Grid Filters for Local Nonlinear Image Restoration

by

Todd Lawrence Veldhuizen

A thesis
presented to the University of Waterloo
in fulfilment of the
thesis requirement for the degree of
Master of Applied Science
in
Systems Design Engineering

Waterloo, Ontario, Canada, 1998

I hereby declare that I am the sole author of this thesis.

I authorize the University of Waterloo to lend this thesis to other institutions or individuals for the purpose of scholarly research.

I further authorize the University of Waterloo to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

The University of Waterloo requires the signatures of all persons using or photocopying this thesis. Please sign below, and give address and date.

Abstract

A new approach to local nonlinear image restoration is described, based on approximating functions using a regular grid of points in a many-dimensional space. Symmetry reductions and compression of the sparse grid make it feasible to work with twelve-dimensional grids as large as 22^{12}. Unlike polynomials and neural networks whose filtering complexity per pixel is linear in the number of filter coefficients, grid filters have O(1) complexity per pixel. Grid filters require only a single presentation of the training samples, are numerically stable, leave unusual image features unchanged, and are a superset of order statistic filters. Results are presented for additive noise, blurring, and superresolution.

Acknowledgements

I thank Dr. Ed Jernigan, my supervisor, for his encouraging words and for allowing me freedom to explore. I am grateful to the Vision and Image Processing Laboratory folks for providing a friendly and relaxing environment. My readers, Dr. Paul Fieguth and Dr. Glenn Heppler, provided many useful suggestions. Special thanks are due to Dr. Gregory V. Wilson for being a tireless mentor and friend.

My partner, Lindsay Patten, provided a context of love, support, and happiness which have seen me through this degree.

Finally, I acknowledge with gratitude the financial support I have received from the National Science and Engineering Research Council of Canada (NSERC), and the Department of Systems Design Engineering, University of Waterloo.

Contents

List of Tables

List of Figures

List of acronyms and symbols

$\|\cdot\|$	2-norm of a vector
(α, β)	(1) Generators for the permutation group associated with D_4 (2) Variables used in the Conjugate Gradient algorithm to determine descent directions.
A	Matrix used to find the MSE-optimal filter coefficients, which are given by **Af = b.**
AWGN	Additive White Gaussian Noise
b	Right-hand side of the least-squares equations used to find the MSE-optimal filter coefficients. See **A.**
dB	Decibels, used to express signal-to-noise ratio (SNR): $10log_{10}P_s/P_n$
D_4	Dihedral group of order 8, which represents the ways a rigid square in the plane can be transformed onto itself through reflections and rotations.
δ	A small shift in the local signal mean
DFT	Discrete Fourier Transform
$E[\cdot]$	Expectation operator. Takes a sample average over a distribution.
f	Vector of grid filter coefficients. Each coefficient gives the value of the function F at an associated grid point.

$F(x_0, \ldots, x_{n-1})$	A nonlinear function which uses pixel values from a local neighborhood $(x_0, \ldots, x_{n-1})$ to estimate the value of the pixel in the center of the neighborhood prior to degradation.
$\Gamma(n)$	The Gamma function: $\Gamma(n+1) = n!$ for integer n
$\mathcal{H}(\rho)$	A radially symmetric Optical Transfer Function (OTF)
iid	Independent, Identically Distributed random variables
$J(\mathbf{f})$	Minimum Mean-Squared Error (MMSE) criterion function
λ	(1) Wavelength of a spatial frequency component, in pixels *(2)* Regularization coefficient for solving least-squares equations
λ^{-1}	Spatial frequency in the range $[0,1/\sqrt{2}]$. Equivalent to p.
$\boldsymbol{L}$	Grid spacing (the distance between grid points)
MMSE	Minimum Mean-Squared Error
MSE	Mean-Squared Error
$\mathrm{orb}_{\mathcal{P}}(\mathbf{x})$	The orbit of x under $\mathcal{P}$, obtained by applying each member $p \in \mathcal{P}$ to x.
$O(f(n))$	f(n) is an asymptotic upper bound.
$\Theta(f(n))$	*f(n)* is an asymptotically tight bound.
OTF	Optical Transfer Function
OSF	Order Statistic Filter
P	An element of a permutation group
P^{-1}	Inverse of a permutation, such that $pp^{-1} = I$, where I is the identity permutation.
$\mathcal{P}$	A permutation group
pdf	Probability density function
PSF	Point Spread Function model of blurring. Same as an impulse response.
PSNR	Peak Signal-to-Noise Ratio (p. 15)

Q_N	An N-dimensional hypercube
R.V.	Random Variable
SNR	Signal-to-Noise Ratio. Generally measured in decibels (dB)
S_N	An N-dimensional hypersphere
s_0	An estimate of a pixel value before degradation
t	Normalized coordinates within an N-dimensional hypercube element: $t \in [0, 1]^N$.
trace(A)	Trace of a matrix is the sum of the diagonal elements
ρ	Radial spatial frequency in the range $[0, 1/\sqrt{2}]$. Equivalent to λ^{-1}.
$\mathbf{w}(\mathbf{x})$	Vector of basis functions (or equivalently, interpolation coefficients) for a point x.
$\omega(f(n))$	A lower bound which is not asymptotically tight.
$\mathbf{x}$	A vector of pixel values from a local neighborhood

Chapter 1

Introduction

1.1 Motivation and structure

Image restoration is the problem of recovering images which have been degraded by blurring and noise. Since imaging devices are never perfect, there are many applications for image restoration: astronomy, medical imaging, remote sensing, and microscopy are but a few.

Techniques for image restoration can be loosely grouped into two categories: local and global. Local filters restore an image one pixel at a time, using information from surrounding pixels. In global restoration techniques, each pixel contributes to the restoration of every other pixel. As a general rule (and there are exceptions), local filters are fast but do not yield very good results; global filters are slow but are capable of astonishingly good results.

In this thesis, a new approach to local image restoration is developed. This method is based on approximating functions of many variables on a multidimensional grid of points, hence the name Grid Filters. These filters generate excellent restoration results and are comparatively fast.

1.1.1 Structure of this thesis

Chapter 1 covers background material which is assumed in subsequent chapters. The origin of blurring and noise in imaging systems is briefly described, as are some common mathematical models. Some popular local image restoration techniques are explained in detail, and a few important global algorithms are mentioned. Common measures for comparing the quality of restored images are explained. The final sections argue that local filters are adequate for image restoration in many common scenarios.

Chapter 2 describes the Grid Filter approach. The theory of Local Minimum Mean-Squared Error (LMMSE) filter design is reviewed, and the major differences between previous approaches and grid filters are pointed out. Grid filters are then described in detail, with sections devoted to feature selection, the structure of the grid, interpolation techniques, symmetry assumptions and training.

Chapter 3 presents results for additive noise and blurring (superresolution). Two approaches for incorporating information from larger neighborhoods (foveated footprints and hybrid filters) are compared. Several properties of grid filters, such as passing outliers unchanged and filtering speed are illustrated. The amount of training data required for adequate filtering results is determined. The performance of grid filters on several noise models is evaluated.

Chapter 4 summarizes the important properties and limitations of grid filters, and points out some areas for future research.

1.2 The image restoration problem

1.2.1 Common sources of blurring and noise

Blurring is present in any imaging system which uses electromagnetic radiation (for example, visible light and X-rays). Diffraction limits the resolution of an imaging device to features on the order of the illuminating wavelength. Scattering of light between the target object and imaging system (for example, by the atmosphere) introduces additional blurring. Lenses and mirrors cause blurring because they have limited spatial extent and optical imperfections. Discretization results in yet more blurring because devices such as CCDs average illumination over regions rather than sampling it at discrete points.

Noise is similarly omnipresent: any imaging device must use a finite exposure (or integration) time, which introduces stochastic noise from the random arrival of photons. Optical imperfections and instrumentation noise (for example, thermal noise in CCD devices) result in more noise. Sampling causes noise due to aliasing of high-frequency signal components, and digitization produces quantization errors. Further noise can be introduced by communication errors and compression.

Blurring and noise processes can be accurately approximated by mathematical models. The next sections review some common models for blurring and noise.

1.2.2 The point-spread function (PSF) model of blurring

Most blurring processes can be approximated by convolution integrals, also known as Fredholm integral equations of the first kind [4]. The blurring is characterized by a Point-Spread Function (PSF) or impulse response. The PSF is the output of the imaging system for an input point source. All the blurring processes considered in this thesis are linear and have a spatially invariant PSF.

For discrete image processing, the convolution integral is replaced by a sum. The blurry image $x(n,m)$ is obtained from the original image $s(n,m)$ by this convolution:

$$x(n,m) = \sum_{a=-\infty}^{+\infty} \sum_{b=-\infty}^{+\infty} s(n+a, m+b)h(-a,-b) \tag{1.1}$$

The function $h(n,m)$ is the discrete Point Spread Function for the imaging system. Also of interest is the Discrete Fourier Transform (DFT) representation of the point-spread function, given by

$$H(u,v) = \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} h(n,m)e^{-2\pi j(\frac{un}{N}+\frac{vm}{M})} \tag{1.2}$$

for $u = \lfloor -N/2 \rfloor + 1, \ldots, \lfloor N/2 \rfloor$ and $v = \lfloor -M/2 \rfloor + 1, \ldots, \lfloor M/2 \rfloor$. $H(u,v)$ gives a set of coefficients for plane waves of various frequencies and orientations. These plane waves, called *spatial frequency components,* reconstruct the PSF exactly when multiplied by the coefficients $H(u,v)$ and summed. The function $H(u,v)$ is referred to as the *transfer function,* or *system frequency response.* By examining $|H(u,v)|$, one can quickly determine which spatial frequency components are passed or attenuated by the imaging system.

As an example, consider this 3x3 mask which can be used to model small amounts of blurring:

$$\frac{1}{15}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 3 & 2 \\ 1 & 2 & 1 \end{bmatrix} \tag{1.3}$$

The DFT of this mask is:

$$H(u,v) = \frac{1}{15}\left[3 + 4\cos\left(\frac{2\pi u}{N}\right) + 4\cos\left(\frac{2\pi v}{M}\right) + 4\cos\left(\frac{2\pi u}{N}\right)\cos\left(\frac{2\pi v}{M}\right)\right] \tag{1.4}$$

Figure 1.1 shows a plot of $|H(u,v)|$. Near $(u,v) \approx (0,0)$, the transfer function has

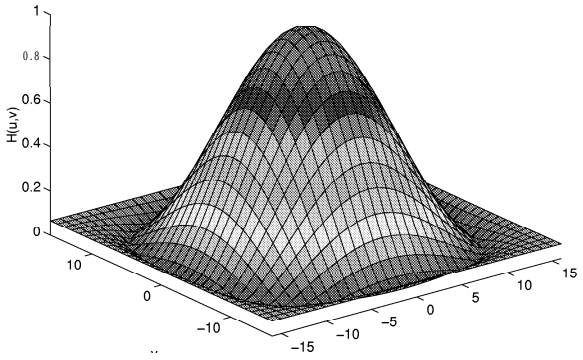


Figure 1.1: $|H(\mathrm{u}, v)|$ for a 3x3 blurring mask, with N = M = 33

$|H(u,v)| \approx$ 1. This indicates that low-frequency components are passed. Near the perimeter of the plot, $|H(u,v)| \approx 0$, meaning that high frequency components are blocked.

1.2.3 Noise models

Noise in imaging systems is usually either additive or multiplicative. This thesis deals only with additive noise which is zero-mean and *white.* White noise is spatially uncorrelated: the noise for each pixel is independent and identically distributed (iid). Common noise models are:

- **Gaussian noise** provides a good model of noise in many imaging systems [5]. Its probability density function (pdf) is:

$$p_n(n) = \frac{1}{\sqrt{\pi\sigma^2}} e^{-\frac{n^2}{\sigma^2}} \tag{1.5}$$

  The Gaussian distribution has an important property: to estimate the mean of a stationary Gaussian random variable, one can't do any better than the linear average. This makes Gaussian noise a worst-case scenario for nonlinear image restoration filters, in the sense that the improvement over linear filters is least for Gaussian noise. To improve on linear filtering results, nonlinear filters can exploit only the non-Gaussianity of the signal distribution.

- **Laplacian noise** (also called biexponential) which has this pdf:

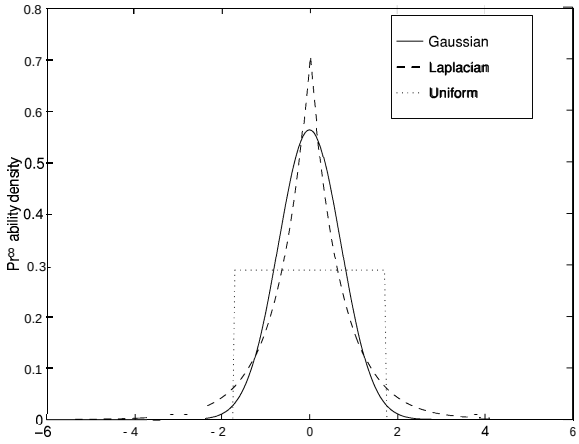


Figure 1.2: Probability density functions of the Gaussian, Laplacian and Uniform distributions

$$p_n(n) = \frac{1}{\sqrt{2}\sigma} e^{-\frac{\sqrt{2}|n|}{\sigma}} \tag{1.6}$$

Nonlinear estimators can provide a much more accurate estimate of the mean of a stationary Laplacian random variable than the linear average [6].

- **Uniform noise** is not often encountered in real-world imaging systems, but provides a useful comparison with Gaussian noise. The linear average is a comparatively poor estimator for the mean of a uniform distribution. This implies that nonlinear filters should be better at removing uniform noise than Gaussian noise. The Uniform pdf is given by:

$$p_n(n) \quad \begin{cases} \frac{1}{\sigma 2\sqrt{3}} & \text{for} |n| \leq \sigma\sqrt{3} \\ 0 & \text{else} \end{cases} \tag{1.7}$$

Figure 1.2 illustrates these PDFs for zero-mean, unit variance noise.

1.3 Literature review

Image restoration is difficult since it is an ill-posed inverse *problem:* there is not enough information in the degraded image to determine the original image unambiguously. The problem has received steady attention since the 1960s, and techniques for its solution continue to be proposed. This section explains the popular

local image restoration techniques in detail. A few important global techniques are described briefly.

1.3.1 Order statistic filters

Given N observations $X_1, X_2, \ldots, X_N$ of a random variable X, the order statistics are obtained by sorting the $\{X_i\}$ in ascending order. This produces $\{X_{(i)}\}$ satisfying:

$$X_{(1)} \leq X_{(2)} \leq \ldots \leq X_{(N)} \tag{1.8}$$

The $\{X_{(j)}\}$ are the order *statistics* of the N observations [8]. An *Order Statistic Filter* (OSF) is a estimator $F(X_1, X_2, \ldots, X_N)$ of the mean of X which uses a linear combination of order statistics:

$$F(X_1, X_2, \ldots, X_N) = \alpha_1 X_{(1)} + \alpha_2 X_{(2)} + \ldots + \alpha_N X_{(N)} \tag{1.9}$$

Order Statistic Filters have long been known to statisticians as L-estimators, but were re-christened and applied to image processing problems by Bovik et. al. [6]. Some common filters which fit the order statistic filter framework are:

- The **linear average,** which has coefficients

$$\alpha_i = 1/N \tag{1.10}$$

- The **median filter,** which has coefficients

$$\text{a;} = \begin{cases} 1 & i = (\text{N} + 1)/2 \\ 0 & \text{otherwise} \end{cases} \tag{1.11}$$

  For image processing applications, N is almost always odd, so the question of how to handle even values of N is avoided.

- The **trimmed mean** filter. which has coefficients

$$\text{a;} = \begin{cases} 1/M & (\text{N} - M + 1)/2 \leq i \leq (\text{N} + M + 1)/2 \\ 0 & \text{otherwise} \end{cases} \tag{1.12}$$

For any distribution, one can determine the optimal coefficients {a;} by minimizing the criterion function

Coefficient	Gaussian	Laplacian	Uniform
α_1	0.11111	-0.01899	0.50000
α_2	0.11111	0.02904	0.00000
α_3	0.11111	0.06965	0.00000
α_4	0.11111	0.23795	0.00000
α_5	0.11111	0.36469	0.00000
α_6	0.11111	0.23795	0.00000
α_7	0.11111	0.06965	0.00000
α_8	0.11111	0.02904	0.00000
α_9	0.11111	-0.01899	0.50000

Table 1.1: OSF filter coefficients for N = 9

$$J(a) = E\left[(\alpha^T \mathbf{X} - \mu)^2\right] \tag{1.13}$$

where a is the vector of order statistic filter coefficients, X is the vector of order statistics, and μ is the mean of the random variable X. It turns out that the linear average is optimal for the Gaussian distribution. Table 1.1 gives OSF coefficients for Gaussian, Laplacian and Uniform noise, for the case N = 9. Bovik [6] lists optimal coefficients for several other distributions.

An aspect of order statistic filters which turns out to be important for Grid Filters is that they are *piecewise linear*. The filter partitions $\Re^N$ into N! regions of the form

$$x_{j_1} \leq x_{j_2} \leq \ldots \leq x_{j_N} \tag{1.14}$$

where $(j_1, j_2, \ldots, j_N)$ is a permutation of $(1, 2, \ldots,$ N). Over each of these regions, the filter output is a linear function (1.9).

To apply an order statistic filter to an image, one typically uses 3x3, 5x5 or 7x7 windows. For non-Gaussian noise, the optimal OSF is superior to taking a local average for flat regions. The main problem with such filters is the underlying stationarity assumption: the derivation of the OSF assumes that X is a stationary point process, an assumption which is grossly violated if there is an edge, line, or other strong signal activity in the window. Figure 1.3 illustrates this for a test image degraded by additive white Gaussian noise with σ^2 = 400. The median filter (lower left) preserves edges (the checkerboard and the bagel), but wipes out fine details (the text and lines). The 3x3 average, which is the optimal OSF for Gaussian noise, blurs too much.

These limitations motivated the development of the Adaptive Trimmed Mean

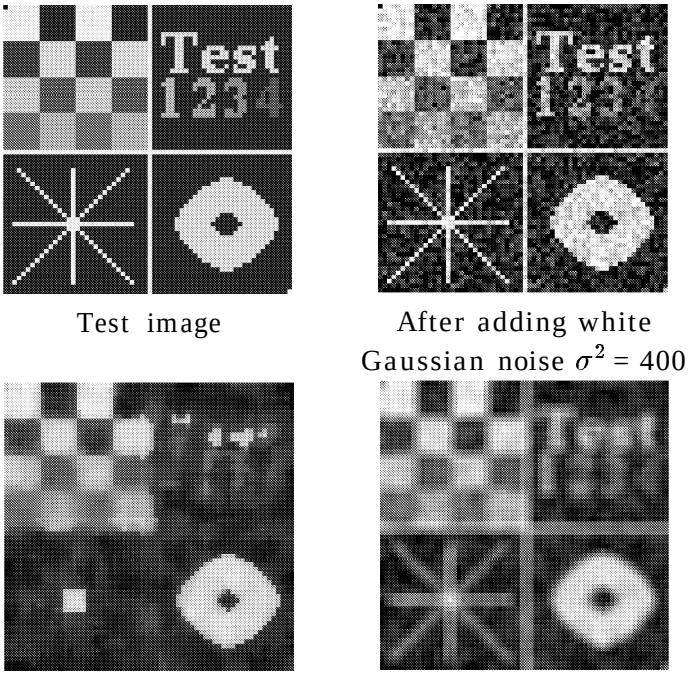


Figure 1.3: Effect of local averages and median filters on fine details

Filter [9], which makes the assumption that the signal is smoothly varying within the local window. When the signal varies slowly within the window, the filter behaves like a trimmed mean. When an abrupt transition is detected, the filter behaves like a median, which preserves edges. Unfortunately, the median filter also destroys fine details, as illustrated in Figure **1.3.** Another extension of OSFs to the nonstationary case, called a Permutation Filter or or p-filter [10], has apparently not yet been tested on image restoration problems.

1.3.2 Lee's local statistics filter

The Lee filter [11] is able to smooth away noise in flat regions, but leave fine details (such as lines and text) unchanged. It uses small windows (3x3,5x5 or 7x7). Within each window. the local mean and variance are estimated:

$$\begin{aligned} \overline{x} &= \frac{\mathbf{1}}{\mathbf{N}} \sum_{i=1}^{N} x_i \\ \sigma_x^2 &= \frac{1}{N-1} \sum_{i=1}^{\mathrm{N}} (x_i - \overline{x})^2 \end{aligned} \qquad \textbf{(1.15)}$$

In regions of no signal activity, the filter outputs the local mean $(\overline{x})$. When signal activity is detected, the filter passes the original signal through unchanged. This is achieved by filters of the form

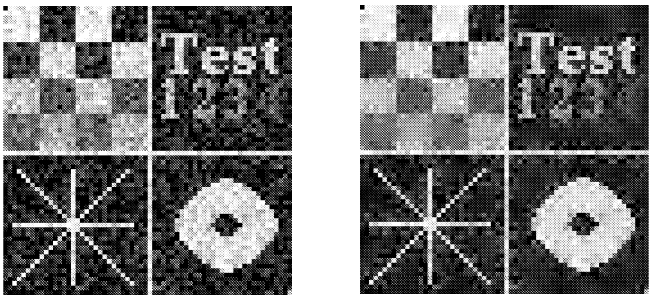
$$F(x_1, \ldots, x_N) = \beta x_1 + (1 - \beta)\overline{x} \qquad \textbf{(1.16)}$$

where x_1 is the central pixel in the window. The parameter β ranges between 0 (for flat regions) and **1** (for regions with high signal activity). For the additive noise case, this formula for β is used:

$$\beta = \max\left(\frac{\sigma_x^2 - \sigma_n^2}{\sigma_x^2}, 0\right) \qquad \textbf{(1.17)}$$

where σ_n^2 is an estimate of the noise variance.

The Lee filter senses when it is being applied to a region which is constant in intensity, and responds by smoothing. In regions which contain signal activity (for example, lines and edges), the Lee filter shuts down its smoothing. The Lee filter can thus smooth in flat regions but still preserve sharp details. Its major drawback is that it leaves noise in the vicinity of edges and lines (Figure 1.4). Variants of this filter handle multiplicative noise and sharpening [11].



Test image with AWGN σ^2 = 400

After Lee filter with 5x5 window. MSE=202

Figure 1.4: Lee filter example

1.3.3 The Wiener filter

The Wiener filter is the MSE-optimal stationary **linear** filter for images degraded by additive noise and blurring. Calculation of the Wiener filter requires the assumption that the signal and noise processes are second-order stationary (in the random process sense).[1] For this description, only noise processes with zero mean will be considered (this is without loss of generality).

Wiener filters are usually applied in the frequency domain. Given a degraded image $x(n,m)$, one takes the Discrete Fourier Transform (DFT) to obtain $X(u,v)$. The original image spectrum is estimated by taking the product of $X(u,v)$ with the Wiener filter $G(u,v)$:

$$\hat{S}(u,v) = G(u,v)X(u,v) \tag{1.18}$$

The inverse DFT is then used to obtain the image estimate from its spectrum. The Wiener filter is defined in terms of these spectra:

$H(u,v)$	Fourier transform of the point-spread function (PSF)
$P_s(u,\ v)$	Power spectrum of the signal process, obtained by taking the Fourier transform of the signal autocorrelation
$P_n(u,\ v)$	Power spectrum of the noise process, obtained by taking the Fourier transform of the noise autocorrelation

The Wiener filter is:

[1] A random process is *second-order stationary* if the expected value of any quadratic function of the process random variables is invariant under shifting. This guarantees that the autocorrelation is well-defined.

$$G(u,v) = \frac{H^*(u,v)P_s(u,v)}{|H(u,v)|^2 P_s(u,v) + P_n(u,v)} \tag{1.19}$$

Dividing through by P_s makes its behaviour easier to explain:

$$G(u,v) = \frac{H^*(u,v)}{|H(u,v)|^2 + \frac{P_n(u,v)}{P_s(u,v)}} \tag{1.20}$$

The term P_n/P_s can be interpreted as the reciprocal of the signal-to-noise ratio. Where the signal is very strong relative to the noise, $P_n/P_s \approx 0$ and the Wiener filter becomes $H^{-1}(u,v)$ – the inverse filter for the PSF. Where the signal is very weak, $P_n/P_s \rightarrow \infty$ and $G(u,v) \rightarrow 0$.

For the case of additive white noise and no blurring, the Wiener filter simplifies to:

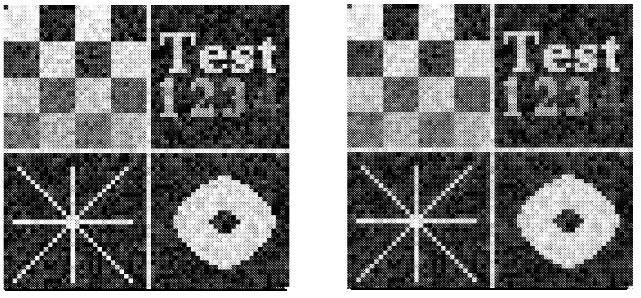
$$G(u,v) = \frac{P_s(u,v)}{P_s(u,v) + \sigma_n^2} \tag{1.21}$$

where σ_n^2 is the noise variance.

Wiener filters are unable to *reconstruct* frequency components which have been degraded by noise. They can only suppress them. Also, Wiener filters are unable to restore components for which $H(u,v) = 0$. This means they are unable to undo blurring caused by bandlimiting of $H(u,v)$. Such bandlimiting occurs in any real-world imaging system.

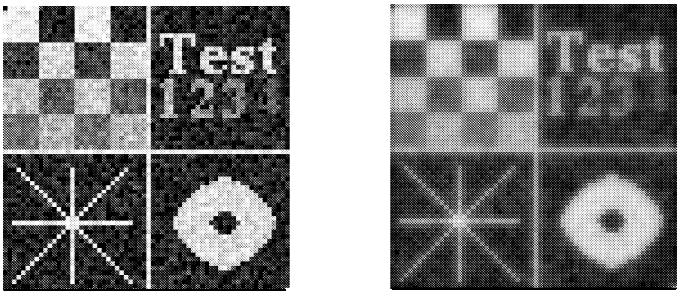
Obtaining P_s can be problematic. One can assume that P_s has a parametric shape, for example exponential or Gaussian. Alternately, P_s can be estimated using images representative of the class of images being filtered. For Wiener results presented in this thesis, P_s was calculated from image to be filtered: P_s was assumed to be radially symmetric, i.e. $P_s(u,v) = P_s(\rho)$ and was estimated by averaging over 30 radial frequency bands. Linear interpolation was used to give P_s a smooth shape.

Figure 1.5 shows a Wiener filter result. The small test image has very strong high-frequency components, so the Wiener filter leaves lots of residual noise. If the test image, which is 64x64, is centered in a 256x256 empty image, the relative power of those high-frequency components is diminished by the large amounts of empty space. The Wiener filter then elects to attenuate high-frequency components to reduce noise in the empty regions. This results in blurring over the small 64x64 subimage (Figure 1.6). Although the MSE over the 256x256 image is quite small, the MSE over the 64x64 test region increases from 400 to 1232. This illustrates an important point about using MSE as a criteria for global filtering: regions are given priority for restoration according to how large they are, rather than their



Test image with AWGN $\sigma_n^2 = 400$ | After Wiener filtering MSE=340

Figure 1.5: Example of Wiener filtering



Test image with AWGN $\sigma_n^2 = 400$ but centered inside a 256x256 empty image | After Wiener filtering MSE=121 (256x256 image) MSE=1232 (portion shown)

Figure 1.6: Another example of Wiener filtering

visual importance.

Wiener filters are comparatively slow to apply, since they require working in the frequency domain. To speed up filtering, one can take the inverse FFT of the Wiener filter $G(u,v)$ to obtain an impulse response $g(n,m)$. This impulse response can be truncated spatially to produce a convolution mask. The spatially truncated Wiener filter is inferior to the frequency domain version, but may be much faster.

1.3.4 Global iterative approaches

Local filters use information from a local neighborhood to restore pixels one at a time. In contrast, global filters use information from the entire image to restore each pixel. To achieve this, there must be a mechanism for information to travel

between every pair of pixels. Wiener filters achieve this by using the frequency domain representation of the image, in which each Fourier coefficient is affected by the value of almost every pixel. Most global filtering approaches achieve it through iteration: at each step, information propagates locally. Many iterations allow information to propagate globally.

A shortcoming of global iterative approaches is that they tend to be quite slow. Some of the algorithms which generate impressive results require hours to filter a single image. For this reason, parallel implementations of these algorithms have been explored.

Most of the algorithms are quite complicated, which makes comparison with them difficult. Implementing some of the newer approaches would be a worthy thesis on its own. For this reason, the most successful global iterative approaches are briefly described here and not mentioned again.

Adaptive filters extend the notion of linear filters by allowing for coefficients which change according to local image properties. The most popular of these are adaptive recursive filters [12, 13, 14, 15, 16] which are based on difference equations with adaptive coefficients. Such filters are able to smooth over very large regions, but can adapt quickly to local signal characteristics. These approaches require many iterations to converge to a solution of the difference equations. Some attempts have been made to adapt multigrid techniques [17, 18] for image restoration. Multigrid methods hold the promise of global algorithms which have complexity $O(N^2)$ for an NxN image (i.e. linear in the number of pixels). Non-multigrid adaptive recursive approaches generally have complexity with a lower bound of $\omega(N^3)$.[2]

Another family of image restoration techniques are based on Markov Random Fields and "annealing" techniques [19, 20, 21, 22]. Annealing techniques are inspired by physical systems which settle into low-energy states as they cool. Loosely, these techniques make small random changes to an image based on a gradually decreasing "temperature" parameter. At initial high temperatures, the changes are very large. As the temperature is lowered, the changes become smaller. These changes are directed toward maximizing an objective function. The objective function is based on the posterior probability and an assumed Markov Random Field (MRF) model for the image. The image tends to settle into a "low energy state" which corresponds to a mode of the MRF model. Quite astonishingly good results have been achieved using these approaches, but they are extremely slow. Parallel versions of these methods have been implemented in an attempt to reduce filtering time [23, 24].

[2] This is because each iteration requires $O(N^2)$ operations, and at least O(N) iterations are required for information to traverse the length of the image.

Regularization methods [4, 25, 26] regard image restoration as an ill-posed inverse problem. Such problems can not be solved by direct inversion, because the solution is highly unstable. To stabilize the inversion, a *stabilizing functional* is introduced. A typical restoration problem involves blurring and additive noise. This process can be written in matrix form as:

$$\mathbf{x} = \mathbf{H}\,\mathbf{s} + \mathbf{n} \tag{1.22}$$

where s is the original image, $\mathbf{H}$ is a matrix representation of the Point Spread Function (PSF), $\mathbf{n}$ is the noise, and x is the observed signal. The matrix $\mathbf{H}$ is typically ill-conditioned or singular, so that evaluating $\mathbf{H}^{-1}\mathbf{x}$ is problematic. In a regularization approach, one introduces additional constraints based on assumptions about the signal model. Some common approaches are:

- **First-order (Tikhonov) regularization finds s to minimize**

$$\|\mathbf{H}\mathbf{s} - \mathbf{x}\|^2 + \lambda\|\mathbf{s}\|^2 \tag{1.23}$$

  This approach selects the solution with minimum energy. Replacing the term $\|\mathbf{s}\|^2$ with more general terms of the form $\|\mathbf{L}\mathbf{s}\|^2$ can introduce restrictions on smoothness and other quantities of interest.

- **Maximum entropy methods** [27, 28, 29] which maximize functionals of the form $-\mathbf{s}^T ln(\mathbf{s})$ subject to

$$\|\mathbf{H}\mathbf{s} - \mathbf{x}\|^2 = \|\mathbf{n}\|^2 \tag{1.24}$$

  Maximum entropy methods have been particularly popular for medical image reconstruction.

1.4 The importance of *priors*

A useful analogy may be drawn between lossless compression and image restoration. In lossless compression, compression of some signals is achieved only at the expense of making other signals longer. One has to design the algorithm so that commonly occurring signals are compressed, and uncommon signals are lengthened. Similarly in image restoration, improvement in some images is obtained by worsening others. To ensure that typical images are improved, the image restoration scheme must incorporate prior knowledge of the statistical properties of the target class of images.

These *priors* are crucial to achieving a good restoration result. The success of a filter depends primarily on the accuracy of its priors. Thus order statistic filters, which make wildly unrealistic assumptions about the signal (namely, that it consists entirely of flat regions) have comparatively poor performance. Wiener filters, which implicitly assume Gaussian priors, generally do a bit better. Lee's filter assumes a mixture of flat and detail regions, which is more realistic. But by far the best results are achieved by filters which *learn* the priors, rather than assuming them. Such filters are able to exploit very detailed, accurate knowledge of the signal statistical properties. Examples include Gauss-Markov Random Fields, Vector Quantization, Neural Networks and the Grid Filters developed in this thesis.

1.5 Measures of image quality

Comparing restoration results requires a measure of image quality. Two commonly used measures are *Mean-Squared Error* and *Peak Signal-to-Noise Ratio* [30]. The mean-squared error (MSE) between two images $g(x,y)$ and $\hat{g}(x,y)$ is:

$$e_{MSE} = \frac{1}{MN} \sum_{n=1}^{M} \sum_{m=1}^{N} [\hat{g}(n,m) - g(n,m)]^2 \tag{1.25}$$

One problem with mean-squared error is that it depends strongly on the image intensity scaling. A mean-squared error of 100.0 for an 8-bit image (with pixel values in the range *O-255)* looks dreadful; but a MSE of 100.0 for a 10-bit image (pixel values in [0,1023]) is barely noticeable.

Peak Signal-to-Noise Ratio (PSNR) avoids this problem by scaling the MSE according to the image range:

$$\text{PSNR} = -10 log_{10} \frac{e_{MSE}}{S^2} \tag{1.26}$$

where S is the maximum pixel value. PSNR is measured in decibels (dB). The PSNR measure is also not ideal, but is in common use. Its main failing is that the signal strength is estimated as S^2, rather than the actual signal strength for the image. PSNR is a good measure for comparing restoration results for the same image, but between-image comparisons of PSNR are meaningless. One image with 20 dB PSNR may look much better than another image with 30 dB PSNR.

MSE and PSNR figures provided in this thesis were calculated after quantization (i.e. after converting floating-point pixel values to integer), but before clipping of the intensity range.

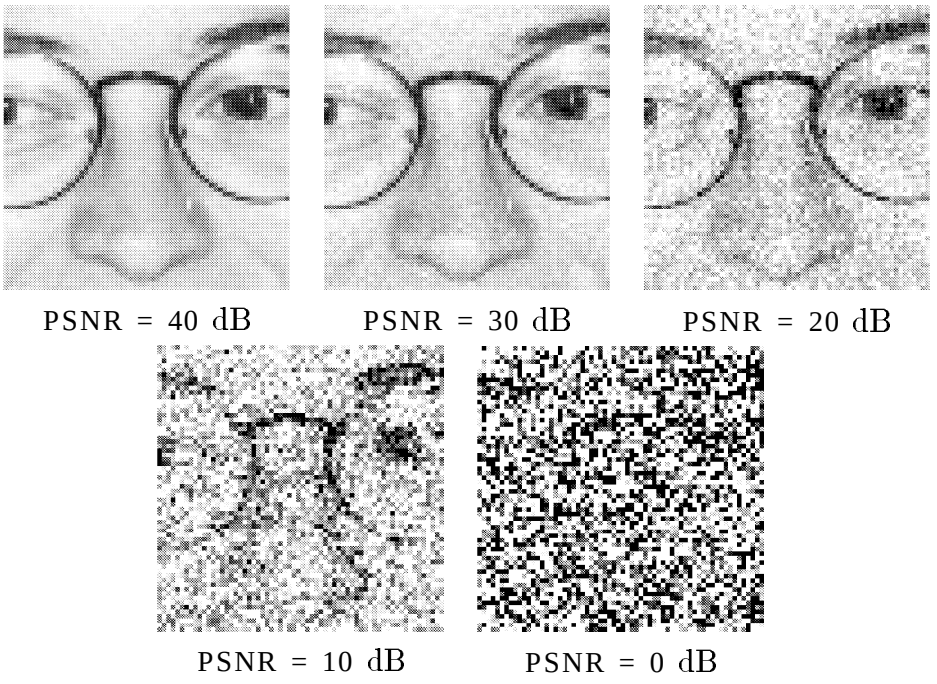


Figure 1.7: Illustration of the PSNR measure

Frequency-domain SNR behaviour

PSNR reduces image quality to a single number. If the number is low, it offers no information about what parts of the signal have been lost. To analyze the restoration more carefully, it is useful to work in the frequency domain. Looking at the signal-to-noise ratio as a function of spatial frequency gives a breakdown of filter performance for features of various scales. One can immediately see whether a filter has trouble with fine or large-scale features.

The Discrete Fourier Transform (DFT) representation of an image $g(n,m)$ is given by:

$$G(u,v) = \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} g(n,m) e^{-2\pi j(\frac{un}{N} + \frac{vm}{M})} \tag{1.27}$$

For the purpose of looking at frequency-domain SNR behaviour, it is useful to lump together the coefficients according to *normalized spatial frequency,* given by:

$$\lambda^{-1} = \sqrt{\frac{N^2v^2 + M^2u^2}{M^2N^2}} \tag{1.28}$$

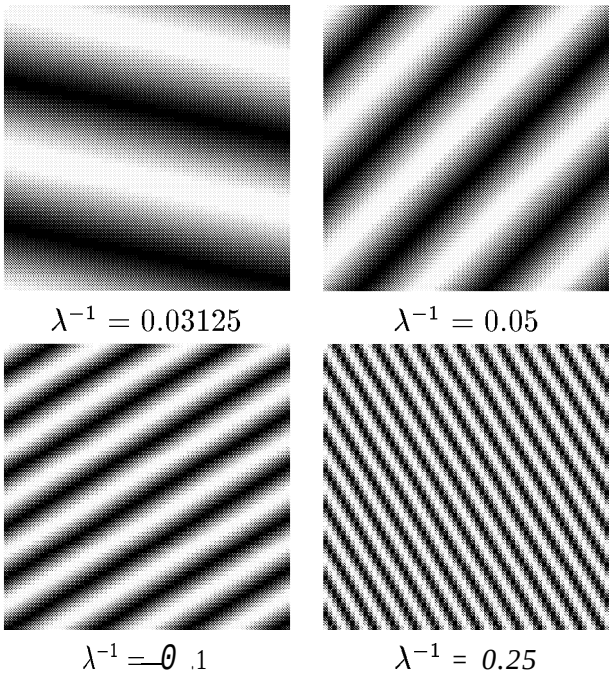


Figure 1.8: Spatial frequency components of various wavelengths and orientations

The range of λ^{-1} is [0, $\frac{1}{\sqrt{2}}$]. The inverse of the normalized frequency (λ) gives the wavelength of the spatial frequency component. The DC component (or average) of an image corresponds to $\lambda^{-1} = 0$. Features on the scale of ten pixels would have $\lambda^{-1} \approx 0.1$ $(\lambda \approx 10)$, and very fine detailed features (on the scale of 2 pixels) have $\lambda^{-1} \approx 0.5$. The highest possible frequency is $\lambda^{-1} = \frac{1}{\sqrt{2}}$ – this corresponds to a checkerboard pattern. Figure 1.8 illustrates some spatial frequency components of various wavelengths and orient ations.

Let $s(n,m)$ be the original image and $x(n,m)$ be a degraded version. To calculate SNR as a function of spatial frequency, the first step is to calculate the difference image:

$$d(n,m) = s(n,m) - x(n,m) \tag{1.29}$$

Then Fourier transforms of $s(n,m)$ and $d(n,m)$ are taken, producing $S(u,v)$ and $D(u,v)$. The frequency domain is then divided into nonoverlapping frequency bands of the form $B_i = [\lambda_i^{-1}, \lambda_{i+1}^{-1})$. Figure 1.9 illustrates this for a sample image. The average band power $P_s(B_i)$ is calculated by averaging $|S(u,v)|^2$ over all (u, v) in the band. The average power for the difference image, $P_d(B_i)$ is calculated similarly.

The signal-to-noise ratio for each band can then be calculated as:

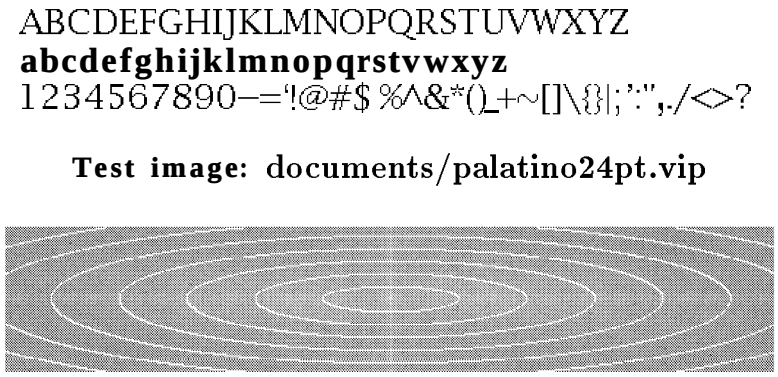


Figure 1.9: Illustration of how the frequency domain is divided into nonoverlapping bands

$$\mathrm{SNR}(B_i) = \frac{P_s(B_i)}{P_d(B_i)} \tag{1.30}$$

This SNR ratio can be plotted as a function of normalized spatial frequency. Figure 1.10 shows such a plot for the image of Figure 1.9 degraded by additive white Gaussian noise with σ^2 = 400. This plot illustrates that there is little difference between the Wiener and Lee filter result for low frequencies ($\lambda^{-1} \approx 0.1$). However, for higher frequencies ($0.3 \leq \lambda^{-1} \leq 0.7$) the Lee filter has much better performance.

These SNR plots can be somewhat deceptive, since they do not convey how *much* noise is left in each band. From Figure 1.10, one might conclude that the very high frequency bands ($0.5 \leq \lambda^{-1} \leq 0.7$) need improvement the most, since they have the lowest SNR. A better approach is to look at how the total MSE is distributed over frequency bands (Figure 1.11). The top plot shows how each band contributes to the total MSE for the raw image. Since additive white noise has a flat power spectrum, the contribution of each band is proportional to its area in the frequency domain. From the bottom plot, it is clear that comparatively little noise remains in the very high and very low frequency bands. The bands $0.1 \leq \lambda^{-1} \leq 0.5$ need the most improvement.

The frequency-domain SNR still does not give a complete picture of the image restoration result. For example, it does not distinguish between visually important and unimportant features. For this reason, it is important to look at the images themselves for a subjective estimate of image quality.

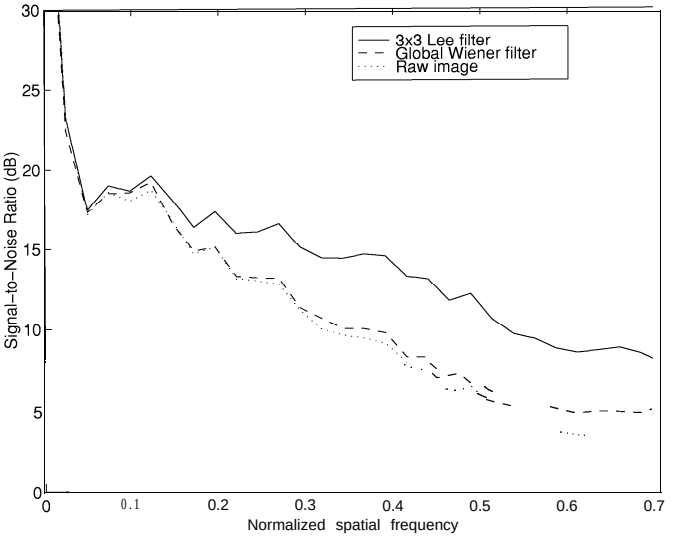


Figure 1.10: Example signal-to-noise ratio plot

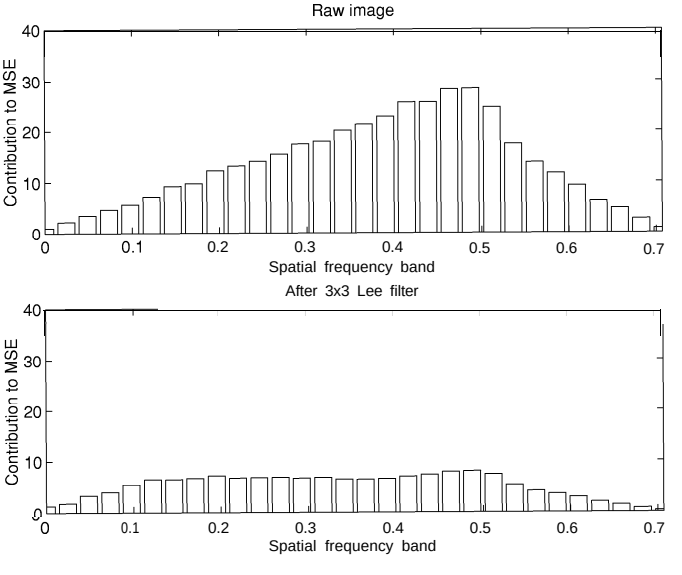


Figure 1.11: Example plot showing how MSE is distributed over frequency bands. The top figure illustrates that noise power is proportional to band area.

1.6 Are local filters good enough?

Most image degradations have global effects. For example, blurring caused by optical limitations of an imaging system has a point-spread function which is (theoretically) infinite in extent: a single point of light gets smeared over the entire image domain. The power spectrum of additive white noise covers the entire frequency domain evenly: there are low, medium and high-frequency components to the noise.

Given these observations. it is reasonable to wonder if local filters are sufficient to restore an image. For the case of additive noise, a strong argument can be made that local filters are sufficient.

1.6.1 Additive noise is primarily a local process

Consider an image degraded by zero-mean, Additive White Gaussian Noise (AWGN) with variance σ_n^2. The power spectrum of the noise is:

$$P_n(u, v) = \sigma_n^2 \tag{1.31}$$

The noise has the same power (σ_n^2) everywhere in the frequency domain. One might conclude from this that all spatial frequencies are affected equally by the noise, but this is misleading.

Figure 1.12 illustrates the noise power spectrum. Low frequencies lie (roughly) inside the circle $\lambda = 10$ (λ is the wavelength of the frequency component, in pixels). The amount of noise power inside this circle can be approximated by comparing its area to the area of the transform. The circle area is $\pi(1/10)^2 \approx 0.0314$. The domain of the transform is $[-0.5, +0.5]^2$, which has area 1. *So* the fraction of noise which lies at wavelengths above 10 pixels is only 0.0314, or about 3%. Similarly, only 12.6% of the noise lies at wavelengths above 5 pixels. The signal strength is typically very strong for wavelengths below 10 pixels, so the effect of the low-frequency noise on the signal-to-noise ratio is minor.

From these arguments, one can conclude that *additive white noise is primarily a local phenomenon.* This implies that the use of a local filter to remove additive noise is reasonable.

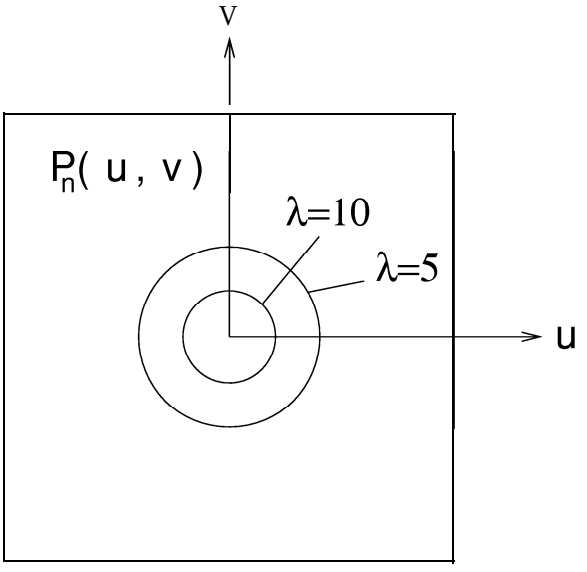


Figure 1.12: Frequency-domain view of additive noise

Chapter 2

Theory and Implementation

2.1 Local MMSE Nonlinear Filters

Local filters use pixels from a local neighborhood to restore an image one pixel at a time. For simplicity, this chapter will work primarily with 3x3 neighborhoods of pixels, labelled according to this scheme:

x_8	x_4	x_5
x_3	x_0	x_1
x_7	x_2	x_6

A local filter restores the central pixel x_0 using the values of the pixels $x_0, x_1, \ldots, x_8$:

$$\hat{s_0} = F(x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8) \tag{2.1}$$

where $\hat{s_0}$ is an estimate of the original central pixel (before degradation). Linear functions F give poor results; to restore images well, a nonlinear function is needed. This nonlinear function might be defined in terms of polynomials, neural networks, or other function approximate schemes. To restore an entire image, the operator is scanned across the image and each pixel is restored individually. To justify this approach, one must assume that the processes generating the image, blurring and noise are all *stationary* – in the random process sense that the probability of a particular behaviour does not depend on the image coordinates. This kind of stationarity is called *strict sense stationarity*. It is easy to find classes of images which violate this assumption – pictures of human faces, for example, are nonstationary in this sense since one is more likely to see eyes in the top of the image than the bottom. However, even images which violate this assumption will only do so mildly on the scale of interest, namely small local windows.

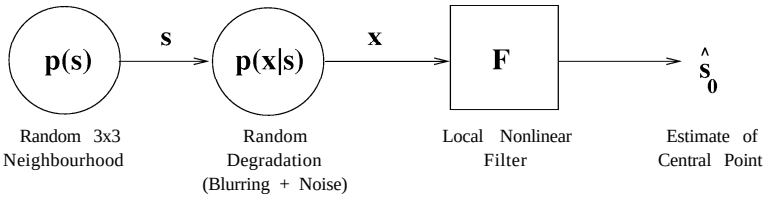


Figure 2.1: Model of the image degradation and restoration scheme.

Figure 2.1 shows the assumed model of the image degradation and restoration scheme. Uncorrupted 3x3 windows are assumed to have some distribution p(s). A particular 3x3 window s is drawn from this distribution. This 3x3 window is transformed into a noisy and blurred version x according to some random degradation $p(\mathbf{x}|\mathbf{s})$.[1] The local nonlinear filter takes the corrupted version x and attempts to estimate the original central point of the window $\hat{s_0}$.

Is there an optimal function F? If one uses mean-squared error (MSE) as a criterion, then there is an optimal F, and it is simply:

$$F(x_0, x_1, \ldots, x_8) = E[s_0|\mathbf{x}] \tag{2.2}$$

Unfortunately, the MSE-optimal F (2.2) requires distribution functions for the signal and degradation processes, involves many-dimensional convolutions, and (for any realistic situation) has no readily expressible form.

The next best thing to finding the MSE-optimal F explicitly is to *approximate* it: choose a representation for F *(e.g.* polynomials), and find the best coefficients. This approximation will be called $\hat{F}$. Explicitly minimizing the error between the optimal F and approximation $\hat{F}$ is not practical since the optimal F is so unwieldy. However, minimizing the expected mean-squared error of the approximate filter also minimizes the distance[2] between the approximate filter and the optimal filter.

2.1.1 What sort of approximation should one use for $\hat{F}$?

Polynomials and neural networks have both been popular choices for defining the function $\hat{F}$. These conventional methods (and their shortcomings) provide the

---

[1] Blurring of course relies on pixels which lie outside a local window. However, this does not pose a problem: the dependence on other pixels can be integrated out. If the blurring depends on a set of pixels $\mathbf{s}^*$ which is a superset of s, it is simple to express $p(\mathbf{x}|\mathbf{s})$ in terms of $p(\mathbf{s}^*)$ and $p(\mathbf{x}|\mathbf{s}^*)$. Naturally, information from pixels outside the 3x3 window useful in restoring the central pixel is lost.

[2] Measured using expected squared difference

motivation for a new approach based on finite element techniques.

Polynomials

In the context of signal processing, polynomial approaches are usually referred to as (discrete) *Volterra series* [31]. Volterra series are an extension of the impulse response model of linear systems to the nonlinear case. A first-order Volterra approximation to F is:

$$\hat{F}(x_0, x_1, \ldots, x_8) = a_1 + a_2 x_0 + a_3 x_1 + \ldots + a_{10} x_8 \tag{2.3}$$

where the $\{a_i\}$ are *basis function coefficients*. Better approximations are obtained by adding higher-order (quadratic, cubic) terms. The main problem with polynomial bases is that evaluation complexity is linear in the number of basis functions. Obtaining high-quality results can require thousands of basis functions; such filters are extremely slow, since restoring each pixel requires evaluating thousands of polynomial terms. An r^{th} order polynomial basis on N variables requires

$$\begin{pmatrix} N+r \\ r \end{pmatrix}$$

basis functions. For example, a 5^{th}-order basis for 9 inputs (a 3x3 window) requires 2002 basis functions. A 5^{th}-order basis for 25 inputs (a 5x5 window) requires 142506 basis functions. Some progress has been made in reducing the number of basis functions using tensor-product bases [32] and adaptive polynomial filters [33].

Training polynomial filters is another challenge. Since the polynomial basis functions are not orthogonal, a least-squares approach is not practical because it leads to a dense system of equations.[3] The least-squares equations for 10000 coefficients of a polynomial basis would require roughly 1 Gb of RAM to store – unrealistic for most workstations. Adding a single training sample to the least-squares equations would require updating every single element in this matrix, making training prohibitively slow. One has to resort to an iterative approach, which means many passes through the training data.

The use of MSE as a criterion function for polynomials leads to troubles: since polynomial basis functions cover the entire domain of x, each filter coefficient affects the performance of the filter for every kind of input signal. This tends to prioritize

[3]Note that although it is possible to create polynomials bases which are orthogonal under uniform measure (e.g. Legendre polynomials), creating such bases which are orthogonal under an unknown density function $p_x(\mathbf{x})$ is impossible.

signal types according to their frequency in the training data, rather than their visual importance. For example, a filter trained on outdoor scenes will tend to do very well for flat regions (sky, water, surfaces), but comparatively poorly on detailed areas which occur less frequently.

Another problem with polynomials is numerical stability. Consider a seventh order polynomial approximation to $\hat{F}$:

$$\begin{aligned}\hat{F}(x_0, x_1, \ldots, x_8) &= a_1 + a_2 x_0 + a_3 x_1 + \ldots \\ &\quad + a_{63359} x_3^4 x_7^3 + a_{63360} x_3^4 x_7^2 x_8 \\ &\quad + \ldots \end{aligned} \tag{2.4}$$

Adding first-order and seventh-order terms together can result in substantial numerical errors, since they may differ by 14 orders of magnitude or more for a typical image. Since floating-point numbers have limited precision, catastrophic cancellations and precision loss result.

A further problem with polynomials is their response to outliers. Polynomial approximations have wild oscillations and run off to $\pm\infty$ away from the regions where their behaviour has been specified. When a polynomial filter encounters something never seen in the training data, the result may be unpredictable (and undesirable).

2.1.2 Neural Networks

It has been shown that neural networks with a single hidden layer are "universal approximators", in the sense that they can approximate any function to any desired degree of accuracy, provided enough hidden units are used [34]. Local nonlinear image restoration is essentially a problem of function approximation, so the feed-forward network is a likely candidate. Neural networks are popular for higher-level image processing tasks, such as segmentation and edge detection, but comparatively little work has been done on local nonlinear image restoration [23, 35, 36, 37]. Feed-forward neural networks share some of the shortcomings of polynomials:

- Evaluation time is linear in the number of coefficients (weights). Achieving high quality results can require networks with tens of thousands of weights, so processing an image using such a network is very slow.
- As with polynomials, the response of neural networks to outliers – unexpected inputs not present in the training set – is unpredictable.

- Training is very slow, requiring many passes through the training data. The complexity of processing a single training sample is linear in the number of network weights. A high-quality result might require tens of thousands of weights, and training sets can contain millions of samples.[4]

- Unlike polynomials, neural networks do not have any guaranteed *rate* of convergence. Although convergence to the optimal F might be guaranteed in the limit as the number of weights approaches infinity, the rate of convergence might be slow enough to make such a guarantee meaningless.

2.1.3 A new approach: grid filters

This thesis proposes a new approach to local nonlinear image restoration: the function F is approximated on a grid of points in an 8 (or more) dimensional space. These Grid *Filters* have some useful properties:

- Evaluation time is small and constant, regardless of the number of filter coefficients. This means you can get arbitrarily close approximations to the optimal F, without paying a substantial performance price.[5]

- Unlike neural networks, which require many presentations of the training samples, grid filters require only a single presentation. This results in faster training times.

- The basis functions used by grid filters affect only a small region of the domain of $\hat{F}$. This means they do not suffer from the problem which polynomials do when trained by MSE – signals are not prioritized by their frequency in the training set.

- When the filter encounters outliers (unexpected inputs), it passes them through unchanged.

- Grid filters contain order-statistic filters as a subset; as a consequence, performance is guaranteed to be better or equivalent to the optimal order statistic filter.

---

[4] A typical image contains about 0.25 million pixels. It does not take many training images to get millions of training samples.

[5] There is a *slight* decrease in filtering time as the number of coefficients increases, due to cache effects.

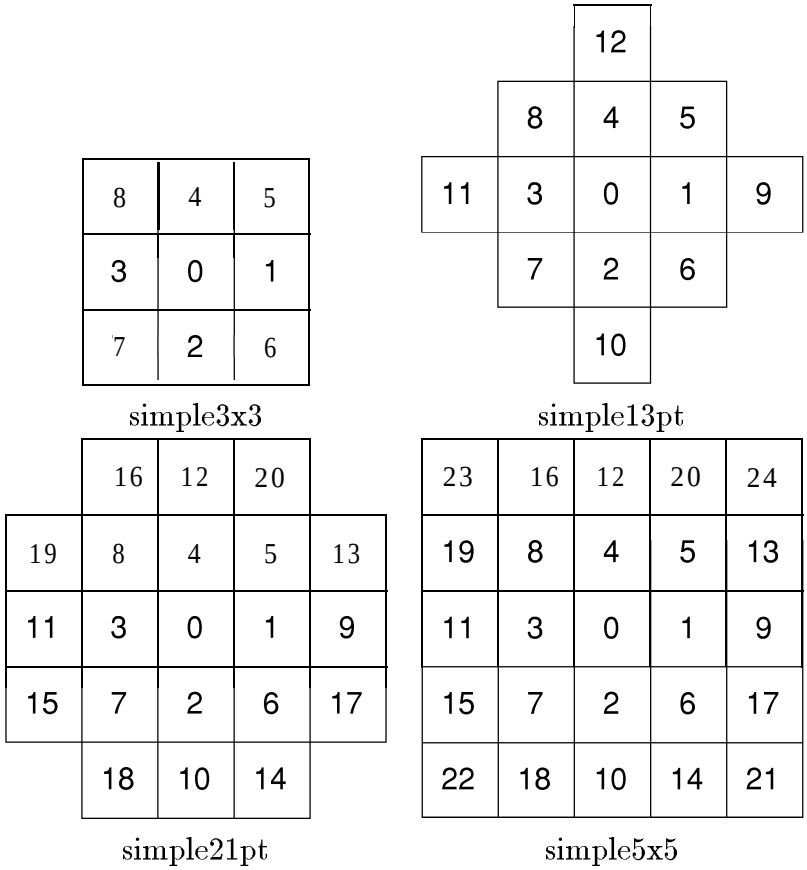


Figure 2.2: Simple operator footprints

- Hybrid Grid filters (described later) contain Lee's local statistics filter for additive noise as a special case.

2.2 Feature selection

Images are restored one pixel at a time, using a nonlinear function of *features* from the surrounding neighborhood. For this thesis, only very simple features have been used: features are either single pixel values, or averages of several pixels. The pattern of pixels used as features is referred to as the operator's *footprint.* Figure 2.2 shows some footprints which use single pixel features.

The simple5x5 footprint of Figure 2.2 has 25 features, which means a 25-dimensional function approximation. The smallest possible grid in 25 dimensions (not counting symmetries, a 2x2x2x. . .x2 grid) contains 33 million points – storing

Footprint name	Number of contributing pixels	Number of features
simple3x3	9	9
simple13pt	13	13
simple21pt	21	21
simple5x5	25	25
fovea5x5	21	9
fovea5x5b	25	10
fovea7x7	49	10
fovea7x7b	25	13
fovea7x7c	49	14
fovea7x7d	49	17
fovea15x15b	225	16
fovea15x15c	113	15
fovea15x15d	113	21
fovea31x31	481	17

Table 2.1: Summary of simple and foveated operator footprints

such a grid would require at least 128 Mb of memory. The grid size is exponential in the number of features – so it is crucial to use as few features as possible.

However, restricting the footprint to a small area – say, 3x3 – will not work either. Noise suppression requires a large footprint, since many pixels are needed to get an accurate estimate of the local signal mean in flat regions. One solution explored in this thesis is *foveated footprints*. Foveated footprints use single-pixel features near the centre of the footprint. Away from the centre, multiple pixels are averaged to create single features.

Figure 2.2 shows some foveated footprints. The larger ones cover a 7x7, 15x15 or even 31x31 area, but use only 10 to 21 features. The single-pixel inputs near the centre of the footprint (the "fovea") are useful for filtering around edges, lines, and fine details. In flat regions, the multiple pixel features (the "periphery") can be used to accurately estimate the local signal mean.

Table 2.1 summarizes the simple and foveated operator footprints.

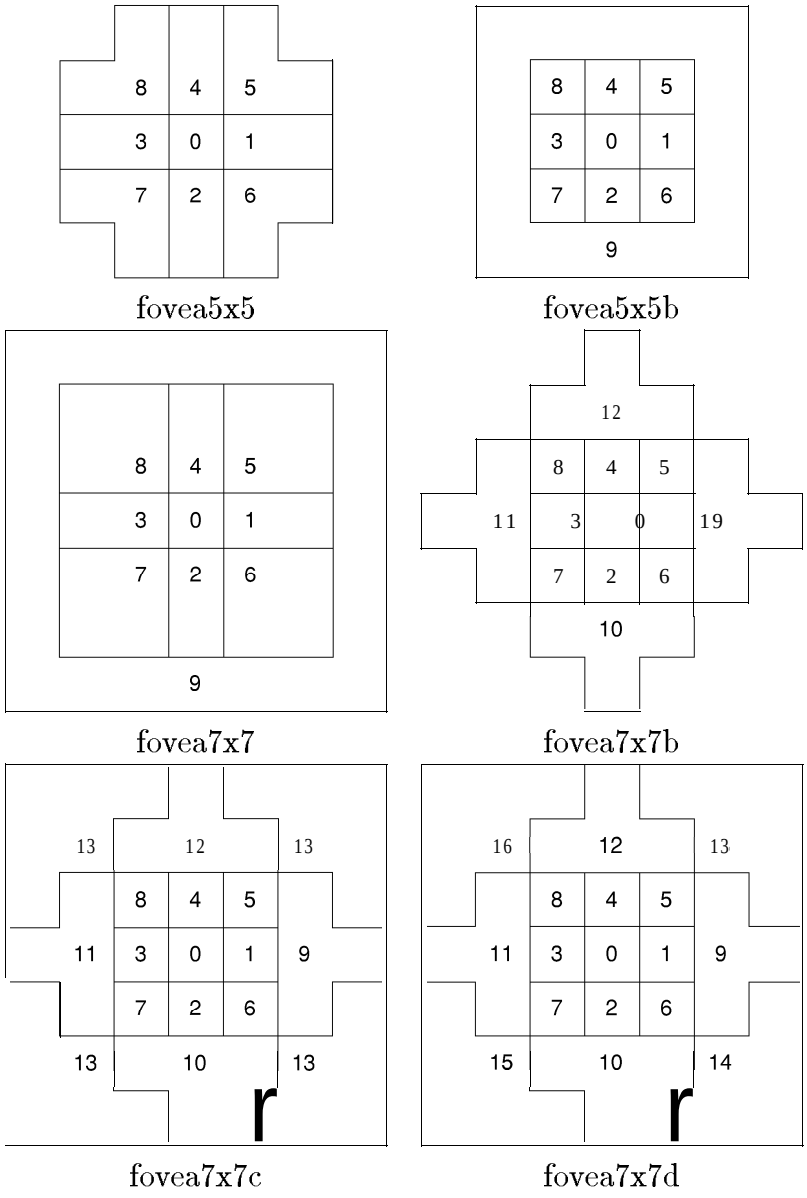


Figure 2.3: 5x5 and 7x7 foveated operator footprints.

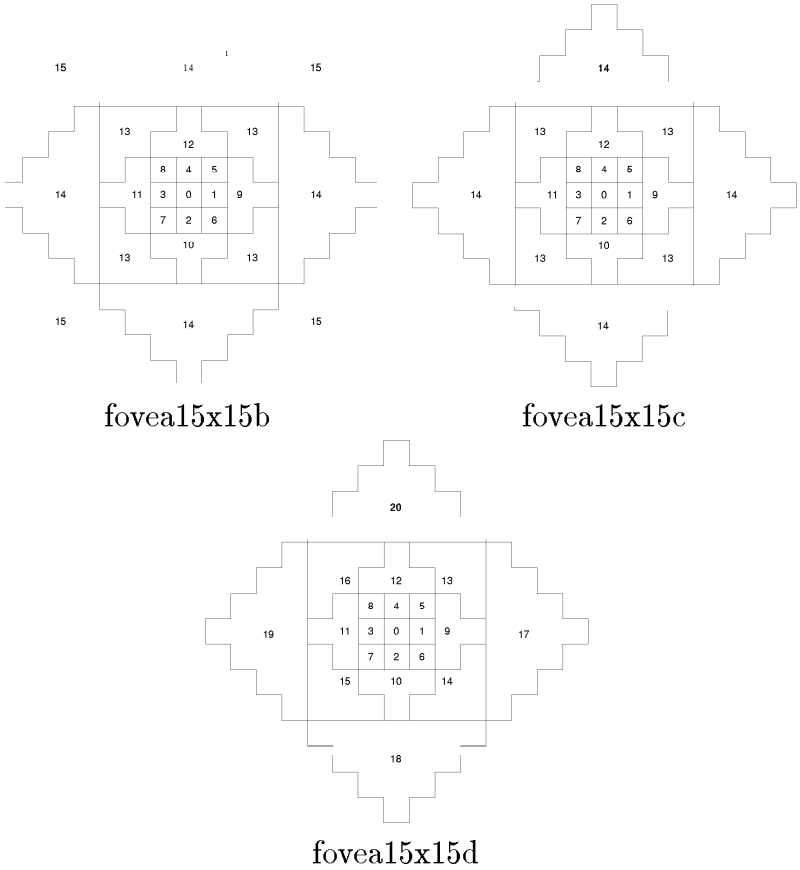


Figure 2.4: 15x15 foveated operator footprints

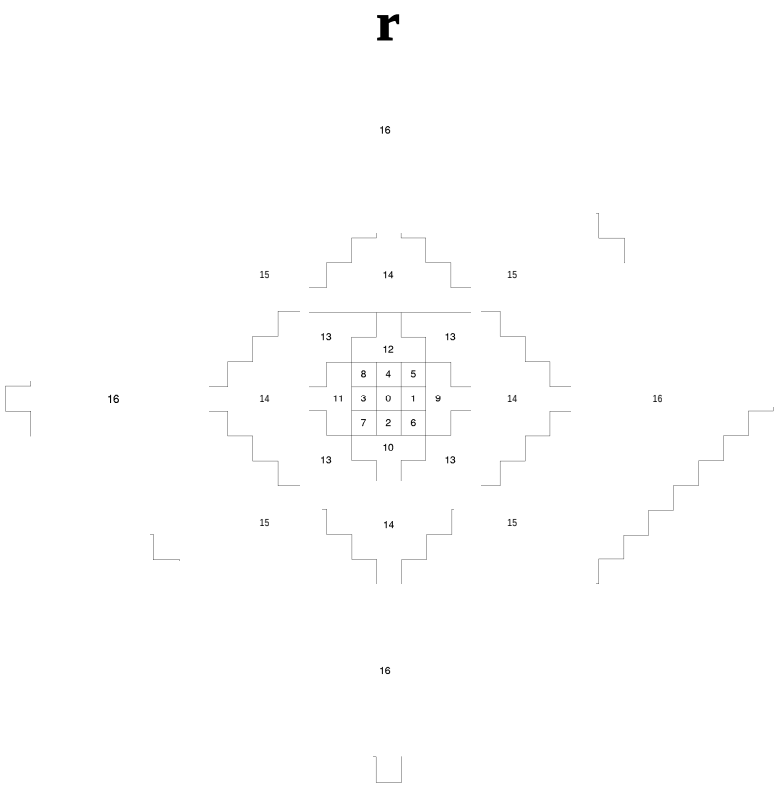


Figure 2.5: A 31x31 foveated operator footprint

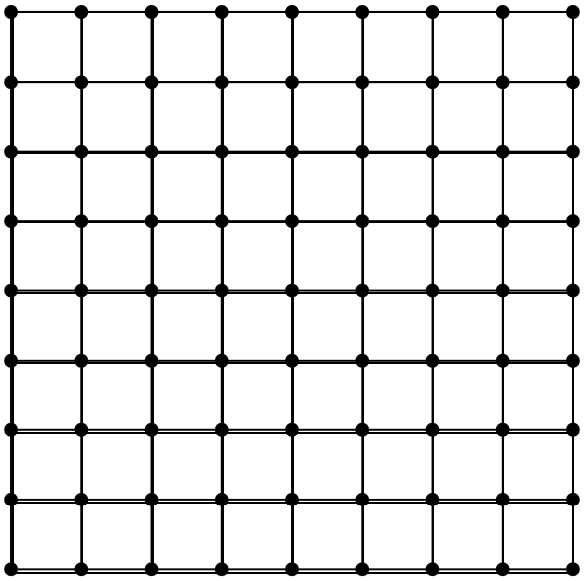


Figure 2.6: A 9^2 grid

2.3 Structure of the grid

The grids described in this thesis are all uniformly spaced: the spacing between grid points is the same in all dimensions. Figure 2.6 illustrates a two-dimensional grid which is used here to describe some terminology.

The grid *extent* is the number of grid points in each dimension. The extent of the grid in Figure 2.6 is 9. The *size* of the grid is the nominal number of grid points; the size of the grid in Figure 2.6 is 9^2. The exponent is the dimensionality of the grid, and the base is the extent.

The grid is uniformly scaled so that it contains all (or almost all) of the domain of the inputs. The width of the scaled grid is referred to as the grid *length*. The grid has the same length in all dimensions.

2.4 Sparse Grid Representation

It turns out that most of the grid points are never needed. To illustrate this, consider a very simple scenario: restoring completely empty images (all pixels set to zero) degraded by additive, zero-mean *iid* Gaussian noise. For features, N single pixel values from the local neighborhood will be used.

Each of the features will be zero-mean, *iid* Gaussian distributed. The grid

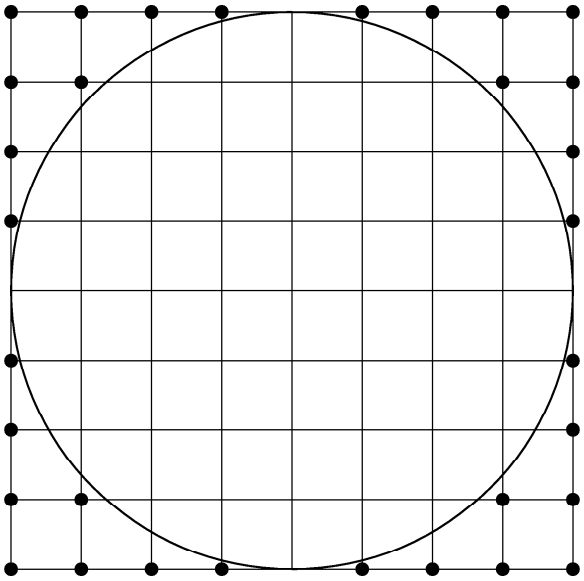


Figure 2.7: Illustration of unnecessary grid points

domain will be $[-R,+R]^N$, with R chosen so that the probability of a feature being outside the hypersphere of radius R is some very small value ϵ. The grid points outside the sphere, but inside the hypercube $[-R,+R]^N$ will be required with probability much less than ϵ, so they can be safely discarded. Figure 2.7 illustrates this scenario for the case N = 2. Grid points lying outside the circle (marked by black dots) are not needed.

If the grid is reasonably fine (i.e. lots of grid points), the proportion of grid points needed can be found by comparing the area of the hypersphere to the area of the bounding hypercube. The volume of the hypersphere is:

$$vol(S_N) = \frac{\pi^{N/2} R^N}{\Gamma(\frac{1}{2}N + 1)} \tag{2.5}$$

and the volume of the hypercube is:

$$vol(Q_N) = (2R)^N \tag{2.6}$$

Their ratio is:

$$\frac{vol(S_N)}{vol(Q_N)} = \left(\frac{\sqrt{\pi}}{2}\right)^N \frac{1}{\Gamma(\frac{1}{2}N + 1)} \tag{2.7}$$

Dimensions	Ratio
2	0.78539816339
8	0.01585434424
12	0.00032599188
16	0.00000359086
20	0.00000002461
24	0.00000000012

Table 2.2: Ratio of hypersphere volume to the bounding hypercube as the number of dimensions increases

Table 2.2 lists the value of this ratio for several N. For a reasonable number of inputs, the vast majority of grid points are unnecessary.

The ratio goes to zero for large N because $\Gamma(\frac{1}{2}\,\mathrm{N}+1)$ is asymptotically bounded to $(\frac{N}{2e})^{\frac{N+1}{2}}$ (by Stirling's approximation). This grows much faster than the$\left(\frac{\sqrt{\pi}}{2}\right)^N$ term."

For realistic images, a similar reasoning applies. Small windows drawn from real images will form clusters in the N-dimensional space, corresponding to edges, flat regions, lines and ramps. These clusters will most likely have hyperellipsoid (or similar) shapes. The ratio of the volume of these clusters to the volume of the grid domain will tend to zero as N gets large.

To exploit this property, a sparse representation of the grid is used. Each grid point is stored in a hash table, keyed by integer grid coordinates. During training, grid points not present in the hash table are created as needed. This is referred to as *dynamic node creation.* Using this approach (plus some symmetry assumptions described later), filters based on grids as large as 21^{12} have been created, requiring only 18000 coefficients.[7]

'Approximating the number of grid points using the volume of the hypersphere works well for a small number of dimensions. However, at a certain critical dimension, hyperspheres stop growing in volume and start shrinking. This is because $lg\mathrm{vol}(S_N)$ is $\Theta(-NlgN)$ due to the Γ term in the denominator. However, the number of grid points does not shrink, but behaves asymptotically as $\Theta(N^r)$, where $r = (L/2)^2$ and L is the extent of the grid in each dimension. This is still much better than the full hypercube grid, in which the number of grid points behaves asymptotically as $\Theta\left(L^N\right)$.

[7]Without the sparse representation and symmetry reductions, this filter would require 7355827511386641 (about $7\mathrm{x}10^{15}$) coefficients.

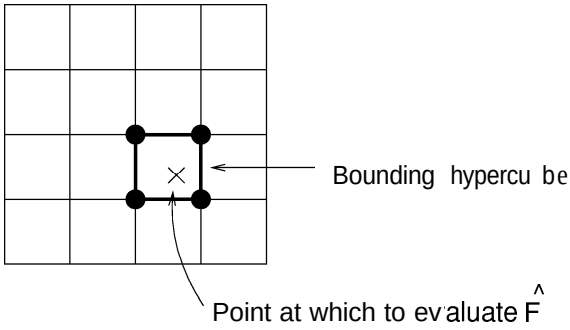


Figure 2.8: A point in the domain of the grid and its enclosing hypercube

2.5 Interpolation

Recall that the nonlinear function F is approximated by values on a regular grid of points. To approximate F at points which do not coincide exactly with a grid point, interpolation is necessary. In this section, two kinds of interpolation are described: multilinear and piecewise linear. The multilinear interpolant is in common use, and is used to point out how conventional interpolation techniques break down in many dimensions. Its failings motivate the development of the piecewise linear interpolant.

2.5.1 Multilinear interpolation

Bilinear and trilinear interpolants are well-known in finite element analysis [38] and computer graphics [39]. Multilinear interpolants are a simple extension to the multidimensional case.

For any given point in the domain of a D-dimensional grid, one can find an enclosing hypercube whose corners coincide with grid points (Figure 2.5.1). Each hypercube has 2^D such corners. To simplify the description, assume the hypercube is over the region $[0, 1]^D$. The aim is to interpolate the value of $\hat{F}$ at some interior point x given its value at the corners. Assume the grid points of the hypercube enclosing x are given by the set $nodeset(\mathbf{x})$, and that node i has coordinates $(z_1^i, z_2^i, z_3^i, \ldots, z_D^i)$. The value of the function at node (grid point) i is f_i.

In multilinear interpolation, the function values associated with all 2^D grid points of the hypercube contribute to every point in the interior. Over the unit hypercube $[0, 1]^D$, the interpolation is given by:

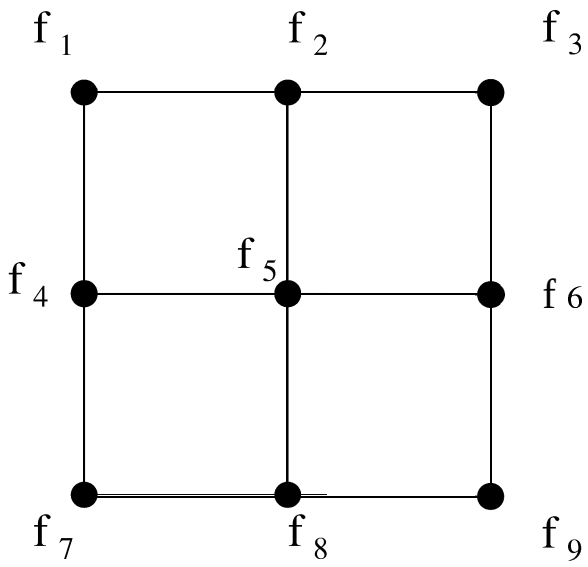


Figure 2.9: Example of a small, two-dimensional grid.

$$\hat{F}(\mathbf{x}) = \sum_{i \in nodeset(\mathbf{x})} f_i \prod_{j=1}^{D} \left(1 - |x_j - z_j^i|\right) \tag{2.8}$$

Interpolating over an element which is not $[0, 1]^D$ requires some scaling factors. Equation 2.8 is linear in the function values $\{f_i\}$. This means that the interpolation scheme for the entire grid can be written as:

$$\hat{F}(\mathbf{x}) = \sum f_i w_i(\mathbf{x}) \tag{2.9}$$

where w_i is a *basis function* associated with grid point i. Figure 2.10 shows the multilinear basis function associated with the grid point f_5 of Figure 2.9.

Interpolating a single value k(x) involves 2^D function values f_i – one function value for each node in the enclosing element. For some of the filters described later, D is 16 or higher. If this interpolation method were used, filtering a single pixel would require a linear combination of more than 60000 function values! This would be prohibitively slow, even on a very fast computer. The situation would be even worse for training: as will be described later, training time is quadratic in the number of function values which contribute to an interpolation. In 16 dimensions, each pixel in a training image would require on the order of $(2^{16})^2$ floating point operations.

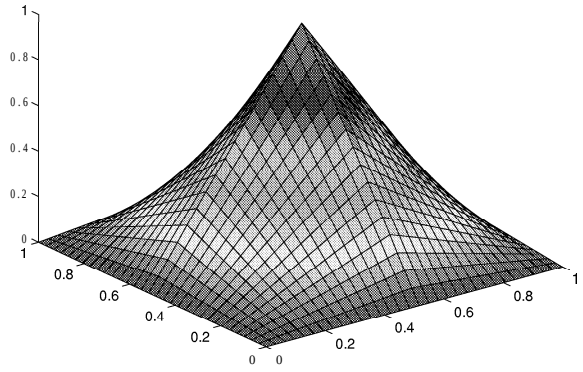


Figure 2.10: Multilinear interpolation basis function for the two dimensional case.

For these reasons, multilinear interpolation is impractical in many dimensions. Spline interpolants are similarly impractical. What is needed is an interpolation scheme with the fewest possible contributing grid points. Piecewise linear interpolation turns out to be the best solution.

2.5.2 Piecewise linear interpolation

In piecewise linear interpolation, each hypercube is sliced into D! smaller regions. These regions are simplexes (convex regions bounded by hyperplanes). The interpolation is linear over each simplex.

As before, only the unit hypercube $[0, 1]^D$ will be considered; the extension to the general case requires some simple scaling factors. Visualization of this interpolation scheme is difficult, so consider the two dimensional case: a unit square (Figure 2.11). The value of $\hat{F}$ is specified at each of the nodes: $\hat{F}(0,0) = f_1, F(1,0) = f_2$, $\hat{F}(1,1) = f_3$, and $\hat{F}(0, 1) = f_4$.

The diagonal line in Figure 2.11 partitions the square into two triangular regions: the lower triangle is $0 \leq x_2 \leq x_1 \leq 1$, and the upper triangle is $0 \leq x_1 < x_2 \leq 1$. In the lower triangle, the function F is approximated by the linear function

$$\hat{F} = f_1 + (f_2 - f_1)x_1 + (f_3 - f_2)x_2 \tag{2.10}$$

And over the upper triangle, the interpolation is

$$\hat{F} = f_1 + (f_4 - f_1)x_2 + (f_3 - f_4)x_1 \tag{2.11}$$

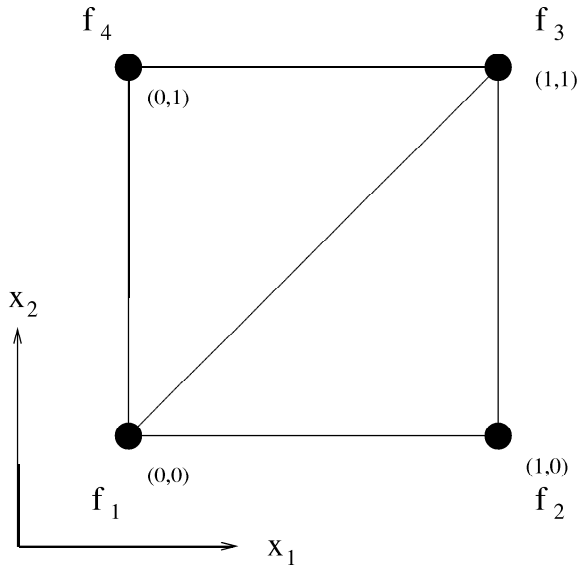


Figure 2.11: Two-dimensional hypercube on $[0, 1]^2$

It is easy to verify that (2.10) and (2.11) are linear and recover the values of F at the corners.[8]

Note the pattern in (2.10) and (2.11): they start at node f_1, and trace out a route from f_1 to f_4 along the perimeter of the triangle, each time adding the difference between consecutive node values (e.g. $f_4 - f_1$) multiplied by the position of the point x in the dimension just travelled (e.g. x_2). This pattern extends to the D-dimensional case: The path followed is determined by always moving along the dimension associated with the largest remaining x_i.

Here is an example to illustrate the procedure for three dimensions. Suppose we want to interpolate over the hypercube $[0, 1]^3$ for the point

$$\mathbf{x}^T = [\ 0.2\ \ 0.6\ \ 0.3\ ]$$

Figure 2.13 illustrates the hypercube and function values. For simplicity, the function values have been labelled so that $\hat{F}(i, j, k) = f_{ijk}$. The path followed is determined by sorting the x_i in ascending order: $x_2 \geq x_3 \geq x_1$ for this situation. This path is illustrated by the arrows in Figure 2.13. The interpolation formula is:

[8] The keen reader will note that the division of the square into two triangles (Figure 2.11) can be done in two ways; rather than drawing the diagonal line from f_1 to f_3, it could be drawn from f_2 to f_4. This type of interpolation introduces an anisotropy: the basis functions have a definite orientation to them. By sacrificing isotropy, substantial computational gains are made.

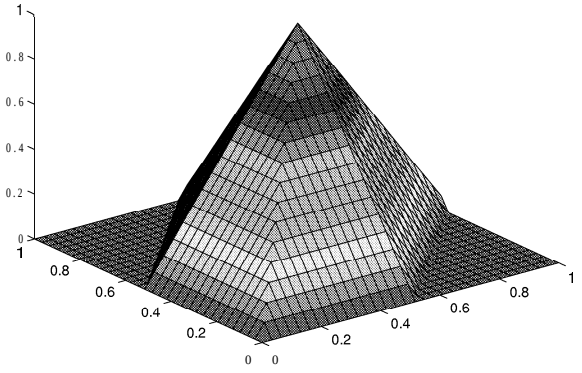


Figure 2.12: Piecewise linear interpolation basis function for the two dimensional case. This basis function corresponds to f_5 of Figure 2.9.

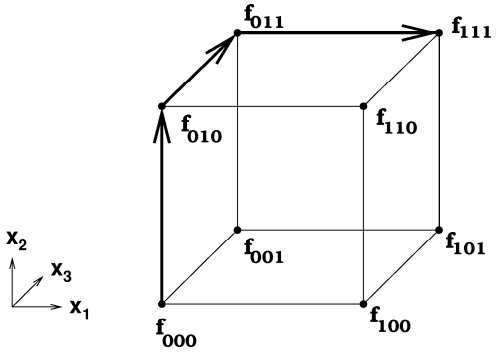


Figure 2.13: Three-dimensional interpolation example

$$\hat{F} = f_{000} + (f_{010} - f_{000})x_2 + (f_{011} - f_{010})x_3 + (f_{111} - f_{011})x_1 \tag{2.12}$$

This interpolation is valid over $1 > x_2 \geq x_3 \geq x_1 \geq 0$. Note that the function values $f_{000}, f_{010}, f_{011}$, and f_{111} are recovered exactly for appropriate choices of x.

Here is the interpolation procedure for D=5 to illustrate the general case.

- Let x $\in [0, 1]^5$ be a point in the unit five-dimensional hypercube. The function has given values $f_{00000}, f_{00001}, f_{00010}, \ldots, f_{11111}$ at the 2^5 corners of the hypercube. For example, at the corner $(0, 1, 1, 0, 1)$, the function $\hat{F}$ has value f_{01101}.

- Find the permutation p which puts the elements of x in descending order. The permutation p maps each of the symbols x_1, x_2, x_3, x_4, x_5 to another symbol from the same set. Denote by px_i the image of x_i under this mapping. The permutation p is chosen so that

$$1 \geq px_1 \geq px_2 \geq px_3 \geq px_4 \geq px_5 \geq 0 \tag{2.13}$$

- Find the inverse permutation p^{-1} (this is the permutation that puts the sorted $p\mathbf{x}$ back in its original order).

- The interpolation is then given by:

$$\begin{aligned} F = f_{p^{-1}(00000)} &+ (f_{p^{-1}(10000)} - f_{p^{-1}(00000)})px_1 \\ &+ (f_{p^{-1}(11000)} - f_{p^{-1}(10000)})px_2 \\ &+ (f_{p^{-1}(11100)} - f_{p^{-1}(11000)})px_3 \\ &+ (f_{p^{-1}(11110)} - f_{p^{-1}(11100)})px_4 \\ &+ (f_{p^{-1}(11111)} - f_{p^{-1}(11110)})px_5 \end{aligned} \tag{2.14}$$

  where p^{-1} *(abcde)* means apply the inverse permutation p^{-1} to the symbols *(abcde).*

Note that this interpolation scheme involves only 6 function values, namely $f_{p^{-1}(00000)}$, $f_{p^{-1}(10000)}$, . . ., $f_{p^{-1}(11110)}$, and $f_{p^{-1}(11111)}$. Figure 2.14 shows a pseudocode implementation for the general D-dimensional case. The interpolation requires $D + 1$ function values.

```
PIECEWISELINEARINTERPOLATE(x)
Interpolate F̂ for a point x using piecewise linear interpolation
    Find the enclosing element and convert to [0, 1]^D coordinates:
    coord is the integer grid coordinate corresponding
          to the point (0, 0, 0,. . . , 0) of the enclosing element.
    t is the normalized [0, 1]^D position inside the element.
    L is the grid spacing.
    do i = 1, D
        coord[i] ←⌊x_i/L⌋
        t_i ←(x_i − coord[i] ∗ L)/L

    Find the permutation p which puts the t_i in
    descending order: t_p[1] ≥ t_p[2] ≥ ... ≥ t_p[D]
    p ← FINDPERMUTATION(t)

    Determine the set of nodes and interpolation coefficients:
        nodes[1..D+1] contains the contributing node numbers
        coeff[1..D+1] contains the interpolation coefficients
    nodes[1] ←FINDNODENUMBER(coord)
    coeff[1] ← 1.0
    do i = 1, D + 1
        j ← p[i]
        coeff[i] ← coeff[i] - t [j]
        coeff[i+1]← t[j]
        coord[j] ← coord[j] + 1
        nodes[i+1] ← FINDNODENUMBER(coord)

    return [coord,coeff]
```

Figure 2.14: Piecewise linear interpolation algorithm

2.5.3 Similarity to order-statistic filters

It turns out that grid filters with piecewise linear interpolation have a close relationship with order statistic filters. To interpolate for a given point x, piecewise linear interpolation first finds a bounding hypercube whose corners coincide with grid points. This hypercube is scaled into the unit hypercube $[0, 1]^D$. Let x' be the point x after being scaled into the unit hypercube. The interpolation scheme slices the hypercube into N! simplexes of the form:

$$0 \leq x'_{i_1} \leq x'_{i_2} \leq \ldots \leq x'_{i_D} \leq 1 \tag{2.15}$$

where $(i_1, i_2, \ldots, i_D)$ is a permutation of $(1, \ldots, D)$. Over each simplex, the interpolation is linear and recovers the values of the grid points at the corners. Compare this to order statistic filters, which are linear over regions of the form

$$x_{i_1} \leq x_{i_2} \leq \ldots \leq x_{i_D} \tag{2.16}$$

Under certain conditions, the grid filter can be a strict superset of the order statistic filter:

- The grid must be centered about the origin $(0, 0, \ldots, 0)$.
- The grid spacing must be isotropic. That is, the spacing between grid points is the same in all dimensions. This is true of all the grid filters described in this thesis.
- The domain of the grid must contain the domain of the order statistic filter.

If these conditions are satisfied, the grid filter contains order statistic filters as a subset. Consequently, the performance of the grid filter will be at least as good as an OSF. If the signal and/or noise are not *iid* (which they are not for any realistic image processing problem), the grid filter will be strictly better than an order statistic filter.

2.6 Symmetry assumptions

The number of grid points can be further reduced by making some reasonable symmetry assumptions.

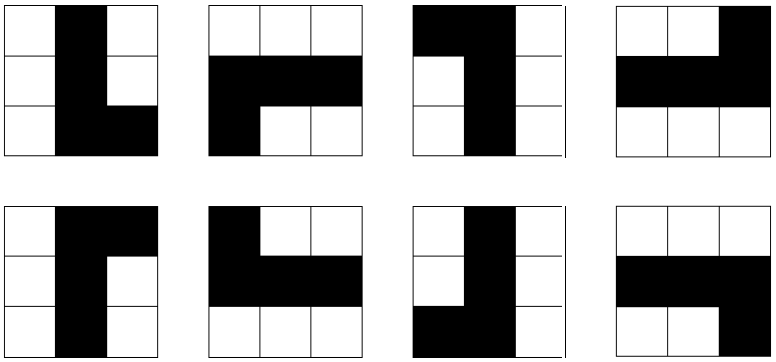


Figure 2.15: Eight orientations of a 3x3 window which should be treated the same by a filter with orientation-invariance.

2.6.1 Orientation invariance

In many applications, the behaviour of the filter should be the same for different orientations of the same 3x3 window. For example, an edge running up and down should be processed the same as an edge running left and right. Figure 2.15 illustrates the 8 orientations of a 3x3 window which should be treated as equivalent (for the purpose of restoring the central pixel).

Working with this type of symmetry is greatly simplified by some rudimentary group theory[40]. A *group* is a set together with a binary operation which maps ordered pairs of elements (a, *b)* to another element, denoted ***ab.*** Simple examples of groups are: the integers under addition; nonsingular real matrices under the matrix product; and real functions under pointwise addition. There are some additional restrictions which groups must satisfy.' The symmetries illustrated in Figure 2.15 form a group. This group is called the dihedral *group of order 8,* and is given the symbol D_4. The group D_4 describes the ways a rigid square in the plane can be transformed onto itself through reflections and rotations. The eight elements of the group are rotation by $0°, 90°, 180°$, 270"; and a flip followed by rotating $0°$, $90°, 180°, 270°$. The group operation is transformation composition; for example,

---

'These properties are

- Associativity: $(ab)c = a(bc)$
- Identity: There is an *identity element* e such that $ae = ea$ = a for all a.
- Inverses: For every element a, there is an *inverse* of a such that $aa^{-1} = a^{-1}a$ = e.
- Closure: For every a and b in the group, ab is also in the group.

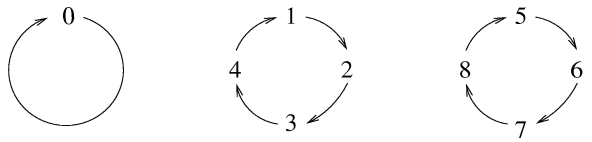


Figure 2.16: Depiction of Cauchy's cycle notation

combining a rotation by 90" with a rotation by 180" produces a rotation by $270°$.

Consider applying the members of this group to a 3x3 array of input pixels, numbered according to this scheme:

8	4	5
3	0	1
7	2	6

Each member of the group produces a permutation of the elements $0,1,\ldots,8$. For example, rotating the array $270°$ clockwise (or equivalently, 90" counterclockwise) results in:

5	1	6
4	0	2
8	3	7

This permutation can be written in "table form" as:

$$\begin{bmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 0 & 2 & 3 & 4 & 1 & 6 & 7 & 8 & 5 \end{bmatrix} \tag{2.17}$$

Each symbol in the first row gets replaced by the corresponding symbol in the second row. The table form does not do a very good job of depicting how things move around under the permutation. Cauchy's cycle notation makes the motion more apparent:

$$(0)(1234)(5678) \tag{2.18}$$

Each set of symbols in parentheses cycles to the left; for example, $1 \leftarrow 2$, $2 \leftarrow 3$, $3 \leftarrow 4$ and $4 \leftarrow 1$. The cycles can be depicted as in Figure 2.16. It is common to omit cycles with only a single element from the notation:

$$(0)(1234)(5678) \equiv (1234)(5678) \tag{2.19}$$

When the 8 elements of the D_4 group are applied to the 3x3 array of input pixels, eight permutations are generated. These permutations form a special kind

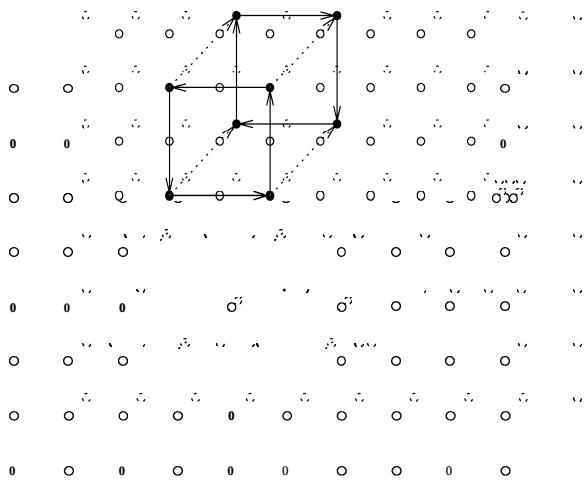


Figure 2.17: The orbits of a grid point under $\mathcal{P}$

of group – a *permutation group.* This permutation group will be referred to as $\mathcal{P}$ from now on.

It turns out that the whole group can be generated from two basic permutations. One such pair is counterclockwise rotation by 90” degrees and a horizontal flip. They can be written using cycle notation as:

$$\begin{aligned} \alpha &= (1234)(5678) \\ \beta &= (13)(58)(67) \end{aligned} \tag{2.20}$$

Any member of the permutation group can be written as $\alpha^i\beta^j$ where $i \in \{0,1,2,3\}$ and $j \in \{0, 1\}$.

What happens when $\hat{F}(\mathbf{x})$ is required to be invariant under this permutation group? For any permutation $p \in \mathcal{P}$,

$$\hat{F}(\mathbf{x}) = \hat{F}(p\mathbf{x}) \tag{2.21}$$

Suppose x is picked to coincide exactly with a grid point. Then $\hat{F}(\mathbf{x})$ is exactly f_j, where j is the node number associated with grid point x. Similarly, $\hat{F}(p\mathbf{x})$ is exactly f_k, where k is another node associated with px. Then from (2.21) $f_j = f_k$. If this exercise is repeated for each $p \in \mathcal{P}$, one finds the eight grid points (recall the size of the permutation group is $|\mathcal{P}| = 8$) whose function values are identical. These eight grid points are the *orbit* of x under $\mathcal{P}$, denoted $\mathrm{orb}_{\mathcal{P}}(\mathbf{x})$. This is illustrated by Figure 2.17.

```
FINDNODENUMBER(coord)
Find the node number residing at the given coordinates
    Apply each member of 𝒫 to coord and look up the
    permuted coordinates in the grid hash table.
    do i = 0, 3
        do j = 0, 1
            newcoord ← APPLYPERMUTATION(α^i β^j, coord)
            N ← GRIDLOOKUP(newcoord)
            if N ≠ ∅ then break

    if currently in training mode
        N ← CREATEGRIDPOINT(coord)

    return N
```

Figure 2.18: FINDNODENUMBER algorithm

Since the eight function grid points are identical, only one representative of $\mathrm{orb}_{\mathcal{P}}(\mathbf{x})$ need be stored. The routine FINDNODENUMBER$(\mathbf{x})$ which looks up nodes in the grid can search the orbit of x under $\mathcal{P}$ to find this representative. Figure 2.18 shows a pseudocode implementation of this approach.

Each footprint has its own permutation group, but they are all isomorphic to D_4, and can be represented by two generators a and β. For example, Figure 2.19 shows the fovea7x7b footprint. Its generators (corresponding to counterclockwise rotation by 90" and horizontal flip) are:

$$\begin{aligned} \alpha &= (1,2,3,4)(5,6,7,8)(9,10,11,12) \\ \beta &= (1,3)(5,8)(6,7)(9,\ 11) \end{aligned} \tag{2.22}$$

2.6.2 Signal mean invariance

Another type of invariance common to image processing filters is *signal mean invariance:* the behaviour of the filter is not affected by shifts in the mean of the signal. If an amount δ is added to all the inputs, then the output should also shift by δ. In symbols,

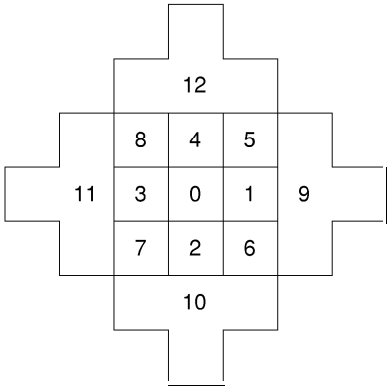


Figure 2.19: The footprint fovea7x7b

$$\hat{F}(x_0 + \delta, x_1 + \delta, \ldots, x_{n-1} + \delta) = \hat{F}(x_0, x_1, \ldots, x_{n-1}) + \delta \tag{2.23}$$

This assumption is valid when the degradation process consists of signal-independent additive noise and/or a point spread function with unit DC gain. Many popular image processing filters possess this property, including local linear filters, Wiener filters," order statistic filters, Lee's local statistics filter for additive noise, and many adaptive recursive schemes which are designed to have unit DC gain.

To exploit this property, consider setting $\delta = -x_0$. Then, by (2.23)

$$\hat{F}(x_0, x_1, \ldots, x_{n-1}) = x_0 + \hat{F}(0, x_1 - x_0, x_2 - x_0, \ldots, x_{n-1} - x_0) \tag{2.24}$$

This property reduces the arguments of F by one, resulting in one fewer dimensions for the problem of approximating F. Instead of approximating a function F over (for example) a nine-dimensional space, only eight dimensions are required. This results in a useful reduction in the number of grid points.

Signal mean invariance also turns the function $\hat{F}$ into an *adjustment* applied to the central pixel value x_0 (2.24). This turns out to be crucial for numerical stability considerations, described in Section 2.7 (p. 52).

2.6.3 Reversed-intensity invariance

Another common type of invariance is *reversed-intensity invariance.* A reversed-intensity version of an image is akin to a photographic negative: it is obtained

[10]Under the mild assumption that $H(0, 0) \approx 1$ for almost all applications

by reversing the pixel values so that dark pixels become bright, and vice versa. Suppose $\mathcal{I}$ is an image, and denote by $\overline{\mathcal{I}}$ the reversed intensity image. A filter $\mathcal{F}$ with the reversed-intensity invariance property satisfies:

$$\mathcal{F}\overline{\mathcal{I}} = \overline{\mathcal{F}\mathcal{I}} \tag{2.25}$$

Filtering the reversed image gives the same result as reversing the filtered image. Linear filters, Lee's local statistics filter for additive noise, Wiener filters, and order statistic filters" all possess this property.

To see the effect of this invariance on the grid, suppose the maximum pixel intensity value is M. Then the reversed-intensity inputs would be $M - x_1, M - x_2$, ..., $M - x_n$. To have reversed-intensity invariance, $\hat{F}$ must satisfy:

$$\hat{F}(x_0, x_1, \ldots, x_{n-1}) = M - \hat{F}(M - x_0, M - x_1, \ldots, M - x_{n-1}) \tag{2.26}$$

Applying the mean-invariance property (2.24) eliminates M from the equation:

$$\hat{F}(x_0, x_1, \ldots, x_{n-1}) = x_0 - \hat{F}(0, x_0 - x_1, x_0 - x_2, \ldots, x_0 - x_{n-1}) \tag{2.27}$$

Compare this equation to (2.24) which states that

$$\hat{F}(x_0, x_1, \ldots, x_{n-1}) = x_0 + \hat{F}(0, x_1 - x_0, x_2 - x_0, \ldots, x_{n-1} - x_0) \tag{2.28}$$

To exploit this relationship, it is necessary that the grid be centered about $(0, 0, \ldots, 0)$. Then (2.27) implies that the grid possesses an anti- (or skew-) symmetry: for each grid point with value f_i, there is an antisymmetric grid point with value $-f_i$.

Figure 2.20 illustrates this symmetry on a grid projected into two dimensions. Due to orientation invariance, there are 8 grid points with value f_i (solid circles, upper right). Reverse-video invariance results in a skew symmetry through the origin. There are another 8 grid points with value $-f_i$ (lower left).

Reverse-video invariance therefore halves the number of grid point values.

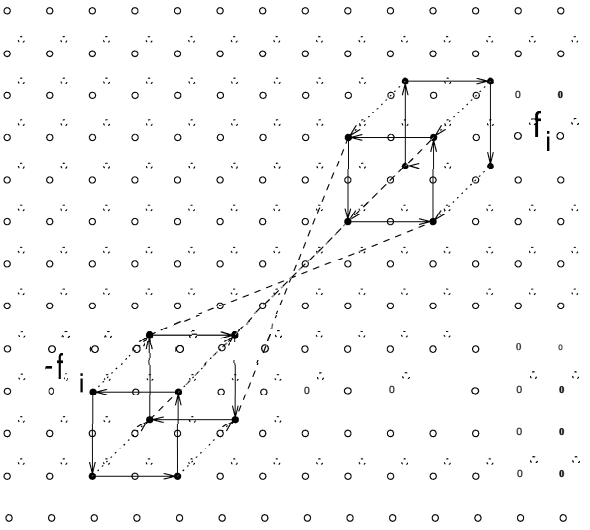


Figure 2.20: Equivalent grid points under reverse-intensity and orientation invariance.

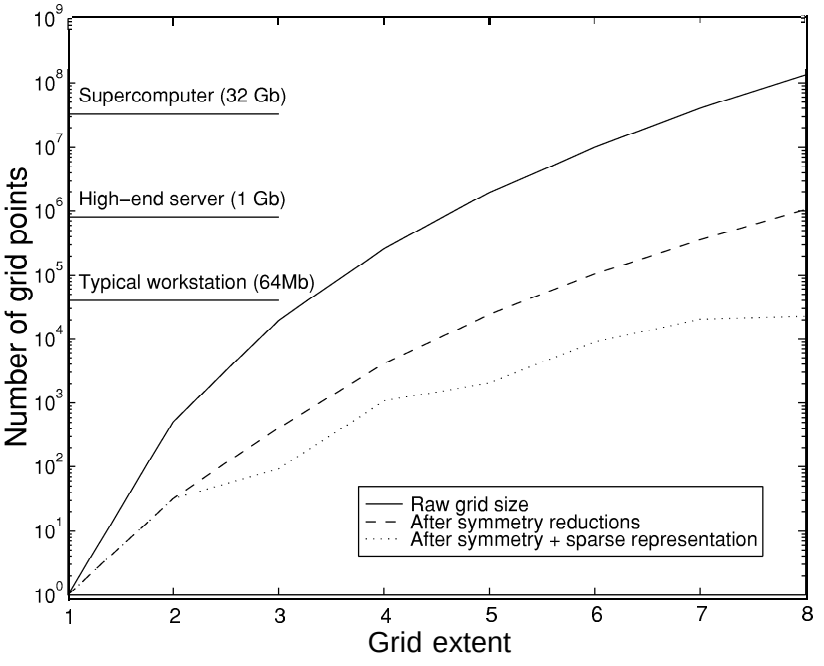


Figure 2.21: Number of grid points for a 3x3 filter

2.6.4 Effect of symmetry reductions

Figure 2.21 shows the number of grid points required for a certain 3x3 filter using various grid extents. The horizontal lines attached to the vertical axis indicate the maximum number of grid points possible for typical computers (circa 1998). Without any reductions in grid size, an 8x8x...x8 grid would require 8^9 coefficients (about 130 million). Training such a filter would tax the abilities of a supercomputer with 32 Gb of memory. After applying symmetry reductions, the number of coefficients drops to roughly 1 million, which would require a high-end computer with about 1 Gb of memory to train. Using a sparse representation for the grid reduces the number of coefficients to a mere 23000. Such a filter can be trained in ten minutes on a typical workstation with 64 Mb of memory.

In addition to reducing the number of filter coefficients, symmetry assumptions help the filter to generalize. Without symmetry assumptions, exhaustive training sets would be required, covering many possible illumination levels and orientations of typical images. A filter with the symmetries described here is able to generalize from a single training sample to similar training samples with varying illumination, orientations, and reversed intensity.

2.7 Training

Training refers to determining the best grid point values $\{f_i\}$ for a class of images. Recall that the MSE-optimal function F is given by

$$F(\mathbf{x}) = E\left[s_0|\mathbf{x}\right] \tag{2.29}$$

where s_0 is the original (noise-free) pixel value. To pick the coefficients $\{f_i\}$, why not simply sample the optimal F at the grid points?

Unfortunately, (2.29) is a completely impractical equation. It requires that distribution functions for the signal and degradation processes be known. Unless one is willing to make very unrealistic assumptions about the signal (e.g. Gaussianity), such distributions are unavailable. Even if a distribution function were available, evaluating (2.29) for a single point x requires convolving functions in 8-24 dimensions (depending on footprint size). Such convolutions are computationally impractical.

A practical alternative is to minimize the filter error over a training ensemble which contains pairs of degraded and pristine images. These images are used to

[11] For noise with symmetric distributions only

provide training samples of original pixels (s_0) and inputs (x) from the degraded version.

Development of the training equations

Let the training ensemble be $\{s_0^j, \mathbf{x}^j\}$ for j = 1,. . . , N. The superscripts are not exponents, but merely number the training samples. The grid consists of coefficients $\{f_i\}$ and associated basis functions $\{w_i(\mathbf{x})\}$, one for each grid point. For simplicity, vector notation will be used: **f** for the coefficients, and w(x) for the basis functions. The grid approximation of F can then be written:

$$\hat{F}(\mathbf{x}) = x_0 + \mathbf{f}^T\mathbf{w}(\mathbf{x}) \tag{2.30}$$

The x_0 term is the pixel value in the centre of the window, and $\mathbf{f}^T\mathbf{w}(\mathbf{x})$ is the adjustment applied to it by the grid filter. The squared-error over the training ensemble is:

$$\mathbf{J(f)} = \sum_{j=1}^{N}\left(x_0 + \mathbf{f}^T\mathbf{w}(\mathbf{x}^j) - s_0^j\right)^2 \tag{2.31}$$

This is a least-squares problem, so the Hessian of (2.31) is semi-positive definite. Minimization of J is therefore achieved by **f** which satisfy $\frac{\partial J}{\partial \mathbf{f}}$ = 0. Applying this to (2.31) results in:

$$\left[\sum_{j=1}^{N}\mathbf{w}(\mathbf{x}^j)\mathbf{w}^T(\mathbf{x}^j)\right]\mathbf{f} = \sum_{j=1}^{N}(s_0^j - x_0^j)\mathbf{w}(\mathbf{x}^j) \tag{2.32}$$

This is a set of linear equations, and can be rewritten as **Af = b,** with

$$\mathbf{A} = \sum_{j=1}^{N}\mathbf{w}(\mathbf{x}^j)\mathbf{w}^T(\mathbf{x}^j)$$

$$\mathbf{b} = \sum_{j=1}^{N}(s_0^j - x_0^j)\mathbf{w}(\mathbf{x}^j)$$

A is a sum of rank-l updates.[12] Since piecewise linear interpolation is used, an interpolation involves only D + 1 function values f_i. This means that w(x) is

[12] A rank-l update of a matrix A has the form A ← A + $\mathbf{w}\mathbf{w}^T$.

```
TRAIN(x, s_0)
Add a training sample to the system of equations
    Find the interpolation for this grid point
    [coord,coeff] = PIECEWISELINEARINTERPOLATE(x)

    Update the sparse matrix (A) and the rhs (b).
        do n = 1, D + 1
            b(coord[n]) ← b(coord[n]) + (s_0 − x_0) * coeff[n]
            do m = n, D + 1
                A(coord[n],coord[m]) ← A(coord[n],coord[m]) + coeff[n]*coeff[m]
    return
```

mostly zero; it has only $D + 1$ nonzero entries. Each rank-l update is therefore sparse, and it turns out that **A** is itself sparse. The right side **b** is a dense vector.

Creating the sparse matrix

For efficiency, the sparse matrix **A** is stored in a hash table during training. This permits quick retrieval of elements, and greatly reduces solution and storage costs.[13] Storage is also saved by only storing the lower-triangular portion of **A,** since it is a symmetric matrix.

Regularization

In practice, the system of equations **Af = b** is singular or ill-conditioned. The reason for this lies in the fact that **(2.32)** is actually a Monte-Carlo approximation to the real MMSE equations:

$$E\left[\mathbf{w}(\mathbf{x})\mathbf{w}^T(\mathbf{x})\right]\mathbf{f} = E\left[(s_0 - x_0)\mathbf{w}(\mathbf{x})\right] \qquad (2.33)$$

That is, instead of forming the system of equations (2.33) by integrating over the signal distributions, it is approximated through random sampling (the training ensemble).

---

[13] Note: the system of equations is typically large enough that storing **A** as a dense matrix is impossible. Some filters described consisted of 16000 grid points, which would require about 2 Gb of RAM were a dense matrix representation used. A sparse matrix representation of the same matrix fits comfortably into 64 Mb of RAM.

Since the grid points are used with highly variable probabilities, multiple grid points may have equations which arise from a single rank-l update. This introduces the singularities. The Monte-Carlo integration also leaves behind residual errors in the matrix and right-hand side.

However, a justifiable stabilization exists. If the system of equations is singular, this means that there are infinitely many choices of coefficients which will perform equally well on the training data. The problem must be further constrained to choose a specific set of coefficients.

The interpretation of the singular system of equations is that the training set is not complete: there are some aspects of the filter's behaviour which have not been specified. A reasonable approach is to select from the equivalent filters the one which makes the least change in the image – otherwise, when the filter encountered something never seen in training, it might turn it into something unpredictable.[14] Recall that when using the mean shift-invariance property, the filter has the form (2.24):

$$\hat{s_0} = x_0 + F(0, x_1 - x_0, x_2 - x_0, \ldots, x_{n-1} - x_0)$$

The change in the pixel value is just $\hat{s_0} - x_0 = F(0, x_1 - x_0, \ldots, x_{n-1} - x_0)$. To minimize this change, the norm of the filter coefficients $||\mathbf{f}||$ can be minimized. Choosing the solution which minimizes $||\mathbf{f}||$ results in a zero-order (or Tikhonov) regularization [4]. A constraint term of the form λI is added to the matrix $\mathbf{A}$:

$$\left[\sum_{j=1}^{N} \mathbf{w}(\mathbf{x}^j)\mathbf{w}(\mathbf{x}^j)^T + \lambda I\right] \mathbf{f} = \sum_{j=1}^{N} (s_0^j - x_0^j)\mathbf{w}(\mathbf{x}^j) \tag{2.34}$$

where λ is chosen to be just large enough to permit a numerically stable solution. For the filters described in this thesis, λ was chosen to be 0.01. This value of λ is large enough to ensure a stable solution, but small enough to not alter the behaviour of the filter substantially.

Choosing this form of regularization drives unused grid coefficients to zero, which permits use of the sparse grid representation.

Iterative solution

The next step is to solve the system of equations to find the coefficients $\{f_i\}$. Since the system is large and sparse, using a factorization (such as Cholesky, QR, PLU)

[14] This is what happens with polynomial filters and outliers: polynomial approximations can have wild oscillations outside the region of training samples, and run off to plus or minus infinity.

```
CONJUGATEGRADIENTSOLVE(A, b)
Solve the sparse system Af = b using CG method
    r^j is the residual at iteration j
    p^j is the descent direction for f
    q^j is the descent direction for r
    f^0 ← 0 (initial solution guess)
    r^0 ← b - MATRIXPRODUCT(A, f^0)
    for j=1,... until convergence
        ρ^j ← DOT(r^(j-1), r^(j-1))
        if j = 1 (first iteration)
            p^j ← r^(j-1)
        else
            β ← ρ^j / ρ^(j-1)
            p^j ← β p^(j-1) + r^(j-1)
        q^j ← MATRIXPRODUCT(A, p^j)
        a ← ρ^j / DOT(p^j, q^j)
        f^j ← f^(j-1) + α p^j
        r^j ← r^(j-1) + α q^j

    return f
```

Figure 2.22: Conjugate Gradient algorithm

is not practical. Factorizations cause too much *fill* – that is, the sparse system becomes dense under factorization, resulting in impractical memory requirements.

To avoid this fill problem, many iterative solution techniques for sparse systems have been developed [41]. For this project, Conjugate Gradient (CG) was selected. It is suitable for symmetric, positive definite systems of equations. CG gets its name from the fact that its consecutive descent directions are conjugate (orthogonal). CG is fast and requires only a small amount of temporary working space. For efficiency, the sparse matrix is converted from hash table storage to compressed row storage (CRS) [42] prior to CG solution.

Figure 2.7 gives an outline of the CG solution algorithm. Convergence was determined by checking the change in the parameter vector f every 20 iterations. The algorithm was halted when

$$\max_i |f_i^j - f_i^{j-20}| \leq 10^{-4} \tag{2.35}$$

i.e. when the maximum coefficient change over 20 iterations was less than 10^{-4}. More precision is not useful, since the matrix $\mathbf{A}$ itself contains large residual errors arising from the Monte-Carlo sampling process. Also, for filtered images, 4-5 accurate digits in the filter coefficients are more than sufficient.

A useful side effect of the first-order regularization is that the parameter λ can be used to speed convergence of the CG solution. Convergence behavior is closely related to the condition number of the matrix $\mathbf{A}$ [41], which can be decreased substantially by choosing a larger λ. Some preliminary experiments indicated that this can be done without measurable impact on the quality of the filter.

2.8 Hybrid filters

The "foveated" footprints were introduced to improve the performance of grid filters in flat regions, where a large region of support is required for good noise suppression. This section describes an alternate approach, in which a hybrid filter combines a signal mean estimator for flat regions with a grid filter to handle detail regions. The primary advantage of this approach is that a larger region of support can be obtained without increasing the dimensionality of the grid.

The outputs from the two filters are mixed according to an indicator function $\beta(\mathbf{x})$. $\beta(\mathbf{x})$ has range $[0, 1]$: the value 0 indicates a flat region, and 1 indicates a region with lots of detail. The filter has the form:

$$\mathrm{F} = \beta F_1 + (1 - \beta) F_2 \tag{2.36}$$

where F_1 is the detail filter, and F_2 is the flat region filter. F_1 is a grid filter with signal mean invariance:

$$F_1 = x_0 + \mathbf{f}^T \mathbf{w} \tag{2.37}$$

Since the grid filter only needs to handle detail regions, a foveated footprint is not necessary – the inputs can be single pixel values from the local window. The footprints simple3x3 and simple13pt are suitable.

The filter F_2 is an estimator for the local signal mean. For Gaussian noise, the best estimator is just a local average over a 5x5, 7x7 or 9x9 window. For other noise models, this estimator might be an order statistic or polynomial estimator. The training procedure is different from the plain grid filters, due to the presence of the β indicator function. As before, mean-squared error is used as a criterion:

$$\begin{aligned} J &= \frac{1}{2}E\left[(F - s_0)^2\right] \\ &= \frac{1}{2}E\left[\left(\beta(x_0 + \mathbf{f}^T\mathbf{w}) + (1-\beta)F_2 - s_0\right)^2\right] \end{aligned} \tag{2.38}$$

The optimal grid coefficients **f** are found by setting $\frac{\partial J}{\partial \mathbf{f}} = 0$. This leads to:

$$E\left[\beta^2 \mathbf{w}\mathbf{w}^T\right] f = E\left[\beta \mathbf{w}\left(s_0 - \beta x_0 - (1-\beta)F_2\right)\right] \tag{2.39}$$

The expectation operators are approximated by summing over a set of training samples $\{s_0^j, \mathbf{x}^j\}$. This leads to a system of linear equations **Af = b,** with

$$\begin{aligned} \mathbf{A} &= \sum_{j=1}^{N} \beta^2(\mathbf{x}^j)\mathbf{w}(\mathbf{x}^j)\mathbf{w}^T(\mathbf{x}^j) \\ \mathbf{b} &= \sum_{j=1}^{N} \beta(\mathbf{x}^j)\mathbf{w}(\mathbf{x}^j)\left(s_0^j - \beta(\mathbf{x}^j)x_0^j - (1-\beta(\mathbf{x}^j))F_2(\mathbf{x}^j)\right) \end{aligned} \tag{2.40}$$

The β^2 term in **A** and the β term in **b** weight the least-squares solution heavily toward performance in detail regions. Since flat regions are handled by F_2, the grid filter can "concentrate its attention" on performing well in detail regions. The presence of the $(1-\beta)F_2$ t **b**

How **to** vary β?

The formula for $\beta(\mathbf{x})$ has been borrowed from the Lee filter [11]. Given a noise estimate σ_n^2, the local mean and variance are estimated over the pixels in a window:

$$\begin{aligned} \bar{x} &= \frac{1}{N}\sum x_i \\ \sigma_x^2 &= \frac{1}{\mathrm{N}\ -\ 1}\sum (x_i - \bar{x})^2 \end{aligned} \tag{2.41}$$

When $\sigma_x^2 >> \sigma_n^2$, this indicates a detail region and $\beta \to 1$ is desired. When $\sigma_x^2 \approx \sigma_n^2$, the region is flat and $\beta \to 0$ is needed to turn on the smoothing filter. These behaviours are achieved by setting

Neighborhood	N	Increase in SNR for flat regions
3x3	9	+ 9.5 dB
5x5	25	+ 14.0 dB
7x7	49	+ 16.9 dB
9x9	81	+ 19.1 dB
11x11	121	+ 20.8 dB

Table 2.3: SNR increase for flat regions achieved by linear smoothing

$$\beta = \max\left(\frac{\sigma_x^2 - \sigma_n^2}{\sigma_x^2}, 0\right) \tag{2.42}$$

A useful side-effect of this choice for β is that it makes Lee's local statistics filter for additive noise a special case of the hybrid grid filter: choosing **f =** 0 recovers the Lee filter exactly. This implies that hybrid grid filters will be ***at least*** as good as Lee's filter for additive noise.

How large a window should be used for F_2?

Consider the case of zero-mean, additive white Gaussian noise (AWGN). Assume that the smoothing filter F_2 takes a simple average of N pixels. In flat regions, this will result in residual variance with an average of:

$$\sigma^2_{\mathbf{residual}} = \frac{\sigma_n^2}{N} \tag{2.43}$$

After smoothing, the signal-to-noise ratio for flat regions (in decibels) is:

$$\begin{aligned} \mathrm{SNR}_{\mathrm{smoothed}} &= 10log_{10}\left(\frac{P_s}{\sigma_n^2}N\right) \\ &= 10log_{10}\left(\frac{P_s}{\sigma_n^2}\right) + 10log_{10}N \\ &= \mathrm{SNR}_{\mathrm{noisy}} + 10log_{10}N \end{aligned} \tag{2.44}$$

where P_s is the signal power, and $\mathrm{SNR}_{\mathrm{noisy}}$ is the signal-to-noise ratio for flat regions in the noisy image. The (decibel) increase in SNR is independent of the amount of noise in the image.

Table 2.3 shows the SNR increase provided by windows of various sizes. Although larger windows mean a greater noise reduction in flat regions, there is a

σ_n^2	N for 30 dB PSNR in flat regions	Recommended window size
100	6	3x3
200	12	5x5
400	24	5x5
800	48	7x7
1600	98	11x11
3200	195	13x13

Table 2.4: Recommended window sizes for $+30\,\mathrm{dB}$ gain in flat regions, assuming $P_s = 128^2$

tradeoff: using too large a window forces the grid filter to do more smoothing, which may decrease its effectiveness at filtering details. The safest approach is to use a window which is just large enough to adequately smooth flat regions.

Solving (2.44) for N, and assuming a desired SNR of 30 dB in flat regions, the proper window size is given by:

$$N \approx 1000\frac{\sigma_n^2}{P_s} \tag{2.45}$$

Table 2.4 lists recommended window sizes based on this equation.

Chapter 3

Results

3.1 Training data sets

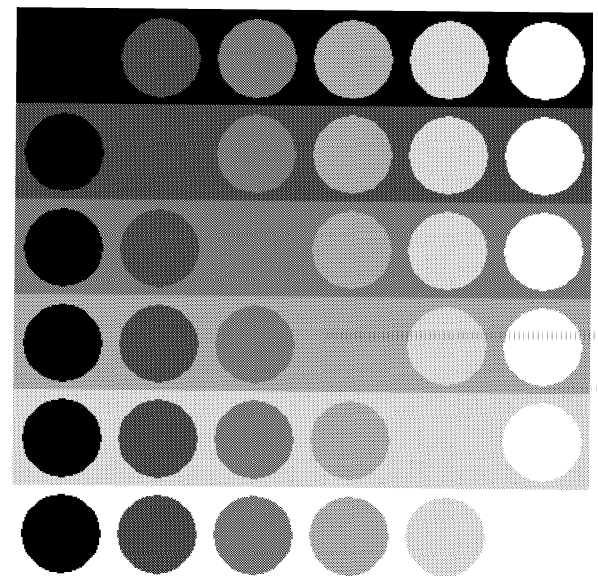
This section describes the images sets used for training and testing the filters.

3.1.1 Synthetic images

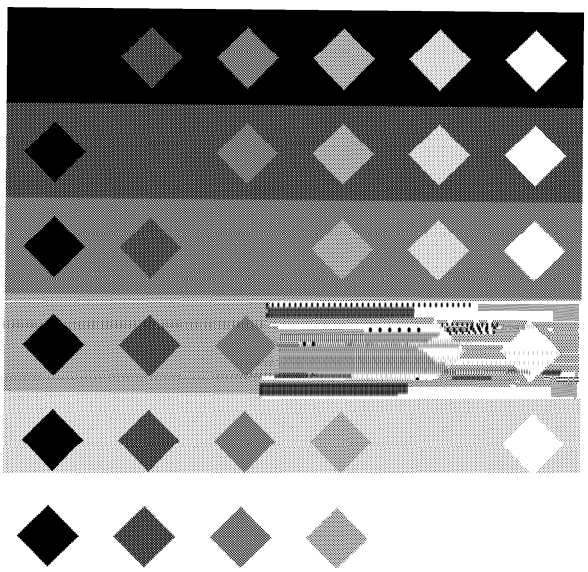
Figure 3.1 shows the synthetic image set. The circles image (upper left) contains solid circles with varying foreground and background intensity. This image is useful because it provides edges of varying orientations and contrasts. The diamonds and squares images are similar, but provide diagonal, horizontal and vertical edges of varying contrast. The lines image contains lines of various orientations. The lines are on a grey background, and vary linearly in intensity from black in the middle to white at the perimeter of the circle. The synthetic images are 8 bit greyscale.

3.1.2 Document images

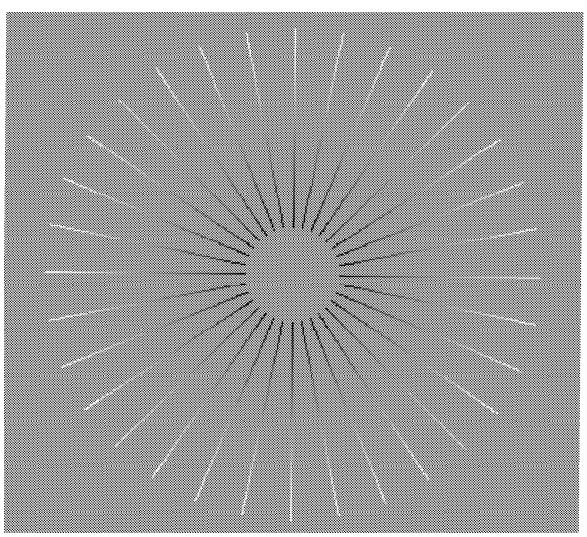
The document image set contains seven images of typefaces. Four of these (for the fonts Helvetica, Times-Roman, Courier, and Palatino-Roman) are shown in Figure 3.2. An additional three images were created by sampling the typefaces Helvetica, Times-Roman and Courier at twice the resolution. The images are 8 bit greyscale, with white (255) background and black (0) text. Table 3.1 summarizes the image sizes.



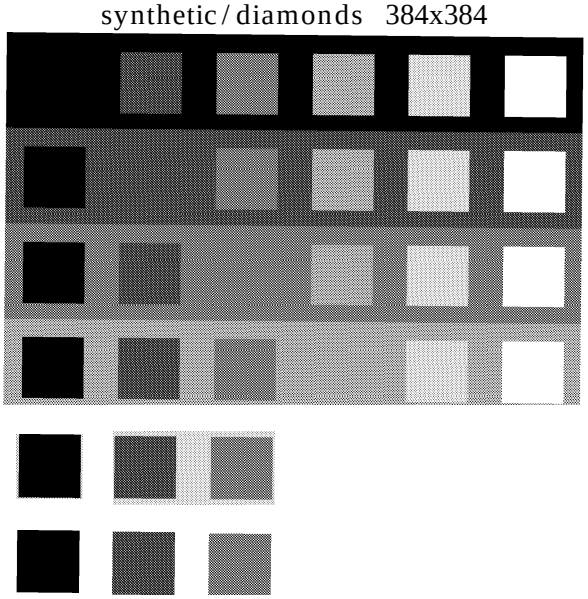
synthetic/circles 384x384



synthetic/diamonds 384x384



synthetic/lines 384x384



synthetic/squares 384x384

Figure 3.1: Synthetic images

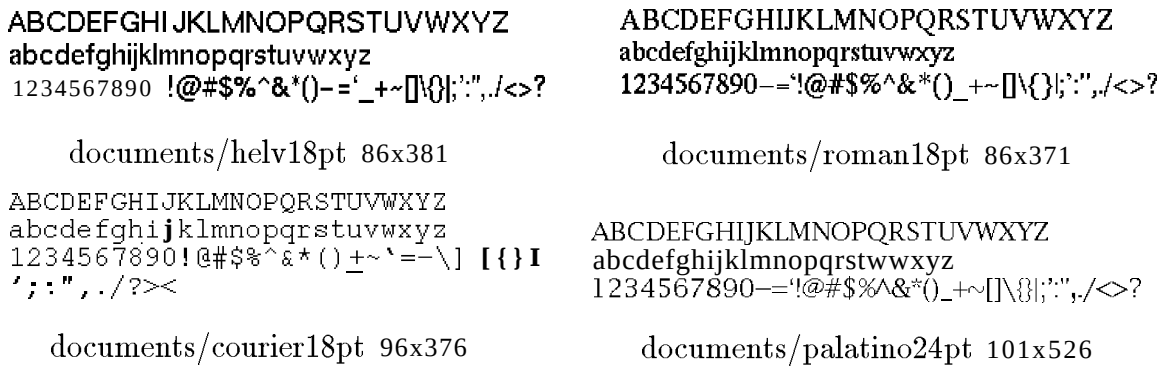


Figure 3.2: Document images

Image name	Size
documents/helv18pt	86x381
documents/helv18ptX2	151x741
documents/roman18pt	86x371
documents/roman18ptX2	151x721
documents/courier18pt	96x376
documents/courier18ptX2	171x731
documents/palatino24pt	101x526

Table 3.1: Document images

3.1.3 Face images

The face image set contains three images of faces. The images were acquired using a 35 mm camera, developing to film and digitized using a good quality flatbed scanner.

3.2 Which footprints are best for additive noise?

As discussed earlier, removing substantial amounts of noise from images requires large footprints. Unfortunately, memory limitations have so far prevented the creation of grid filters with more than 14 dimensions. Two techniques have been described for achieving large footprints without increasing the dimensionality of the grid substantially: foveated footprints (2.2) and hybrid filters (2.8).

To determine which of these approaches are better for removing additive noise, a variety of filters were trained on images of text with σ^2 = 400 additive white Gaussian noise. Filters based on all of the footprints described in (2.2) were trained. Hybrid filters based on the simple3x3 and simple13pt footprints, and using 5x5 linear averagers for F_2 were also trained.

The training data set consisted of images of Courier, Times-Roman and Helvetica text at 18 point and 36 point size. The filters were tested on images of Palatino text at 24 point size.

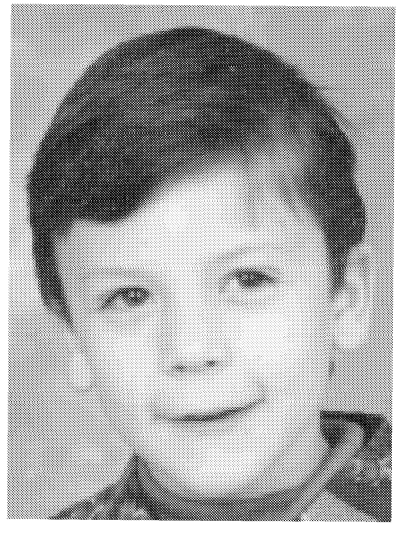
Each filter was trained on a variety of grid sizes from 2^N to 8^N. For many of these filters, there was insufficient memory available to construct the least-squares equations.

Of the foveated footprints, only the footprints fovea5x5, fovea5x5b, and fovea7x7 were successfully trained for a large grid size. The other foveated footprints resulted in too many dimensions for the grid; in some cases, it was not even possible to create a 2^N grid for the filter.

Figure 3.4 shows results for the four best filters. Over the range of grid sizes, the hybrid filter based on the simple3x3 footprint was superior to the simple3x3, fovea5x5 and fovea5x5b filters.

Within current memory restrictions, it appears that hybrid grid filters are superior for removing substantial amounts of additive noise. However, it turns out that for undoing blurring, foveated grid filters are better than hybrid filters; t hese results are described in a later section.

faces/ann-mallory 212x228



faces/christopher 178x132

faces/elise 217x163

Figure 3.3: Face images

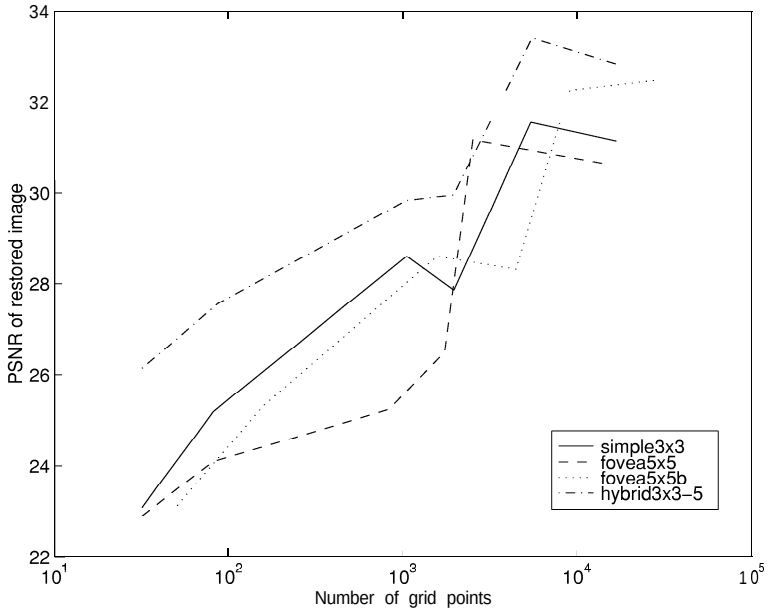


Figure 3.4: PSNR results for selected filters trained to remove σ^2 = 400 AWGN from text images

3.3 Synthetic images with additive noise

Synthetic images allow the behaviour of filters to be studied for typical "building blocks" of images, such as lines and edges. This section compares the performance of a hybrid grid filter to Lee and Wiener filters, for edges and lines under additive noise. The hybrid filter used a 3x3 grid filter for F_1 and a 7x7 linear average for F_2. The grid size used was 12^8. The filter was trained on all the synthetic test images (Figure 3.1). Each test image was repeated 8 times, each time with a different noise field, for a total of 4.6 million training samples. While this was an extreme amount of training data, it excluded the possibility of artifacts or poor results due to undertraining.

Four filters were trained for additive white Gaussian noise with σ^2 = 100, 200, 400 and 800. Training times were 26 to 77 minutes (including CG iteration time). Filtering rates were 3600 pels/s to 6200 pels/s. The filters are summarized by Table 3.2.

Table 3.3 shows MSE results for the circles image. The hybrid grid filter performed substantially better than both the Wiener and Lee filters for all noise levels. The Lee filter results shown are for the best window size, which was 5x5 for all noise amounts. Figure 3.5 plots output versus input PSNR for the four filters. The hybrid

Noise Variance	Grid points	Matrix entries	Training time (minutes)	Filtering rate (pels/s)
100	6987	178437	25:49	6170
200	10925	307987	32: 10	5771
400	21247	638769	46:40	5101
800	48540	1497491	76:55	3665

Table 3.2: Summary of filters for removing AWGN from synthetic images

Noise Variance	MSE		
	Grid filter	Wiener filter	Best Lee filter
100	83	67.6	24.6 (5x5)
200	15'9	106.7	44.1 (5x5)
400	31'3	156.8	77.0 (5x5)
800	60'7.	217.5	131.5 (5x5)

Table 3.3: Results for the synthetic circles image with AWGN

grid filter was able to consistently increase the PSNR by $+11\,\mathrm{dB}$. This corresponds to decreasing the noise power by a factor of 12-13.

Figures 3.6 and 3.7 show excerpts of the circles restoration for σ^2 = 100 and $\sigma^2 = 800$, respectively. In these excerpts, the circle edge in the upper right corner is high contrast. The circle edge in the lower left corner is low contrast. The upper left and lower right circle edges are medium contrast. The residual noise left by the Lee filter around edges is apparent in both figures. The hybrid grid filter does well on the high and medium contrast edges, but the low contrast edge in Figure 3.7 is smudged. This is because the noise variance is quite large compared to the edge strength; the grid filter is unable to localize the edge accurately. In Figure 3.6, the Wiener filter leaves lots of noise behind; in Figure 3.7, it smoothes the noise at the expense of blurring the edges.

Table 3.3 shows MSE results for the lines image. Again, the grid filter was much better than both the Wiener and Lee filters for all noise levels. These results are summarized by Figure 3.8, which plots the output versus input PSNR for the filters.

Figures 3.9 and 3.10 show excerpts of the lines restoration for σ^2 = 100 and $\sigma^2 = 800$, respectively. The grid filter successfully smoothes around the lines, although in Figure 3.10 some noise remains. Note that portions of the lines are missing in the grid filter result of Figure 3.10. This is likely because there is insufficient contrast to distinguish the lines from noise within a 3x3 window. A

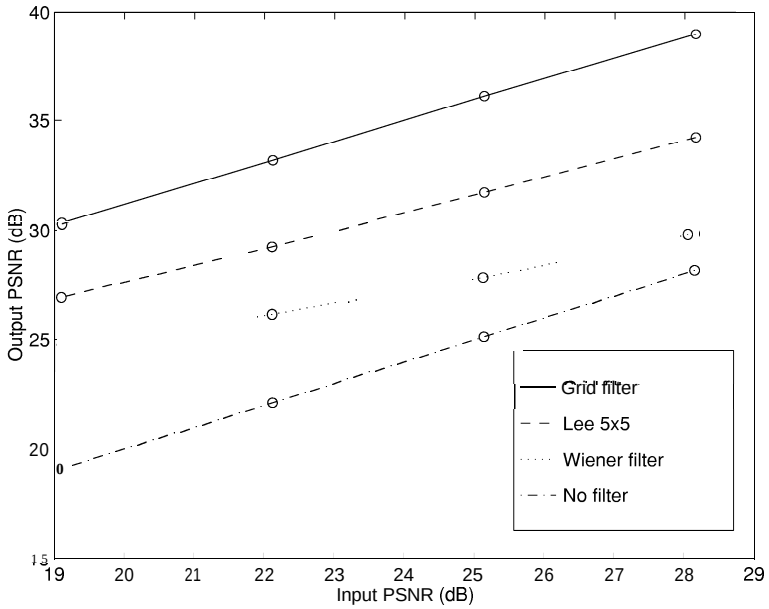
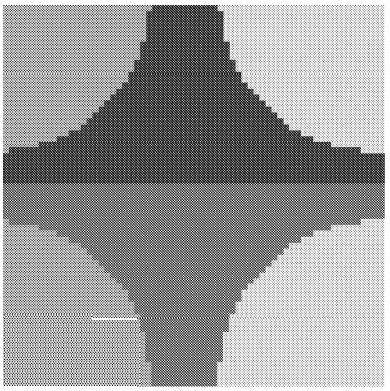


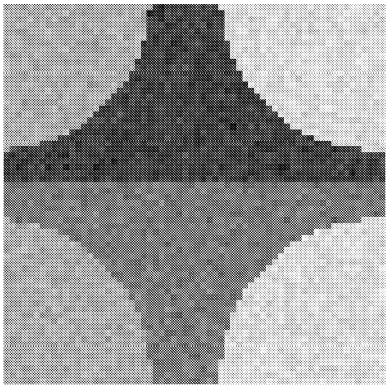
Figure 3.5: Input/Output PSNR plot for the synthetic circles image with AWGN

Noise Variance	MSE		
	Grid filter	Wiener filter	Best Lee filter
100	95	55.6	19.7 (5x5)
200	17'0	77.6	34.5 (7x7)
400	30'7	98.6	56.6 (7x7)
800	60'5.	115.6	88.1 (9x9)

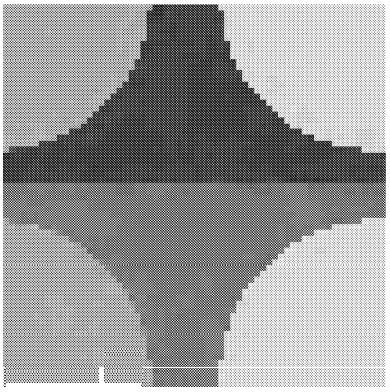
Table 3.4: Results for the synthetic lines image with AWGN



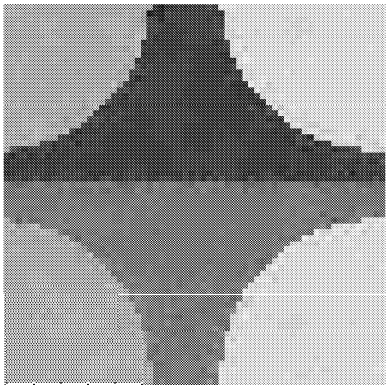
Original



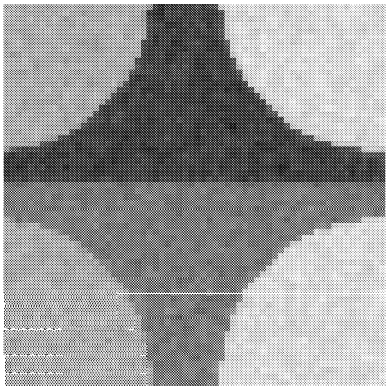
AWGN $\sigma^2 = 100$
MSE=99.5 PSNR=28.2 dB



Grid filter
MSE=8.3 PSNR=39.0 dB

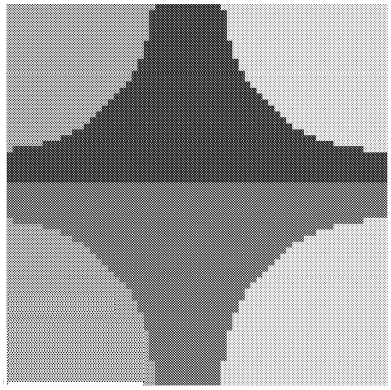


5x5 Lee filter
MSE=24.6 PSNR=34.2 dB

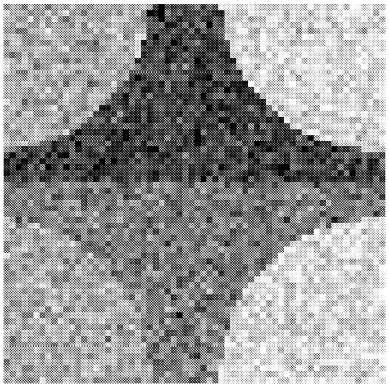


Wiener filter
MSE=67.6 PSNR=29.8 dB

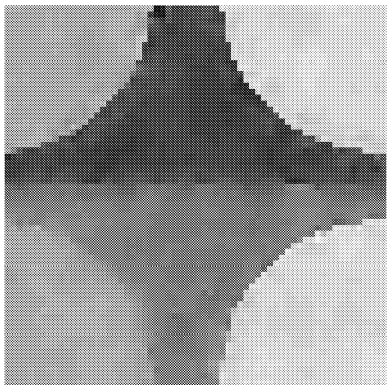
Figure 3.6: Results for the synthetic circles image with AWGN $\sigma^2 = 100$



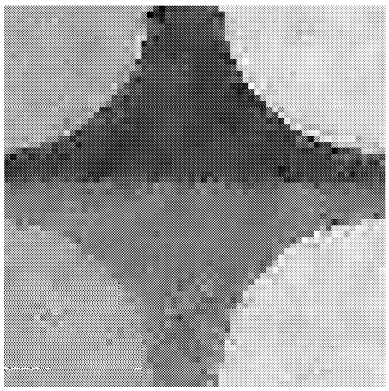
Original



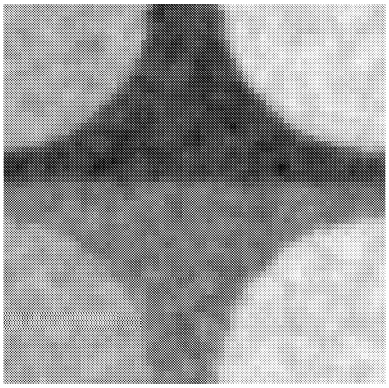
AWGN σ^2 = 800
MSE=799.3 PSNR=19.1 dB



Grid filter
MSE=60.7 PSNR=30.3 dB



5x5 Lee filter
MSE=131.5 PSNR=26.9 dB



Wiener filter
MSE=217.5 PSNR=24.8 dB

Figure 3.7: Results for the synthetic circles image with AWGN σ^2 = 800

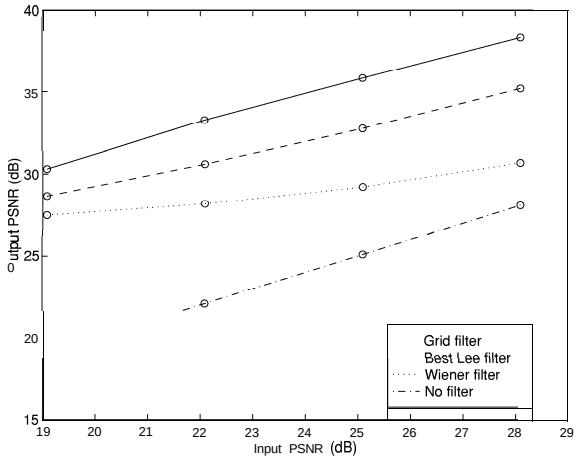


Figure 3.8: Input/Output PSNR plot for the synthetic lines image with AWGN

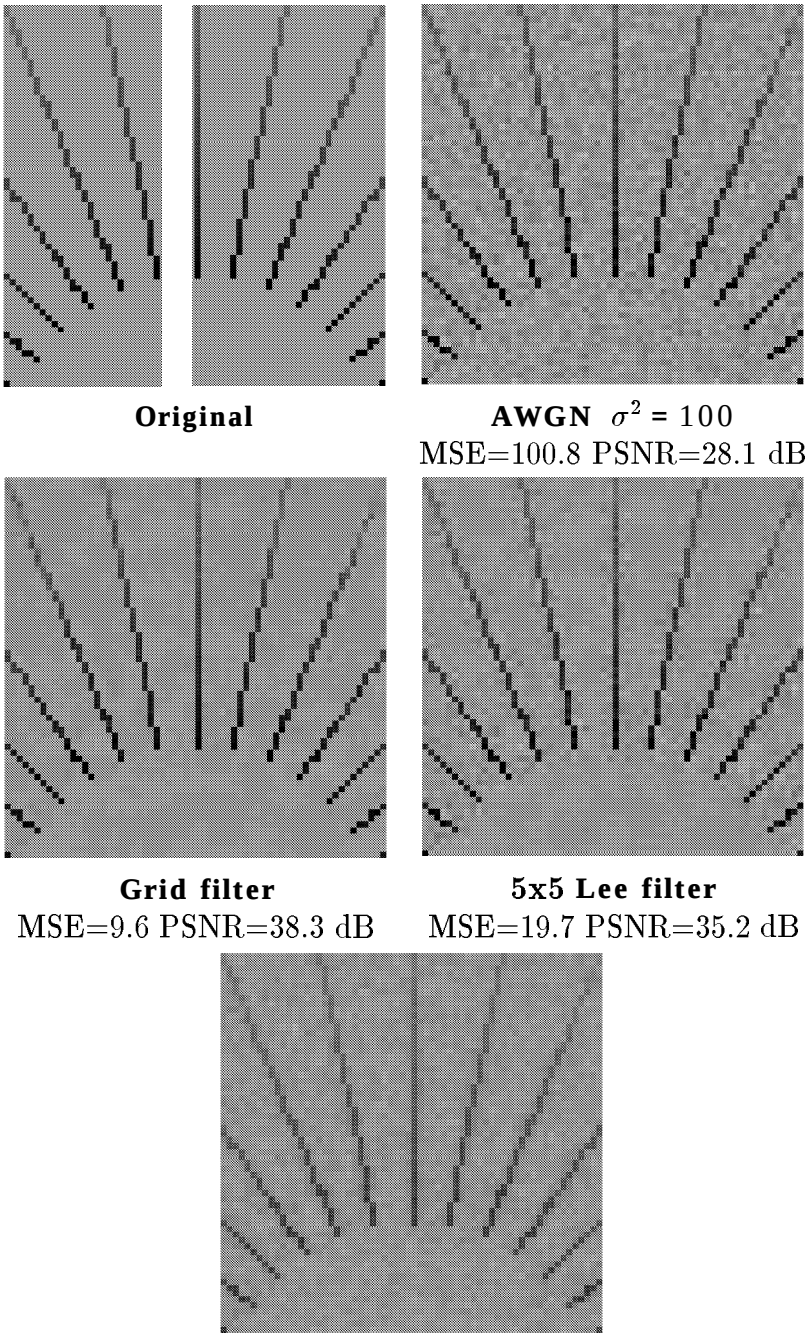
larger footprint (e.g. $\mathrm{simple13pt}$) might solve this problem. The Lee filter leaves substantial noise surrounding the lines. The Wiener filter smoothes out the noise in flat regions at the expense of the lines; this is most apparent in Figure 3.10.

3.4 Text with additive noise

One of the properties of a grid filter is that the filtering rate is (theoretically) independent of the number of grid coefficients. To illustrate this property, a hybrid filter with a 3x3 grid filter for F_1 and 7x7 linear average for F_2 was trained on images of text using various grid sizes. The text images were synthetically generated, and contained samples of Helvetica, Courier and Times-Roman at 18 and 36 point resolution (Figure 3.2). Filters using grid sizes from 2^8 up to 8^8 were trained. The degradation model was Additive White Gaussian Noise with σ^2 = 400.

The filters were tested on a synthetically generated sample of Palatino font at 24 point resolution. Note that both the typeface and size were different from the training data. This testing data verifies that the filter does not just memorize those particular fonts, and is able to generalize to a different type face.

Table 3.5 summarizes the training and testing results for the filters. Note that the training rates exclude CG solution time. Both training rates (excluding CG iteration time) and testing rates are in theory independent of the number of grid points, but in practice cache effects causes a gradual drop-off. This is illustrated in Figure 3.11. Note that a neural network or polynomial has training and filtering



Original

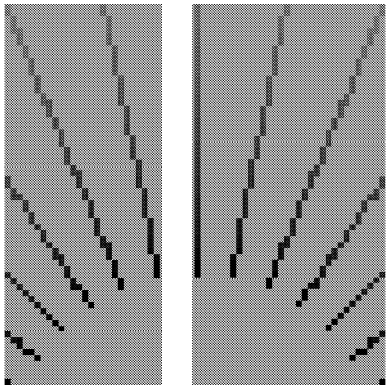
AWGN $\sigma^2 = 100$
MSE=100.8 PSNR=28.1 dB

Grid filter
MSE=9.6 PSNR=38.3 dB

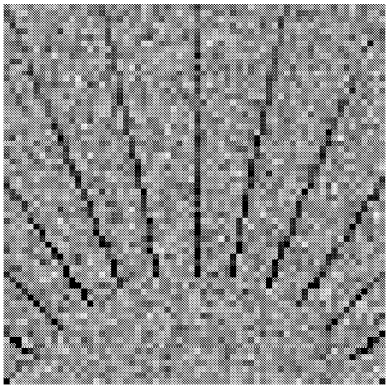
5x5 Lee filter
MSE=19.7 PSNR=35.2 dB

Wiener filter
MSE=25.8 PSNR=34.0 dB

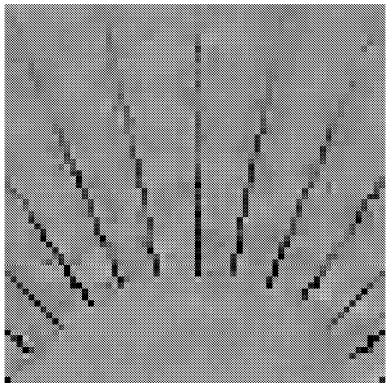
Figure 3.9: Results for the synthetic lines image with AWGN $\sigma^2 = 100$



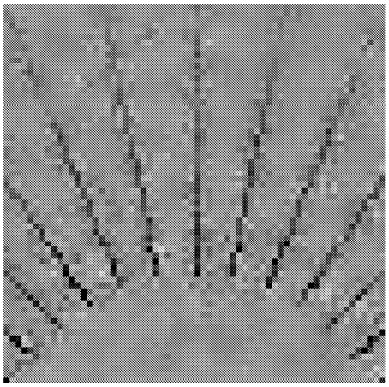
Original



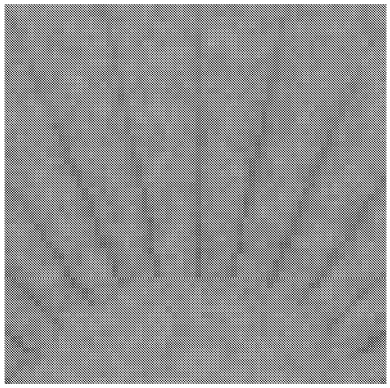
AWGN σ^2 = 800
MSE=803.7 PSNR=19.1 dB



Grid filter
MSE=60.5 PSNR=30.3 dB



9x9 Lee filter
MSE=88.1 PSNR=28.7 dB



Wiener filter
MSE=115.6 PSNR=27.5 dB

Figure 3.10: Results for the synthetic lines image with AWGN σ^2 = 800

Grid extent	Grid points	Matrix entries	Training rate (pels/s)	Testing rate (pels/s)	MSE	PSNR (dB)
2	32	336	4229	6793	186.4	25.4
3	93	2210	3728	6575	124.6	27.2
4	1076	53703	3541	6149	73.7	29.4
5	2052	133171	3353	6051	70.3	29.8
6	8968	240866	3733	5982	28.4	33.6
7	20545	748607	2785	5432	30.9	33.4
8	22922	601530	3259	5870	27.5	33.9

Table 3.5: Training and testing results for text degraded by AWGN $\sigma^2 = 400$

rates which are inversely proportional to the number of coefficients. If plotted on Figure 3.11, the rates of these filters would behave as $O(N^{-1})$, with N is the number of coefficients.

Figure 3.12 shows excerpts from the text image restored by filters with varying grid extents. The change in quality of the restoration result is obvious as the size of the grid is increased. It is clear from these results that the filter is able to generalize from the training data to typefaces not seen in training.

Figure 3.13 shows how the PSNR of the restored Palatino text image changed as a function of grid extent. For grids larger than 2^8, the restoration result was superior to both the best Lee filter (3x3) and the Wiener filter. The similarity of PSNR for the last three filters suggests that the results are as good as can be obtained.

Figure 3.14 shows the SNR as a function of spatial frequency for various grid extents. In the higher frequency bands, the hybrid grid filter is able to boost the SNR by up to $+25\,\mathrm{dB}$. This is possible because of the tightly constrained statistical properties of text images. The grid filter acquires extensive prior knowledge of what text ought to look like during training; it is able to apply these priors to excellent effect when filtering.

Figure 3.15 shows how MSE is distributed over spatial frequency bands in the noisy image (top) and the image restored by the 8^8 hybrid grid filter. What error remains is concentrated around $\lambda^{-1} \approx 0.1$. One explanation for this is that the grid filter is able to exploit its knowledge of the priors only on the scale of a 3x3 window. This results in excellent noise reduction in the high frequency bands. However, it is not able to apply knowledge of priors for features on the scale of $\lambda^{-1} \approx 0.1$. Another possible explanation is that the residual noise can be attributed to the fact that the image is very similar to the training data on the small scale. At the scale of $\lambda^{-1} \approx 0.1$, the test image is quite different from the training data, because

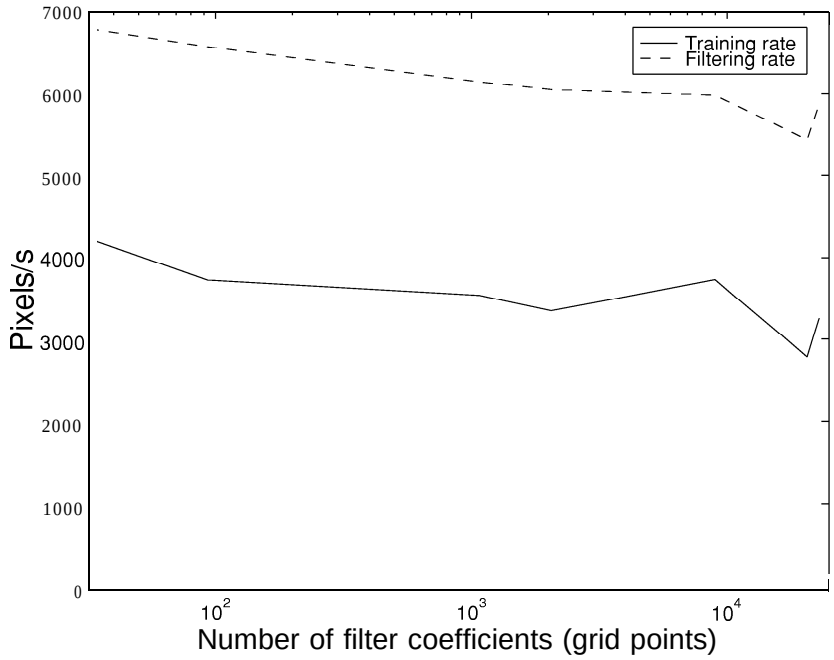


Figure 3.11: Training and filtering rates as a function of grid extent

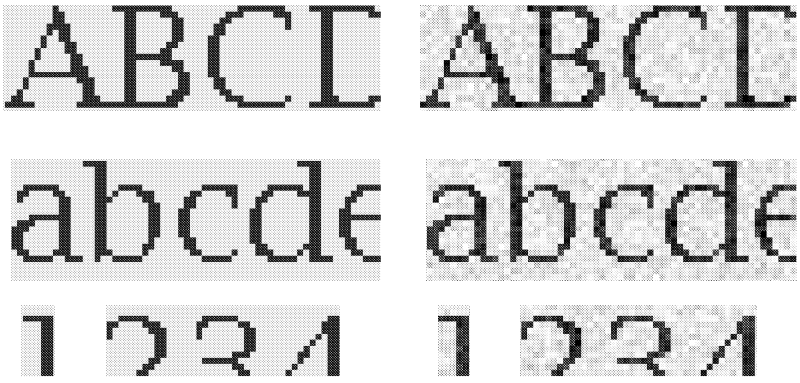
the text is 24 point (whereas the training data consisted of 18 and 36 pt text).

3.5 Faces with additive noise

Images of faces are harder to restore than text or synthetic images, because they contain gentle gradients and large scale features. To test grid filters on images of faces, a hybrid filter with a 8^{11} grid filter for F_1 and a 5x5 linear averager for F_2 was trained on images of faces degraded by AWGN. The training set consisted of 8 repetitions of the images Christopher and ann-mallory (Figure 3.3). Each repetition used a different noise field, for a total of 550912 training samples. Four filters were trained, using noise variances of $\sigma^2 =$ 100, 200, 400 and 800. Table 3.6 summarizes the training results.

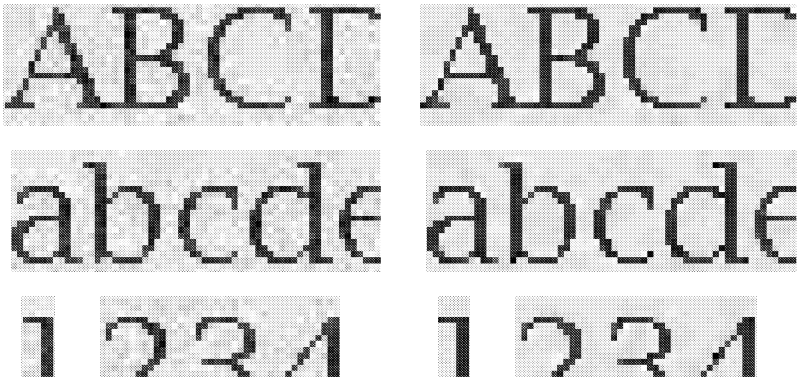
The filters were tested on the elise image (Figure 3.3). Table 3.5 summarizes the results. The grid filter outperformed the Wiener filter and the best Lee filter. Figure 3.16 plots input and output PSNR based on the data of Table 3.5.

Figures 3.17 and 3.18 show extracts of the restoration results for $\sigma^2 = 100$ and $\sigma^2 = 800$, respectively. Occasional bright pixels are noticeable in the grid filter results. These are artifacts due to missing grid points. A larger training data set



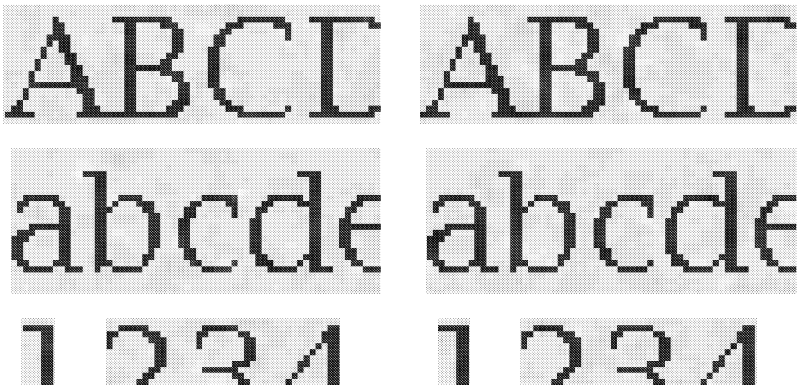
Original

Noisy: AWGN σ^2 = 400
MSE=400, PSNR=22.2



Grid size 2^8
MSE=186.4, PSNR=25.4

Grid size 4^8
MSE=73.7, PSNR=29.5



Grid size 6^8
MSE=28.4, PSNR=33.6

Grid size 8^8
MSE=27.5, PSNR=33.8

Figure 3.12: Restoration results for 24 pt Palatino text using varying grid extents

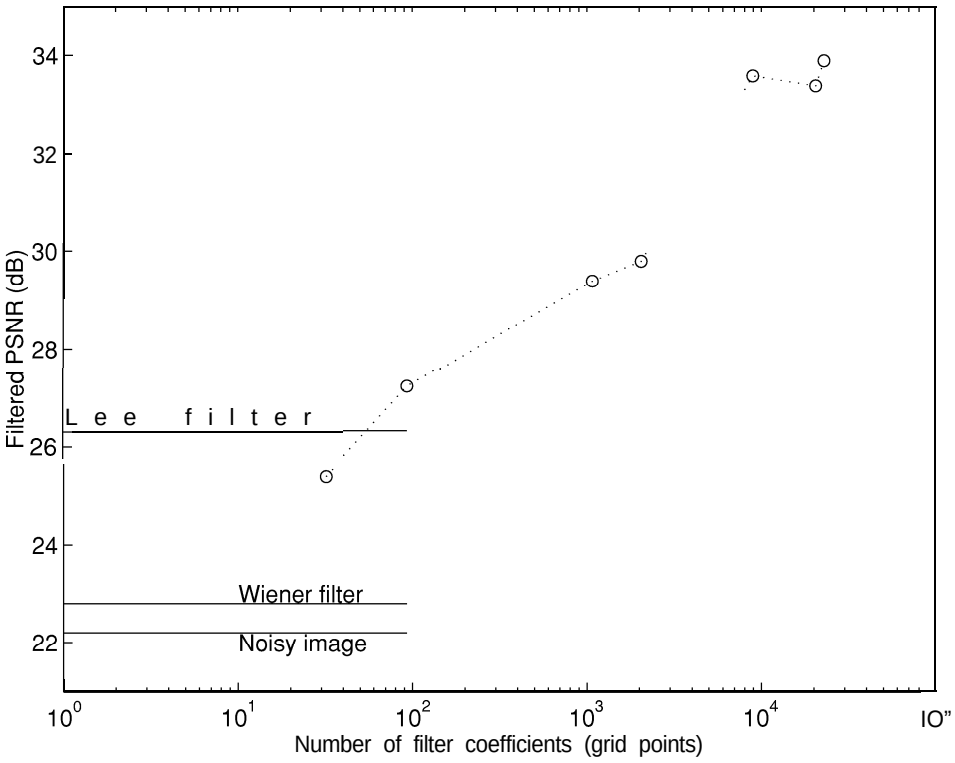


Figure 3.13: Restoration quality for 24 pt Palatino text using varying grid extents

Noise Variance	Grid points	Matrix entries	Training time (minutes)	Filtering rate (pels/s)
100	4471	123340	4:22	6690
200	5770	163480	5:06	6450
400	8556	248314	6:21	6010
800	15170	460491	8:15	5170

Table 3.6: Summary of filters for removing AWGN from face images

Noise Variance	MSE		
	Grid filter	Wiener filter	Best Lee filter
100	36.4	58.4	45.2 (3x3)
200	56.3	93.5	76.3 (3x3)
400	88.4	142.4	129.3 (3x3)
800	139.2	208.8	208.2 (5x5)

Table 3.7: Results for the elise image with AWGN

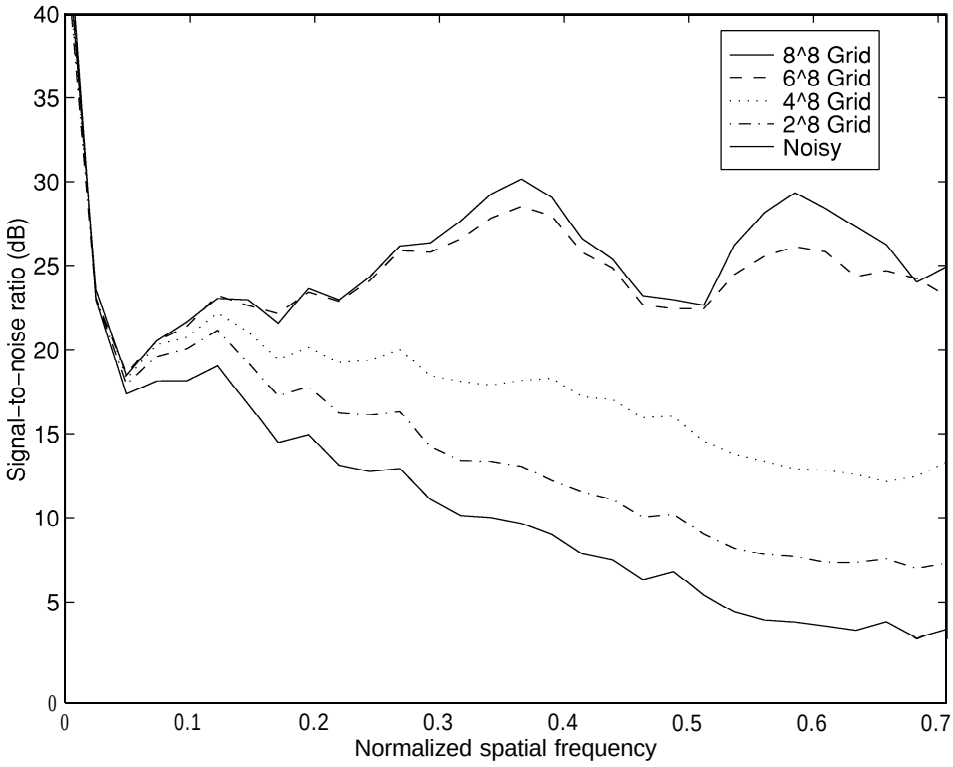


Figure 3.14: Signal to noise ratio

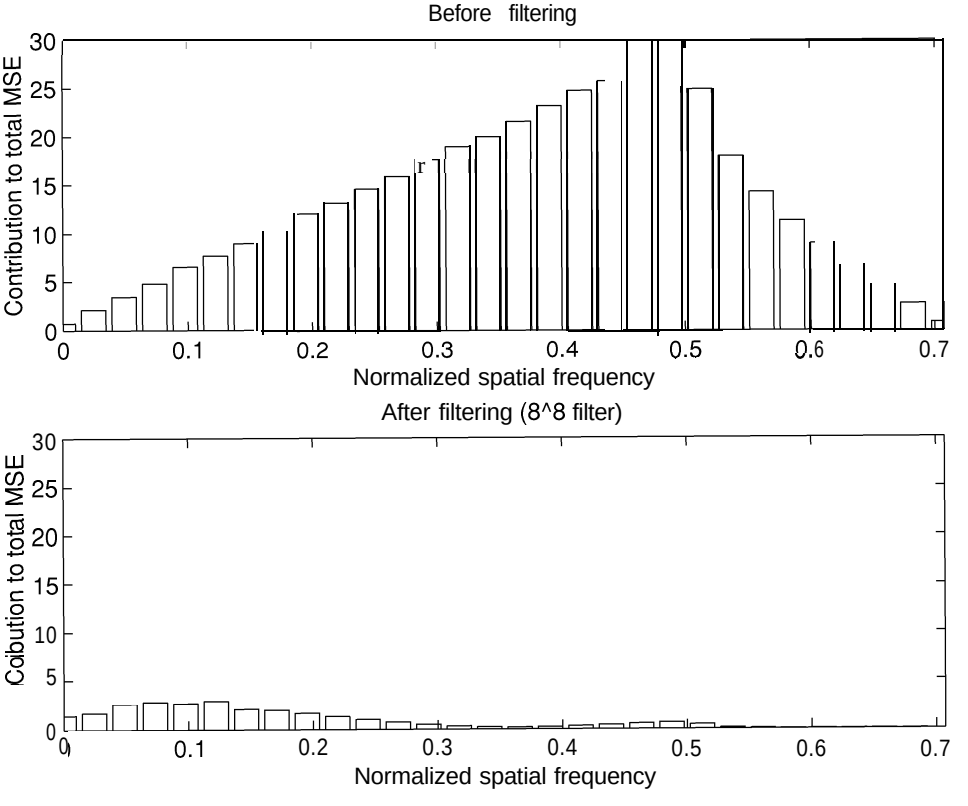


Figure 3.15: How various frequency bands contribute to total MSE

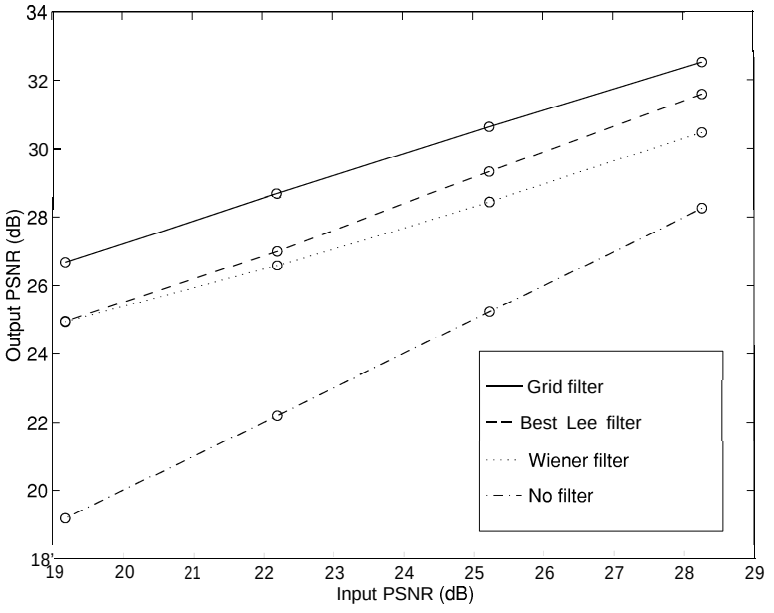


Figure 3.16: Input/Output PSNR plot for the faces/elise image with AWGN

would likely avoid this problem. The residual noise left by the Lee filter is quite distracting, especially in the σ^2 = 800 case (Figure 3.18). The Wiener filter leaves the skin with a blotchy appearance.

3.6 Response to unusual features

One of the benefits of grid filters is that they tend to pass unusual inputs (outliers) unchanged. This is in contrast to polynomials and neural networks, which can have unpredictable (and possibly undesirable) responses to unusual inputs.

To illustrate this property, a grid filter with footprint simple13pt (Figure 2.2) was trained on images of circles (Figure 3.1) degraded by additive white Gaussian noise with σ^2 = 100. The filter was then tested on an image of lines of varying contrast (Figure 3.1). The training set (circle image) contained flat regions, but no lines.

The grid filter had 37433 coefficients, and filtered at a rate of 1260 pixels/s. When tested on the circle image, the filter increased the PSNR from 28.2 to 37.8 dB. Figure 3.19 shows an excerpt from this image for low-contrast circles. The smudging of the lower-left circle in the filtered image of Figure 3.19 illustrates a

Original

AWGN $\sigma^2 = 100$
MSE=97.2 PSNR=28.3 dB

Grid filter
MSE=36.4 PSNR=32.5 dB

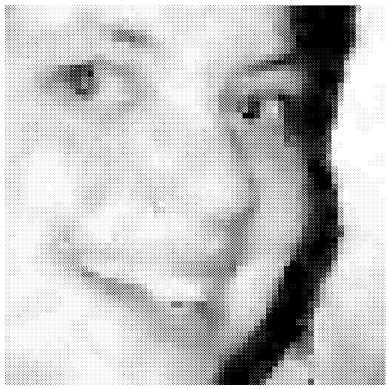
3x3 Lee filter
MSE=45.2 PSNR=31.6 dB

Wiener filter
MSE=58.4 PSNR=30.5 dB

Figure 3.17: Results for the elise image with AWGN $\sigma^2 = 100$

Original

AWGN σ^2 = 800
MSE=785.2 PSNR=19.2 dB



Grid filter
MSE=139.2 PSNR=26.7 dB

5x5 Lee filter
MSE=208.2 PSNR=25.0 dB

Wiener filter
MSE=208.8 PSNR=25.0 dB

Figure 3.18: Results for the elise image with AWGN σ^2 = 800

consequence of using Mean-Squared Error (MSE) as the training criterion: where the signal is ambiguous, the filter tends to average over possibilities. In the smudged regions, the noise makes it difficult for the filter to determine exactly where the edge should be, so the possibilities are averaged together. It is debatable whether this behaviour is desirable or not. Maximum-likelihood filters, in contrast, would boldly choose one of the possibilities. This would result in a crisp (but possibly incorrect) location for the edge.

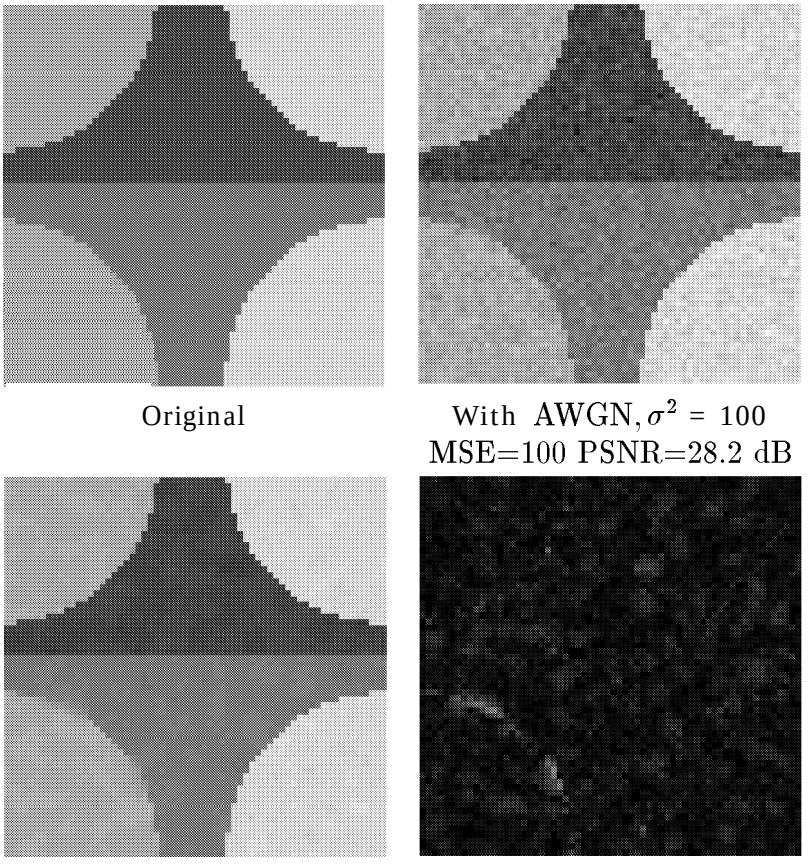
When the filter was tested on the lines image, the PSNR increased from 28.1 to 33.9 dB (Figure 3.20). The lines, which were unlike anything the filter had seen in the training set, were passed through mostly unchanged. Some noise surrounding the lines was amplified.

Passing outliers unchanged is very important for some applications. The danger is that when a filter encounters something unexpected, it will either react unpredictably, or erase the unusual feature by trying to make it look more like the training data. For example, a filter trained on images of spiral galaxies might try to make everything look like a spiral galaxy; when it saw something highly unusual (and for an astronomer, perhaps significant), the filter might erase (or obscure) the unusual feature. Grid filters avoid this problem.

Another interesting ability of grid filters is that they can detect unusual features. During training, points are added to the grid as required. If a missing grid point is encountered during filtering, this suggests that the local neighborhood contains an unusual feature unlike anything seen in training. Figure 3.21 shows the result of flagging unusual features in the lines image. The high-contrast portions of the lines are unlike anything seen in training, and are flagged.

Note that this "unusual feature detector" only works for features which are unusual on the scale of the filter footprint. Large-scale unusual features which are comprised of small, typical features would not be detected.

The count of missing grid points serves as a useful indicator. When a filter is applied to an image typical of its training set, none or very few missing grid points are encountered. If the number of missing grid points is large, this indicates that the training data was inadequate to handle the image being filtered. For example, when the "circles" filter was applied to the circles image, 3345 missing grid points (about 0.17%) were encountered. When it was applied to the "lines" image, the number of missing grid points jumped to 44514 (about 2.3%), indicating that the image was not typical of the training set.



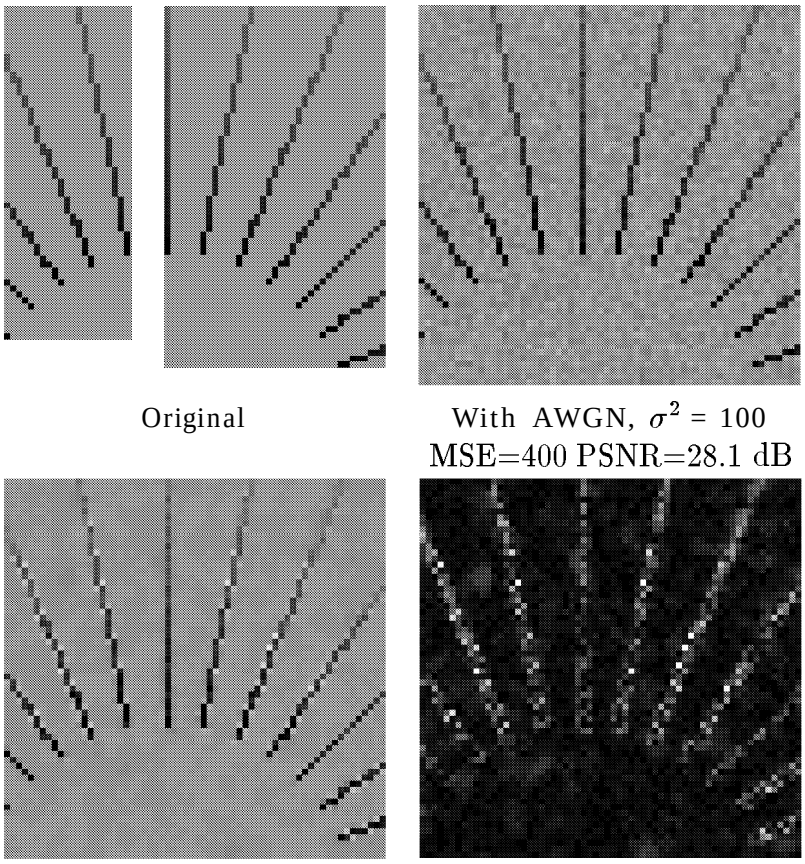
Original

With AWGN, σ^2 = 100
MSE=100 PSNR=28.2 dB

Filtered
MSE=10.9 PSNR=37.8 dB

Difference image

Figure 3.19: Result for simple13pt filter trained and tested on synthetic/circles with AWGN, σ^2 = 100



Original

With AWGN, σ^2 = 100
MSE=400 PSNR=28.1 dB

Filtered
MSE=26.6 PSNR=33.9 dB

Difference image

Figure 3.20: Result for simple13pt filter trained on synthetic/circles with AWGN σ^2 = 100 but tested on synthetic/lines

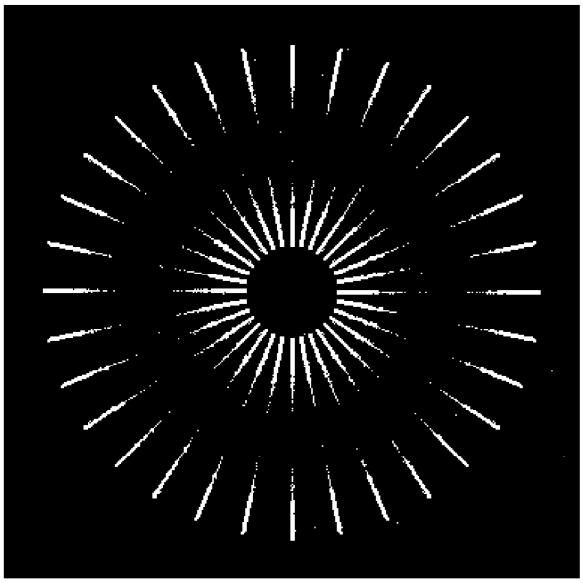


Figure 3.21: Neighborhoods which the grid filter marked as containing unusual features

3.7 How much training is necessary?

An important question about grid filters is the quantity of training required. Based on experience gained so far, the following observations seem to be true:

- The training set must be large enough to be representative of the image class.
- For little or moderate noise, a single presentation of the training images is sufficient. For extreme amounts of noise, it is sometimes necessary to present the training set several times, each time with a different noise pattern.
- The amount of training data required increases rapidly if the dimensionality of the grid is increased. For example, filters based on the footprint simple13pt (12 dimensions) require more training data than comparable filters based on the foot print simple3x3 (8 dimensions).
- If additive noise is used, the number of grid points (and matrix entries) increases as a function of the number of training samples. If trained using a very large number of training samples, eventually every possible point in the grid would be used, since the noise distributions used are all long-tailed. However, only a small number of grid points contribute appreciably to the restoration

	Training			Testing		
Repetitions	Training Samples	Grid Points	Matrix Entries	Missing grid points	MSE	PSNR (dB)
1	3600	12830	172812	18.4%	111.9	27.6
2	7200	25471	299939	11.3%	98.9	28.2
3	10800	37833	408996	7.5%	87.5	28.7
4	14400	49532	511480	5.5%	91.4	28.5
5	18000	61354	618117	4.5%	87.1	28.7

Table 3.8: Results for simple13pt and σ^2=400

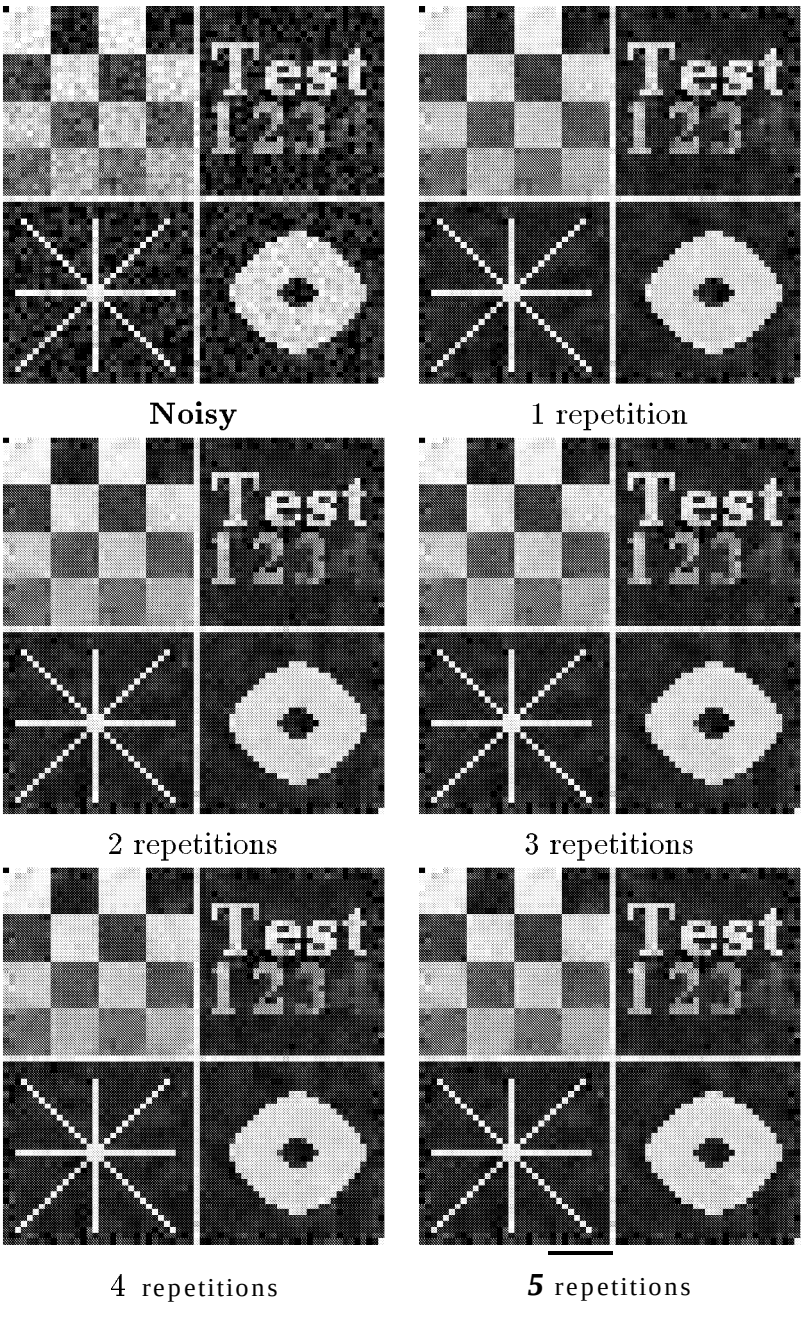
result. There is a definite advantage to using small, representative training sets, because they generate fewer grid points.

To illustrate these ideas quantitatively, grid filters were trained and tested on a synthetic test image (Figure 1.3) with varying amounts of AWGN (σ^2 = 400, σ^2 = 200 and σ^2=100). Two kinds of filters were tested: an 8^{12} grid filter using the simple13pt footprint, and a 8^8 hybrid grid filter. The test image size was 64x64 and contained 3600 training samples. The filters were trained using a variable number of repetitions of the test image. Each repetition had a different noise field. The filters were evaluated by testing on the same image used in training.

3.7.1 Results for the simple13pt filter

Table 3.8 summarizes the results for σ^2 = 400. Five filters were trained, using 1 to 5 repetitions of the image (with different noise fields). It was not possible to use more than 5 repetitions, because the size of the least-squares equations exceeded available memory. The filters were then tested on the same image. The missing grid points column shows the percentage of grid points which were required during testing but not present in the grid. Note that after approximately 10000 training samples, there was no increase in PSNR, but the number of grid points and matrix entries continued to grow. When additive noise is used, the number of grid points continues to grow as the training set is enlarged. If the noise distribution is long-tailed, eventually every possible grid point would be required. For this filter, the number of possible grid points is approximately 27 million. However, only a small number of grid points contribute appreciably to reducing the MSE.

To illustrate this point, the last filter in Table 3.8 was used to filter the training image many times using different noise fields. The frequency with which each grid point was used was tabulated, and used to approximate the probability of the


Noisy
1 repetition
2 repetitions
3 repetitions
4 repetitions
5 repetitions

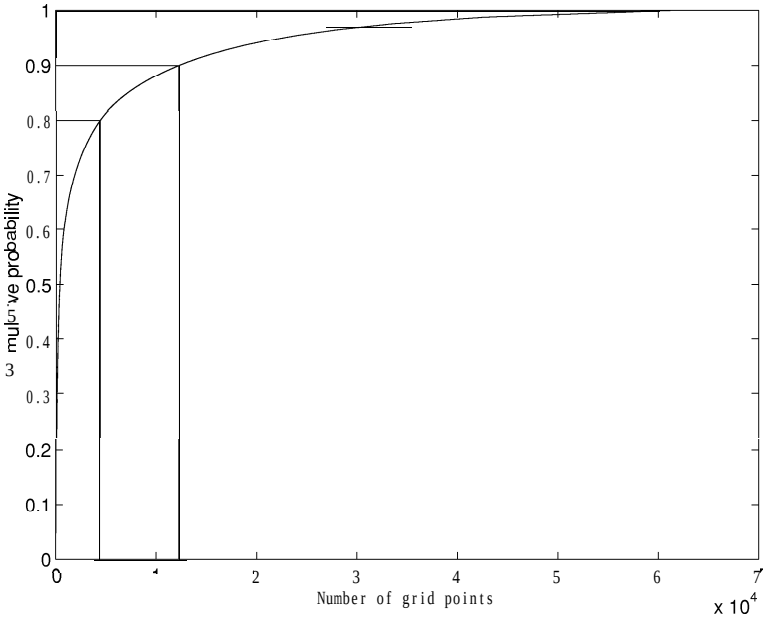


Figure 3.22: Cumulative Probability distribution of grid points

grid point being used. The grid points were then sorted in descending order of probability, and the cumulative sum was taken (Figure 3.22). From this graph, one can determine that 343 grid points account for 50% of the probability. About 4350 grid points account for 80% of the probability (Table 3.9).

From this data, it is apparent that the probability distribution of grid points is highly non-uniform. The data also suggest that it should be possible to delete large numbers of grid points without appreciably affecting the MSE of the filter. A two-pass training approach would be able to take advantage of this nonuniform distribution:

- During the first pass through the training data, grid points would be created as required. The frequency with which each grid point was used would be tabulated. However, the least-squares matrix would not be constructed – training data would be used only to determine the probability distribution of the grid points.

- Grid points would then be sorted according to frequency of occurrence, and only the first (say) 95% of the grid points (by cumulative frequency) would be kept. The remainder would be deleted.

Cumulative Probability	Number of Grid Points
05.	343
06.	749
07.	1818
08.	4350
0.95 09	22450 12250

Table 3.9: Key data points from Figure 3.22

	Training			Testing		
Repetitions	Training Samples	Grid Points	Matrix Entries	Missing grid points	MSE	PSNR (dB)
1	3600	11054	154219	14.9%	52.7	30.9
2	7200	21063	261487	8.8%	45.7	31.5
3	10800	30453	350191	5.8%	37.6	32.4
4	14400	39148	432797	4.0%	39.1	32.2
5	18000	47676	517295	3.2%	35.6	32.6
6	21600	55516	585352	2.6%	37.0	32.5

Table 3.10: Results for simple13pt and σ^2=200

- In the second pass of training, the least-squares matrix would be constructed. No new grid points would be added during this pass.

This approach would circumvent the problem illustrated in Table 3.8 in which the number of grid points grows as the size of the training data set is increased. This approach has not yet been implemented.

Tables 3.10 and 3.11 give results for AWGN with σ^2 = 200 and σ^2 = 100. Similar behaviour is apparent for these results: after about 10000 training samples, the PSNR is fairly stable, but the number of grid points and matrix entries continues to increase. Note that this number of training samples is quite small compared to the size of a typical training image, which contains hundreds of thousands of training samples.

3.7.2 Results for the Hybrid grid filter

The Hybrid filter used an 8^8 grid filter with footprint simple3x3 for F_1 and a 5x5 linear average for F_2. Unlike the simple13pt filter described in the previous section,

	Training			Testing		
Repetitions	Training Samples	Grid Points	Matrix Entries	Missing grid points	MSE	PSNR (dB)
1	3600	9819	142124	12.5%	28.6	33.6
2	7200	17985	233906	7.2%	23.5	34.4
3	10800	25743	312259	4.7%	19.4	35.2
4	14400	32686	378599	3.1%	19.7	35.2
5	18000	39408	447198	2.4%	18.3	35.5
6	21600	45463	503281	2.0%	19.2	35.3

Table 3.11: Results for simple13pt and σ^2=100

	Training			Testing		
Repetitions	Training Samples	Grid Points	Matrix Entries	Missing grid points	MSE	PSNR (dB)
1	3600	2666	34316	2.517%	201.7	25.1
2	7200	3642	53938	1.590%	182.3	25.5
3	10800	4282	69310	1.175%	173.9	25.7
4	14400	4767	81551	0.897%	169.1	25.9
5	18000	5152	92119	0.774%	167.9	25.9
10	36000	6485	133850	0.423%	151.2	26.3
100	360000	11621	371019	0.019%	134.8	26.8
1000	3600000	17812	791986	0.010%	132.3	26.9

Table 3.12: Results for simple3x3 and σ^2=800

which used a 12-dimensional grid, this filter used an 8-dimensional grid. The smaller number of dimensions allowed the use of very large training sets (up to 3.6 million training samples) and a noise variance of σ^2 = 800.

Tables 3.12 and 3.13 show results for AWGN with σ^2 = 800 and σ^2 = 100, respectively (results for σ^2 = 200 and σ^2 = 400 are not given in table form). The number of grid points required grew quite slowly as the size of the training set was increased. In theory, this grid filter would reach about 1 million grid points, assuming an extremely large training set. However, even with 3.6 million training samples, the filter for σ^2 = 800 required only 17812 grid points. The percent of missing grid points was much lower than the simple13pt filters; when 3.6 million training samples were used, the percentage of missing grid points was 0.01% for the σ^2 = 800 filter and exactly 0% for the σ^2 = 100 filter.

Figure 3.23 shows how the MSE of the filters changed as the training set size was

	Training			Testing		
Repetitions	Training Samples	Grid Points	Matrix Entries	Missing grid points	MSE	PSNR (dB)
1	3600	1441	17101	1.015%	23.2	34.5
2	7200	1833	24210	0.613%	20.8	35.0
3	10800	2079	29673	0.415%	19.8	35.2
4	14400	2209	33339	0.331%	19.7	35.2
5	18000	2344	36872	0.252%	19.6	35.2
10	36000	2690	47687	0.130%	18.9	35.4
100	360000	3641	89266	0.002%	17.9	35.6
1000	3600000	4483	128423	0.000%	17.7	35.6

Table 3.13: Results for simple3x3 and σ^2=100

increased. Some rough guidelines for training set size can be extracted from this graph. For σ^2 = 100 and σ^2 = 200, about 10000 training samples were sufficient to train the filter. Larger training sets did not significantly reduce the MSE. Note that most training images contain far more pixels than this, so for small amounts of noise, a single pass through the training data appears sufficient. For σ^2 = 400, about 10^5 training samples were sufficient. The largest noise amount, σ^2 = 800 required roughly 10^6 training samples.

3.8 Performance for different noise types

As described in an earlier section (1.2.3), Gaussian noise is a worst-case for nonlinear image restoration filters, in the sense that improvement over linear filters is least for Gaussian noise. For non-Gaussian noise models, nonlinear filters should perform better than linear filters.

To verify this, a grid filter was trained on Gaussian, Laplacian, and Uniform noise. The grid filter used a 8^{12} grid and the simple3x3 footprint (Figure 2.2). The filters were trained on the synthetic squares image (Figure 3.1). This image was chosen because it consists largely of flat regions.

Training consisted of 1.5 million training samples (10 repetitions of the synthetic/squares image, each time with a different noise field). Training time averaged 7-10 minutes for each filter (including CG iteration time). Results were compared to the Wiener filter and optimal 3x3 Order Statistic Filters (with coefficients as in Table 1.1).

Table 3.14 shows results for σ^2 = 100 noise. Note that using a linear average,

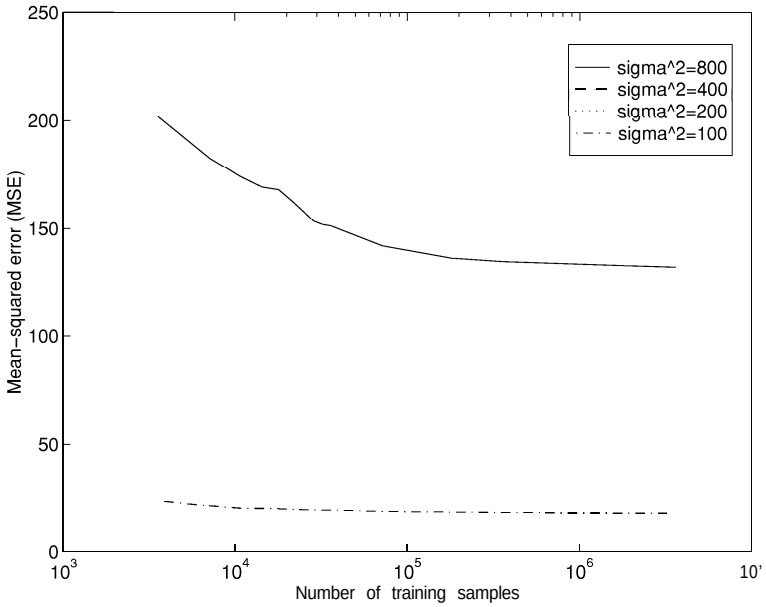


Figure 3.23: Effect of training set size on MSE for a hybrid 8^8 grid filter

	MSE		
Noise model	Grid filter	Wiener filter	Optimal OSF
Gaussian	12.53	59.3	155.3
Laplacian	10.18	59.4	32.0
Uniform	8.02	59.9	314.0

Table 3.14: MSE results for various noise models. $\sigma^2 = 100$

the MSE for a mean estimate of 9 observations of a stationary R.V. is 11.11. The grid filter does slightly worse than this for Gaussian noise because it has to cope with edges. For all noise types, the grid filter outperforms both the Wiener filter and the optimal OSF filter. Note that for Gaussian and Uniform noise, the OSF filters *increase* the noise amount. This is because OSF filters are derived assuming a stationary point process, which the squares image is not.

Table 3.15 shows results for $\sigma^2 = 800$ noise. The theoretical best a 3x3 linear average can do is an MSE of 88.89 for a stationary signal.' The grid filter is able to do better than this for both Laplacian and Uniform noise, despite the

'This is because a 3x3 window contains 9 observations; the residual variance is therefore $800/9 = 88.89$.

	MSE		
Noise model	Grid filter	Wiener filter	Optimal 3x3 OSF
Gaussian	106.7	206.8	232.6
Laplacian	80.9	203.5	109.2
Uniform	64.3	205.0	328.4

Table 3.15: MSE results for various noise models, $\sigma^2 = 800$

nonstationarity of the signal.

3.9 Superresolution

All imaging systems have an upper limit on resolution. These limitations can arise in several ways:

- **Diffraction** of light limits resolution to the wavelength of the illuminating light.
- **Lenses** in optical imaging systems truncate the image spectrum in the frequency domain [44].
- **Sampling** of images limits the maximum spatial frequency to a fraction of the sampling rate.

Superresolution refers to reconstructing frequency components which lie above the cutoff frequency of the imaging system. Grid filters, which learn statistical properties of an image class, are able to exploit their prior knowledge to perform superresolution. This section explores two scenarios: coherent (laser) illumination, and incoherent (non-laser) illumination. Images of text are used as a test case, since they have tightly constrained structural properties.

Coherent imaging systems

A coherent imaging system with a circular exit pupil acts as an ideal low-pass filter [44]. The frequency-domain representation of such a filter is a disc (Figure 3.24): all frequency components inside the disc are passed, and components outside the disc are blocked. To simulate such a system, images were Fourier transformed and all frequency components $\lambda^{-1} \geq \frac{1}{3}$ were zeroed. A mild amount of noise (AWGN, σ^2=10) was added.

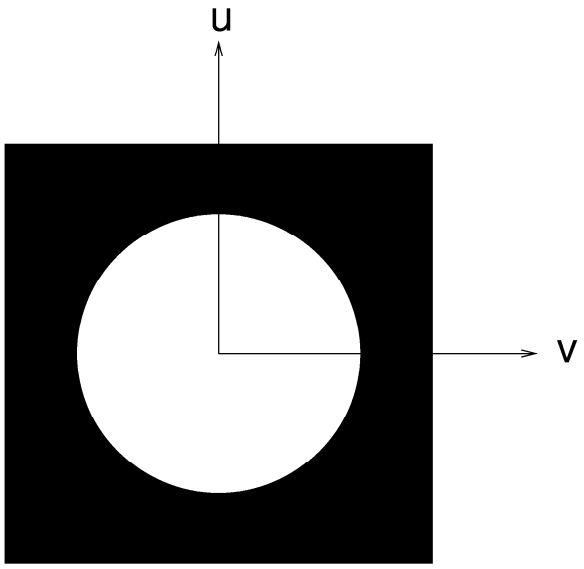
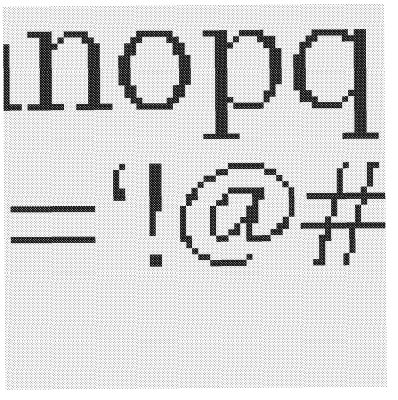


Figure 3.24: Frequency-domain representation of an ideal low-pass filter

Coherent imaging systems are subject to a noise process known as *speckle*, which can be approximated by multiplicative exponential noise [45]. Speckle only appears when the surface being imaged has roughness at the scale of the illuminating wavelength. Its omission here implies an assumption that the object being imaged is smooth at that scale.

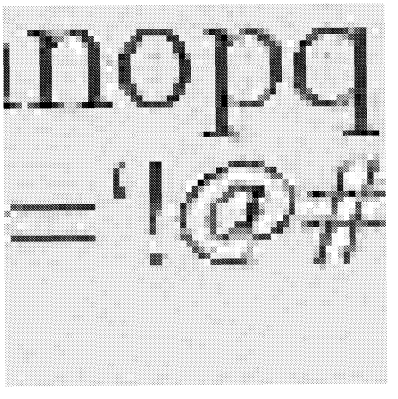
A hybrid filter with a 13-point detail filter for F_1 and 5x5 averager for F_2 was trained on images of Times-Roman, Helvetica and Courier text at 18 and 36 point size. Due to memory restrictions, the largest possible grid was 7^{12}. Figure 3.25 shows sample results for 24 pt Palatino text (not in the training set). The grid filter was able to reconstruct missing high-frequency components based on information in medium-frequency bands and its own knowledge of the structure of text images. It was also able to suppress "ringing" around edges which is caused by abrupt truncation in the frequency domain.

Figure 3.26 shows the signal-to-noise ratio for the raw and filtered images. Although there is no signal power present for $\lambda^{-1} \geq \frac{1}{3}$, the grid filter is able to reconstruct the missing components to the 4-7 dB level. This reconstruction comes at the cost of decreasing the SNR for lower frequencies. However, the actual contribution to MSE from the low frequency bands is comparatively low (Figure 3.27). The decreased SNR for low frequencies could be repaired by combining the filtered and raw images in the frequency domain to get the best of both signals.

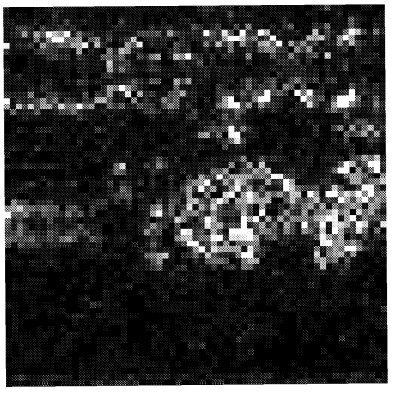


Original

After low-pass filtering
MSE=1306 PSNR=17.0 dB



Restoration
MSE=359.3 PSNR=22.6 dB



Absolute difference ∗4

Figure 3.25: Superresolution for a simulated coherent imaging system

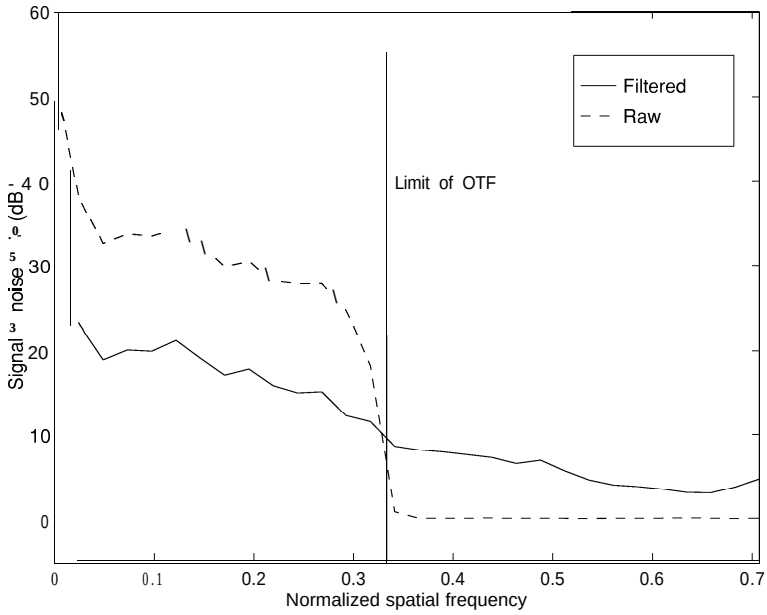


Figure 3.26: Signal-to-noise ratio for superresolution on a simulated coherent imaging system

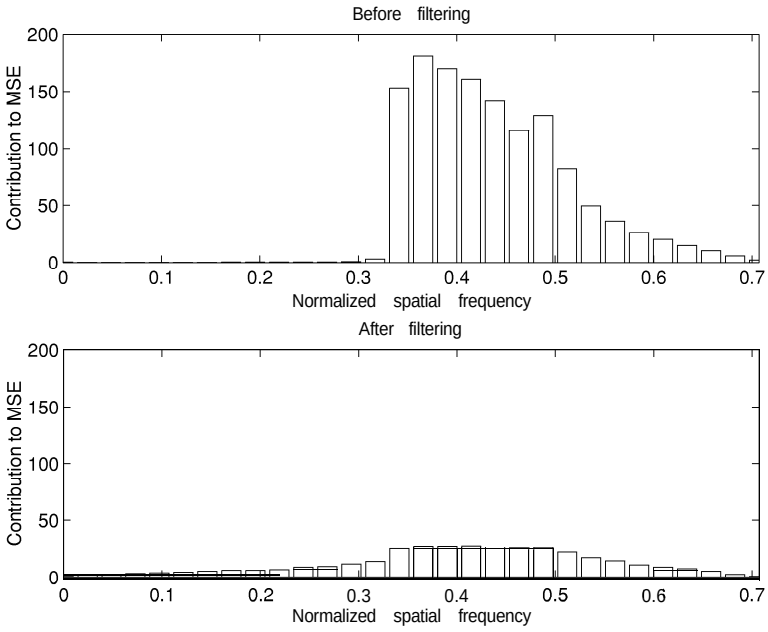
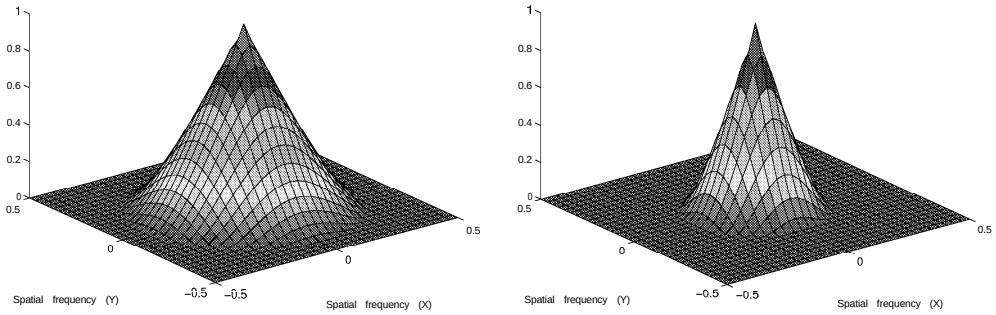


Figure 3.27: Distribution of MSE over spatial frequency bands, before and after filtering



$\rho_0 = 0.2$ $\rho_0 = 0.125$

Figure 3.28: Optical Transfer Function for a diffraction-limited, incoherent imaging system

Incoherent imaging systems

Changing from coherent to incoherent illumination changes the blurring process substantially. A coherent imaging system has an abrupt cut-off in the frequency domain, which results in "ringing" around edges. Incoherent illumination produces a smooth drop-off in the frequency domain which blurs edges gradually.

The Optical Transfer Function (OTF) for a diffraction-limited incoherent imaging system is radially symmetric. It can be written as $\mathcal{H}(\rho)$, where ρ is the radial frequency (λ^{-1})[44]:

$$\mathcal{H}(\rho) = \frac{2}{\pi}\left[cos^{-1}\left(\frac{\rho}{2\rho_0}\right) - \frac{\rho}{2\rho_0}\sqrt{1-\left(\frac{\rho}{2\rho_0}\right)^2}\right] \tag{3.1}$$

for $\rho \leq 2\rho_0$. The parameter ρ_0 is a cutoff frequency: frequencies greater than $2\rho_0$ have $\mathcal{H}(\rho) = 0$. Between 0 and $2\rho_0$, frequencies are gradually attenuated.

To simulate an incoherent imaging system, the OTF of (3.1) was applied to images, and noise was added (AWGN, $\sigma^2 = 10$). Two values of ρ_0 were used: 0.2 and 0.125. For $\rho_0 = 0.2$, **50%** of the spectrum (by area) is completely removed. For $\rho_0 = 0.125$, 80% of the spectrum is removed. A plot of the OTFs is shown in Figure 3.28.

A filter with a 13pt grid filter for F_1 and 5x5 averager for F_2 was trained on images of 36 point Helvetica, Times-Roman and Courier text. The filter was tested on 36 point Palatino text. The severe amount of blurring rendered smaller text indecipherable. Figures 3.29 and 3.30 show results for $\rho_0 = 0.2$. Although all signal

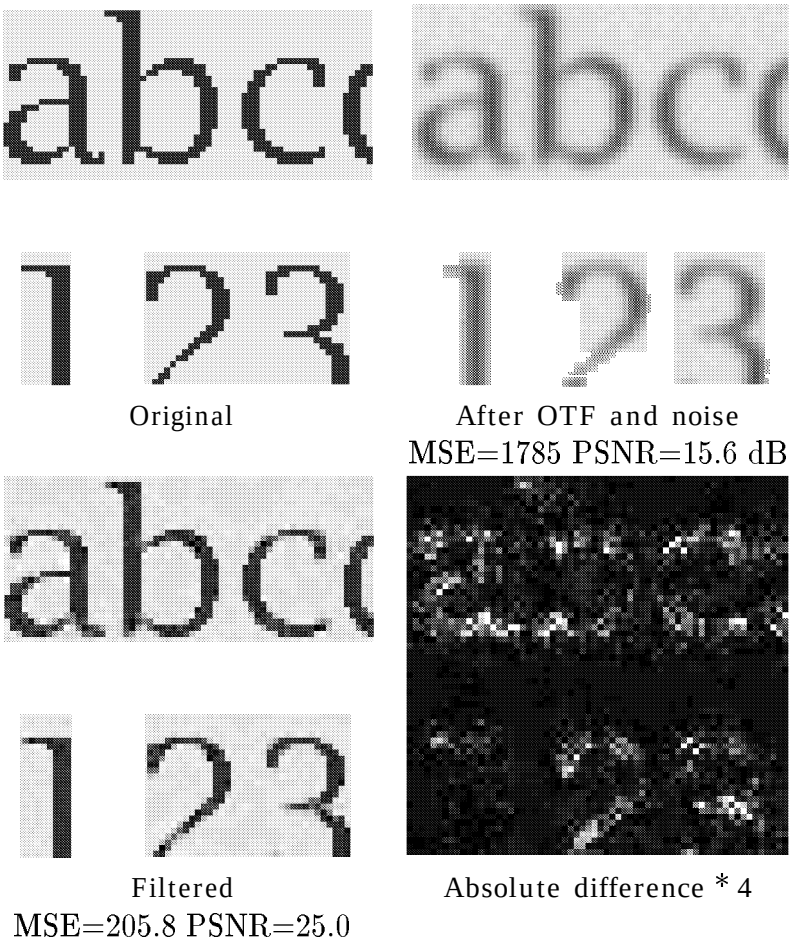


Figure 3.29: Results for $\rho_0 = 0.2$

components beyond $\rho \geq 0.4$ are completely eliminated by the OTF, the grid filter is able to reconstruct the missing components based on information in the medium frequency bands. Note that any *linear* filter would be unable to achieve a positive SNR for any frequencies above this limit.

Figures 3.31 and 3.32 show results for $\rho_0 = 0.125$. Experimentation revealed that the footprint fovea15x15c did better than the hybrid filters. The grid size was 13^{14}, but only 8772 grid points were used. The OTF for $\rho_0 = 0.125$ eliminates all components beyond $\rho \geq 0.25$. This means that there are no signal components with wavelength 4 pixels or less; about 80% of the spectrum is missing. The grid filter is able to partially reconstruct missing components in the range $0.25 \leq \rho \leq 0.5$. This means the filter is able to *double* the resolution of the image from $\lambda_{\min} = 4$ to

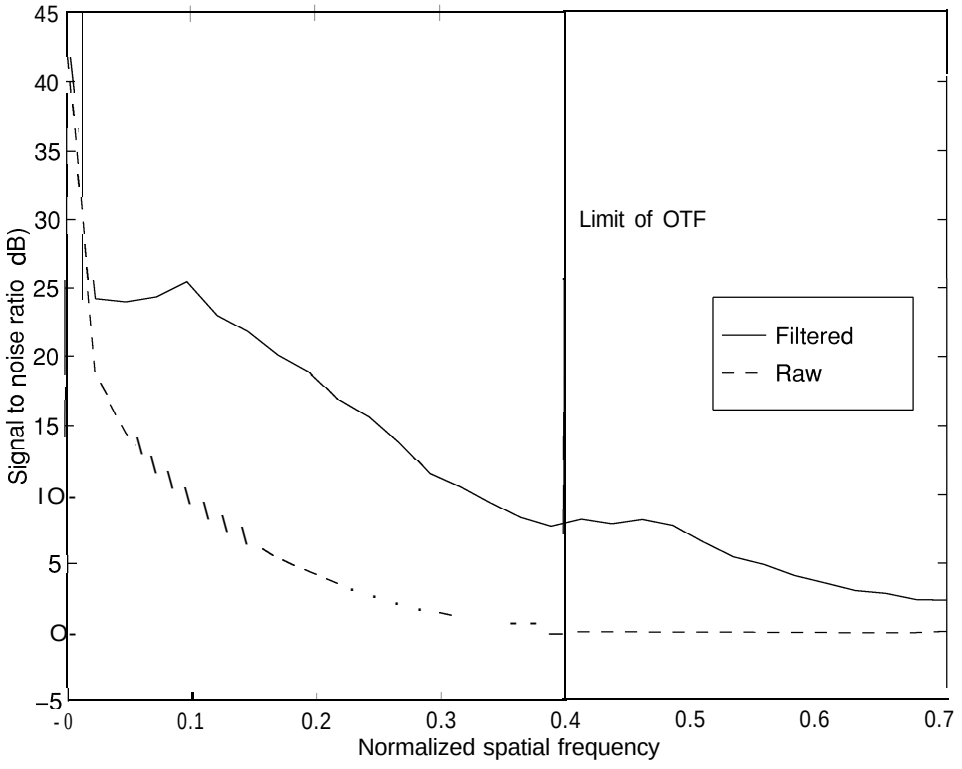


Figure 3.30: Signal-to-noise ratio for $\rho_0 = 0.2$

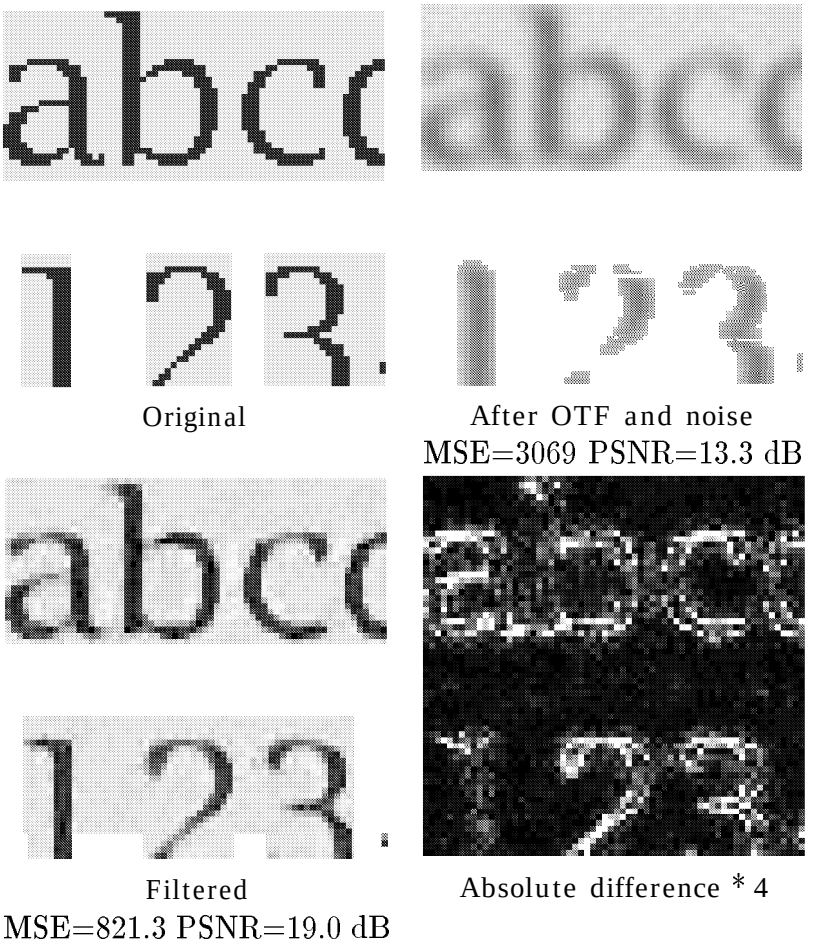


Figure 3.31: Results for $\rho_0 = 0.125$

$\lambda_{min} = 2$. The term resolution is used here to mean the smallest wavelength with positive SNR.

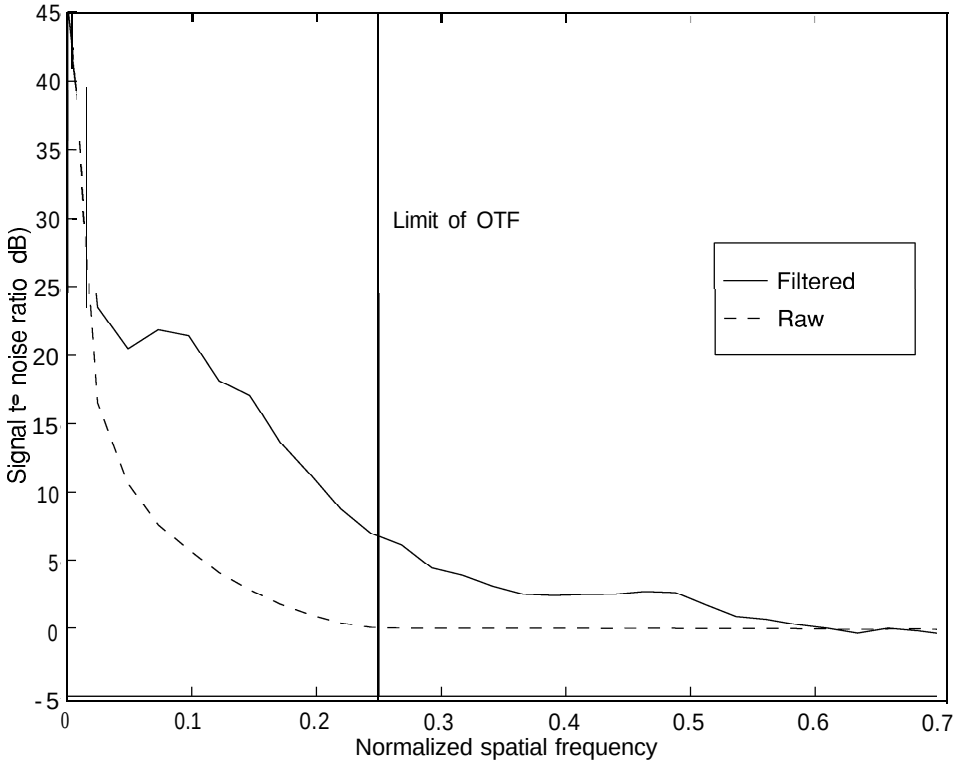


Figure 3.32: Signal-to-noise ratio for $\rho_0 = 0.125$

Chapter 4

Summary, Limitations and Future Work

4.1 Summary

Grid filters are a new type of filter for local nonlinear image processing. They have many properties which make them appealing; this section reviews the most important of these.

The evaluation time of grid filters is small and roughly constant, regardless of the number of filter coefficients. Arbitrarily close approximations to the optimal local nonlinear filter can be obtained without paying a significant performance price. This is in contrast to neural networks and polynomials, whose evaluation time is linear in the number of coefficients. Unlike many global approaches to image restoration which require iteration, grid filters restore images in a single pass.

Symmetry assumptions allow grid filters to generalize from training images to similar images with different orientations, illuminations, and reversed intensity values. These symmetry assumptions reduce both the number of filter coefficients and the amount of training data required. Unlike neural networks, grid filters require only a single presentation of the training data set.

When grid filters encounter unusual inputs, they tend to pass them unchanged. Grid filters can detect and flag regions of images which are unlike the training images.

Certain types of grid filters contain order statistic filters as a subset. Hybrid grid filters contain Lee's local statistics filter for additive noise as a special case. In practice, the performance of grid filters is superior to order statistic filters, Lee's filter, and the global Wiener filter.

Preliminary results are tantalizingly good. Grid filters are able to exploit structural characteristics of images to achieve excellent results for both noise suppression and superresolution.

4.2 Limitations

Grid filters are limited to a small number of inputs (roughly 13 if a typical workstation is used for training). We have had success with footprints as large as 15x15, with feature selection used to reduce the number of inputs.

Unlike Wiener, Lee, and order statistic filters, grid filters need to be trained for a target class of images. Training grid filters requires pairs of pristine and degraded images. These training pairs can be acquired experimentally (for example, pairing low-quality images with those acquired by a high-fidelity imaging system). If the degradation process is well understood, degraded images can be simulated using appropriate synthetic images. Grid filters are not suitable for blind deconvolution.

Grid filters are good at removing local degradations, for example small amounts of blurring and white noise. They are not suitable for removing large-scale blurring or low-frequency noise, although they might prove useful for cleaning up after a global inverse filter.

They perform best when the image class and degradation is tightly constrained. For example, a filter trained on document images with a specific amount of noise will perform much better than a filter trained to handle a wide variety of image types and degradations.

4.3 Future work

4.3.1 Improvements to the filter design

Feature selection

In the current implementation, features (inputs to the grid filter) are limited to single pixel values, or averages of pixel values from the local region. It may be possible to improve the performance of grid filters by using different kinds of features:

- Principle components (or Karhunen-Lòeve transformation) of the local neighborhood might allow larger window sizes to be used, while still using a small number of features. A shortcoming of principle components for this application is that they provide an optimal basis for reconstructing the entire

neighborhood, rather than a basis which gives maximal information about the central pixel. For this reason, a feature selection technique based on information theoretic principles might yield better results.

- Features which incorporate information from larger scales might solve the problem of smoothing over large regions in the presence of extreme noise. Features based on IIR filters or wavelets might serve this purpose.

- Currently filters must be specific to a particular degradation model. It might be possible to create filters which handle a variety of degradations by including parameter estimates for the degradation model as filter inputs. For example, a filter for additive noise could take an estimate of the noise variance as an input. The filter could then be trained on a variety of noise variances. This would effectively make the filter coefficients a function of the noise variance estimate. For blurring, one could include an estimate of the width of the point-spread function as a filter input.

- Hybrid grid filters make use of the variance and mean from a large neighborhood to switch between the grid filter and a point estimator. Better results might be obtained if the variance and mean estimates were fed into the grid filter as inputs. The grid filter could then turn itself on and off in an optimal fashion.

Grid design

- Currently the grid has explicit spatial bounds, and grid spacing is an integer fraction of the grid length. With the sparse representation, there is no need to have a limit on the spatial length of the grid. It might be useful to throw away this notion, and make the grid spacing a real-valued parameter which could be continuously varied.

- In several of the results, it is clear that there is a dramatic difference between grid spacings which are an even and odd fraction of the total grid length. This can be seen as an oscillation in PSNR in (for example) Figure 3.4 (p. 64). It likely relates to whether grid points coincide with the origin or are spaced symmetrically around it. It appears that grids which are spaced symmetrically around the origin perform better. The reasons for this should be investigated.

- The current grids are highly anisotropic. Staggered grids [46] might overcome this problem, and produce better results with fewer grid points. In two dimensions, staggered grids are hexagonal, and this geometry extends nicely to

many dimensions. On a staggered grid, the linear interpolation described in Section 2.5.2 (p. 37) becomes more closely isotropic. Staggered grids are also easy to implement: mapping from a staggered grid to a regular grid requires a simple linear transformation.

The anisotropy of regular grids is a substantial problem: the resolution of the grid is $\sqrt{N}$ times as large in certain directions as in others. For N = 2 this is not much of a problem, so it is ignored. But for N = 16, it means that in certain directions the grid has 4 times the resolution as in other directions. By developing grids which are more closely isotropic, there are reasons to believe that the complexity of many-dimensional function approximation will go from being exponential to polynomial in N.

- Many of the nodes in a highly-trained filter do not contribute much to reduction of the MSE. As described in Section 3.7.1 (p. 87), it may be beneficial to implement a two-pass training approach. The first pass would identify important grid points, and the second pass would form the least-squares equations to determine their value.

  There may be useful insights to be gained from regarding many-dimensional function approximation using grids as a compression problem. For example, throwing away unimportant grid points is reminiscent of rate-distortion ideas from coding theory.

- It is possible to train a grid filter to not only estimate the central pixel, but also the variance of that estimate. This would allow grid filters to place rough confidence intervals on filtering results. For example, in regions where there is insufficient information to localize an edge, for example in Figure 3.7 (p. 68), the grid filter might be able to indicate that there was information lost in that region.

- Better results might be obtained if the grid spacing were able to differ in each dimension. For example, one might use a very fine grid spacing for dimensions corresponding to nearby pixels, and a coarse spacing for dimensions corresponding to pixels fart her away.

- Symmetries of the grid are based primarily on the group D_4. This group is related to the symmetries of a rigid square in the plane. It can be written as the direct product of two subgroups: a reflection group and a cyclic rotation group. The cyclic rotation group is actually an approximation to the continuous rotation group SO(2) on the image sampling grid. With larger windows,

it may be possible to devise groups larger than D_4 which more closely approximate SO(2). Constructing larger symmetry groups would reduce the number of grid points. It may be useful to drop the group requirement, and work instead with sets or trees of permutations.

Training

- It may be possible to train grid filters without solving least-squares equations. The approach is akin to sampling the optimal function F on the grid points. The training equations which accomplish this are:

$$f_i = \frac{\sum_j w_i(\mathbf{x}^j)(s_0^j - x_0^j)}{\lambda + \sum_j w_i(\mathbf{x}^j)} \tag{4.1}$$

  where λ is a regularization parameter, and the other symbols are as described for (2.32, p. 51). This approach works marvelously well for two-dimensional test functions. Unfortunately, it appears to not work at all for practical image filtering problems in 8 or 12 dimensions. No reasonable explanation for this failure has yet been found.

- The time required to solve the least-squares equations using Conjugate Gradient might be reduced by the use of a preconditioner (see for example [41]).

- A detailed analysis of the effect of the regularization parameter λ on filter performance and convergence time of CG is needed. It is possible that some of the noise surrounding outlying data points, for example in Figure 3.20 (p. 83), might be due to too small a parameter λ.

- It may be possible to adopt multigrid methods (see for example [47]) to solve the least-squares equations. Multigrid methods are generally much faster than CG for large problems.

- Convergence of the CG method might be quicker if a succession of regularization parameters λ were used, starting with large values and ending with small values. Larger values of the parameter λ reduce the spectral condition of the least-squares equations, which speeds convergence. Starting with a large value of λ would allow quick convergence to an approximate solution, with subsequent values of λ used to refine the solution.

- Instead of stopping convergence by looking at residuals or changes in the solution vector, it should be possible to explicitly compute the criterion function

J(f) from the least-squares system. This would provide a better measure of when iteration should be stopped. It might also provide a good way of determining how large λ can be without affecting MSE substantially.

- The computational resources required by grid filters are far greater for training than filtering. Filters which require (for example) 100 Mb of RAM for training require less than 10 Mb of RAM for filtering. It may be useful to port the training software to a supercomputer, so that very large filter training problems can be tackled. Although such filters would require a supercomputer to train, they would still fit comfortably into a workstation's memory for filtering .

- Rather than explicitly constructing the least-squares equations, it might be useful to adopt a neural-network style of training. This would greatly reduce memory requirements at the expense of longer training times.

4.3.2 Speed improvements

The current implementation is written in the C++ language using a framework which was designed for flexibility rather than speed. It should be possible to substantially increase filtering speeds by making the code more efficient:

- Profiling of the current implementation has revealed that most of the filtering time is spent in isomorph selection (i.e. handling permutation group symmetries). The current implementation of isomorph selection is grossly inefficient. It should be possible to devise a faster method to handle these symmetries.

- Selection of the interpolation nodes (the routine FINDPERMUTATION of Figure 2.14 (p. 41)) is currently implemented using selection sort. This is highly inefficient, and should be replaced with a fast sorting network.

- Better use of cache memory might be obtained by implementing tiling [48, 49] of the image.

- It might be useful to implement a filter compiler which would take the filter specification generated by the current implementation, and turn it into highly optimized C-language code. This would permit specialization of critical algorithms, which has been shown to have a large effect on performance [50].

- For hybrid grid filters, there are large regions of the image where the output of the grid filter is not used at all, because the parameter β is zero (for the explanation of this parameter, see Section 2.8 on p. 55). It may be possible to speed up the performance of these filters by first testing the parameter β. If it is sufficiently close to zero, evaluation of the grid filter can be avoided.

4.3.3 Applications

In this thesis, grid filters were evaluated for their ability to undo additive noise and blurring (superresolution). It is reasonable to expect that grid filters may do well for a variety of other problems:

- Grid filters could be adapted for multiplicative noise. In this scenario, the signal mean invariance assumption (Section 2.6.2, p. 46) would have to be discarded. The grid filter would still be designed as an adjustment to the original pixel value, but a dimension would not be eliminated from the grid.

- Grid filters could be trained for robust edge detection. This would require paired samples of images and the desired edge detection image.

- Grid filters should be capable of edge enhancement (sharpening) in the presence of noise. Training data could be synthesized by sharpening an image, and training a grid filter on the noisy version.

- Binary Tree Predictive Coding (BTPC) [51] uses a multilevel approach to image compression. Each level is used to predict the representation of the image at the next finer level. Grid filters can be trained to do this type of prediction, and it is possible that they will outperform current prediction techniques.

- Grid filters could be trained to remove artifacts left by compression algorithms such as JPEG. Preliminary tests using BTPC have shown roughly a $+3$dB gain when heavily compressed images were passed through a grid filter.

- It may be useful in some situations to train a cascade of grid filters. The first filter would do the best it could to undo the degradation. The second filter would clean up after the first filter, and contribute its own thoughts to the restoration. Unfortunately, it will not be possible to train such a cascade simultaneously. The first filter will have to be fully trained before the second filter training can be started.

- Grid filters might be useful for audio signal processing. Their ability to suppress noise, extrapolate missing frequency components and undo compression artifacts might be useful for telephony.

- The ideas behind grid filters extend nicely to supervised pattern recognition problems in a moderate number of dimensions. They might be appropriate for image segmentation, texture recognition, and general pattern classification problems.

Bibliography

[1] John Hertz, Anders Krogh, and Richard G. Palmer. *Introduction to the Theory of Neural Computation,* volume 1 of *Santa Fe Institute Studies in the Sciences of Complexity, Lecture Notes.* Addison-Wesley, Reading, Massachusetts, 1991.

[2] Simon Haykin. Neural networks expand SP's horizons. *IEEE Signal Processing Magazine,* 13(2):24–49, March 1996.

[3] K. Sam Shanmugan and A. M. Breipohl. *Random Signals: Detection, Estimation and Data Analysis.* John Wiley & Sons, Toronto, 1988.

[4] I. J. D. Craig and J. C. Brown. *Inverse problems in astronomy: a guide to inversion strategies for remotely sensed data.* Adam Hilger Ltd., Boston, 1986.

[5] Kenneth R. Castleman. *Digital Image Processing.* Prentice Hall, Toronto, 1996.

[6] A. C. Bovik, T. S. Huang, and D. C. Munson. A generalization of median filtering using linear combinations of order statistics. *IEEE Transactions on Acoustics, Speech, and Signal Processing,* 31(6):1342–1349, December 1983.

[7] I. Pitas and A. N. Venetsanopoulos. *Nonlinear digital jilters: principles and applications.* Kluwer Academic Publishers, Boston, 1990.

[8] H. A. David. *Order statistics.* Wiley, Toronto, 1981.

[9] A. Restrepo and A. C. Bovik. Adaptive trimmed mean filters for image restoration. *IEEE Trans. Acoustics, Speech, and Signal Processing,* ASSP-36(8):1326, 1988.

[10] Kenneth E. Barner and Gonzalo R. Arce. Permutation filters: A class of nonlinear filters based on set permutations. *IEEE Transactions on Signal Processing,* 42(4):782–798, 1994.

[11] J. S. Lee. Digital image enhancement and noise filtering by use of local statistics. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* PAMI-2:165–168, 1980.

[12] Kevin Erler and M. Ed Jernigan. Adaptive image restoration using recursive image filters. *IEEE Trans. on Signal* Processing, 42(7):1877–1881, July 1994.

[13] Zhi-Qiang Liu and Terry Caelli. A sequential adaptive recursive filter for image restoration. *Computer Vision, Graphics, and Image Processing,* 44(3):332–349, December 1988.

[14] Y. H. Lee, S. J. Ko, and A. T. Fam. Efficient impulsive noise suppression via nonlinear recursive filtering. *IEEE Transactions on Acoustics, Speech, and Signal Processing,* 37:303–306, 1989.

[15] Philippe Saint-Marc, Jer-Sen Chen, and Gerard Medioni. Adaptive smoothing: A general tool for early vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 13(6):514–529, June 1991.

[16] Klaus Rank and Rolf Unbehauen. An adaptive recursive 2-D filter for removal of Gaussian noise in images. *IEEE Transactions on Image Processing,* 1(3):431–436, July 1992.

[17] A. H. Tewfik and H. Garnaoui. Multigrid implementation of a hypothesis testing approach to parametric blur identification and image restoration. *J. Opt. Soc. Am. A, Opt. Image Sci.,* 8:1026–1037, *1991.*

[18] K. Zhou and C. K. Rushforth. Image restoration using multigrid methods. *Appl. Opt.,* 30:2906–2912, *1991.*

[19] S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 6:721–741, 1984.

[20] S. Z. Li. *Markov Random Field Modeling in Computer Vision.* Computer Science Workbench. Springer-Verlag, Tokyo, first edition, 1995.

[all] M. R. Bhatt and U. B. Desai. Robust image restoration algorithm using Markov random field model. *CVGIP: Graphical Models and Image Processing,* 56(1):61–74, January 1994.

[22] J. Zerubia and R. Chellappa. Mean field annealing using compound gauss-markov random fields for edge detection and image restoration. Technical Report RR-1295, Institut National de Recherche en Informatique et Automatique (National Institute for Research in Computer and Control Sciences), October 1990.

[23] D. W. Murray, A. Kashko, and H. Buxton. A parallel approach to the picture restoration algorithm of Geman and Geman on a SIMD machine. *Image, Vision and Computing,* 4: 1333142, 1986.

[24] A Kashko. Image restoration by simulated annealing on the DAP at QMC. Technical Report QMW-DCS-1985-366, Queen Mary College, Department of Computer Science, December 1985.

[25] N. B. Karayiannis and A. N. Venetsanopoulos. Regularization theory in image restoration-the stabilizing functional approach. *IEEE Transactions on Acoustics, Speech, and Signal* Processing, 38(7):1155, 1990.

[26] Alan M. Thompson, John C. Brown, Jim W. Kay, and D. Michael Titterington. A study of methods of choosing the smoothing parameter in image restoration by regularization. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* PAMI-13(4):326–339, April 1991.

[27] H. J. Trussell. The relationship between image restoration by the maximum A posteriori method and a maximum entropy method. *IEEE Trans. Acoustics, Speech, and Signal Processing,* ASSP-28(1):114, 1980.

[28] Jan Myrheim and Håavard Rue. New algorithms for maximum entropy image restoration. *CVGIP: Graphical Models and Image Processing,* 54(3):223–238, May 1992.

[29] S. F. Burch, S. F. Gull, and J. Skilling. Image restoration by a powerful maximum entropy method. *Computer Vision, Graphics, and* Image Processing, 23(2):113–128, August 1983.

[30] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing.* Addison-Wesley, New York, 1992.

[31] Martin Schetzen. *The Volterra and Wiener theories of nonlinear systems.* Wiley, New York, 1980.

[32] Robert D. Nowak and Barry D. Van Veen. Tensor product basis approximations for Volterra filters. *IEEE Transactions on Signal Processing,* 44(1):36–50, January 1996.

[33] V. John Mathews. Adaptive polynomial filters. *IEEE Signal Processing Magazine,* pages 10-26, July 1991.

[34] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. Neural *Networks,* 2:359–366, 1989.

[35] L. Guan. Image restoration by a neural network with hierarchical cluster architecture. *Journal of Electronic Imaging,* 3(2):154–63, 1994.

[36] Y. T. Zhou, R. Chellappa, A. Vaid, and B. K. Jenkins. Image restoration using a neural network. *IEEE Trans. Acoustics, Speech, and Signal Processing,* ASSP-36(7):1141, 1988.

[37] Y.-T. Zhou, R. Chellappa, A. Vaid, and B. K. Jenkins. Image restoration using a neural network. *IEEE Transactions on Acoustics, Speech, and Signal Processing,* 36(7):1141–1151, July 1988.

[38] David S. Burnett. *Finite Element Analysis.* Addison-Wesley, Reading, Massachusetts, 1987.

[39] Steve Hill. Tri-linear interpolation. In Paul Heckbert, editor, *Graphics Gems IV,* pages 521-525. Academic Press, Boston, 1994.

[40] Joseph A. Gallian. *Contemporary Abstract Algebra.* D. C. Heath and Company, Toronto, 3rd edition, 1994.

[41] R. Barrett, M. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. Van der Vorst. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, 2nd Edition.* SIAM, Philadelphia, PA, 1994.

[42] U. Schendel. *Sparse Matrices: Numerical Aspects with Applications for Scientists and Engineers.* Halsted Press, Toronto, 1989.

[43] C. C. Paige and M. A. Saunders. LSQR: An algorithm for sparse linear equations and sparse least squares. *ACM Trans. Math. Soft.,* 8:43–71, *1982.*

[44] Joseph W. Goodman. *Introduction to Fourier optics.* McGraw-Hill, 1968.

[45] Anil K. Jain. *Fundamentals of Digital Image Processing.* Prentice Hall, 1989.

[46] Bengt Fornberg. High-order finite differences and the pseudospectral method on staggered grids. *SIAM Journal on Numerical Analysis,* 27(4):904–918, August 1990.

[47] William L. Briggs. *A Multigrid Tutorial.* Society for Industrial and Applied Mathematics, Philadelphia, Pennsylvania, 1987.

[48] Monica S. Lam, Edward E. Rothberg, and Michael E. Wolf. The cache performance and optimizations of blocked algorithms. In *Fourth Intern. Conf. on Architectural Support for Programming Languages and Operating Systems (APLOS IV), Palo Alto, California,* pages 63-74, April 9-11 1991.

[49] Larry Carter, Jeanne Ferrante, and Susan Flynn Hummel. Hierarchical tiling for improved superscalar performance. In *Proceedings of the 9th International Symposium on Parallel Processing (IPPS'95*, pages 239-245, Los Alamitos, CA, USA, April 1995. IEEE Computer Society Press.

[50] Andrew Berlin and Daniel Weise. Compiling scientific code using partial evaluation. *Computer,* 23(12):25–37, Dec 1990.

[51] John A. Robinson. Efficient general-purpose image compression with binary tree predictive coding. *IEEE Transactions on Image Processing, 6,* 1997.

[52] Ronald W. Schafer, Russel M. Mersereau, and Mark. A. Richards. Constrained iterative restoration algorithms. *Proceedings of the IEEE,* 69(4):432–450, April 1981.

[53] G. Thomas and R. Prost. Iterative constrained deconvolution. *Signal Processing,* 23:89–98, 1991.

[54] 0. C. Zienkiewicz. *Finite Element Method.* McGraw-Hill, London, 4th edition, 1989. In two volumes.

[55] George M. Zloković. *Group Supermatrices in Finite Element Analysis.* Ellis Horwood/Simon & Schuster, New York, 1992.

[56] Reiner Lenz. *Group Theoretical Methods in Image Processing,* volume 413 of *Lecture Notes in Computer Science.* Springer-Verlag, Berlin, 1990.

[57] Martin Hanke and James G. Nagy. Restoration of atmospherically blurred images by symmetric indefinite conjugate gradient techniques. *Inverse Problems*, 12:157–173, 1996.

[58] Lloyd Allison. Generating coset representatives for permutation groups. *Journal of Algorithms*, 2:227–244, *1981.*

[59] *G.* Butler and C. W. H. Lam. A general backtrack algorithm for the isomorphism problem of combinatorial objects. *Journal of Symbolic Computation*, 1:363–381, 1985.

[60] C. C. Paige and M. A. Saunders. Solution of sparse indefinite systems of linear equations. *SIAM J. Numer. Anal.*, 12:617–629, 1975.

[61] G. G. Lorentz. *Approximation of Functions.* Chelsea, New York, 2nd edition, 1986.

[62] E. R. Davies. On the noise suppression and image enhancement characteristics of the median, truncated median and mode filters. *Pattern Recognition Letters*, 7:87–97, 1988.

[63] A. P. King and R. G. Wilson. Multiresolution image analysis based on local symmetries. Research Report CS-RR-248, Department of Computer Science, University of Warwick, Coventry, UK, September 1993.

[64] S. E. Reichenbach and S. K. Park. Smallconvolution kernels for high-fidelity image restoration. *IEEE Transactions on Signal Processing*, 39(10):2263, 1991.

[65] Jonathon D. Victor. Nonlinear systems analysis in vision: overview of kernel methods. In Robert B. Pinter, editor, *Nonlinear Vision*, chapter 1, pages l-37. CRC Press, London, 1992.

[66] Sheldon M. Ross. *Introduction to probability models.* Academic Press, San Diego, 5th edition, 1993.

[67] E. L. Lehmann. *Theory of Point Estimation.* John Wiley & Sons, New York, 1983.

[68] V. G. Voinov and M. S. Nikulin. *Unbiased Estimators and Their Applications*, volume 1. Kluwer Academic Publishers, Boston, 1989.

[69] D. F. Andrews, P. J. Bickel, F. R. Hampel, P. J. Huber, W. H. Rogers, and J. W. Tukey. *Robust Estimates of Location.* Princeton University Press, 1972.

[70] Bernard Picinbono. *Random Signals and Systems.* Prentice Hall, Englewood Cliffs, 1993.

[71] J. Wood. Invariant pattern recognition: A review. *Pattern Recognition,* 29(l):l-17, 1996.