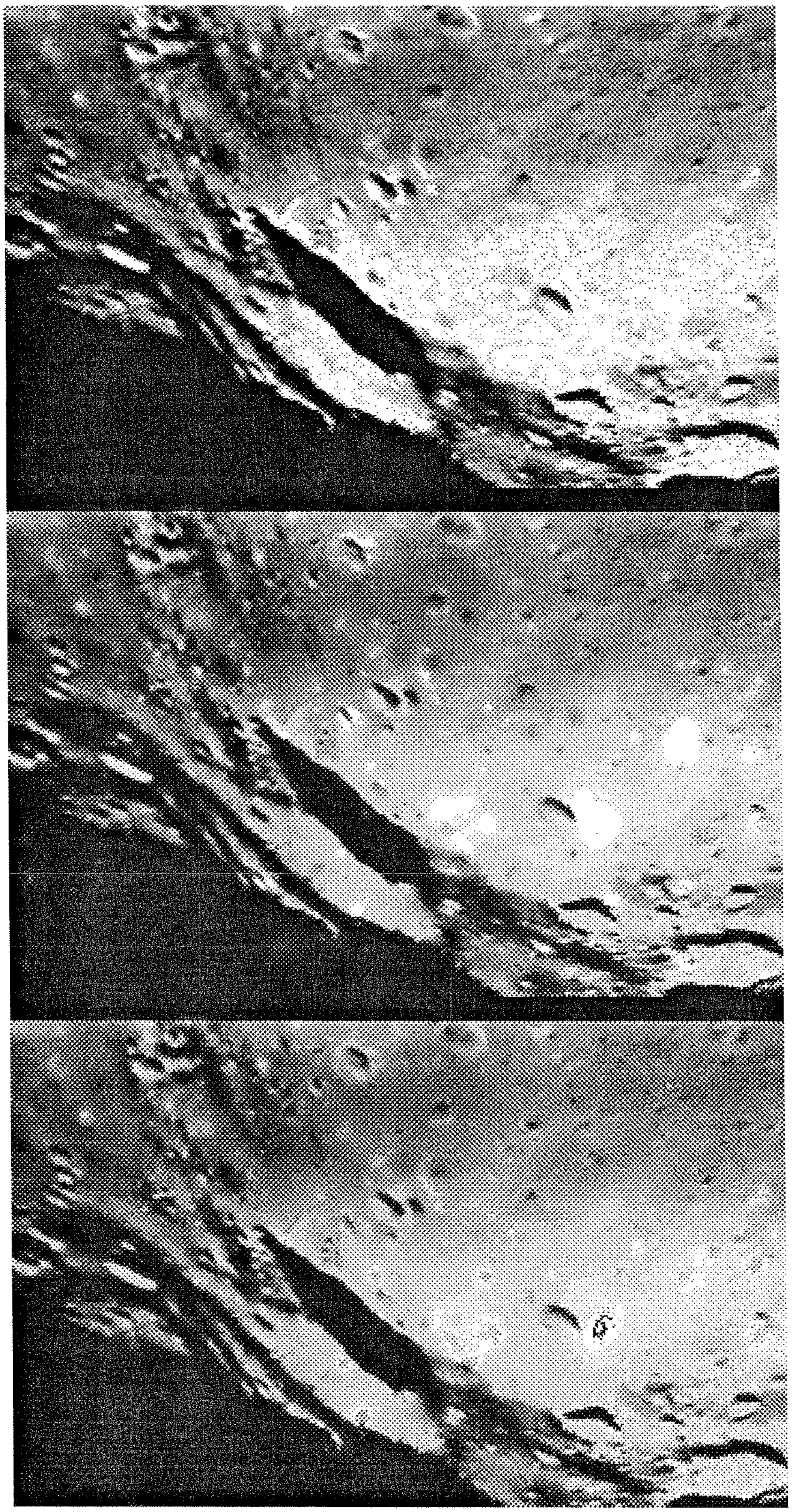

Figure 4.7: Dense low freq noise in pyramid

this type of noise is shown in Fig. 4.6 and Fig. 4.7. In the first case (Lena) 0.1 % of the pixels were made blob centers and average size of blobs was 5×5 thereby altering about 0.25 % of the pixels (about 217) while for Fig. 4.7 (Phobos) 0.3 % of the pixels were made blob centers. In the Phobos image, the noise blobs look identical to some of the genuine features in the image thereby ruling out filter based techniques for noise removal.

The speed of decoding depends on the number of iterations required, which in turn depends on the rate at which th is reduced. For the results shown in this paper, the overall time for decoding a pyramid corresponding to a 320x240 image and generating the clean image was 2.5mins on a Sparc2. The decoding algorithm allows one to trade-off accuracy for speed depending on the application.

4.6 Summary

The Burt pyramid is a robust 2D error correcting code with a 33% redundancy. The error correcting properties of the pyramid is based on this redundancy and not on *a priori* assumptions about the image spectrum. This makes it possible to detect and correct errors even if they are frequency matched with the image. Sharp details in the images are preserved since the error correcting process does not remove part of the signal spectrum and blur the images. Moreover, the errors that are detected are removed by subtraction and not smeared by a low-pass filter. As was pointed out in the introduction, this is an imortant feature for many image processing applications.

The error correction algorithm is an iterative algorithm that performs well for bit error rates of about 10^{-3}. For noisier channels, the pyramid encoding can be used in conjunction with other conventional error correcting codes by using them to reduce the BER for transmission of the pyramid pixels to about 10^{-3} or the number of iterations can be increased thereby trading off speed for accuracy. The pyramid code can also be used for analog storage or transmission of 2D images as images, i.e., without having to unraster them into 1D data stream. The Burt pyramid can also be generalized to arrays of images by creating a 3D pyramid by a process of filtering and decimation along x, y, z. These 3D pyramids could then be used to transmit or store large arrays of images as are generated by MRI and PET scans. Since the redundancy of the 3D pyramid is only 14% this makes it a very compact code.

Chapter 5

Conclusions

The desirability of redundancy in image processing has often been underestimated. The traditional approach has been to look upon the redundancy of natural images as scope for data compression and this outlook has carried over to the design of image analysis systems like filter banks which were usually designed to make the output of each filter as independent of the others as possible. The format of input images was set by the needs of storing and transmitting with minimum cost, which meant critically sampled representations like the wavelet representations. Motion detection algorithms or other image analyses were expected to work best within this framework since it was the most compact. However, the key issue for image analysis is not the size of any representation but the manner in which information is encoded in it.

The estimation of optical flow field using derivatives provides a dense estimate of the flow field for a variety of inputs, like shaded objects, textures and diffuse blobs. The algorithm requires interpolation and estimation of derivatives but no feature extraction or recognition. The cost of interpolating images, estimating derivatives and optical flow fields was analyzed using the basis function formalism. This formalism makes explicit the relationships between frequency, computation cost and accuracy of results. Since the (digital) frequency of the sampled image depends on the ratio of the sampling rate to the highest frequency in the analog image, the result links the cost and accuracy of estimation to the redundancy in the representation. It was shown how oversampling and reducing the frequency content of the images greatly reduces computation costs and increases accuracy. The difference is significant enough to justify interpolating and resampling a critically sampled image by a large separable kernel before doing a flow field estimation. This indicates that the optimum overall strategy for handling huge image sequences would be to use wavelets to store and transmit the images. When motion estimation is required, the desired sections can be interpolated and upsampled before being passed on to the flow field estimation algorithm.

The use of a multiresolution framework like the gaussian pyramids can enhance the performance of an optical flow field estimation algorithm. The coarse to fine approach serves two purposes: it reduces the computation costs by doing much of the analysis on smaller versions of an image and it increases the range of displacements that can be estimated by a first-order model. The gaussian pyramids are to be preferred over the band pass laplacian pyramids since their low frequency dominated spectra are more conducive to estimating derivatives and they incorporate all the information of the higher levels of the pyramid. This eliminates the need to iterate through the levels of the pyramid. Any level of a gaussian pyramid can be constructed given the laplacian pyramid, which is an overcomplete representation. It was shown how this pyramid could be used as an error correcting code for images. Low frequency and high frequency noise added to the pyramid can be detected and corrected. The results for different images show how the noise can be removed without attenuating similar looking image features.

The issues of information extracted and cost are central to the design of filter banks whose purpose is to estimate the frequencies present in the input. A method for designing filter banks under such constraints, given the type of filters and the *prior* for the inputs, is presented. The decoding of filter outputs was treated as a distinct problem. This allows the design of the filters to be governed by the above constraints while the codebook creation and decoding can be tailored to the desired trade off between handling multiple inputs and achieving high accuracy. The implementation of such a filter bank showed how 60 filters could be used to discriminate 1296 different inputs. A probability measure was developed for the filter outputs so that information could be integrated across parameters and the higher levels of the system could be given a more informative description of the inputs than just the maximally probable input parameter value.

Appendix A

Decoding L_0

Decoding the first laplacian band requires special care since it is the widest band (width $= \pi/2$). Isolated noise pixels added to L_0 will thus have most of their energy within the frequency range natural to this band. Moreover, being the first band, it has only one neighboring band, the L_1. The net result is that isolated noise pixels in L_0 do not decay significantly after **CR**.

The reconstruction process adds the L_0 to $\mathbf{exp}G_1$ to create G_0. All the single pixel noise in L_0 thus gets translated into single pixel noise in G_0. Now consider a G_0 with an isolated non-zero pixel. During construction, this pixel will produce a characteristic pattern in L_0 with a peak value close to the pixel value in G_0. Since the loss for the center pixel is not much more than the gain by the neighboring pixels, the usual decoding algorithm fails to single out the center pixel as the sole cause for noise. However, if we consider the difference of the two L_0 then all the isolated errors show up as the characteristic center-surround pattern. The center of the pattern indicates the position of the noise pixel and the peak value can be used to calculate the value of the noise pixel. The decoding of single pixel noise in L_0 can be done using this single step process before using the more general technique for the whole pyramid.

The subsampling process in the construction phase creates four different sub-lattices and each of these have their own characteristic center-surround pattern. The detection of the patterns in $L_{0_{diff}} = L_{0_{orig}} - L_{0\mathbf{CR}}$ must be done with four different templates, with each template operating on one fourth of the pixels. The size of these kernels is dependent on the size of the low-pass filter used during $\mathbf{C}$: theoretically it should be twice that but practically a center extract of equal size does as good a job. This approach fails if there are two or more isolated noise pixels closer to each other than the size of the kernel being used and, aside from considerations of time, this provides an incentive for using smaller kernels. On the other hand, the kernel size cannot be made too small as it would not be able to reliably identify the characteristic center-surround patterns from other patterns that will be present due to the noise in other bands.

References

[Anandan, 1989] Anandan, P. (1989). A computational framework and an algorithm for the measurement of visual motion. *Intl. Journal of Computer Vision*, 2(3):283–310.

[Anderson, 1990] Anderson, C. (1990). Pyramids in machine vision at jpl. In *Proc First International Symposium on Measurement and Control in Robotics*, pages F1.3.1 – F1.3.4, Houston, Texas.

[Anderson and Rakshit, 1992] Anderson, C. and Rakshit, S. (1992). *Shape in Pictures - Mathematical Descriptions of Shape in Greylevel Images*, chapter Interpolation in Mutiscale Representations. Ed- Ying-Lie, O and Toet, A and Heijmans, H and Foster, D and Meer, P.

[Bartolini et al., 1993] Bartolini, F., Cappelini, V., Columbo, C., and Mecocci, A. (1993). Multiwindow least-squares approach to the estimation of optical flow with discontinuities. *Optical Engineering*, 32(6).

[Bergen et al., 1992] Bergen, J., Burt, P., Hingorani, R., and Pleg, S. (1992). A three-frame algorithm for estimating two-component image motion. *IEEE Trans on Pattern Analysis and Machine Intelligence*, 14(9):886–895.

[Burt, 1983] Burt, P. (1983). Fast algorithms for estimating local image properties. *Computer Graphics and Image Processing*, 21:386–382.

[Burt and Adelson, 1983] Burt, P. and Adelson, E. (1983). The laplacian pyramid as a compact image code. *IEEE Trans. on Comm.*, 31:532–540.

[Burt et al., 1992] Burt, P., Anandan, P., and Hanna, K. (1992). An electronic front end processor for active vision. *SPIE, Intelligent Robots in Computer Vision*, 1825(IX):769–780.

[Coogan et al., 1993] Coogan, T., Kumar, S., and VanEssen, D. (1993). Macaque cortex cross sections. The author would like to thank the above for providing the image sequences.

[Daubechies, 1990] Daubechies, I. (1990). The wavelet transform, time-frequency localization and signal analysis. *IEEE Trans on Information Theory*, 36:961–1005.

[Fleet and Jepson, 1989] Fleet, D. and Jepson, A. (1989). Hierarchial construction of orientation and velocity selective filters. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 11(3):315–324.

[Freeman and Adelson, 1991] Freeman, W. T. and Adelson, E. H. (1991). The design and use of steerable filters. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 13(9):891–906.

[Gallant et al., 1993] Gallant, J., Connor, C., Rakshit, S., and Van Essen, D. (1993). Selectivity for polar and hyperbolic gratings in macaque visual area v4: Multivariate classification and anatomical distribution. In *Soc. Neuroscience Abstracts, vol. 19, Part 1*, page 771, Washington D.C. Society for Neuroscience. abs no. 315.13.

[Greenspan et al., 1994] Greenspan, H., Bologne, S., Perona, P., Goodman, R., Rakshit, S., and Anderson, C. (1994). Overcomplete steerable pyramid filters and rotation invariance. Submitted for IEEE Conference on Computer Vision and Pattern Recognition, June 94.

[Haralick and Lee, 1983] Haralick, R. and Lee, J. (1983). The facet approach to optical flow. In Baumann, L., editor, *Proceedings Image Understanding Workshop*, pages 84–93.

[Heeger, 1987] Heeger, J. (1987). Model for the extraction of image flow. *J. Optical Society of America A.*, 4(8):1455–1471.

[Horn, 1986] Horn, B. (1986). *Robot Vision.* The MIT Press.

[Horn and Schunck, 1981] Horn, B. and Schunck, B. (1981). Determining optical flow. *Artificial Intelligence*, 17:185–204.

[Mallat, 1989] Mallat, S. (1989). A theory for multiresolution signal decomposition: The wavelet representation. *IEEE Pattern Analysis and Machine Intelligence*, 11:674–693.

[McEliece, 1977] McEliece, R. (1977). *The Theory of Information and Coding*, volume 3 of *Encyclopedia of Mathematics and its Applications.* Press Syndicate of Univ. of Cambridge.

[Nagel, 1987] Nagel, H. (1987). On the estimation of optical flow: Relations between different approaches and some new results. *Artificial Intelligence*, 33:299–324.

[Namazi and Lipp, 1992] Namazi, M. and Lipp, J. (1992). Nonuniform motion estimation using map principle. *IEEE Trans on Image Processing*, 1(4).

[Oagata and Sato, 1992] Oagata, M. and Sato, T. (1992). Motion-detection model with two stages: spatiotemporal filtering and feature matching. *J. Optical Society of America A.*, 9(3):377–387.

[Oppenhiem and Schafer, 1975] Oppenhiem, A. and Schafer, R. (1975). *Digital Signal Processing.* Prentice-Hall Inc.

[Perona, 1994] Perona, P. (1994). Deformable kernels for early vision. To appear in IEEE Trans on Pattern Analysis and Machine Intelligence.

[Simoncelli, 1993] Simoncelli, E. (1993). *Distributed Representation and Analysis of Visual Motion.* PhD thesis, MIT.

[Simoncelli and Adelson, 1991] Simoncelli, E. and Adelson, E. (1991). *Subband Image Coding*, chapter Subband Transforms. Kluwer Academic Publishers, Norwell, Massachusetts.

[Simoncelli et al., 1992] Simoncelli, E., Freeman, W., Adelson, E., and Heeger, D. (1992). Shiftable multiscale transforms. *IEEE Trans on Information Theory*, 38(2).

[Tretiak and Pastor, 1984] Tretiak, O. and Pastor, L. (1984). Velocity estimation from image sequence with second order differential operators. In *Proc. Intl Conf on Pattern Recognition, Montreal*, pages 20–22.

[Weber and Malik, 1992] Weber, J. and Malik, J. (1992). Robust computation of optical flow in a multiscale differential framework. UCB/CSD 92/709, EECS, UC Berkeley, jweber@eecs.berkeley.edu.