Digital Image Processing
using
Local Segmentation

Torsten Seemann
B. Sc (Hons)

School of Computer Science and Software Engineering
Faculty of Information Technology
Monash University
Australia.

Submission for the degree of Doctor of Philosophy

April 2002

Contents

List of Figures

List of Tables

Listings

Abstract

A unifying philosophy for carrying out low level image processing called "local segmentation" is presented. Local segmentation provides a way to examine and understand existing algorithms, as well as a paradigm for creating new ones. Local segmentation may be applied to range of important image processing tasks. Using a traditional segmentation technique in intensity thresholding and a simple model selection criterion, the new FUELS denoising algorithm is shown to be highly competitive with state-of-the-art algorithms on a range of images. In an effort to improve the local segmentation, the minimum message length information theoretic criterion for model selection (MML) is used to select between models having different structure and complexity. This leads to further improvements in denoising performance. Both FUELS and the MML variants thereof require no special user supplied parameters, but instead learn from the image itself. It is believed that image processing in general could benefit greatly from the application of the local segmentation methodology.

Declaration

This thesis contains no material that has been accepted for the award of any other degree or diploma in any university or other institution. Furthermore, to the best of my knowledge, this thesis contains no material previously published or written by another person, except where due reference is made in the text of the thesis.

Torsten Seemann

January 28, 2003

Acknowledgments

"I love deadlines. I love the whooshing sound they make as they fly by."

— Douglas Adams, 1952—2001

I wish to acknowledge the following people for their role in the completion of this thesis:

- My supervisor, Peter Tischer, for his guidance, constructive proofreading, and many fruitful discussions, some of which even stayed on topic. Thank you for helping me produce a thesis I can be proud of.
- David Powell, for many productive white-board exchanges over 6 years of sharing an office, and for actually understanding most of my thesis after a single proofreading.
- My parents, Sandra and Günther Seemann, for always supporting my choices in life, and only occasionally nagging me to finish up and get a real job that pays money.
- My partner, Naomi Maguire, for her devoted love, support and encouragement. What else could you ask for?
- Naomi's parents, Brenda and Bill Maguire, for accepting an unemployed doctoral student "bum" into their family.
- Bernd Meyer, for a handful of valuable thesis related discussions we had when he occasionally blessed our office with a visit.
- The staff in the School of Computer Science and Software Engineering at Monash University. In particular, Trevor Dix, for regularly checking in on me, and commiserating with the plight of the doctoral student.

This thesis was typeset using $\LaTeX\,2_\varepsilon$ [KD95]. Graphs were generated by `gnuplot` [WKL$^+$], and diagrams were constructed using `XFig` [SKYS].

Chapter 1

Introduction

Image processing is a rapidly growing area of computer science. Its growth has been fueled by technological advances in digital imaging, computer processors and mass storage devices. Fields which traditionally used analog imaging are now switching to digital systems, for their flexibility and affordability. Important examples are medicine, film and video production, photography, remote sensing, and security monitoring. These and other sources produce huge volumes of digital image data every day, more than could ever be examined manually.

Digital image processing is concerned primarily with extracting useful information from images. Ideally, this is done by computers, with little or no human intervention. Image processing algorithms may be placed at three levels. At the lowest level are those techniques which deal directly with the raw, possibly noisy pixel values, with denoising and edge detection being good examples. In the middle are algorithms which utilise low level results for further means, such as segmentation and edge linking. At the highest level are those methods which attempt to extract semantic meaning from the information provided by the lower levels, for example, handwriting recognition.

The literature abounds with algorithms for achieving various image processing tasks. However, there does not appear to be any unifying principle guiding many of them. Some are one dimensional signal processing techniques which have been extended to two dimensions. Others apply methods from alternative disciplines to image data in a somewhat inappropriate manner. Many are the same basic algorithm with parameter values tweaked to suit the problem at hand. Alternatively, the parameters are optimized with respect to a suitable training set, without thought on how to vary them for images with different properties. There do exist well considered methods, but unfortunately a large proportion of new ideas have been *ad hoc*, without any central guiding principle.

This thesis proposes a unified approach to low level image processing called "local segmentation". The local segmentation principle states that the first step in processing a pixel should be to segment explicitly the local region encompassing it. On a local scale, this has the effect of making clear which pixels belong together, and which pixels do not. The segmentation process results in a local approximation of the underlying image, effectively separating the signal from the noise. Thus higher level algorithms can operate directly on the signal without risk of amplifying the noise. Local segmentation can be seen as providing a common framework for constructing image processing algorithms.

Many existing image processing algorithms already make partial use of the local segmentation concept. It is possible to examine these algorithms with respect to the local segmentation model they use. This helps to make their strengths and weaknesses more apparent. Even popular techniques, such as linear and rank filters, can be framed in terms of their application of local segmentation. In most cases the segmentation is implicit rather than explicit. That is, the choice of which pixels belong together is performed in a systematic, but sometimes roundabout manner. The SUSAN image processing system, developed by Smith and Brady, originally considered using explicit segmentation, but eventually chose to allow all pixels to have partial membership in the centre pixel's segment.

Image denoising is particularly suited to demonstrating the utility of local segmentation. Denoising is the process of removing unwanted noise from an image. A denoised image is an approximation to the underlying true image, before it was contaminated. A good denoising algorithm must simultaneously preserve structure and remove noise. Obviously, to do this the algorithm must be able to identify what structure is present. Local segmentation specifically attempts to separate structure from noise on a local scale. Denoising would therefore be a good application with which to test different approaches to local segmentation.

Local regions only contain a small number of pixels. It is unlikely that there would be more than a few segments present at such a scale, so unconnected, homogeneous groups of pixels are likely to part of the same global segment. Traditional threshold-based segmentation techniques, such as k-means, perform well with this sort of data. This technique is used to develop FUELS, an algorithm for denoising greyscale images affected by additive noise. FUELS requires only one parameter, the noise variance, which can be supplied by the user or estimated automatically from the image. The noise variance and pixel values are used by a model order selection criterion for deciding the optimal number of segments present in the local region.

FUELS has two additional original features. If the optimal segmentation is deemed unfit, the noisy pixels are passed through unmodified. This is called the "do no harm" principle. Also, local segmentation produces multiple overlapping estimates for the value of each pixel in the image. FUELS is able to combine these estimates to further improve its denoising performance. It will be shown that FUELS is competitive with state-of-the-art denoising algorithms over a range of noise levels.

Although FUELS uses a simple segmentation algorithm, it still achieves good results. It does, however, have some limitations. The nature of FUELS' model selection criterion makes it unable to distinguish segments having a contrast difference less than a specified multiple of the noise variance. The criterion is also not flexible enough to allow FUELS to compare different models having the same number of segments. This is because the thresholding algorithm does not incorporate any spatial information.

Local segmentation may be considered an inductive inference problem. We wish to infer the underlying image structure given only the noisy pixel values. Bayesian and information theoretic techniques are recognised as some of the most powerful for these types of problems. One such technique is Wallace's minimum message length information theoretic criterion (MML), which has been developed continually since 1968. MML is essentially a Bayesian method, but provides a sound way to choose a point estimate from the posterior distribution over models. It works well with small amounts of data, and is invariant under non-linear transformations of the parameter space.

Under MML, a candidate model is evaluated using its message length. A message is an efficient lossless encoding of the data, and consists of two parts. The first part encodes the model and its parameters. The second part encodes the data given the model from the first part. The candidate model with the shortest overall message length is deemed the best model for the data. This is similar, but not identical, to the maximum *a posteriori* (MAP) estimate used in Bayesian analysis. When the model contains continuous parameters, MML optimally quantizes the prior density such that the MAP estimate is invariant under non-linear transformations of the data and model parameters.

The model selection criterion used by FUELS is replaced by a more flexible MML one. MML makes it straightforward to include extra models in the pool of candidate models being considered. For a 3×3 window, there exists 256 unique ways to divide the 9 pixels into one or two segments. The MML version of FUELS is modified to evaluate and consider all of these. This allows spatial information to be exploited, and overcomes the minimum contrast difference that FUELS requires between segments. The "do no harm" principle is re-interpreted, and shown to fit naturally within the MML framework.

Using MML for model selection is shown to improve results relative to FUELS, and to outperform other good denoising algorithms. It produces better local approximations, especially for very noisy images. Evaluating and comparing large numbers of models using MML, however, is much more computationally intensive than FUELS. In terms of RMSE, the improvements are not large. This indicates that simpler techniques like FUELS and SUSAN are already good trade-offs between efficiency and effectiveness. In critical applications where the best possible modeling is required, the use of MML methods could be warranted.

The local segmentation paradigm is not limited to denoising applications. It is shown that a variety of image processing tasks may be addressed using local segmentation. Breaking an image into its low level structural components could be considered an essential first step, from which many other tasks are derived. It is shown how local segmentation may be used for edge detection, pixel classification, image enlargement and image compression. The extension to different noise models, such as impulse noise, image models, such as planar segments, and higher dimensional data, such as volume images, is also discussed.

The FUELS local segmentation algorithm has the desirable feature of being simple, while still producing good results. This makes it well suited to robotic and computer vision applications. It can be implemented using low amounts of memory and processing power, ideal for putting into hardware or embedded microcontrollers. Local segmentation is inherently parallelizable, because each pixel's local region is processed independently. Thus a highly concurrent implementation would be possible. This could be useful in real time applications where many images per second need to be analysed.

I believe that local segmentation provides a unifying philosophy for carrying out low level image processing. It provides a way to examine and understand existing algorithms, as well as a paradigm for creating new ones. A local segmentation analysis of an image can be re-used by a wide range of image processing tasks. Using a traditional segmentation technique in intensity thresholding and a simple model selection criterion, the FUELS denoising algorithm is shown to be highly competitive with state-of-the-art algorithms on a range of images. In an effort to improve the local segmentation, MML is applied to select between a larger set of models having different structure and complexity. This leads to further improvements in denoising performance. Both FUELS and the MML variants thereof require no special user supplied parameters, but instead learn from the noisy image itself. I believe that image processing in general could benefit greatly from the application of the techniques proposed in this thesis.

Chapter 2

Notation and Terminology

2.1 Digital images

A *digital image* is a discrete two-dimensional function, $f(x, y)$, which has been quantized over its domain and range [GN98]. Without loss of generality, it will be assumed that the image is rectangular, consisting of Y rows and X columns. The *resolution* of such an image is written as $X \times Y$. By convention, $f(0, 0)$ is taken to be the top left corner of the image, and $f(X-1, Y-1)$ the bottom right corner. This is summarized in Figure 2.1.

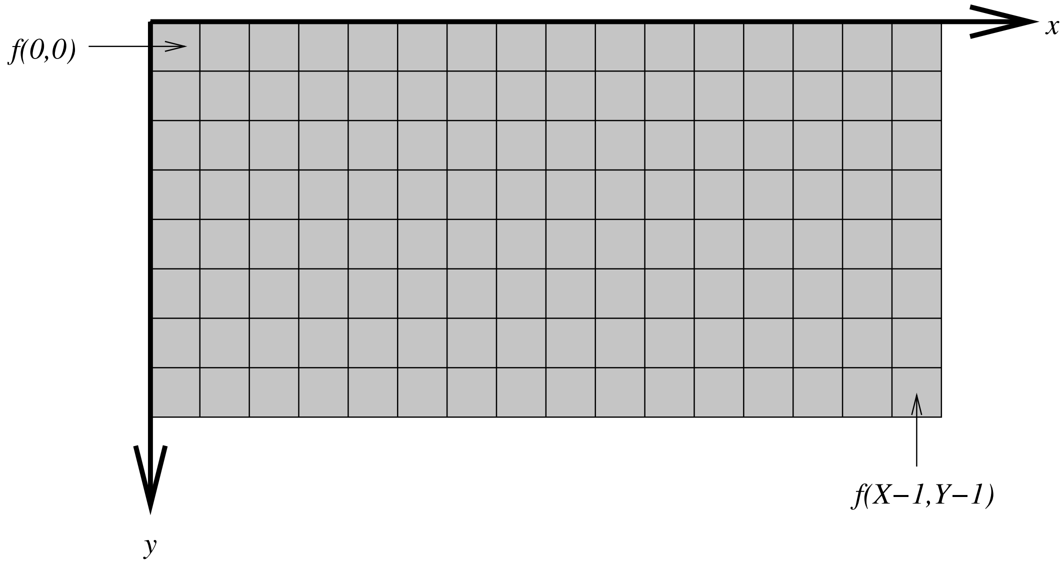

Figure 2.1: A rectangular digital image of resolution 16×8.

Each distinct coordinate in an image is called a *pixel*, which is short for "picture element". The nature of the output of $f(x, y)$ for each pixel is dependent on the type of image. Most images are the result of measuring a specific physical phenomenon, such as light, heat, distance, or energy. The measurement could take any numerical form.

A *greyscale* image measures light intensity only. Each pixel is a scalar proportional to the brightness. The minimum brightness is called black, and the maximum brightness is called white. A typical example is given in Figure 2.2. A *colour* image measures the intensity and chrominance of light. Each colour pixel is a vector of colour components. Common colour spaces are RGB (red, green and blue), HSV (hue, saturation, value), and CMYK (cyan, magenta, yellow, black), which is used in the printing industry [GW92]. Pixels in a *range* image measure the depth of distance to an object in the scene. Range data is commonly used in machine vision applications [KS00].

Figure 2.2: A typical greyscale image of resolution 512×512.

For storage purposes, pixel values need to be quantized. The brightness in greyscale images is usually quantized to Z levels, so $f(x, y) \in \{0, 1, \cdots, Z - 1\}$. If Z has the form 2^L, the image is referred to as having *L bits per pixel*. Many common greyscale images use 8 bits per pixel, giving 256 distinct grey levels. This is a rough bound on the number of different intensities the human visual system is able to discern [Jäh93]. For the same reasons, each component in a colour pixel is usually stored using 8 bits.

Medical scans often use 12–16 bits per pixel, because their accuracy could be critically important. Those images to be processed predominantly by machine may often use higher values of Z to avoid loss of accuracy throughout processing. Images not encoding visible light intensity, such as range data, may also require a larger value of Z to store sufficient distance information.

There are many other types of pixels. Some measure bands of the electromagnetic spectrum such as infra-red or radio, or heat, in the case of thermal images. Volume images are actually three-dimensional images, with each pixel being called a *voxel*. In some cases, volume images may be treated as adjacent two-dimensional image slices. Although this thesis deals with greyscale images, it is often straightforward to extend the methods to function with different types of images.

2.2 Image statistics

2.2.1 The histogram

A *histogram* plots the relative frequency of each pixel value that occurs in a greyscale image. Figure 2.3 shows the intensity histogram for the image from Figure 2.2. The histogram provides a convenient summary of the intensities in an image, but is unable to convey any information regarding spatial relationships between pixels. In this example, the image does not contain many very low or very high intensity pixels. It is possible that peaks in the histogram correspond to objects in the image, but it is difficult to be certain without visually examining the image.

2.2.2 The mean

The *image mean* is the average pixel value of an image. For a greyscale image this is equal to the average brightness or intensity. Let the image $f(x, y)$ be referred to using the shorthand $\mathbf{f}$. The mean of this image, $\mathrm{E}[\,\mathbf{f}\,]$, may be calculated using Equation 2.1.

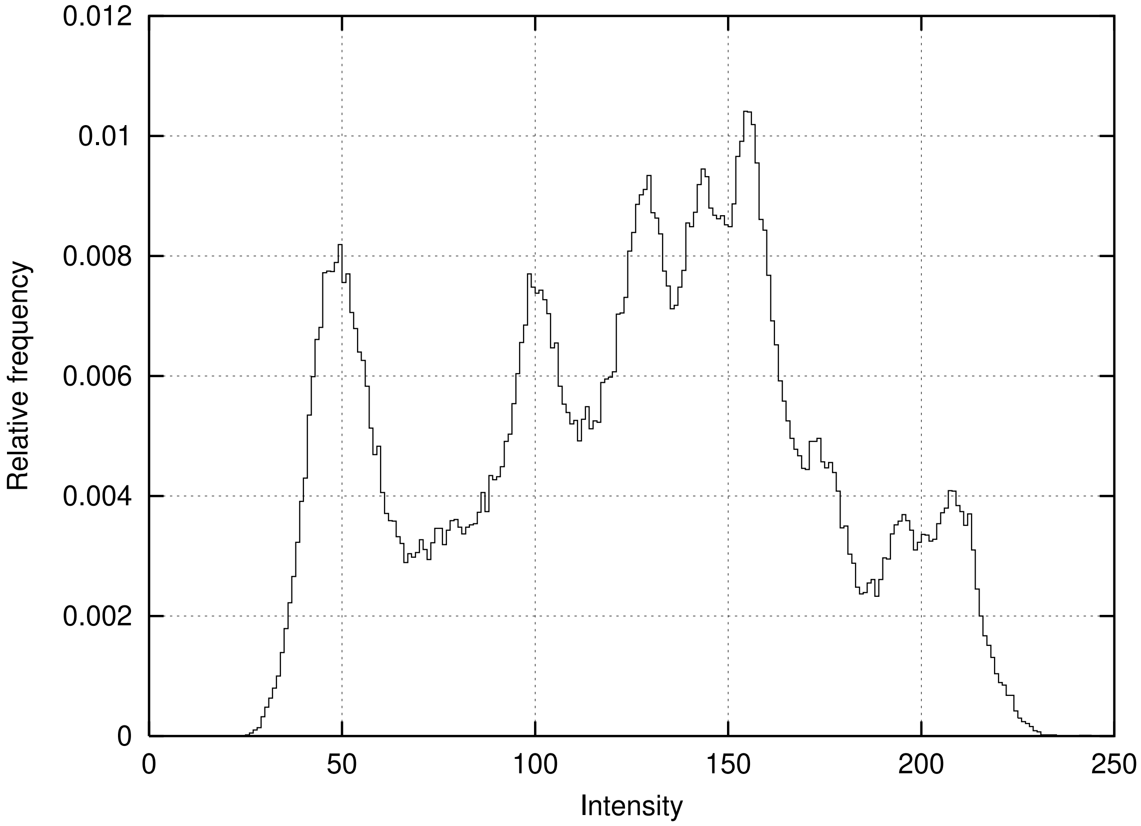

Figure 2.3: The histogram for the greyscale image in Figure 2.2.

$$\mathrm{E}[\,\mathbf{f}\,] = \frac{1}{YX}\sum_{y=0}^{Y-1}\sum_{x=0}^{X-1} f(x,y) \tag{2.1}$$

2.2.3 The variance

The *image variance*, $\mathrm{Var}[\,\mathbf{f}\,]$, gives an estimate of the spread of pixel values around the image mean. It can be calculated using either Equation 2.2 or Equation 2.5. The latter has the advantage of requiring only one pass through the image. The *standard deviation* is simply $\sqrt{\mathrm{Var}[\,\mathbf{f}\,]}$.

$$\begin{aligned} \mathrm{Var}[\,\mathbf{f}\,] &= \mathrm{E}[\,\mathbf{f} - \mathrm{E}[\,\mathbf{f}\,]\,]^2 && (2.2)\\ &= \frac{1}{YX}\sum_{y=0}^{Y-1}\sum_{x=0}^{X-1} (f(x,y) - \mathrm{E}[\,\mathbf{f}\,])^2 && (2.3) \end{aligned}$$

$$= \frac{1}{YX}\sum_{y=0}^{Y-1}\sum_{x=0}^{X-1}\left(f(x,y) - \frac{1}{YX}\sum_{y'=0}^{Y-1}\sum_{x'=0}^{X-1} f(x',y')\right)^2 \tag{2.4}$$

$$\begin{aligned} \mathrm{Var}[\,\mathbf{f}\,] &= \mathrm{E}[\,\mathbf{f}^2\,] - \mathrm{E}[\,\mathbf{f}\,]^2 && (2.5)\\ &= \left(\frac{1}{YX}\sum_{y=0}^{Y-1}\sum_{x=0}^{X-1} f(x,y)^2\right) - \left(\frac{1}{YX}\sum_{y=0}^{Y-1}\sum_{x=0}^{X-1} f(x,y)\right)^2 && (2.6)\end{aligned}$$

2.2.4 The entropy

The image histogram may be considered a probability distribution over pixel values. For the case of a Z level greyscale image, the histogram entry for intensity z may be written as $\Pr(z)$. The *entropy* of an image, $\mathbf{f}$, is given by Equation 2.7. The units of entropy are *bits* when using logarithms to base 2.

$$\mathrm{H}(\mathbf{f}) = -\sum_{z=0}^{Z-1} \Pr(z)\log_2 \Pr(z) \quad \text{bits} \tag{2.7}$$

The entropy has a maximum value of $\log_2 Z$ when all intensities occur will equal frequency, corresponding to a uniform histogram. It has a minimum value of 0 when all pixels have the same intensity. The entropy is one measure of the information content of an image. Because it is calculated from the histogram, it is unable to take spatial factors into consideration.

2.3 Image algebra

2.3.1 Image-scalar operations

Various useful arithmetical operations may be defined for images. Let $\otimes$ represent the binary operator for addition, subtraction, multiplication, or division. Equation 2.8 shows how to combine a scalar, c, and an image, $\mathbf{g}$, to produce a new image, $\mathbf{f}$. This is a pixel-wise operation — each pixel in $\mathbf{g}$ is operated on using $\otimes$ with c, and the result put in $\mathbf{f}$.

$$\mathbf{f} = \mathbf{g} \otimes c \qquad \equiv \qquad \forall(x, y) \quad f(x, y) = g(x, y) \otimes c \tag{2.8}$$

This idea could be used to enhance an image which is too dark. Consider the image in Figure 2.4a which uses 8 bits per pixel (256 levels), but only contains pixels with intensities from 64 to 191. One may consider enhancing it to use the full intensity range. This can be achieved using Equation 2.9, where $\lfloor\cdot\rfloor$ denotes integer truncation, and floating point precision is used for all pixels during the calculation. The result is given in Figure 2.4b.

$$\mathbf{f} = \left\lfloor \frac{\mathbf{g} - 64}{128} \times 255 \right\rfloor \tag{2.9}$$

Figure 2.4: (a) low contrast image; (b) after enhancement.

2.3.2 Image-image operations

The image-scalar operation may be extended to the combination of two images, $\mathbf{g_1}$ and $\mathbf{g_2}$, having the same resolution. Instead of combining a scalar with each pixel, two pixels with the same coordinate in different images are used instead. Equation 2.10 describes this process.

$$\mathbf{f} = \mathbf{g_1} \otimes \mathbf{g_2} \qquad \equiv \qquad \forall(x, y) \;\; f(x, y) = g_1(x, y) \otimes g_2(x, y) \tag{2.10}$$

Imagine wanting to generate a blended version of two greyscale images of identical resolution. This could be achieved using Equation 2.11, where α determines the mixing proportion. *Alpha blending* is a simple form of morphing, and is often used to dissolve between scenes in film and television. A visual example for $\alpha = 0.5$ is given in Figure 2.5.

$$\mathbf{f} = \lfloor \alpha \times \mathbf{g_1} + (1 - \alpha) \times \mathbf{g_2} \rfloor \tag{2.11}$$

Figure 2.5: Alpha-blending example: (a) first image; (b) second image; (c) blended image using $\alpha = 0.5$.

2.4 Image acquisition

Image acquisition is the process of obtaining a digitized image from a real world source. Each step in the acquisition process may introduce random changes into the values of pixels in the image. These changes are called *noise*. Assume you want to send a photo of your new house to a friend over the Internet. This may be achieved by taking a photograph with a conventional camera, having the film made into a print, scanning the print into a computer, and finally emailing it to your friend. Figure 2.6 shows the many potential sources of noise.

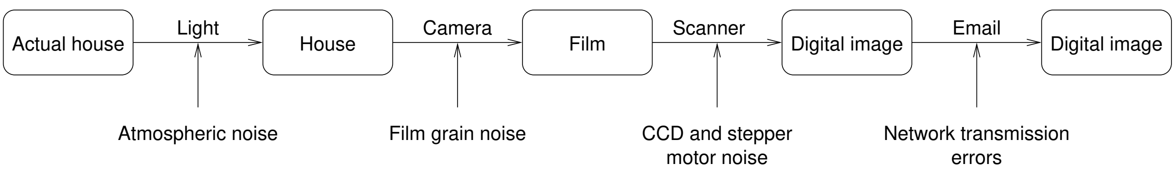

Figure 2.6: Noise may be introduced at each step in the acquisition process.

The air between the photographer and the house may contain dust particles which interfere with the light reaching the camera lens. The silver-halide crystals on the film vary in size and are discontinuous, resulting in film grain noise in the printing process [MJ66]. Most scanners use a CCD array to scan a row of the print, which may introduce photo-electronic noise. The scanner's CCD array is controlled by a fine stepper motor. This motor has some degree of vibration and error in its movement, which may cause pixels to be mis-aligned. The scanner also quantizes the CCD signal, introducing quantization noise [GN98]. Transmitting the image over the Internet is nearly always a bit preserving operation thanks to error checking in network protocols. However, an image transmitted to Earth from a remote space probe launched in the 1970's is almost guaranteed to contain errors.

2.5 Types of noise

The previous example illustrated the manner in which an image may be affected by noise during the acquisition process. The properties of the noise introduced at each capture step are likely to vary. However, there are three standard noise models which model well the types of noise encountered in most images: additive, multiplicative, and impulse noise. Figure 2.7 shows how these types of noise affect a typical greyscale image.

2.5.1 Additive noise

Let $f'(x,y)$ be the noisy digitized version of the ideal image, $f(x,y)$, and $n(x,y)$ be a "noise function" which returns random values coming from an arbitrary distribution. Then *additive noise* can be described by Equation 2.12.

$$f'(x,y) = f(x,y) + n(x,y) \tag{2.12}$$

Additive noise is independent of the pixel values in the original image. Typically, $n(x,y)$ is symmetric about zero. This has the effect of not altering the average brightness of the image, or large parts thereof. Additive noise is a good model for the thermal noise within photo-electronic sensors [Pit95].

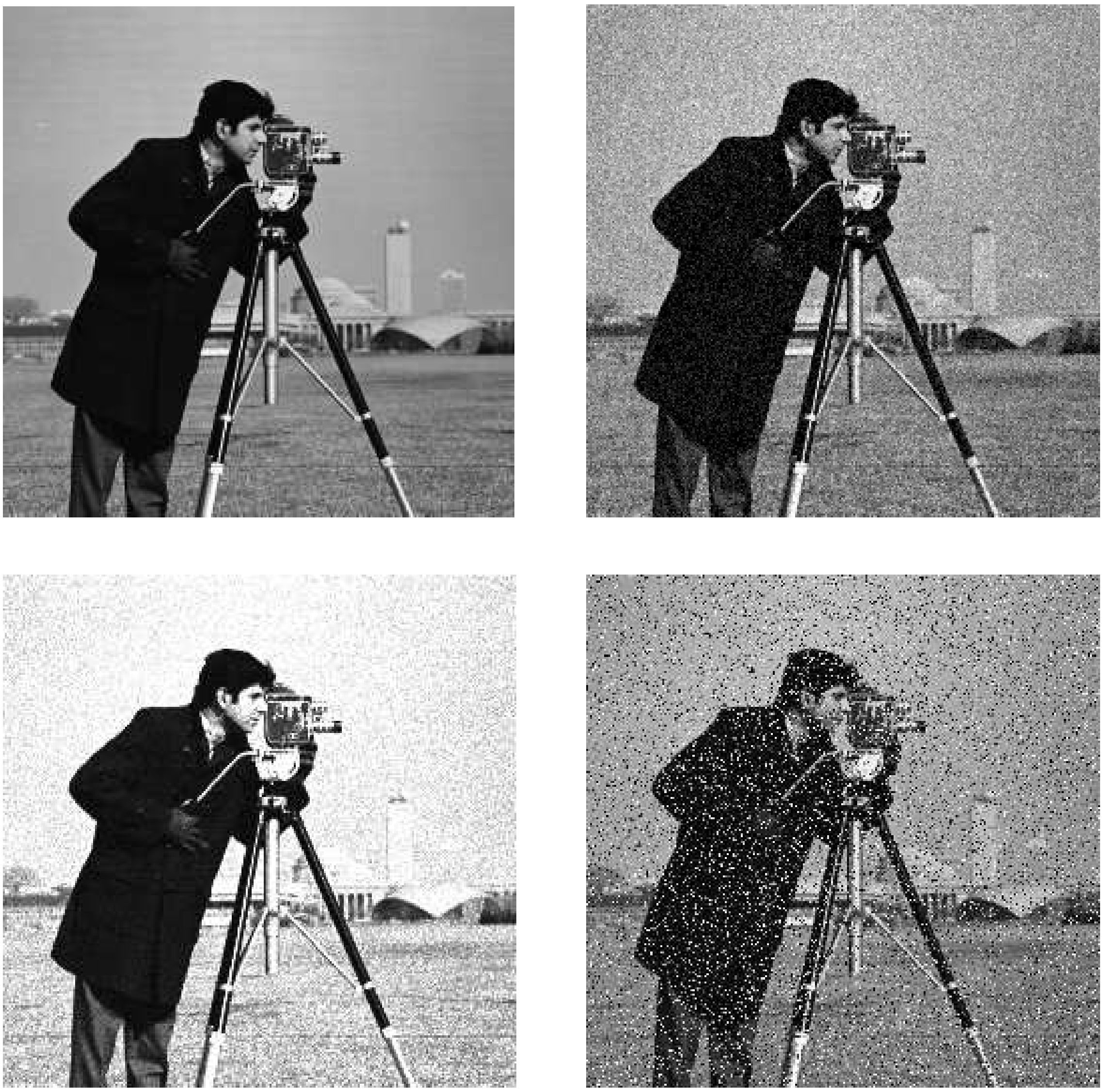

Figure 2.7: Different types of noise: (a) original image; (b) additive noise; (c) multiplicative noise; (d) impulse noise.

2.5.2 Multiplicative noise

Multiplicative noise, or *speckle noise*, is a signal dependent form of noise whose magnitude is related to the value of the original pixel [KSSC87]. Equation 2.13 describes one simple form it can take, but a more complex function of the original pixel value is also possible. Multiplicative noise is an approximation to the noise encountered in images recorded on film slides [Jai89] and from synthetic aperture radar [Lee81a, Cur91].

$$f'(x,y) = f(x,y) + n(x,y)f(x,y) = f(x,y)[1 + n(x,y)] \tag{2.13}$$

2.5.3 Impulse noise

Impulse noise has the property of either leaving a pixel unmodified with probability $1 - p$, or replacing it altogether with probability p. This is shown in Equation 2.14. Restricting $n(x, y)$ to producing only the extreme intensities 0 or $Z - 1$ results in *salt-pepper* noise. The source of impulse noise is usually the result of an error in transmission or an atmospheric or man-made disturbance [PV90].

$$f'(x, y) = \begin{cases} n(x, y) & \text{with probability} \quad p \\ f(x, y) & \text{with probability} \quad 1 - p \end{cases} \tag{2.14}$$

2.5.4 Quantization noise

Quantization noise is due to the quantization of pixel values during the analog to digital conversion. For example, imagine an analog image with brightness values ranging from 0 to 10. If it is quantized to accuracy 0.1, the digitized image will have 101 distinct grey levels. A given intensity, z, could have originally been anywhere in the range $z \pm 0.05$. This uncertainty in the true value of z is called quantization noise [STM97].

2.5.5 The noise function

Usually the properties of the noise function, $n(x, y)$, do not vary with x and y. A spatially invariant stochastic process is referred to as being *stationary*. The noise function could theoretically take any form, but many standard probability distributions have been found useful. For additive noise, the Gaussian and Laplacian distributions are often used [JRvdH93, AG98]. The standard case of impulse noise uses a uniform distribution on $[0, Z - 1]$.

The most common noise model used in this thesis is an additive zero-mean Gaussian of unknown variance, independently and identically distributed for each pixel. The application to alternative noise models is also considered. Some algorithms developed for additive noise can be adapted to multiplicative noise by logarithmically transforming $f'(x, y)$, applying the algorithm, and then applying the inverse transform [MPA00].

2.6 Segmentation

Segmentation involves partitioning an image into groups of pixels which are homogeneous with respect to some criterion. Different groups must not intersect each other and adjacent groups must be heterogeneous [PP93]. The groups are called *segments*. Figure 2.8a shows a noisy image containing three objects on a background. The result of segmentation is given in Figure 2.8b. Four segments were discovered, and are shown with a dashed outline.

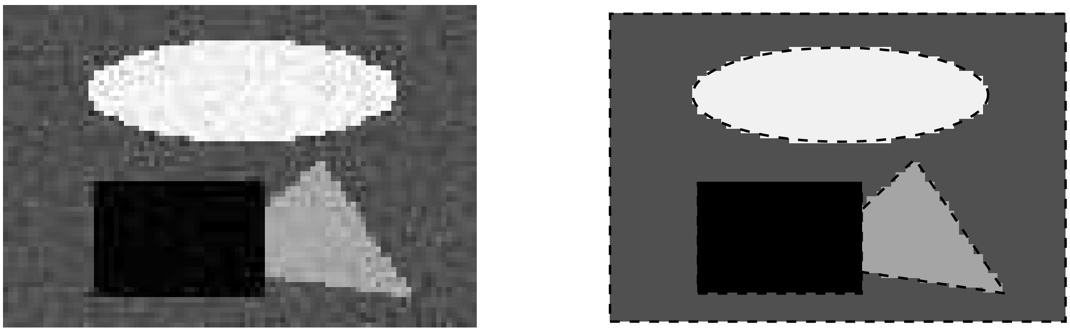

Figure 2.8: (a) original image; (b) segmented into four segments.

The homogeneity criterion used for segmenting Figure 2.8 was based only on the similarity of pixel intensities. For images containing large amounts of noise or fine structure, this criterion may be insufficient for successful segmentation. In those cases, some information regarding the spatial relationship between pixels is required. In particular, the assumption that pixels belonging to the same segment are expected to be spatially connected is exploited.

A pixel only has eight immediate neighbours. If symmetry is required, there exists only three forms of pixel connectedness which make sense, shown in Figure 2.9. There is one type of 8-connectedness, and two types of 4-connectedness. Type I is more popular than Type II due to the improved proximity of the four neighbouring pixels. In this thesis, only 8-connectedness and Type I 4-connectedness are considered.

Most digital images exist on a rectangular grid. This is primarily due to the arrangement of image sensors on camera and scanning equipment. Research has been done on the superior properties of hexagonal grids [Sta99], but they are unlikely to displace the square grid in the near future. In this thesis we deal only with images sampled on a square grid.

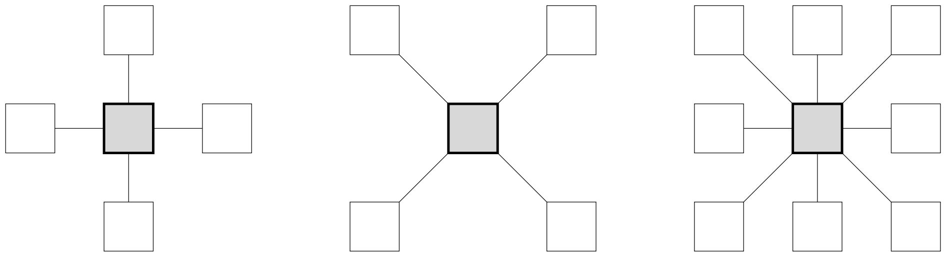

Figure 2.9: Pixel connectedness: (a) 4-connected [Type I]; (b) 4-connected [Type II]; (c) 8-connected.

2.7 Local windows

Most of the algorithms discussed in this thesis operate in an adaptive *local* manner, processing each pixel in turn. Only a small number of neighbouring pixels are included in any calculation. Local processing can produce simpler algorithms with lower algorithmic time and space complexity. They may be parallelizable, theoretically to one computer processor per pixel. There is also evidence to suggest that the human visual system functions in a parallel local manner, combining many local interpretations into an overall image [BBB00].

The union of the pixel being processed and its neighbouring pixels may be collectively referred to as a *window*, a *mask*, or the *local region* surrounding the pixel. Local windows typically involve fewer than 50 pixels, on images with up to 10^7 pixels. The only unbiased window configuration is an *isotropic* one — symmetrical and centered on the pixel to be processed. A circular window meets this requirement, but because pixels reside on a rectangular grid, some pixels would cover an area only partially within the circle. Figure 2.10 shows five local windows which are commonly used in image processing. Each is an approximation to a circular window, with the square windows being simplest to implement.

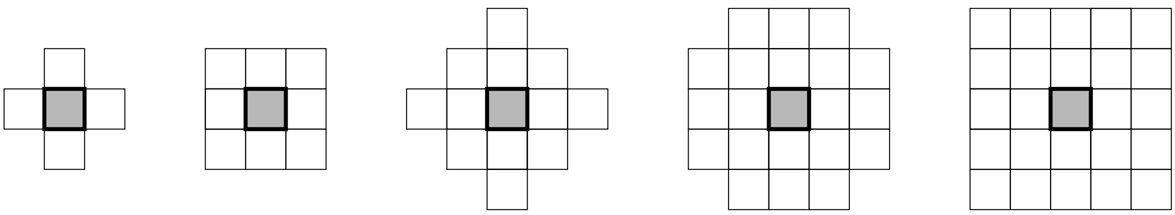

Figure 2.10: Common local neighbourhood configurations: 5, 9, 13, 21, 25 pixels.

Chapter 3

Local Segmentation in Image Processing

3.1 Introduction

This thesis proposes the use of local segmentation as an effective way to achieve a variety of low level image processing tasks. The local segmentation principle states that the first step in processing a pixel should be to segment the local region encompassing that pixel. This provides a snapshot of the local structural features of the image, with the signal clearly separated from the noise. It is hoped that the identified structural information could be used to implement many image processing tasks including, but not limited to, image denoising [Mas85], pixel classification [Cho99], edge detection [HSSB98], and pixel interpolation [AW96].

Local segmentation can be seen to belong to a continuum of approaches to image understanding, as shown in Figure 3.1. At the lowest level is local segmentation which operates in a purely local manner using only a small number of pixels. At a higher level is global segmentation which attempts to group together related pixels from throughout the image. The highest level is object recognition, whereby global segments are combined into logical units representing real world objects of interest.

The fundamental component of the local segmentation approach is the segmentation algorithm itself. Most segmentation algorithms are designed to operate upon a whole image, or a large portion thereof. Local segmentation can only utilise a small number of pixels belonging to fragments of larger segments. Thus a local segmentation algorithm differs in that it

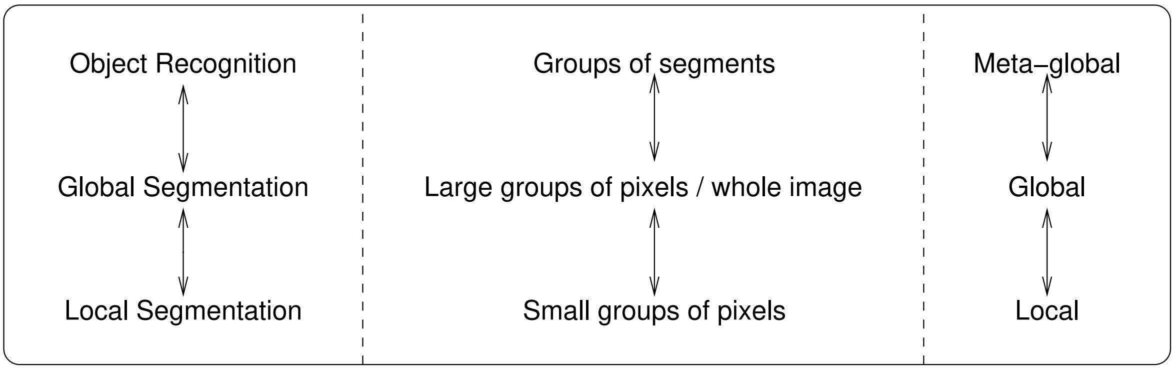

Figure 3.1: An image processing hierarchy.

has less data and less context to work with. In Section 3.2 the suitability of applying global segmentation algorithms to local segmentation will be examined.

Some image processing techniques can be seen or interpreted as exploiting the principle of local segmentation in some way. In most cases the local segmentation principle is not stated explicitly, nor used as a guiding principle for developing related algorithms. One example is the lossy image compression technique Block Truncation Coding, or BTC [FNK94]. BTC uses simple thresholding to segment 4×4 blocks of pixels into two classes, but it was many years before alternative segmentation algorithms were considered. Further instances of existing local segmentation-based algorithms will be explored in Section 3.3.

Those image processing tasks suited to local segmentation are often the first to encounter the raw image data. This data is usually contaminated with one or more forms of noise. The fundamental attribute of a local segmentation based algorithm should be to preserve as much image structure (useful information) as possible, and to suppress or remove as much noise (useless information) as possible. These goals are complementary and inseparable — the ability to identify structure implies the ability to identify noise, and *vice versa*.

Good image denoising algorithms specialize in extracting structure from noisy images. This application is probably the most appropriate low level technique for demonstrating local segmentation. In Section 3.4, the extent to which existing denoising algorithms utilise the principles espoused in this thesis will be explored. It is shown that the trend in state-of-the-art denoising algorithms has been toward a local segmentation perspective.

Upon reading this chapter, certain themes will become apparent. Image processing is an enormous field consisting of many different algorithms. Within a specific field it is often difficult to compare results. This is due to the rarity of objective criteria for comparison, a lack of standard test data, and simply the widely differing goals and needs of each system. Some techniques are *ad hoc* in their approach, often having been inappropriately adapted from other fields without thought to the validity of the underlying assumptions. Others have one or more tunable parameters which, although data dependent, must be supplied by the user, rather than learned automatically from the image itself. It is hoped that the work in this thesis will go some way to improving this situation.

3.2 Global segmentation

Segmentation involves partitioning an image into groups of pixels which are homogeneous with respect to some predicate. Each group is called a *segment*. Different segments must not intersect and adjacent segments must be heterogeneous [PP93]. Pixels within a segment need not necessarily be spatially connected. Clustering is usually used to refer to segmentation techniques using a homogeneity predicate which does not consider spatial information [HS85]. The segments produced by a clustering algorithm are sometimes called *clusters*, but they are still legitimate segments.

Global segmentation is concerned with segmenting a whole image. Local segmentation deals with segmenting sub-images which are small windows on a whole image. Although a sub-image is still a valid image, it is also a fragment of a larger scene being processed in isolation. A side-effect of this is shown in Figure 3.2. The image consists of two global segments: a light cross on a dark background. Each of the segments is homogeneous and fully connected. The 3×3 sub-image is also a cross on a background, but its "background" consists of four spatially disjoint pixels. Without a larger context it is difficult to ascertain whether they should be treated as one or four segments. A clustering algorithm, guided only by pixel intensities, would group them into a single segment.

The number of pixels available to local segmentation is much lower than what most global segmentation algorithms would expect. This has an effect on the typical number of distinct segments expected to be encountered. Figure 3.3a is a typical greyscale image called `lenna`. Figure 3.3b shows an image which plots, for each pixel from `lenna`, the standard deviation

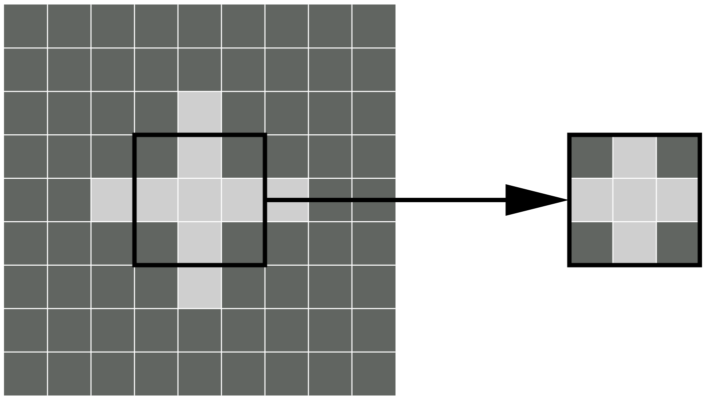

Figure 3.2: A 3×3 sub-image taken from the centre of a 9×9 image with two segments.

of the 3×3 sub-image centered on that pixel. The standard deviations range from 0 to 72, and are represented here using intensities from black to white.

Figure 3.3: (a) original image; (b) local 3×3 standard deviations, with black representing 0 and white 72.

Homogeneous regions in the original image produce dark areas in the standard deviation image, while edges are responsible for lighter areas. The majority of Figure 3.3b is dark. Figure 3.4 shows a histogram of the standard deviations, which is expected to be unimodal and skewed to the right [RLU99]. The peak at 2.4 corresponds roughly to the natural level of variation within homogeneous regions. The skew toward larger standard deviations is caused by heterogeneous edge and texture regions varying above the natural level.

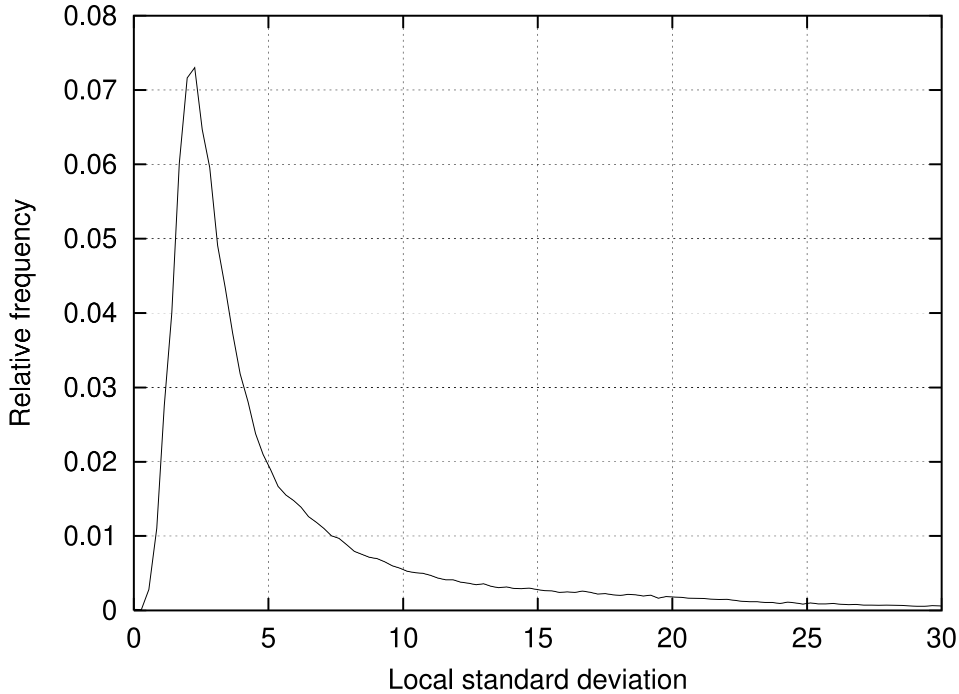

Figure 3.4: Histogram of local 3×3 standard deviations shown in Figure 3.3b.

The area under the histogram near the natural level of variation is much higher than in the skew. This suggests that it is highly likely that a randomly selected 3×3 sub-image will be homogeneous, consisting of a single segment. This is in stark contrast to global segmentation, where the diagnosis or even consideration of a single segment result is rare. A local segmentation algorithm must therefore be able to determine automatically the number of segments present in the sub-image. Fortunately, this task is made easier because the number of segments is likely to be small.

A small sub-image implies small segments. Global segmentation algorithms, in an attempt to reduce over-segmentation, often disallow segments containing fewer than a specified number of pixels. A local segmentation algorithm should expect to diagnose many small segments, including those consisting of a single pixel. Figure 3.5 shows a alternate 3×3 sub-image taken from a whole image. The one pixel segment may be considered noise by global segmentation, but local segmentation must be more lenient and allow for the fact that a lone pixel may be part of a larger global segment.

Global segmentation deals mostly with segments consisting of a relatively large number of pixels. This makes estimated parameter values for global segments naturally more robust.

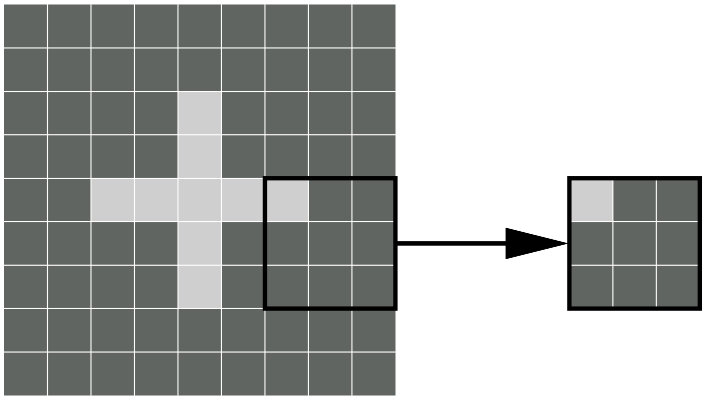

Figure 3.5: A 3×3 sub-image taken from the right of the 9×9 image with two segments.

Local segmentation must be frugal in its demands for pixel data. An important factor is the processing of pixels from the image margins because they have incomplete neighbourhood information. For a 3×3 window applied to an $N\times N$ image, the proportion of affected pixels is $4(N-1)/N^2$. For whole images ($N \gg 50$), this is usually insignificant, but for sub-images ($N \ll 10$) being locally segmented, this proportion can be very high. For the common 3×3 window the proportion is $8/9$ — nearly every pixel in the block.

Although the underlying assumptions of local and global segmentation are different, any good local segmentation algorithm will still owe a debt to its global counterparts. By 1994 over 1000 segmentation algorithms had been published [ZG94] and this number has likely doubled since. The sheer variety of algorithms makes it difficult to neatly classify them, so in this thesis they will be examined in terms of the following attributes:

- whether or not spatial information is used
- suitability to small sub-images
- ability to detect the number of segments automatically
- underlying segment and noise model
- region growing (identify homogeneity) or edge following (identify heterogeneity)
- attempt to optimize an objective criterion
- time and space complexity
- parallel or sequential execution [Ros81]

In the following sections those global segmentation algorithms relevant to the development of good local segmentation algorithms are examined. The discussion is broken into two main parts. Section 3.2.1 deals only with clustering techniques (non-spatial segmentation), while Section 3.2.2 covers methods which incorporate spatial information.

3.2.1 Clustering

Segmentation which does not use spatial information is sometimes called clustering. Clustering was used in numerical taxonomy and multivariate analysis long before electronic computers existed [Fis36]. Image processing has adapted clustering algorithms from other disciplines by treating pixel values as independent variables. For example, colour pixels have 3 attributes and each pixel can be considered a point in 3-dimensional space. Clustering involves grouping pixels in this multivariate space.

For the case of greyscale intensity data, clustering into M groups reduces to the simpler case of estimating $M-1$ *thresholds*. Each threshold T_1 to T_{M-1} is an intensity value which acts as a dividing point between neighbouring clusters. Typically, each pixel in the thresholded image, $t_M(x,y)$, is set to a unique intensity, L_i, associated with each of the M clusters. Equation 3.1 expresses this formally.

$$t_M(x,y) = \begin{cases} L_1 & \text{if} & & & f(x,y) & \leq & T_1 \\ L_2 & \text{if} & T_1 & < & f(x,y) & \leq & T_2 \\ & & & \vdots & & & \\ L_{i+1} & \text{if} & T_i & < & f(x,y) & \leq & T_{i+1} \\ & & & \vdots & & & \\ L_M & \text{if} & T_{M-1} & < & f(x,y) & & \end{cases} \tag{3.1}$$

Thresholding assumes that pixels from different segments form separate populations based solely on the disparity of their intensities. It is well suited to images containing relatively few objects and low amounts of noise. If there is a large number of segments, or large variation within segments, it is more likely that the segments' pixel value distributions will overlap, rendering valleys in the histogram less obvious or even non-existent [LCP90]. Thresholding would be a good candidate for local segmentation, because on a small scale only a small number of segments are expected. Thresholding is suited also to a variety of homogeneity criteria, but with one caveat: an intensity used by one cluster may not be used by another.

Binary thresholding

Thresholding is one of the oldest, simplest and most popular techniques used in image processing [PP93]. Most of the thresholding literature is concerned with classifying pixels into object or background classes [Wes78, FM81, SSW88]. This is known as *binary* or *bi-level* thresholding. Algorithms which deal with three or more classes are called *multilevel* thresholding techniques [RRK84, PG94]. On a local scale, most sub-images are expected to be homogeneous. This implies that the next most likely situation is sub-images with two segments. Thus an examination of binary clustering will be useful.

If an image consists of two or more clear objects, the histogram should have a corresponding number of peaks. The thresholds should be chosen at the valleys of the histogram. For bi-level thresholding, Prewitt *et al* [PM66] repeatedly smoothed the histogram until a single minimum existed between two maxima, and chose T as the intensity corresponding to the minimum. An example of this method is given in Figure 3.6.

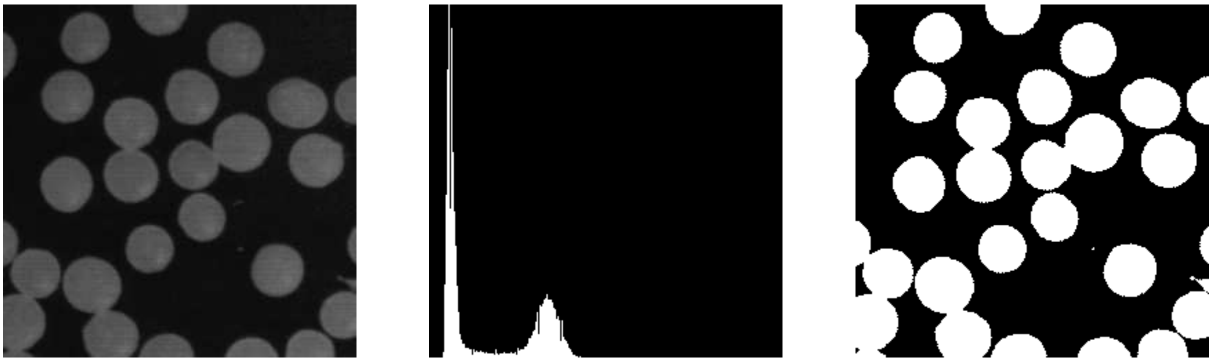

Figure 3.6: (a) the 8 bpp pellets image; (b) its histogram; (c) thresholded using $T = 46$.

Entropic methods treat the histogram as a probability distribution of Z symbols. Kapur *et al* [KSW85] split the histogram into two parts and compute the entropy[1] of each distribution. The optimal threshold, T, is chosen to maximize the sum of the entropies of the two parts.

[1] The entropy measures the average information content of a set of symbols, assuming the probability of each symbol is known. If the probability of symbol s_i is $\Pr(s_i)$, and there are N symbols, the entropy (in bits) is given by the expression: $-\sum_{i=1}^{N} \Pr(s_i) \log_2 \Pr(s_i)$.

The aim is to retain as much information as possible in the binarized image by choosing a split in which each of the two distributions are as uniform as possible. This method has the advantage of not having to estimate any parameters, but does not help us decide on the bimodality of the histogram.

The trend in thresholding has been toward mixture modeling, which treats an image histogram as a linear blend of parameterized statistical distributions [EH81, TSM85, MP00]. Kittler's minimum error method assumes the histogram is a mixture of Gaussians with separate means and variances [KI86]. The chosen objective criterion minimizes the classification error. The resulting threshold corresponds to the intersection point of the two Gaussians, shown in Figure 3.7. This point is also known as the Bayes minimum error threshold, and Kittler provides both exhaustive and iterative search procedures for determining it [KI86].

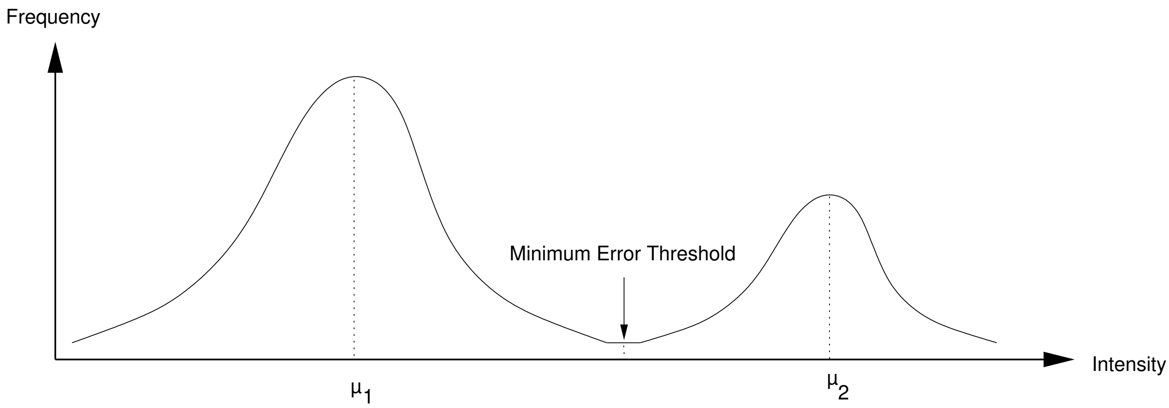

Figure 3.7: Determination of the threshold in mixture modeling.

The minimum error method replaced the previously popular method of Otsu [Ots79] and its fast implementation [RRK84]. It was shown by Kurita *et al* [KOA92] that Otsu's method is actually equivalent to the Kittler's minimum error method if each distribution in the mixture has the same form and the same variance. An adjustment to remove a mild bias of the variance estimates due to overlapping of the distributions was provided by Cho *et al* [CHY89].

In the minimum error paper, Kittler *et al* describe the problem of encountering a homogeneous image. In this case the histogram would be unimodal, causing the optimal threshold to always be chosen at either the extreme left or extreme right of the histogram. To some extent this condition can be used to distinguish between homogeneous and heterogeneous images. This is important in terms of local segmentation as a large proportion of the sub-images are expected to be homogeneous and should be treated as such.

Methods to evaluate binary thresholds objectively do exist [WR78]. Sahoo *et al* [SSW88] use "uniformity" and "shape" measures to compare eight different algorithms. The uniformity measure is concerned only with minimizing the sum of intra-class variances, while the shape measure incorporates spatial coherence of edge boundaries. Using real images, which were not bimodal, they found the minimum error method and the entropic method to perform best. They also point out that it would be trivial to design a new thresholding algorithm which jointly optimizes itself with respect to the shape and uniformity measures. This severely limits the usefulness of these objective criteria.

Lee *et al* [LCP90] examined five global thresholding algorithms. They used two test images for which the correct binary segmentation was known. In addition to the shape and uniformity measures, they measured the probability of mis-classification relative to the correct segmentation. They stated that no algorithm obviously stood out, and lamented on the difficulty of comparing algorithms across different types of data. Despite this, they decided that Otsu's method [Ots79] was the best overall. This suggests that the minimum error method would have done as well if not better. Glasbey [Gla93] also performed a similar experiment by generating artificial histograms from mixtures of Gaussian distributions with different means and variances. He found the iterated form of the minimum error method to do best. This is not surprising given the way the data was generated.

Local thresholding

Global thresholding methods use the same threshold for every pixel in an image. Difficulties arise when the illumination of a scene varies across the image [GW92]. In Figure 3.8 the mug has been thresholded at $T = 171$, corresponding to the most prominent valley in its histogram. The resulting segmentation is poor, because the shadow and the darker background area have been grouped with the mug. To counteract this, a different threshold, $T(x, y)$, can be used for each pixel in (or region of) the image. This *dynamic threshold* could be based on the histogram of an appropriately sized sub-image encompassing each pixel. This is known as adaptive, variable or *local* thresholding [CK72].

Nakagawa *et al* [NR79] divide an image into non-overlapping windows. The histogram is assumed to be a mixture of Gaussians, but the mixture parameters are estimated under the

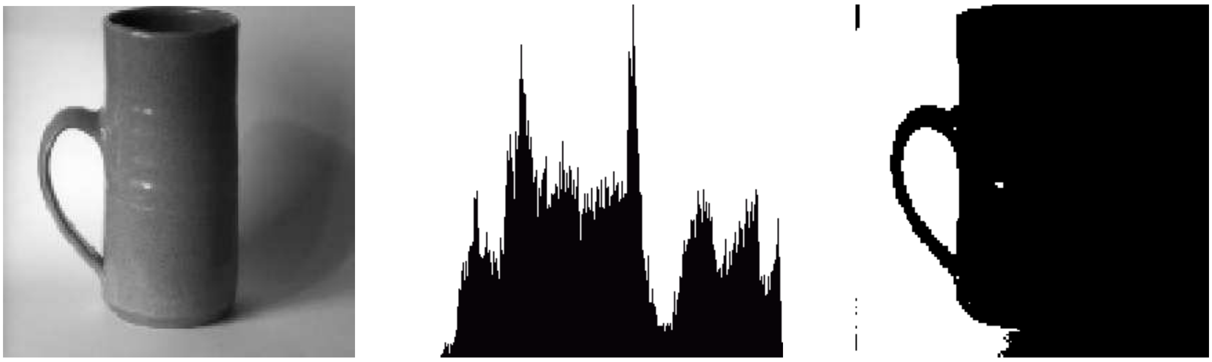

Figure 3.8: (a) the mug image; (b) its histogram; (c) thresholded using $T = 171$.

assumption of no overlap. A test for bimodality (two segments) versus unimodality (one segment) is then applied. The test involves three criteria and four user supplied numbers, and is designed to ensure good separation and meaningful variances. If the window is bimodal, the threshold is chosen to minimize the probability of mis-classification. For unimodal windows, the threshold was chosen by linearly interpolating neighbouring thresholds. An extension to windows of three segments is also described, where neighbouring threshold information aids in choosing which of the two thresholds to choose.

Local thresholding is clearly exploiting the principle of local segmentation. Obviously, any global thresholding technique could be used for determining each local threshold. Smaller windows increase the chance of obtaining a unimodal histogram, corresponding to a homogeneous (or extremely noisy) region. If a method is unable to detect this situation, the threshold it computes could be nonsensical. It is important to be able to determine the number of segments present in a local window.

Multilevel thresholding

Some binary thresholding techniques can be adapted to function with more than two clusters [RRK84, KI86]. The mixture modeling approach extends easily, because it models a histogram as a blend of distributions, choosing thresholds at the valleys between the peaks of distributions. Equation 3.2 gives the general form of a mixture model with k classes. The π_is are the mixing weights which must sum to unity, $g_i(\cdot)$ is the distribution function for class i, and $\vec{\theta_i}$ holds the parameters of distribution i.

$$h(x) = \sum_{i=1}^{k} \pi_i \cdot g_i(x; \vec{\theta}_i) \quad (3.2)$$

Often each class is assumed normally distributed with unknown mean and variance. In this situation $h(x)$ has $3k - 1$ degrees of freedom. These parameters are usually estimated using the Expectation-Maximization (E.M.) algorithm [DLR77]. The E.M. algorithm is an iterative process which is guaranteed to increase, at each step, the statistical likelihood [Fis12] of the data given the mixture model probability density function, $h(x)$. Although the E.M. algorithm always converges, it may do so only to a local optimum. Usually, multiple runs at different starting conditions are required to overcome this. However, greyscale pixel data is a simple one-dimensional case, and these problems usually only occur in multivariate situations, such as colour pixels.

The general E.M. algorithm is computationally very expensive because it calculates k integrals per datum per step. The integrals are used to determine the partial membership of each datum to each distribution. For the normal distribution this must be done numerically as no closed form exists. If the distributions are well separated, it is possible to assign each pixel wholly to one class without biasing the parameter estimates. If, additionally, the distributions are assumed normal with a common variance, the maximum likelihood criterion optimized by the E.M. algorithm reduces to minimizing the sum of squares of each datum to its centroid [Alp98]. This criterion may be optimized using simpler techniques.

The k-means algorithm really refers to two different algorithms, both designed to minimize the sum of squared errors of each datum to its cluster centroid. The simpler form, referred to as H-means by Hansen *et al* [HM01], is given in Listing 3.1. Hansen *et al* refers to the more complex form as k-means [McQ67, DH73, HW79], given in Listing 3.2.

1. Choose k initial class means.
2. Assign each pixel to the class with the closest mean.
3. Recompute the class means using the new assignments.
4. If any pixels have switched class, go to Step 2.

Listing 3.1: The H-means algorithm.

1. Choose k initial class means.
2. Assign each pixel to the class with the closest mean.
3. Reassign one pixel to a new class such that the move best reduces the overall sum of square distances criterion.
4. Recompute the class means using the new assignments.
5. If the criterion was improved, go to Step 3.

Listing 3.2: The k-means algorithm.

In H-means, every pixel is reassigned per iteration, whereas k-means only performs one optimal re-assignment. For one-dimensional data like greyscale pixels, both algorithms should produce the same clustering if provided with sensible starting conditions. This is especially true when k is low, as it is expected to be in local segmentation.

The ISODATA [RC78] algorithm is similar to k-means but introduces various heuristics for improving the clustering result. These heuristics include forcing a bound on the minimum and maximum number of classes, requiring a minimum class membership amount, and restricting the maximum variance within a class. Although these heuristics must be supplied as parameters by the user, they do allow the ISODATA algorithm to go some way to estimating the number of classes and ensuring they have sensible properties. They also allow the user to incorporate *a priori* information into the clustering process. The natural level of variation within the image, if known, could be used to restrict the maximum variance within a class.

Fuzzy c-means [Bez81] is a variation on k-means which allows each datum *partial membership* in each cluster, similar to a mixture model. Compared to the sudden jumps during re-assignment in k-means, the cluster centroids in fuzzy c-means move more smoothly. Its objective function alss differs, claiming to better suited to non-spherically distributed data [HB88]. Fuzzy c-means must calculate partial memberships for each datum at each step, which, from a computational perspective, may make it unsuited to local segmentation.

Summary

Thresholding techniques produce segments having pixels with similar intensities. They can handle any underlying image model as long as each intensity is associated with only one segment. The disregard of spatial information means that pixels within a segment may not be connected. In terms of local segmentation this may be an advantage, because the disconnected components of a local segment could actually be part of the same global segment. Global thresholding will suffer when pixels from different segments overlap in their use of intensities. If this is due to noise, a technique such as the minimum error method can estimate the underlying cluster parameters and choose the thresholds to minimize the classification error. If the overlap is due to variation in illumination across the image, variable thresholding could be used. This can be seen as a form of local segmentation.

The likelihood of encountering unimodal histograms is high when thresholding sub-images. Within the local segmentation framework it is extremely important to be able to detect the number of segments in the window. Most of the thresholding literature either ignores this situation, or handles it with special parameters which must be provided by the user. It would be better if these parameters were estimated from the image itself.

Thresholding has low space complexity. Only one pass through the image is required to build a histogram. All further calculations are performed using only the histogram, as no spatial information is required. For this reason its time complexity is also low. If the number of pixels in the image is low compared to the number of possible grey levels ($X \times Y \ll Z$), the same benefits apply without using a histogram. Local segmentation must be applied independently to each pixel, making thresholding an attractive option.

3.2.2 Spatial segmentation

Global segmentation algorithms which take spatial information into consideration usually outperform their clustering-based counterparts. Spatial information is useful because most segments corresponding to real world objects consist of pixels which are spatially connected. Evaluating the quality of segmentation algorithms is an inherently difficult problem [PP93, Zha97]. This section will discuss the main approaches to spatial segmentation and assess their suitability to local segmentation.

Region based segmentation

Region growing algorithms tackle segmentation by identifying homogeneous, spatially connected sets of pixels within an image. *Splitting algorithms* begin with large image regions and break them up into smaller, more homogeneous ones. *Merging algorithms* compare neighbouring regions (or pixels) and merge them if they have similar enough properties. Some techniques use a combination of splitting and merging. What they all have in common is the use of a measurable image property as a criterion for splitting and merging [YG01].

Region growing requires a seed to begin. Ideally the seed would be a region, but it could be a single pixel. A new segment is grown from the seed by assimilating as many neighbouring pixels as possible that meet the homogeneity criterion. The resultant segment is then removed from the process. A new seed is chosen from the remaining pixels. This continues until all pixels have been allocated to a segment. As pixels are aggregated, the parameters for each segment have to be updated. Thus the resulting segmentation could depend heavily on the initial seed chosen and the order in which neighbouring pixels are examined.

Region splitting begins with a whole image and divides it up such that the parts are each more homogeneous than the whole. The main difficulty is in deciding on where to make a partition. Early approaches used a regular decomposition such as a quad tree [FKN80]. Splitting alone is insufficient for reasonable segmentation as it severely limits the shapes of segments. Horowtiz *et al* [HP74] suggest a merging phase after splitting is complete. This can be done efficiently using tree searching algorithms [Knu73], but the resulting regions may have unnatural blocky boundaries.

A popular image model used today is the *piece-wise constant* segment model with additive Gaussian noise. Under this model, pixels from the same segment are normally distributed with mean μ and variance σ^2. The variance provides a measure of homogeneity within the segment. The sample mean and variance may be used to estimate μ and σ. Fisher's criterion [YG01] in Equation 3.3 is an example of a homogeneity predicate for splitting or merging two classes, A and B.

$$\lambda = \frac{|\mu_A - \mu_B|^2}{\sigma_A^2 + \sigma_B^2} \tag{3.3}$$

Fisher's λ is maximized by classes with widely separate means and low variance. Typically λ would be compared to a threshold — if it is lower, merging would proceed. The value of the threshold impacts the number and size of segments accepted. Ideally, its value should depend on the natural level of variation of pixel values in the image. From a local segmentation point of view, Fisher's criterion could useful for distinguishing between the existence of one or more clusters in a sub-image.

Edge based segmentation

Edge based segmentation is the dual to region growing. It exploits spatial information by detecting the edges in an image, which correspond to discontinuities in the homogeneity criterion for segments. Edge detection is usually done with local linear gradient operators, such as the Prewitt [PM66], Sobel [Sob70] and Laplacian [GW92] filters. These operators work well for images with sharp edges and low amounts of noise. For noisy, busy images they may produce false and missing edges. The detected boundaries may not necessarily form a set of closed connected curves, so some edge linking may need to be required [Can86].

There are many edge based algorithms in the literature, particularly in the area of active contours and snakes [KWT87]. However, the application of edge models to local regions is unlikely to be successful as local regions contain too few pixels, of which most have incomplete neighbour information.

Hybrid region-edge approaches

Watershed segmentation [MB90, HRA96] has become popular recently, assisted by the existence of an efficient implementation [VS91]. It is a hybrid approach which uses edge detection to determine seeds for growing regions, which are then merged. First an edge detector is applied to the image, resulting in an "elevation map" which quantifies the edge magnitude around each pixel. Seeds for region growing are taken from the points of lowest elevation, the so-called "basins". Region growing proceeds in a manner similar to how the basins would fill if the elevation map was slowly and uniformly covered with water. When the "immersion" process completes, the basins are merged into suitably sized global segments.

The edge detection and region growing steps in watershed segmentation are only loosely coupled, because the elevation map can be provided beforehand. Other hybrid techniques tightly integrate these steps, for example, Tabb's multi-scale image segmentation approach [TA97]. Tabb suggests that scale selection and edge and region detection can not be separated, and that good segmentation algorithms effectively have to perform a Gestalt analysis. Although this approach produces pleasing results on whole images, it is difficult to apply on a local level where only a small number of pixels are available. His algorithm does have the admirable feature of automatically determining all required parameters from the image itself.

Relaxation labeling

Relaxation labeling can be applied to many areas of computer vision. In the past it has been applied to scientific computing applications, particularly to the solution of simultaneous nonlinear equations [RHZ76]. The basic elements of the relaxation labeling method are a set of features belonging to an object and a set of labels. In the wider context of image processing, these features are usually points, edges and surfaces. For the case of image segmentation each pixel has a feature describing which segment it belongs to, and there is a different label for each segment in the image.

The labeling schemes are usually probabilistic in that each pixel is partially assigned to all segments. A pixel's assignment takes the form of normalized probabilities which estimate the likelihood of it belonging to each segment. Different techniques are used to maximize (or minimize) the probabilities by iterative adjustment, taking into account the probabilities associated with neighbouring features. Relaxation strategies do not necessarily guarantee convergence, thus pixels may not end up with full membership in a single segment.

Two main strategies are used for optimizing the iteration process [YG01]. Simulated annealing, or stochastic relaxation, introduces a random element into the iteration scheme, to reduce the chance of becoming stuck in secondary optima. Geman *et al* [GG84] applied this to a local Markov random field model [Li01]. In contrast to stochastic relaxation, which makes random changes, deterministic relaxation only makes changes which improve the configuration, and thus converges much faster. Besag's Iterated Conditional Modes (ICM) [Bes86] algorithm uses this approach for image segmentation. Due to its greedy nature, ICM does not find a global minimum, and its results are dependent on the initial conditions chosen.

Extending clustering methods

Haralick *et al* [HSD73] used the *co-occurrence matrix* to incorporate the spatial relationship between pixels into the choice of threshold in texture analysis. This matrix is constructed as a 2-D histogram from *pairs* of pixels at a fixed orientation and distance apart, similar to a Markov model. Figure 3.9 gives an example of two co-occurrence matrices for a simple 4×4 image consisting of three noiseless segments. Pixels within segments should dominate the diagonal entries of the matrix (similar grey levels) while the off-diagonal entries should be made up of edge pixels (differing grey levels). By creating two separate histograms from these groups and finding where the valley in the first matches a peak in the second, a threshold can be obtained. The interested reader is directed to the paper for further information. This approach is not suited to small sub-images as there are too few pixels to generate a meaningful co-occurrence matrix.

0	0	1	1
0	0	1	1
0	0	2	2
0	0	2	2

a \ b	0	1	2
0	4	2	2
1	0	2	0
2	0	0	2

a b

a \ b	0	1	2
0	6	0	0
1	0	2	2
2	0	0	2

a
b

Figure 3.9: (a) 4×4 image with 3 segments; (b) co-occurrence matrix for horizontal pairs of pixels; (c) co-occurrence matrix for vertical pairs of pixels.

Leung and Lam [LL96] introduce the concept of *segmented-scene spatial entropy* (SSE) to extend the entropic thresholding methods to make use of spatial information. Thresholds are chosen to maximize the amount of information in the "spatial structure" of the segmented image. Their spatial model relates a pixel to its four immediate neighbours. This method is too complicated to apply to small sets of pixels with little connectedness information.

An interesting approach to bi-level thresholding was taken by Lindarto [Lin96]. Traditional entropic methods try to maximize the information needed to encode the pixel values once the segmentation is determined. Lindarto, however, chooses a threshold which minimizes the combined lossless encoding length (information content) of the binary segment map *and* the original pixels. The grey level pixels are encoded using a least-entropy linear predictor [Tis94] optimized for each segment, while the bitmap is efficiently stored using the JBIG algorithm [PMGL88]. This approach uses Wallace's minimum message length (MML) inductive inference principle [WB68, OH94] to choose the model (threshold) which minimizes the overall compressed length of the model plus the data given the model. The versatile MML model selection technique will be applied to local segmentation in Chapter 5.

Summary

Segmentation algorithms which use spatial information can produce better results than those which do not. This is especially true for noisy images, where pixel intensity alone is insufficient for distinguishing between segments. However, spatial segmentation algorithms would struggle with the low amount of spatial information available in a small window. Local segmentation must be applied to every pixel, and hence a simple and fast algorithm is also desirable. Thresholding techniques are therefore an attractive option for local segmentation.

3.3 Local segmentation

Local segmentation is concerned with segmenting small sub-images in isolation. The main difference between local segmentation and global segmentation is that the former has only a small number of pixels available to it. This increases the chance that the sub-image is homogeneous. If an edge is present, it may not be as obvious without the context of a whole image. The concept of modeling and segmenting local regions is not new. Many algorithms already use it in some form, but its application may be implicit and non-obvious.

3.3.1 The facet model

Most image processing applications assume that the pixel grid is a discrete approximation to an underlying intensity surface. The *facet model* of Haralick *et al* [HW81] assumes that the true surface is piece-wise continuous and that the image is a noisy sampled version of it. In particular, it assumes that each segment in the image is well modeled using a two-dimensional polynomial function. The *flat facet* model assumes each piece has a constant intensity. The *sloped facet* model allows the intensity to vary linearly in both directions, like a plane. The idea extends easily to higher degree polynomials. It would be possible to fit facet models to sub-images.

There are two main problems with the facet model. Firstly, it does not provide a criterion for deciding which order polynomial best fits the local region. It is important to strike a balance between over-fitting the noise and under-fitting the structure. Secondly, polynomial functions are not well suited to modeling sharp edges between arbitrarily shaped segments. Segment boundaries are visually very important and it is important to handle them correctly.

3.3.2 Block truncation coding

One of the oldest appearances of local segmentation is the lossy, fixed bit rate image compression technique *block truncation coding*, or BTC [DM79, FNK94]. In one form, BTC divides an image into non-overlapping 4×4 blocks pixels. The pixels are segmented into two clusters using thresholding at the block mean. The two cluster centroids, μ_0 and μ_1, are set average value of the pixels in each cluster. Figure 3.10 provides an example.

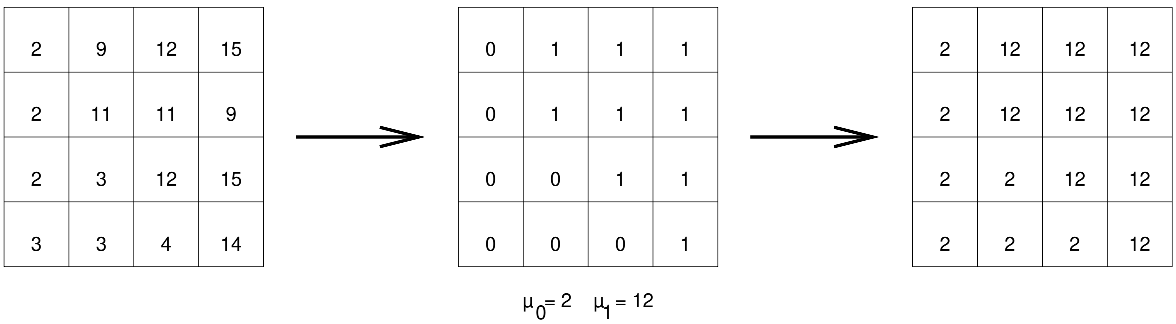

Figure 3.10: BTC example: (a) original 4×4 pixel block; (b) decomposed into a bit-plane and two means; (c) reconstructed pixels.

From a local segmentation perspective, BTC assumes that segments consist of pixels of a common grey level separated by step edges aligned to pixel boundaries. The assumption of two segments implies the existence of a bimodal histogram, and the use of the mean as the threshold implies that each segment contains approximately the same number of pixels.

The original BTC algorithm has been modified repeatedly over the years [FNK94]. In standard BTC, the number of segments is always two. Some techniques allow the number of segments to be varied on a block by block basis. For example, if the difference between the two means, $|\mu_1 - \mu_0|$, is low enough, the block is considered homogeneous, so no bitmap and only one mean are encoded. Conversely, if the block variance is very high then multilevel thresholding can be used. This approach to automatically detecting the number of clusters is rudimentary, but still effective. In Chapter 4, this idea will be extended to form the basis of an effective local segmentation technique for removing noise from images.

Some BTC variants replace thresholding with spatial techniques designed to better preserve image structure. Gradient based BTC [QS95] and correlation based BTC [Cai96] choose a different threshold for each *pixel* in the block. Each threshold depends on neighbouring gradient and pixel difference information. This itself is a crude form of relaxation labeling. For most images the gain is minimal, suggesting that spatial information may be of limited value when segmenting small regions.

3.3.3 SUSAN

SUSAN [Smi95, Smi96, SB97], much like the local segmentation framework of this thesis, evolved to be a general approach to low level image processing. One of SUSAN's design constraints was efficiency, so that it could be run on real time data in a robotic vision system. SUSAN was originally designed for edge and corner detection, but was also adapted for structure preserving denoising. The denoising component differs a little from the SUSAN framework presented here, and is treated separately in Section 3.4.5.

SUSAN processes greyscale images in a local manner using an approximately circular window containing 37 pixels. The centre pixel in the window is called the *nucleus*. Neighbouring pixels similar in intensity to the nucleus are grouped into an USAN, or *Univalue Segment Assimilating Nucleus*. The USAN creation process may appear to be a form of region growing using the nucleus as a seed, except that pixels in the USAN are not necessarily connected. The formation of an USAN is more related to clustering than segmentation.

The pixel similarity function is controlled by a brightness threshold, t. Smith first describes the USAN as accepting only pixels with an intensity difference of t units from the nucleus.

This can be considered *crisp clustering*, where similar pixels receive full weight, and dissimilar pixels receive no weight. The *size* of the USAN is equal to the sum of weights given to pixels in the window, which for crisp clustering, equals the number of pixels assimilated. Smith, however, found a *fuzzy* membership function to give better feature detection. This fuzzy function was "optimally" derived to have the form $e^{-(\Delta/t)^6}$, where Δ is the intensity difference from the nucleus. Figure 3.11 compares the crisp and fuzzy membership functions as a function of Δ. When the fuzzy form is used, all pixels have partial contribution to the size of the USAN. This is no longer an explicit segmentation of the local neighbourhood.

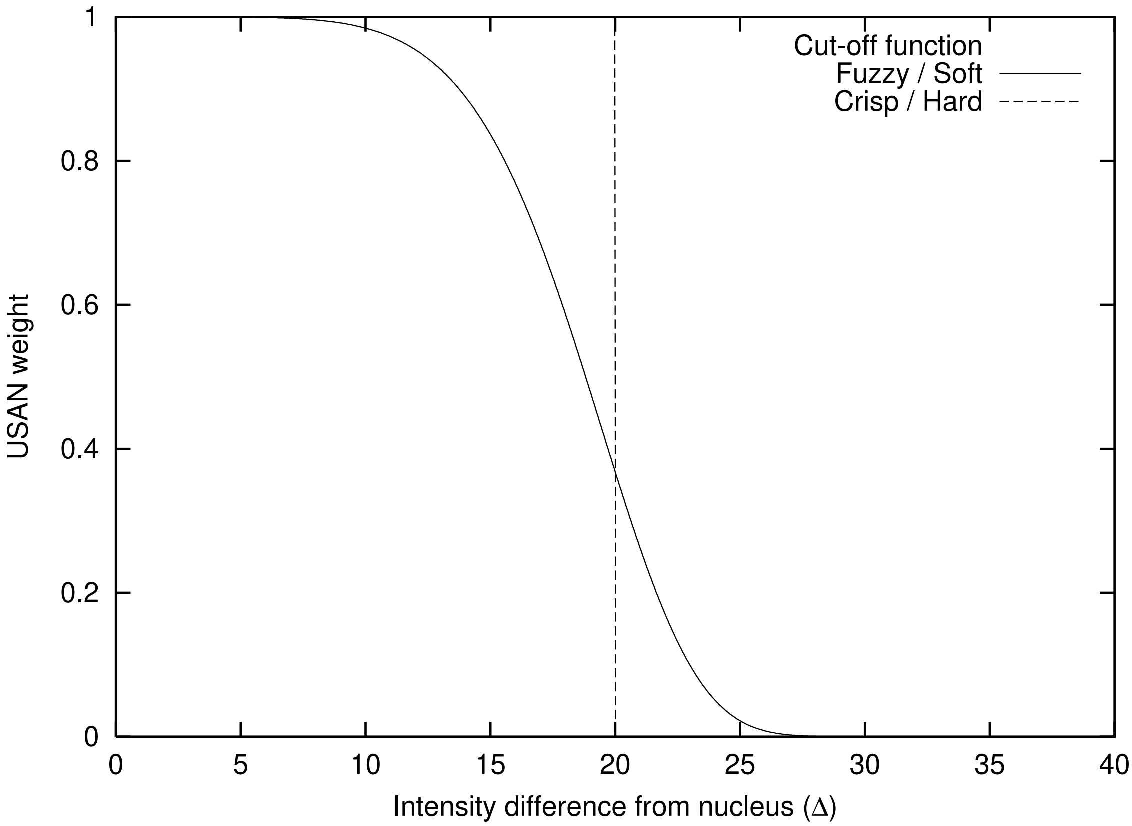

Figure 3.11: Comparison of the hard and soft cut-off functions for pixel assimilation in the SUSAN algorithm.

To perform feature detection, the USAN size for each pixel is plotted. On this surface homogeneous regions correspond to plateaus, edges to valleys, and corners to deeper valleys. SUSAN processes this USAN surface to determine the directions and strengths of edges and the positions of corners. Smith finds the SUSAN feature detector to give results on a par with Canny's famous edge detection algorithm [Can86]. Smith states that SUSAN is suited to images with segments of smoothly varying intensity and both step and ramp edges.

The main difficulty with SUSAN is that its performance is dependent on a user supplied brightness threshold, t. Smith claims that SUSAN is relatively insensitive to the choice of t, and advocates a default of 20. He states that t may be used to vary SUSAN's sensitivity to features. The value of t really controls the minimum feature contrast that can be detected. My experiments show that t should be proportional to the natural level of intensity variation expected within image segments. This is the only way for the USAN to assimilate pixels in homogeneous regions and not produce false edge output.

3.4 Denoising

Image denoising is the process of removing unwanted noise from an image [Mas85, BC90]. The noise can take a variety of forms and is introduced in differing amounts at each step during the acquisition of the image. The literature abounds with denoising techniques, but they may be classified broadly into temporal and spatial approaches.

3.4.1 Temporal filtering

Temporal filtering averages multiple sampled versions of *the same scene* [She90]. If the true image is $\mathbf{f}$, and N samples, $\mathbf{g}_1 \ldots \mathbf{g}_N$, are taken, the temporally filtered image, $\mathbf{f}'$, can be calculated using Equation 3.4.

$$\mathbf{f}' = \frac{1}{N} \sum_{i=1}^{N} \mathbf{g}_i \tag{3.4}$$

If each pixel in $\mathbf{f}$ was corrupted by the addition of symmetric zero-mean noise of variance σ^2, then for any one pixel the expected noise variance, $\mathrm{E}[\,(\mathbf{f} - \mathbf{g}_i)^2\,]$, is σ^2. However, the expected noise for the *ensemble average*, $\mathbf{f}'$, reduces to σ^2/N. If the noise was not additive in nature, for example impulse noise, then a better "averaging" function, such as the median (discussed later), should be used.

Temporal filtering is ideal if multiple version of the same image can be obtained. In practice this does not usually happen, because the objects in the scene move, or the capturing equipment wobbles. Even slight variations can displace the pixels in each sampled image, causing the ensemble average to become a blurred version of the original.

In this thesis it is assumed that only a single noisy version of an original image is available, so temporal filtering of the form in Equation 3.4 can not be used. However, it will be shown how overlapping local segmentation results for the same pixel coordinate can be combined to suppress noise in a fashion similar to temporal filtering.

3.4.2 Spatial filtering

Where temporal filtering uses multiple versions of each pixel value at the same position but different "times", *spatial filtering* uses pixels at the same "time" but at different spatial coordinates in the image. When temporally filtering a static scene, pixels from the same position but different times are expected to have the same noiseless value. When those pixels are not available, the best alternative is pixels *near* the current pixel. When only a single noisy image is available, spatial filtering is the only option. This is the basis of nearly all image denoising algorithms.

The simplest spatial filter is the *averaging filter* which replaces a pixel with the average of itself and the pixels in the local neighbourhood. The most common averaging filter is the *box filter* [McD81], which gives equal weight to the 9 pixels in a 3×3 window. This choice of weights maximally reduces the noise variance when the noise is additive. Box filtering is similar to temporal filtering except that samples from adjacent pixel positions are used as approximations to multiple samples of the pixel at the same position. Figure 3.12 gives an example of its application.

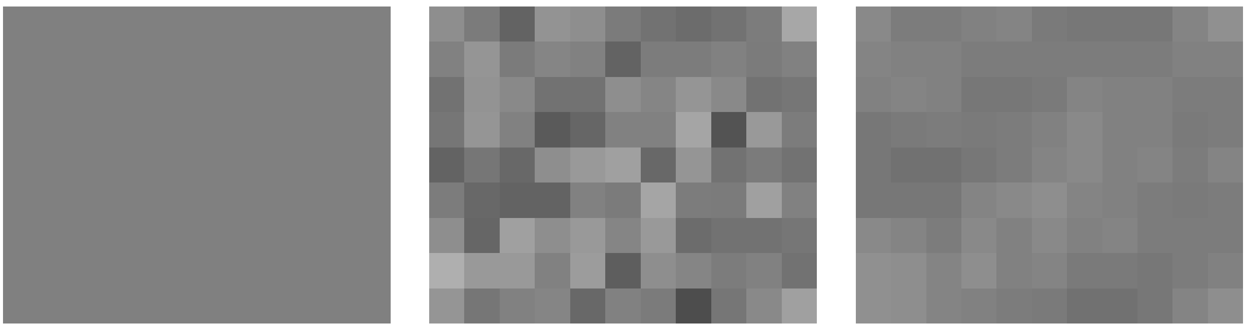

Figure 3.12: Application of the 3×3 box filter: (a) original; (b) noisy; (c) filtered.

Although an averaging filter performs well for additive noise in homogeneous regions, it tends to blur edges and other image structure in heterogeneous regions. Figure 3.13 gives an example of this undesirable behaviour. To combat this deficiency, most research is concerned with *structure preserving* denoising algorithms.

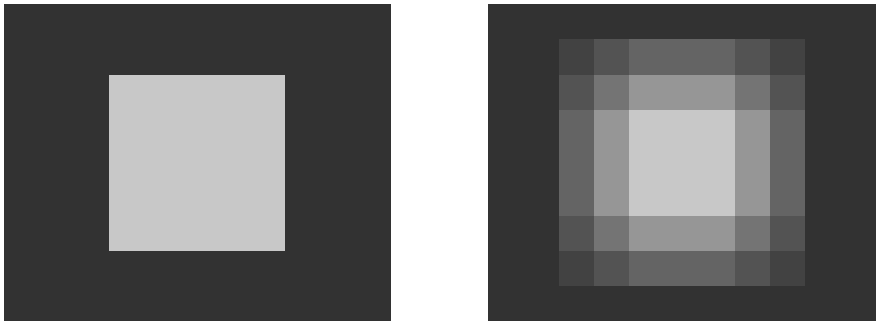

Figure 3.13: Box filtering in the presence of edges: (a) original image; (b) filtered.

To preserve image structure while removing noise implies the ability to distinguish between the two. It will be shown in later chapters that local segmentation is a good basis for modeling the relationship between structure and noise in an image. Segments correspond to homogeneous regions and boundaries between segments are structure that needs to be preserved. Existing spatial denoising algorithms can be examined in terms of the way they use local segmentation. Some use no local segmentation at all, some use it implicitly, and some explicitly use a local segmentation model.

3.4.3 Fixed linear filters

The simplest denoising filters are linear ones. For a square window of side length $2N+1$, the operation of a linear filter is described by Equation 3.5, where the weights, $w(\cdot)$, are assumed to sum to one.

$$f'(x,y) = \sum_{\delta_x=-N}^{N} \sum_{\delta_y=-N}^{N} w(x',y') f(x+\delta_x, y+\delta_y) \tag{3.5}$$

The box filter described in Section 3.4.2 is a linear filter with $N = 1$ and equal weights $w(\delta_x, \delta_y) = 1/(2N+1)^2$. Its *configuration* is illustrated in Figure 3.14a. It shows the spatial arrangements of the weights used by the linear filter. Interestingly, McDonnell [McD81] developed a recursive algorithm which can implement box filtering with only five operations per pixel, *independent of* N.

$$\frac{1}{9}\begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array} \qquad\qquad \frac{1}{16}\begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 2 & 4 & 2 \\ \hline 1 & 2 & 1 \\ \hline \end{array}$$

Figure 3.14: (a) 3×3 box filter; (b) approximate 3×3 Gaussian filter.

Gaussian filters use normalized weights of the form in Equation 3.6. This filter varies the weights such that those pixels further from the centre contribute less to the overall average. The spatial extent of a linear Gaussian filter is determined by σ, usually chosen to be around $(2N+1)/6$, as to be mostly encapsulated by the bounding square. Figure 3.14b illustrates the configuration of a 3×3 Gaussian filter in this case. Gaussian filters are sometimes preferred over box filters because their effect on the Fourier spectrum of the image is better [GW92].

$$w(\delta_x, \delta_y) \propto e^{-\frac{\delta_x^2+\delta_y^2}{2\sigma^2}} \tag{3.6}$$

From a local segmentation point of view, each linear filter weight may be interpreted as being proportional to the believed *probability* that the pixel is related to the centre pixel. “Relatedness” really means “in the same segment”. The centre pixel usually has the highest weight, as it should be the best estimator of its own original value when the noise is additive. If the centre pixel is at coordinate (x, y), and another pixel is at $(x + \delta_x, y + \delta_y)$, the implied probability that they are in the same segment is given by Equation 3.7.

$$\Pr(f'(x, y) \text{ and } f'(x + \delta_x, y + \delta_y) \text{ in same segment}) = \frac{w(\delta_x, \delta_y)}{w(0, 0)} \tag{3.7}$$

For example, the box filter trusts all the pixels in the window equally, giving them probability one of being from the same source. It has been stated that the correlation between pixels is often on the order of 0.9 to 0.95 [NH88]. From this point of view, the coefficients of the box filter are quite reasonable. For the approximate 3×3 Gaussian filter in Figure 3.14b, the side pixels are considered $2/4 = 0.5$ correlated to the centre pixel, and the corner pixels

$1/4 = 0.25$ correlated. This may be considered a primitive form of local segmentation, but one which is unable to adapt to changing image structure.

All fixed linear filters will defocus an image, blurring edges as in Figure 3.13, and attenuating narrow lines and other fine structure. The behaviour worsens as N increases. A positive feature is that smoothing is very good in homogeneous regions corrupted by additive noise, reducing the noise variance by up to a factor of $(2N+1)^2$. What is required is a method to distinguish between homogeneous and heterogeneous regions, to get the benefits of smoothing without the disadvantage of blurring edges.

3.4.4 Rank filters

Rank filters are designed to operate on a numerically ordered set of pixels. The pixels $p_1 \ldots p_N$ from the local neighbourhood are gathered and *sorted* into a new set $p_{(1)} \ldots p_{(N)}$, where $\forall_i \; p_{(i)} \leq p_{(i+1)}$. For greyscale images, the ordering is determined by pixel intensity, but there is no obvious analogue for higher dimensional data such as colour pixels. Manipulation of the ordered set of data originated as a way to improve the statistical robustness of traditional estimates (such as the mean) when the data was contaminated [MR74, Hub81]. Digital images are often contaminated by noise. Thus, to some, it seemed natural to apply rank techniques to image denoising.

The median

The simplest rank filter is the *median filter*. The median is the middle datum in a sorted data sequence [Tuk71]. Just as the mean minimizes the average squared error, the median minimizes the average absolute error. If the number of pixels is odd, the median is uniquely defined. When it is even, there are an infinite number of solutions between the two middle pixels. In this case, the average of the two middle values is usually taken as the median. Equation 3.8 summarizes the calculation of the median, p_{MED}, of M pixels.

$$p_{MED} = \begin{cases} p_{(\frac{M+1}{2})} & \text{if} \quad M \quad \text{odd} \\ \frac{1}{2}\left[p_{(\frac{M}{2})} + p_{(\frac{M}{2}+1)}\right] & \text{if} \quad M \quad \text{even} \end{cases} \tag{3.8}$$

Figure 3.15 gives an example of the median applied to a 3×3 sub-image. The pixels values are homogeneous with a small amount of additive noise. In this situation the median would choose the filtered value to be 6. The mean, $\bar{p}$, of these pixels is 5.78, which would usually be rounded to 6. When the noise is additive and symmetric, the mean and median will usually agree on the filtered value, as both estimate the centre of a distribution.

6	5	8
5	7	6
4	6	5

(4 5 5 5 $\underline{6}$ 6 6 7 8) $\quad p_{MED} = 6 \quad \bar{p} = 5.78$

Figure 3.15: Application of the median to a homogeneous block of pixels.

Consider the situation in Figure 3.16, whereby the centre pixel from Figure 3.15 has been contaminated by impulse noise. The pixel values are no longer homogeneous. A box filter would produce a filtered estimate of 13.1, the outlying pixel value 73 having pulled the mean away from the true centre. The *breakdown point* of the median is 50% because it is still able to produce useful estimates when up to 50% of pixels are contaminated. In this case, the median is 6 again — a much better estimate than the mean.

6	5	8
5	73	6
4	6	5

(4 5 5 5 $\underline{6}$ 6 6 8 73) $\quad p_{MED} = 6 \quad \bar{p} = 13.1$

Figure 3.16: Application of the median to a homogeneous block with impulse noise.

In the previous two examples the pixel blocks were meant to be homogeneous. When the noise is additive, the mean and median behave similarly. In the presence of impulse noise the median is superior to the mean, as it can cope with up to 50% contamination. A good structure preserving filter must also function well in the presence of edges. Figure 3.17 shows a 3×3 pixel block containing two *noiseless* segments. As expected, the mean averages across the edge. The median correctly chooses 3 as its filtered estimate, making it appear to have special structure preserving abilities.

Figure 3.18 gives a similar example, but now the centre pixel is on the corner of a noiseless segment. The median in this case is 8 — the value belonging to the alternate segment. This illustrates that the median is implicitly performing primitive local segmentation. In the

3	3	8
3	3	8
3	3	8

$(3\ 3\ 3\ 3\ \underline{3}\ 3\ 8\ 8\ 8)$ $\qquad p_{MED} = 3 \qquad \bar{p} = 4.67$

Figure 3.17: Application of the median to a noiseless edge block.

presence of *two distinct segments*, the median chooses a pixel from the segment containing the *majority* of the pixels from the window. For a 3×3 window, this decision rule fails whenever the intersection of the local window with the centre pixel's segment consists of fewer than five pixels. When the window contains more than two segments, the median's behaviour is a more complicated function of the segment populations.

3	3	8
3	3	8
8	8	8

$(3\ 3\ 3\ 3\ \underline{8}\ 8\ 8\ 8\ 8)$ $\qquad p_{MED} = 8 \qquad \bar{p} = 5.77$

Figure 3.18: The median (incorrectly) implicitly segmenting a noiseless corner block.

The weighted median

The simple approach to local segmentation that the median takes causes it to fail when the centre pixel belongs to a minority segment. Unfortunately, thin lines and other fine structure correspond to this situation. Instead of moving toward a local segmentation analysis of the problem, the concept of the median has been generalized to handle various special cases. A popular variant is the *centre weighted median* [Bow84]. The philosophy of the centre weighted median is to increase the size of the ordered set of pixels by adding c extra duplicates of the centre pixel, before taking the median. The value of c depends on the number of pixels in the window, and the type of structures one wishes to preserve [KL91].

For 3×3 windows, $c = 2$ could be used to preserve corner structures like the one in Figure 3.18. Figure 3.19 shows that when the centre weighted median is used, the corner pixel is correctly filtered as 3. The duplication of the centre pixel simply improves the probability that the centre pixel will belong to the majority segment. For the plain median filter, the centre pixel needs to be in a segment having at least $\lceil N/2 \rceil$ members, where $\lceil \cdot \rceil$ denotes rounding to the nearest integer. The weighted median reduces this requirement to $\lceil (N-c)/2 \rceil$ pixels. This improvement comes at the cost of reducing its breakdown point.

3	3	8
3	3	8
8	8	8

$$(3\ 3\ 3\ 3\ 3\ \underline{3}\ 8\ 8\ 8\ 8\ 8) \qquad p_{MED} = 3 \qquad \bar{p} = 5.77$$

Figure 3.19: Using the centre weighted median ($c = 3$) on a noiseless corner block.

Further variations on the median

Figure 3.20 illustrates the application of the median and centre weighted median filters to a greyscale image. Despite their obvious failure on such a basic, albeit contrived, example, these median based filters continue to be used as components in image processing systems. Many extensions to the median have been suggested to overcome its defects [Tag96, GO96, EM00]. These techniques suffer from not seriously attempting to diagnose the structural features present within the local region [KL91], or require sets of parameters to be estimated from a training set [CW00].

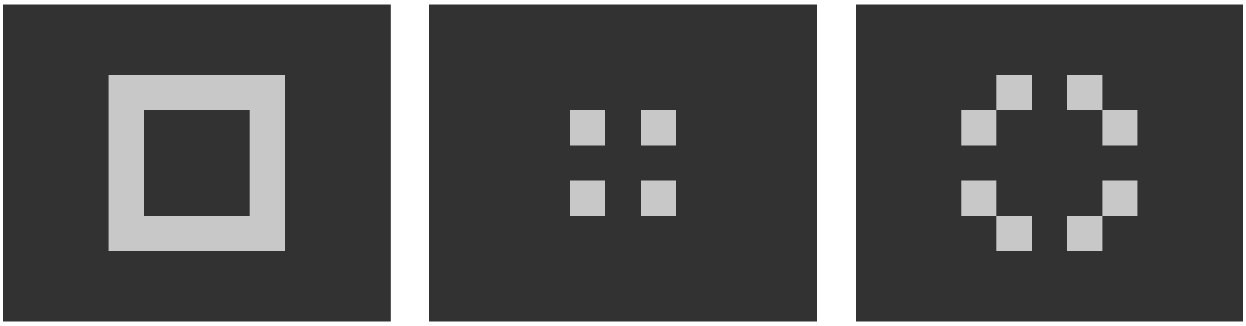

Figure 3.20: (a) original image; (b) 3×3 median filtered; (c) 3×3 weighted median, $c = 3$.

The limitations of median based approaches are not due to the median operator itself, but to its *misapplication* to heterogeneous data. The time spent on generalizing the median to two dimensional image data may have been better spent concentrating on modeling the underlying image from a local segmentation point of view. Local segmentation would suggest that pixels should first be appropriately segmented into homogeneous groups. The median could then be applied just to the pixels belonging to the segment containing the centre pixel.

3.4.5 Locally adaptive filters

An adaptive denoising algorithms adjusts the functionality of the filter for each pixel in response to an image's local properties. One would expect this to improve performance over the static methods already discussed. Most adaptive algorithms are based on either or both of the following two principles, whether they acknowledge it or not:

1. Pixels that are spatially close are more likely to be in the same segment.

2. Pixels that are similar in intensity are more likely to be in the same segment.

Adaptive denoising algorithms apply these principles in order to use only those pixels in the same segment as the centre pixel for denoising. This grouping can be seen as a partial step of local segmentation, whereby a homogeneous segment involving the centre pixel is formed. Perhaps surprising is that the same two principles are also the basis for most global image segmentation algorithms. This suggests that segmentation and denoising are highly related tasks. In Chapters 4 and 5, this fact will be used to develop local segmentation based denoising algorithms.

The Kuwahara filter

Given a $5{\times}5$ window, the Kuwahara filter [Kuw76] examines the nine unique $3{\times}3$ regions which incorporate the centre pixel, shown with a bold outline in Figure 3.21. The mean of the $3{\times}3$ window with the lowest variance is used as the denoised estimate. In terms of local segmentation, utilizing the $3{\times}3$ window with lowest variance increases the chance that the mean was computed from a homogeneous part of the image. If all global segments in the image are at least $3{\times}3$ in size, then at least one of the nine candidate regions containing the centre pixel will be truly homogeneous. Unfortunately this will not necessarily be the case.

There are many obvious cases where the Kuwahara filter is forced to choose a mean from nine heterogeneous regions, resulting in blurring. For example, in Figure 3.21 the window surrounded by a dotted line has the lowest variance. Unfortunately, that window's mean is also highly biased to an intensity not equal to that of the centre pixel.

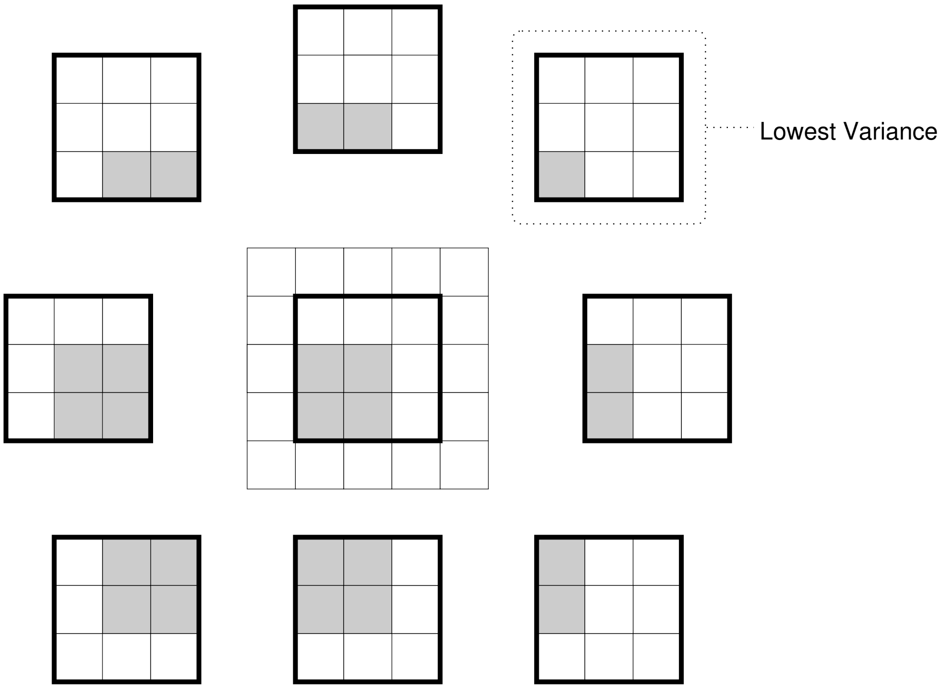

Figure 3.21: The Kuwahara filter considers nine 3×3 regions within a 5×5 window.

Despite its failings, the Kuwahara technique illustrates two important points. Firstly, the case where the pixel being processed is at the centre of the window is not the only useful case; each pixel participates in many overlapping windows. Secondly, the Kuwahara filter chooses the best fitting model from a set of candidate models using an objective criterion. Both these ideas are exploited by local segmentation denoising algorithms in Chapters 4 and 5.

Gradient inverse weighted smoothing

Gradient inverse weighted smoothing [WVL81], or GIWS, is an adaptive linear filter for greyscale images. The weight for each pixel is inversely proportional to the intensity difference between it and the centre pixel. This is shown in Equation 3.9, where $\mathbf{f}'$ is the noisy image. The $\max$ operator prevents division by zero when two pixels have the same value.

$$w(\delta_x, \delta_y) \propto \frac{1}{\max\{\frac{1}{2}, |f'(x+\delta_x, y+\delta_y) - f'(x,y)|\}} \tag{3.9}$$

There are two variants of the GIWS filter: the first includes the centre pixel in the weighted average, while the second one does not. If the centre pixel is included it will always be given a weight of 2. If not included, GIWS becomes resistant to single pixel impulse noise [Hig91]. If the image is unaffected by impulse noise, the former variant should be used, as it will give more smoothing in the presence of additive noise.

Figure 3.22 gives an example of GIWS applied to a noiseless 3×3 corner block. The resultant weights are higher for those pixels in the centre pixel's segment (intensity 3) than those in the other segment (intensity 8). The smoothed estimate $\hat{p}$ is 3.56 when the centre pixel is included in the linear combination, and 3.71 when omitted. Ideally the estimate should be 3, but in both cases it has been polluted with values from the other segment. The extent to which this occurs may be controlled by modifying the $\frac{1}{2}$ in the denominator of Equation 3.9.

(a)

3	3	8
3	3	8
8	8	8

(b) $w(\delta_x, \delta_y) \propto$

2	2	0.2
2	2	0.2
0.2	0.2	0.2

(c) $\hat{p} = 3.56$

Figure 3.22: Gradient inverse weighted smoothing (GIWS): (a) original pixels; (b) computed weights, unnormalized; (c) smoothed value when centre pixel included.

The failure of GIWS in Figure 3.22 may be attributed to its refusal to make absolute decisions about which pixels belong together. Its *soft cut-off* function, shown in Figure 3.23, allows information from neighbouring segments to diffuse into the current segment. A local segmentation approach would suggest the use of a *hard cut-off* to explicitly separate pixels from different segments. The cut-off point would be related to the natural level of variation within image segments. For the example in Figure 3.22, a hard-cut off function would have filtered the block exactly, given any reasonable cut-off point.

Sigma filter

Lee's Sigma filter [Lee83] is based on the alpha-trimmed mean estimate for the centroid of a data set [RL87]. Firstly, the standard deviation, s, of the pixels in the window is calculated. The filtered value is then set to the average of those pixels within $2s$ of centre pixel. If the number of pixels taking part in the average is too small, Lee recommends taking the average of all the non-centre pixels. Lee claims that the Sigma filter performs better than the median and GIWS with the centre pixel included.

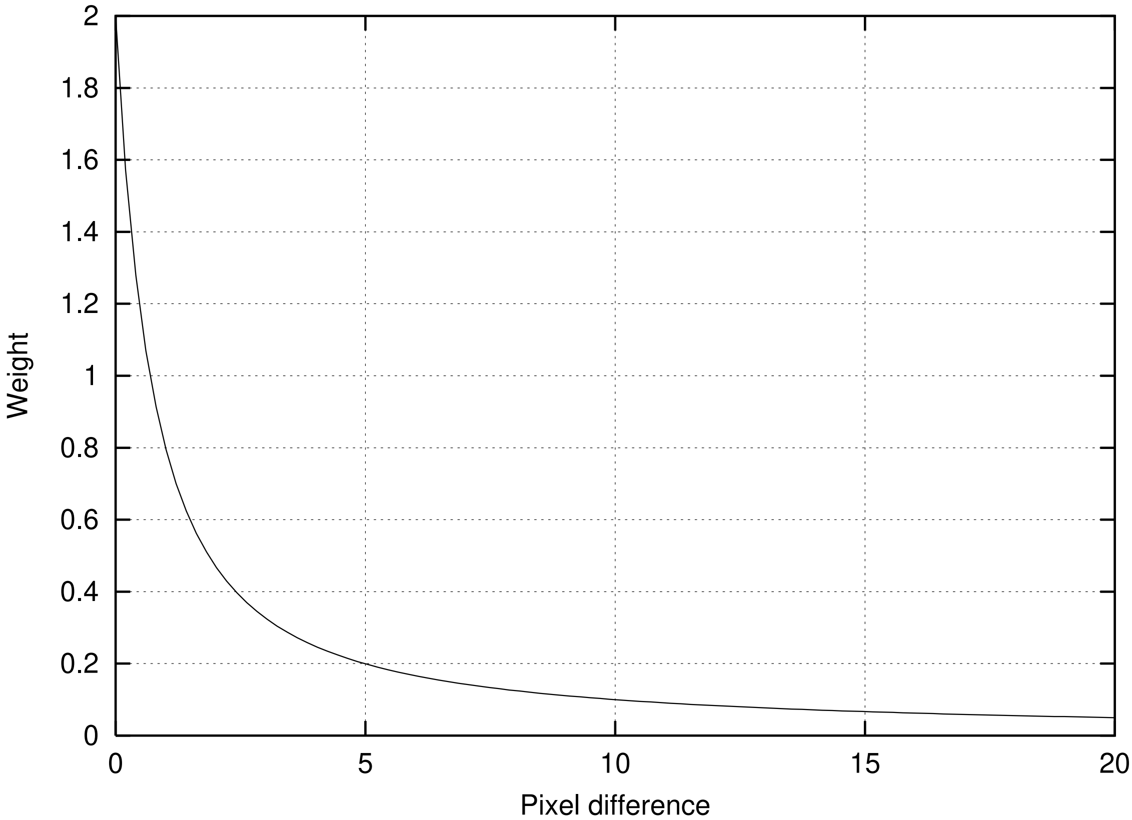

Figure 3.23: Pixel weighting function for the GIWS adaptive denoising algorithm.

Figure 3.24 demonstrates the Sigma filter on a corner block. The standard deviation of the block is 2.5, resulting in the averaging of all pixels within 5 intensity units of the centre pixel. Unfortunately, *all* the pixels are in this range, causing the Sigma filter to revert to a box filter. This produces a blurred estimate of $\hat{p} = 5.78$.

3	3	8
3	3	8
8	8	8

$$\sigma = 2.50 \qquad \hat{p} = \tfrac{4\times3+5\times8}{9} = 5.78$$

Figure 3.24: Demonstration of the Sigma filter on a noiseless corner block.

The Sigma filter tries to average only those pixels similar enough to the centre pixel. This is an explicit form of local segmentation using a hard membership function. The similarity measure uses the local variance to control the maximum intensity deviation a pixel may have from the centre pixel while still being considered part of the same segment. In terms of local segmentation, the local variance is an estimate of the natural level of variation within segments. In homogeneous regions it is a good estimate, but around edges it will be too high. Thus the variation estimate is poor for those cases where its value is crucial, and acceptable for those cases where it is not overly important.

The local variance is very sensitive to contamination from sources such as impulse noise. To a lesser extent it also suffers from the small amount of pixel data used to calculate it. The Sigma filter could be improved by replacing the local variance with a globally estimated or *a priori* supplied natural level of variation. This is the approach taken by the local segmentation based denoising algorithms of Chapters 4 and 5. The fact that the Sigma filter also falls back to an overall average when too few pixels are available for averaging implies that it does not allow segments to consist of a single pixel.

Anisotropic diffusion

Anisotropic diffusion [PM90, BSMH98] is a highly iterative process which slowly adjusts pixel intensities to make them more like their neighbours. In its original form it has been shown to be very similar to GIWS [Wei97]. Equation 3.10 describes the anisotropic diffusion process: t is the iteration number; $\mathcal{Q}$ is the set of neighbouring coordinates, usually the four nearest ones; M is the number of neighbours, usually four; λ controls the rate of smoothing; ψ is a user-supplied function; and σ is a user supplied parameter to be described later.

$$f_{t+1}(x,y) = f_t(x,y) + \frac{\lambda}{M} \sum_{(x',y') \in \mathcal{Q}} \psi\left(f_t(x',y') - f_t(x,y), \sigma\right) \tag{3.10}$$

The ψ function is the most important influence on behaviour of the diffusion process. Equation 3.11 lists the three most commonly used functions. Figure 3.25 plots these three functions with respect to the intensity difference, d, for $\sigma = 5$. Note that the functions have been scaled to fit them together. Only the relative weights for a particular function are meaningful.

$$\begin{array}{lrcl} \text{Standard} & \psi(d,\sigma) & = & 2d \\ \text{Lorentz} & \psi(d,\sigma) & = & \frac{2d}{2\sigma^2+d^2} \\ \text{Tukey} & \psi(d,\sigma) & = & d(1-(\frac{d}{\sigma})^2)^2 \quad \text{if} \quad d < |\sigma| \quad \text{else} \quad 0 \end{array} \tag{3.11}$$

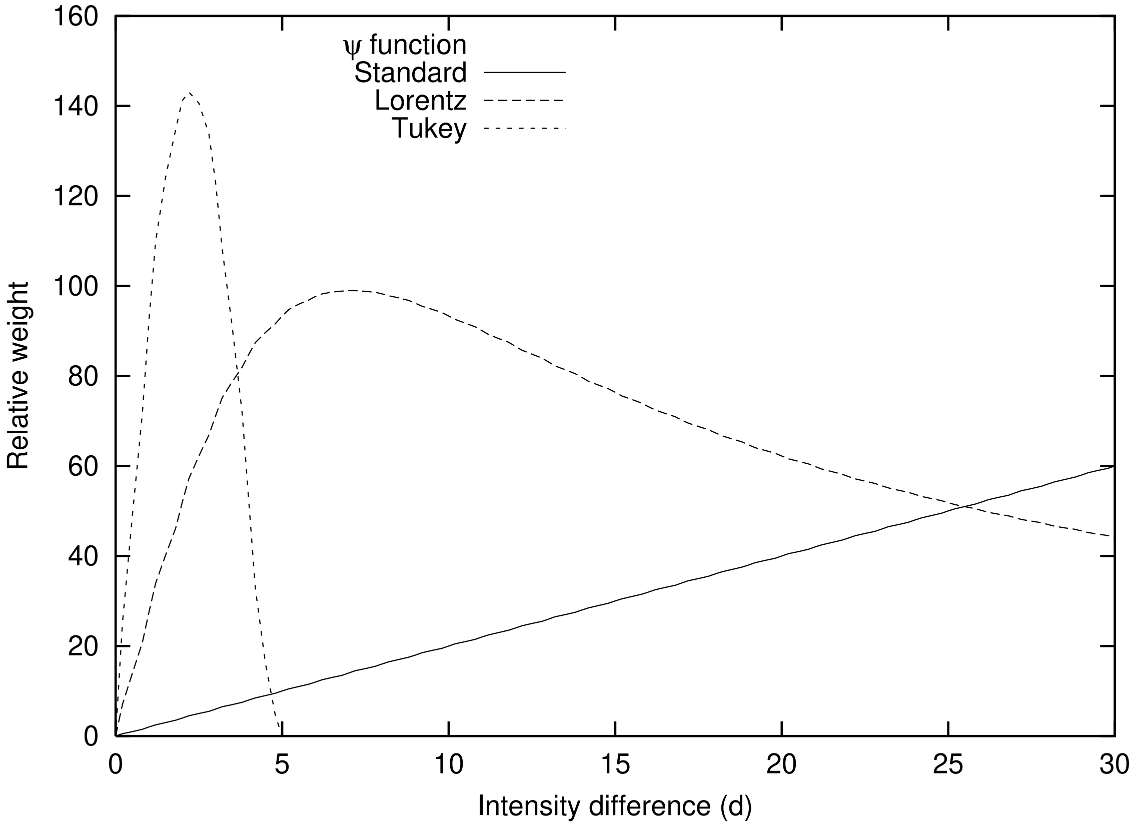

Figure 3.25: Anisotropic diffusion ψ weighting functions when $\sigma = 5$. Each function has been scaled for comparison purposes.

The Standard formulation forces a pixel value toward the intensity of each neighbour, at a rate directly proportional to its intensity difference from that neighbour. Equation 3.12 shows that Standard diffusion can be interpreted as static linear filter. Equal averaging of all five pixels in the local window occurs when $\lambda = 0.4$.

$$f_{t+1}(x,y) = f_t(x,y) + \frac{2\lambda}{M} \sum_{(x',y')\in\mathcal{Q}} [f_t(x',y') - f_t(x,y)] \tag{3.12}$$

$$= (1-2\lambda) f_t(x,y) + \frac{2\lambda}{M} \sum_{(x',y')\in\mathcal{Q}} f_t(x',y') \tag{3.13}$$

The problem with Standard diffusion is that, if left for many iterations, the denoised image would become completely homogeneous. In a fashion similar to the GIWS in Section 3.4.5, all neighbouring pixels receive some weight, no matter what intensity they have. Therefore, information from unrelated segments can diffuse across segment boundaries.

The Lorentz formulation attempts to reduce pixel leakage between segments by slowing the diffusion process at some point, defined by the parameter σ. This parameter provides a sense of how different pixel values can be while still being in the same segment. The plot of the Lorentz weighting function in Figure 3.25 reaches a maximum when $d = \sqrt{2}\sigma$, and decays thereafter. The decay is quite slow, so a large difference will still be given a significant weight. Thus pixels from different segments can still influence each other, eventually producing a homogeneous image.

The Tukey formulation takes the Lorentz idea further and completely shuts down diffusion when $d \geq \sigma$, giving maximum weight when $d = \sigma/\sqrt{5}$. A neighbouring pixel value deviating more than σ intensity levels from the centre pixel will have no influence whatsoever. In local segmentation, the σ parameter may be considered proportional to the natural level of variation of pixel values within a segment. For the simple case of constant intensity segments with additive noise, this would relate to the standard deviation of the noise. The use of the Tukey function ensures that the final image will never become a homogeneous image, with the side-effect of being unable to preserve contrast differences less than σ.

The λ parameter controls the speed at which diffusion occurs. It is recommended that λ be made small and the number of iterations large, as to more closely approximate the theoretically continuous diffusion formulae. It must be noted that the diffusion process is *very* slow, with thousands of iterations being common. This also usually requires that all intermediate results be kept to floating point accuracy.

Adaptive smoothing

Adaptive smoothing [SMCM91] claims to be strongly related to anisotropic diffusion and GIWS. It was designed to denoise both intensity and range images. Instead of using the intensity difference to weight neighbouring pixels, a more thorough estimate of the *gradient* is used, shown as the ∇ operator in Equation 3.14. The parameter σ is user-supplied, and controls the scale at which the filter works.

$$w(\delta_x, \delta_y) \propto e^{-\frac{\nabla(x+\delta_x, y+\delta_y)}{2\sigma^2}} \tag{3.14}$$

From a local segmentation perspective, the gradient term discourages the incorporation of pixels from different segments, as the gradient between them would cross a segment boundary and therefore be very high. Unfortunately, gradient functions are usually too coarse too accurately control smoothing around fine image details [Smi95].

SUSAN

SUSAN was introduced in Section 3.3.3 in regard to its use of local segmentation principles for edge and corner detection. SUSAN gathers the set of neighbouring pixels which are most similar in intensity and closest in position to the centre pixel. This set is called the USAN. The topographic pattern revealed by plotting the size of each USAN may be used to determine the position of edges. Smith realized that a weighted average of the pixels within an USAN form a denoised estimate of the centre pixel [Smi95, Smi96, SB97]. Unlike the denoising algorithms described so far, SUSAN exploits both the similarity of pixel intensities *and* their spatial proximity to the centre pixel. Equation 3.15 describes the resulting SUSAN weighting function, where σ is a spatial threshold and t is a brightness threshold. The centre pixel is excluded from the average.

$$w(\delta_x, \delta_y) \propto e^{-\frac{\delta_x^2+\delta_y^2}{2\sigma^2} - \frac{(f'(x+\delta_x, y+\delta_y) - f'(x,y))^2}{t^2}} \qquad \text{but with} \quad w(0,0) = 0 \tag{3.15}$$

Due to its exponential-of-sums form, the weighting function is separable into its spatial and intensity components. Ignoring that fact that the centre pixel receives no weight, the spatial component is an isotropic 2-D Gaussian distribution with variance σ^2. Figure 3.26 plots this function for $\sigma = 4$, the default value used by Smith's implementation[2]. By default, SUSAN uses a 37 pixel circular 7×7 mask (see Figure 4.46e). The choice of $\sigma = 4$ prevents the Gaussian spatial weighting function from extending very far. Thus the weights for the closest and furthest pixels differ by less than a factor of two when using a 37 pixel mask.

The intensity difference component is the positive half of a Gaussian function with variance $t^2/2$. Figure 3.27 plots this function for $t = 20$, the value Smith claims works well for all images [Smi95]. SUSAN's Gaussian intensity weighting function effectively cuts off after

[2]Available from `http://www.fmrib.ox.ac.uk/~steve/susan/`

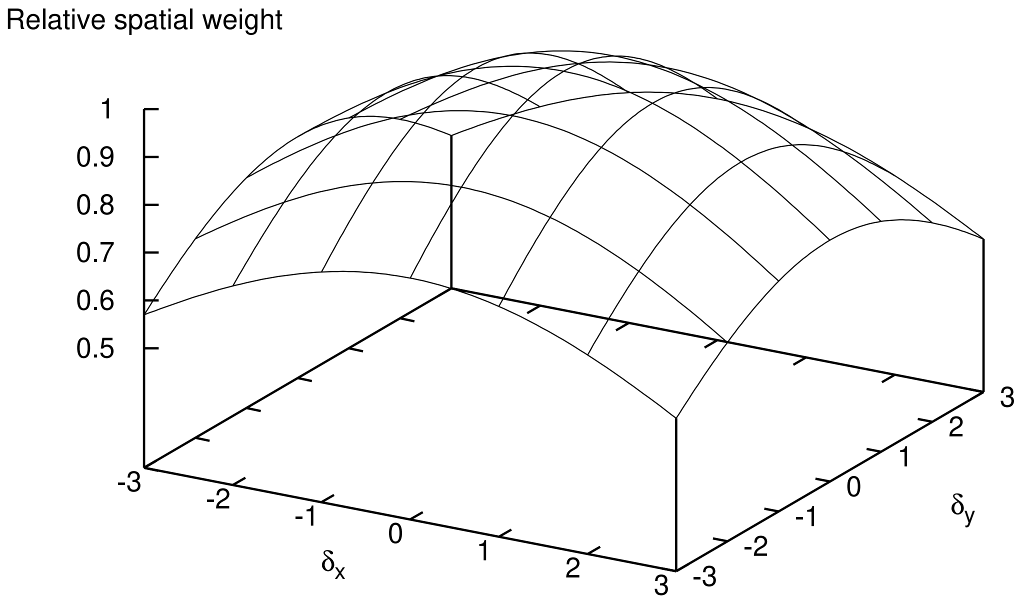

Figure 3.26: The SUSAN spatial weighting component when $\sigma = 4$, ignoring that $w(0, 0) = 0$.

three standard deviations. This corresponds to a intensity difference of $3t/\sqrt{2}$, which is 42 when $t = 20$. SUSAN's intensity weighting is much closer to a hard cut-off than the hyperbolic one used by GIWS, already seen in Figure 3.23.

Figure 3.28 gives an example of SUSAN denoising a 3×3 window. Due to the smaller mask, σ was reduced to 2. For this example $t = 2.5$ was chosen, resulting in a good filtered value of $\hat{p} = 3.14$. The SUSAN algorithm is very sensitive to the choice of t. If the default values had been used, the denoised estimate would have been about 6. If the slightly lower $t = 2$ was used, it would have been 3.02. If t is chosen poorly, it seems that SUSAN introduces noise into its filtered estimate. My experiments have shown that t should be proportional to the natural level of variation within the image. For the case of additive zero-mean Gaussian noise with variance s^2, setting $t = 3s$ works well.

When denoising, SUSAN gives no weight to the centre pixel. This has the advantage of helping to suppress impulse noise, but the disadvantage of reducing the amount of smoothing when the centre pixel is legitimate. SUSAN has an additional mode which has not been

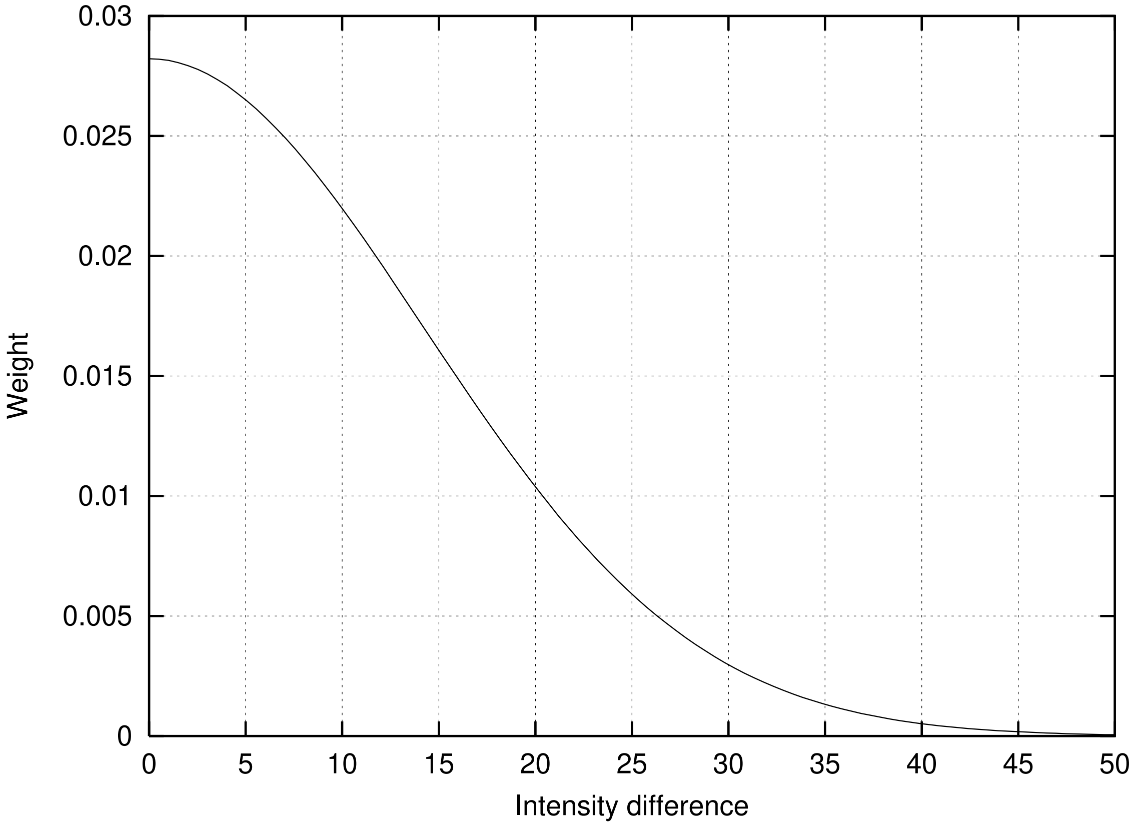

Figure 3.27: The SUSAN intensity difference weighting component when $t = 20$.

3	3	8
3	3	8
8	8	8

0.78	0.88	0.78
0.88	0	0.88
0.78	0.88	0.78

1.00	1.00	0.02
1.00	0	0.02
0.02	0.02	0.02

$\hat{p} = 3.14$

Figure 3.28: SUSAN denoising: (a) original pixels; (b) spatial weights, unnormalized, $\sigma = 2$; (c) intensity difference weights, unnormalized, $t = 2.5$; (d) denoised value.

mentioned yet. When the weights for all the neighbouring pixels are too small (corresponding to a weak USAN), SUSAN ignores them and uses the median of its eight immediate neighbours instead. In terms of local segmentation, this mode is invoked when no pixels appear to belong in the centre pixel's segment.

SUSAN's denoising outperforms the box filter, the median, GIWS, and the Saint-Marc's adaptive smoothing method [Smi95]. It takes the intensity weighting idea from GIWS, but modifies it to behave more like a hard cut-off function. Although the intensity weighting function is continuous, it effectively cuts off to zero for differences greater than $3t$ or so. The introduction of a spatial weighting component, ignored by most denoising algorithms, also helps it to perform better. The main drawback is that the brightness threshold must be supplied. Despite this, SUSAN is one of the best and fastest local denoising algorithms available, in part due to its implicit application of the local segmentation principle.

3.4.6 Filtering based on global segmentation

It is possible to use a global segmentation algorithm to perform image denoising. Imagine a piece-wise constant image corrupted by additive noise, and a segmentation algorithm which can segment it correctly. The average of the pixels in each segment should be close to the constant value that originally coloured each segment. By replacing each pixel with the mean of the segment it belongs, the image is denoised. The problem with this scheme is simply the difficulty of achieving perfect global segmentation.

Hussain *et al* [HR94] take a hybrid approach, using a combination of a Gibbs-Markov random field and relaxation labeling to isolate segment boundaries globally. A local filter is then used to average the centre pixel with neighbours on the same side of a boundary. In regions with no boundaries, all pixels are averaged and thus it behaves exactly as the box filter in Section 3.4.3. Without providing numerical results, they claim to have better subjective results when compared to simple averaging. Unfortunately, when the high noise is high, their algorithm suffers because the distinction between noise and structure is less obvious. It is unclear why they did not just average all the pixels from global segments rather than take local averages. I expect it is due to the fact that most large segments will exhibit some spatial variation in intensity. In these cases a local average could be a better estimate of the underlying pixel value when compared to a segment-wide average.

Watershed segmentation has also been adapted to denoising [HH95, HH99]. The image is first segmented into "catchment basins". These basins are smaller subsets of larger segments, and are represented by the mean of the pixels they contain. For a given basin, B, those neighbouring (spatially connected) basins with means considered close enough to B's mean are incorporated into B. The pixels in B are then set to the aggregated mean. The criteria for closeness can vary. In the early paper [HH95], the closest 50% neighbouring basin means are used, whereas later [HH99], only those means within some difference threshold, T, were incorporated. They claim to outperform various filters, including anisotropic diffusion. The only test image used was a simple 3-class piece-wise constant synthetic image containing very high contrast edges corrupted by additive zero-mean Gaussian noise with variance 30^2. They admit their technique depends on the choice of T, which is related to the noise level. The challenge is to estimate accurately the natural intensity variation one might expect to observe within each segment.

3.5 Conclusions

Image segmentation algorithms attempt to partition an image into separate groups of related pixels, called segments. Pixels are usually considered related if they have similar values, or are located near each other. Clustering is a form of segmentation which ignores spatial information. Segmentation is considered and essential component of image understanding systems. Although segmentation of arbitrary images is inherently difficult, the principles described in this chapter have been applied successfully to many tasks.

Local image processing algorithms assume that most of the information about a pixel can be found within the small neighbourhood of pixels surrounding it. Local algorithms are often used for low level tasks, such as denoising and edge detection. The local neighbourhood may be considered a tiny image, but with different expected properties. It contains fewer pixels, is made up of fewer distinct regions, and has a higher proportion of pixels in the margins.

Segmentation is a fundamental image processing task. Pixels within segments are homogeneous with respect to some predicate, and hence can be used to denoise the image while preserving structure. The boundaries between segments correspond directly to discontinuities in the image, hence edge detection is achievable in the same framework. It would seem obvious to apply these ideas directly to the sub-images used in local image processing. Strangely, much of the literature seems to treat local techniques differently, preferring to use *ad hoc* methods instead, which usually require image dependent parameters to be supplied, or estimated from training sets.

Image denoising algorithms are a good basis for examining the state of the art for modeling images on a local level. The SUSAN algorithm is currently one of the best local, one-pass image denoising algorithms available. During its initial development, SUSAN used a form of explicit local segmentation. Pixels similar enough to the centre pixel were assimilated using a hard cut-off function, and then averaged. But rather than refine the local segmentation criterion used, SUSAN moved to an implicit local segmentation, admitting all the local pixels to the local segment, but allowing their contribution to vary. This formulation made it difficult to link the brightness threshold parameter to a specific image model.

This thesis advocates a progression toward an explicit local segmentation model, via the suitable application of global segmentation techniques to local image data. The local segmentation principle states that the first step in processing a pixel should be to segment the local region encompassing that pixel. The segmentation algorithm should automatically determine the number of segments present in an efficient manner, and any parameters should be estimated from the image itself. The information obtained from local segmentation should sufficiently separate the noise and structure within the region. The structural information may then be used to achieve goals such as denoising and edge detection.

Chapter 4

Local Segmentation applied to Structure Preserving Image Denoising

4.1 Introduction

The local segmentation principle may be used to develop a variety of low level image processing algorithms. In this chapter it will be applied to the specific problem of denoising greyscale images contaminated by additive noise. The best image denoising techniques attempt to preserve image structure as well as remove noise. This problem domain is well suited to demonstrating the utility of the local segmentation philosophy.

A multilevel thresholding technique will be used to segment the local region encompassing each pixel. The number of segments will be determined automatically by ensuring that the segment intensities are well separated. The separation criterion will adapt to the level of additive noise, which may be supplied by the user or estimated automatically by the algorithm. The resulting segmentation provides a local approximation to the underlying pixel values, which may be used to denoise the image.

The denoising algorithm presented is called FUELS, which stands for "filtering using explicit local segmentation". FUELS differs from existing local denoising methods in various ways. The local segmentation process clearly decides which pixels belong together, and does so democratically, without using the centre pixel as a reference value. If the computed local approximation suggests changing a pixel's value by too much, the approximation is ignored, and the pixel is passed through unmodified. The fact that each local approximation overlaps with its neighbour means that there are multiple estimates for the true value of each pixel. By combining these overlapping estimates, denoising performance is further increased.

FUELS will be shown to outperform state-of-the-art algorithms on a variety of greyscale images contaminated by additive noise. FUELS' worst case error behaviour will be shown to be proportional to the noise level, suggesting that it is quite adept at identifying structure in the image. The denoised images produced by FUELS will be seen to preserve more image structure than algorithms such as SUSAN and GIWS.

4.2 Global image models

An image model is a mathematical description of the processes affecting the final pixel values in an image. As described in Section 2.5, these processes may include atmospheric effects in the scene, noise in the capture device, and quantization of pixel values. It may be possible to construct a model which accounts for all these steps, but it would probably consist of many difficult to determine parameters. Often a simpler model can incorporate the main factors and still achieve good results.

Modeling the steps from acquisition to a final digital image is sufficient, but not necessary, for successful image processing. In many cases the full acquisition history may not even be known. In any case, some assumptions or prior beliefs regarding the properties of the original scene are also required. The characteristics of the light source and the object surfaces will influence greatly how we expect the image pixel values to be distributed spatially and spectrally. The number, distance and size of objects in the scene will also affect the proportion, scale and sharpness of edges in the image.

For example, consider the artificial images in Figure 4.1[1]. The first is greyscale light intensity image of a simple object. The second image is also of the same object, except that the intensity of each pixel represents the *distance* from the camera to each point in the scene.

[1]These images are part of the Range Image Segmentation Comparison Project at the University of South Florida. `http://marathon.csee.usf.edu/range/seg-comp/images.html`

This is an example of a *range image*, for which the "light source" is actually a distance measurement device. It has been shown that pixel intensities in a range image usually vary in a spatially linear manner, due to the polyhedral nature of most objects [KS00]. However this may not be an appropriate assumption to make when analyzing low resolution fingerprint images in a criminal database. There one would expect many ridges and high contrast edges. This thesis focuses on greyscale light intensity images. The generalization to other types of images is considered in Chapter 6.

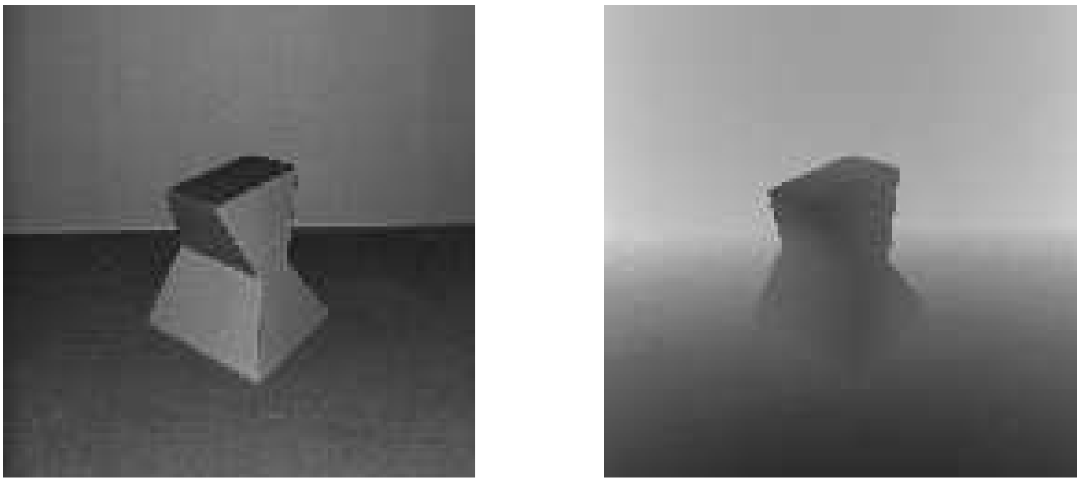

Figure 4.1: Two images of the same scene: (a) light intensity; (b) range, darker is closer.

4.2.1 The facet model

For greyscale image data, variations on the *facet model* [HW81] are most commonly used. The facet model assumes that a digital image is a discrete approximation of a piece-wise continuous intensity surface. Each piece is disjoint and is called a facet or segment. The underlying pixel values in each facet are assumed to follow a two dimensional polynomial function. This polynomial fit can be of any order. Figure 4.2 provides an example of a one dimensional discrete signal approximated by increasingly higher order polynomial curves.

The simplest facet approximation is a zeroth order polynomial of the form $f(x, y) = a$, where a is a constant. Each pixel in a constant facet is assumed to have the same value. For greyscale data this would be a scalar representing the intensity, but for colour data it would be an RGB vector. An image containing constant facets referred to as being *piece-*

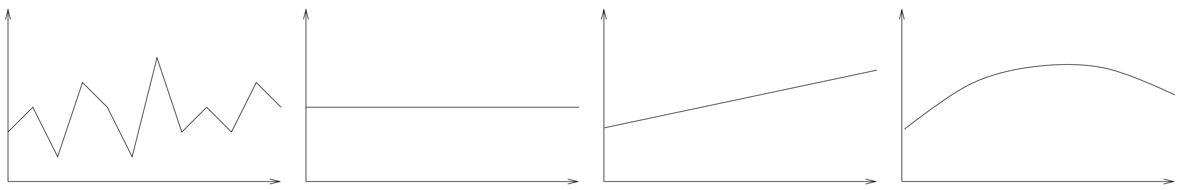

Figure 4.2: One dimensional polynomial approximation: (a) the original signal; (b) constant; (c) linear; (d) quadratic.

wise constant. Piece-wise constant image models are commonly used in image processing. They only have one parameter to estimate, and are simple to manipulate.

First order polynomial approximation in two dimensions has the mathematical form of a plane, namely $f(x, y) = a + bx + cy$. An image containing facets of this type is *piece-wise planar*. Pixel values in a planar facet are linearly dependent on the their position within the facet. Planar facets are more flexible than constant facets, but at the expense of having needing three parameters to be estimated for them. If b and c are small enough, and we are concerned only with a small area within a larger planar facet, then a constant approximation may be sufficiently accurate.

4.2.2 Generalizing the facet model

The facet model is the basis for the more general *segmentation-based* approach to image modeling. The main difference is that the requirement for a *polynomial* approximation is relaxed. Instead, the interior of a segment may be modeled using any mathematical description which defines a *homogeneity criterion* for the segment. For example, texture is often modeled, not as a functional surface, but as a low order Markov model [Hua84]. A first order Markov model is probability distribution over pairs of pixel values. This model is quite different from a polynomial one, but it can still be used to define a homogeneity criterion.

An important attribute of any homogeneity criterion used in image segmentation is whether or not it takes spatial coherence into consideration. As discussed in Section 3.2.1, techniques such as thresholding consider only pixel values, and not their relative spatial positions. This can produce unconnected segments, which may or may not be desirable.

4.2.3 Image sampling

Consider the simple undigitized monochrome scene in Figure 4.3a. It consists of a square object of uniform high intensity on a dark background of uniform low intensity, and is perfectly piece-wise constant. Figure 4.3b illustrates the discretization of this scene into a 9×11 digital image. The intensity of each pixel in the final image, shown in Figure 4.3c, is set to the average light intensity of the area each pixel covers in the original scene. The resulting image is also piece-wise constant, consisting of two distinct segments. This is because the boundary between object and background coincided with the alignment of the pixel grid.

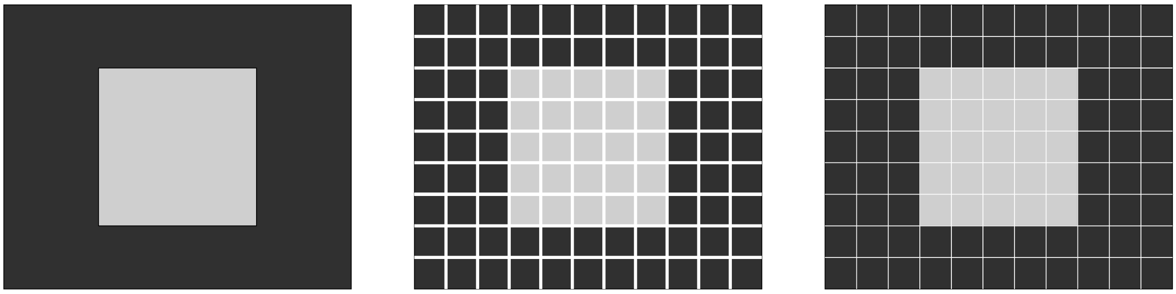

Figure 4.3: Discrete sampling of an object aligned to the pixel grid: (a) original scene; (b) superimposed sampling grid; (c) digitized image.

Figure 4.4 shows what occurs when an object in a scene does not align exactly with the sampling grid. The sampling process has produced a range of pixel intensities in the digitized image. Each pixel on the object boundary has received an intensity which is a blend between the intensities of the two original segments. In fact, there are now seven unique pixel intensities compared to the original two.

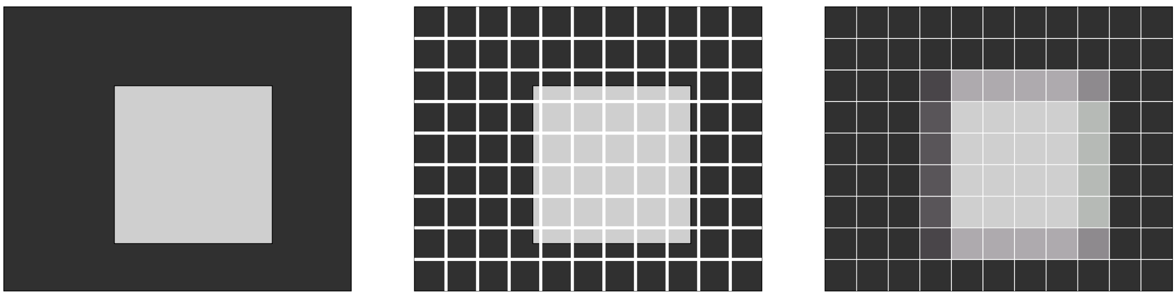

Figure 4.4: Discrete sampling of an object mis-aligned with the pixel grid: (a) original scene; (b) superimposed sampling grid; (c) digitized image.

Most observers would still assert the existence of only two segments, but would have some difficulty assigning membership of each boundary pixel to a specific segment. One interpretation is that those pixels with intermediate intensities have partial membership to both segments. If an application requires a pixel to belong only to one segment, that segment in which it has maximum membership could be chosen. Alternatively, it could be claimed that the image is still piece-wise constant, but now consists of ten segments. These differing interpretations highlight the fact that an image model for a continuous scene may no longer apply to its digitized counterpart.

4.2.4 Edges and lines

Edges are the boundaries between segments. Lines may be regarded as very narrow segments, having two edges situated very close together. Figure 4.5 gives a one dimensional example of two types of edges and the two corresponding types of lines. The image models used for edges and lines are not unrelated to the models used for the interior of segments.

Figure 4.5: (a) step edge; (b) line; (c) ramp edge; (d) roof.

The *step edge* defines a perfect transition from one segment to another. If segments are piece-wise constant and pixels can only belong to one segment, then a step edge model is implicitly being used. If a segment is very narrow, it necessarily has two edges in close proximity. This arrangement is called a *line*. An arguably more realistic model for edges is the *ramp edge*. A ramp allows for a smoother transition between segments. This may be useful for modeling the blurred edges created from sampling a scene containing objects not aligned to the pixel grid. Two nearby ramp edges result in a line structure called a *roof*.

Edge profiles may be modeled by any mathematical function desired, but steps and ramps are by far the most commonly used. If a ramp transition occurs over a large number of pixels, it may be difficult to discriminate between it being an edge, or a planar facet. If pixels along the ramp are assigned to a particular segment, the values of those pixels may be dissimilar to the majority of pixels from inside the segment.

4.2.5 A useful noise model

As described in Section 2.4, different types of noise may be introduced at each step of the image acquisition process. The general functional form for the noise component, $n(x, y)$, at each pixel is given in Equation 4.1. Parameters x and y are the spatial position of the pixel in question, $\vec{\theta}$ is a vector of fixed parameters determining some properties of the noise such as its intensity and spread, and **f** is the original image which may be required if the noise term is data dependent. This formulation has scope for a huge number of possible noise functions.

$$n(x,\ y\ ;\ \vec{\theta},\ \mathbf{f}) \tag{4.1}$$

A simple, but still useful and versatile noise model is additive zero-mean Gaussian noise which is independently and identically distributed (i.i.d.) for each pixel. Under this model the noise *adds* to the original pixel value before digitization. The noise term may be written like Equation 4.2, where "$\sim \mathcal{N}(\mu, \sigma^2)$" denotes a random sample from a normal distribution of mean μ and variance σ^2. Figure 4.6 plots the shape of this noise distribution when $\sigma^2 = 1$.

$$n(x, y\ ;\ \sigma^2) \sim \mathcal{N}(0, \sigma^2) \tag{4.2}$$

Because the noise is additive and symmetric about zero, it has the desirable effect, on average, of not altering the mean intensity of the image. It only has one parameter, the variance σ^2, which determines the spread or strength of the noise. Although the work in this thesis assumes that the noise variance is constant throughout the image, it would be possible to vary it on a per pixel basis. This and other extensions are discussed in Chapter 6.

Consider a *constant* facet containing pixels with intensity z. After $\mathcal{N}(0, \sigma^2)$ noise is added, it is expected that 99.7% of pixels will remain in the range $z \pm 3\sigma$. This is called the 3σ

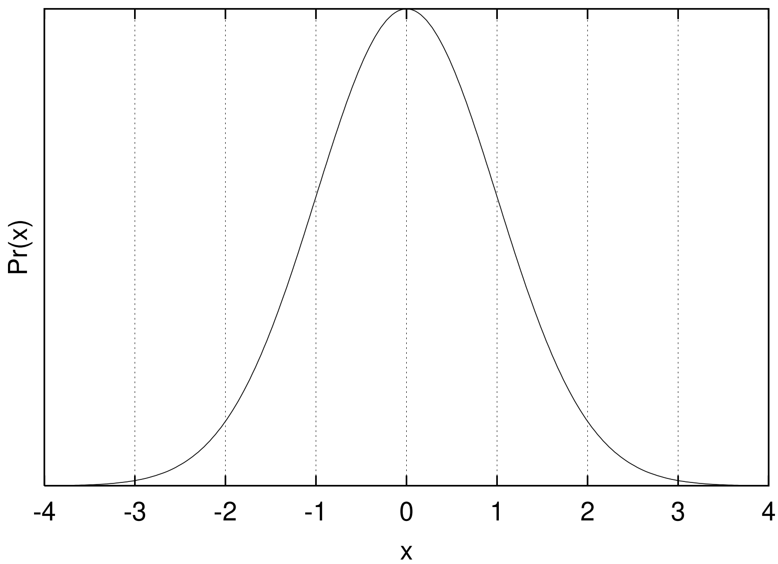

Figure 4.6: The standard Gaussian distribution: $z \sim \mathcal{N}(0, 1)$

confidence interval for z [MR74]. Examination of Figure 4.6 shows that very little probability remains for values of z outside the confidence interval. Table 4.1 lists the number of standard deviations from the mean for which a given proportion of a normally distributed data is expected to lie.

Fraction of data (%)	Number of standard deviations from mean
50.0	0.674
68.3	1.000
90.0	1.645
95.0	1.960
95.4	2.000
98.0	2.326
99.0	2.576
99.7	3.000

Table 4.1: Confidence intervals for normally distributed data.

Figure 4.7 shows the effect of two different noise variances on the `square` image, which has segments of intensity 50 and 200. When $\sigma = 10$, the two segments remain distinct. When σ is quadrupled, the square obtains some pixels clearly having intensities closer to the original *background* intensity. After clamping, the 3σ confidence interval for the background is $50 \pm 120 = (0, 170)$, and $200 \pm 120 = (80, 255)$ for the square. These limits have an overlap of 90, or approximately 2σ. On average, this situation would result in about 5% of pixels being closer to the opposing segment mean. For the 11×9 `square` image this would affect about 5 pixels, roughly corresponding with what is observed.

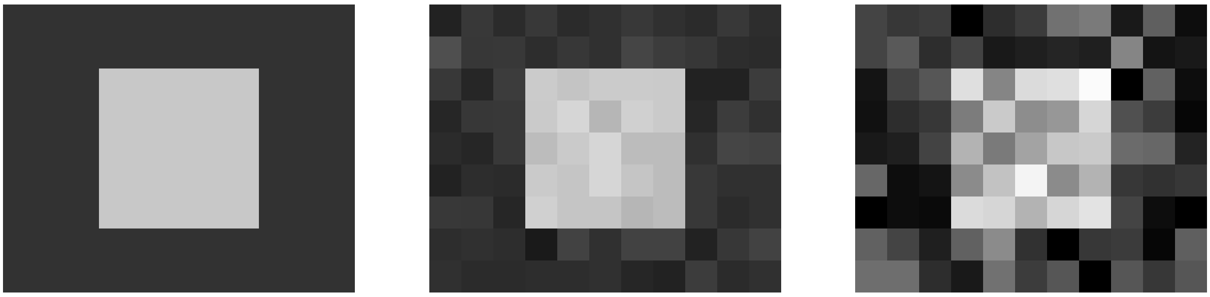

Figure 4.7: The effect of different Gaussian noise levels: (a) no noise; (b) added noise $\sigma = 10$; (c) added noise $\sigma = 40$.

For piece-wise constant segments, the noise standard deviation defines a *natural level of variation* for the pixel values within that segment. The natural level of variation describes the amount by which pixel values may vary while still belonging in same segment. For the case of planar segments, the natural level of variation depends on both the noise level and the coefficients of the fitted plane. If a global planar segment only has a mild slope, then the variation due to the signal may be negligible for any local window onto that segment. The natural level of variation in the window will be dominated by the noise rather than the underlying image model.

4.2.6 Pixel quantization

Section 4.2.3 discussed the side-effects of sampling a continuous scene on a discrete image grid. The sampled pixel intensity is computed as an average or integral over a small area of the scene. This intensity must also be quantized if it is to be stored in digital form [GN98]. The number of bits per pixel (bpp), L, is typically around 8 to 16. Most "everyday" images use 8 bpp per band, but medical and scientific applications usually store data with higher accuracy. The number of unique intensities that can be stored in L bpp is $Z = 2^L$, which comes to $Z = 256$ for 8 bpp images.

Consider the one-dimensional signal drawn with a dotted line in Figure 4.8. Quantization to 8 levels produces the signal plotted with the solid line. Due to the rounding processing, a quantized value, z, could have originally had any value in the range $[z - 0.5, z + 0.5)$. This *quantization noise* may be approximated using the standard deviation of a uniform distribution of width 1, which is $\sqrt{1/12}$, or about 0.29. Usually the other forms of noise are at least an order of magnitude higher than this, so quantization noise can safely be ignored by most applications.

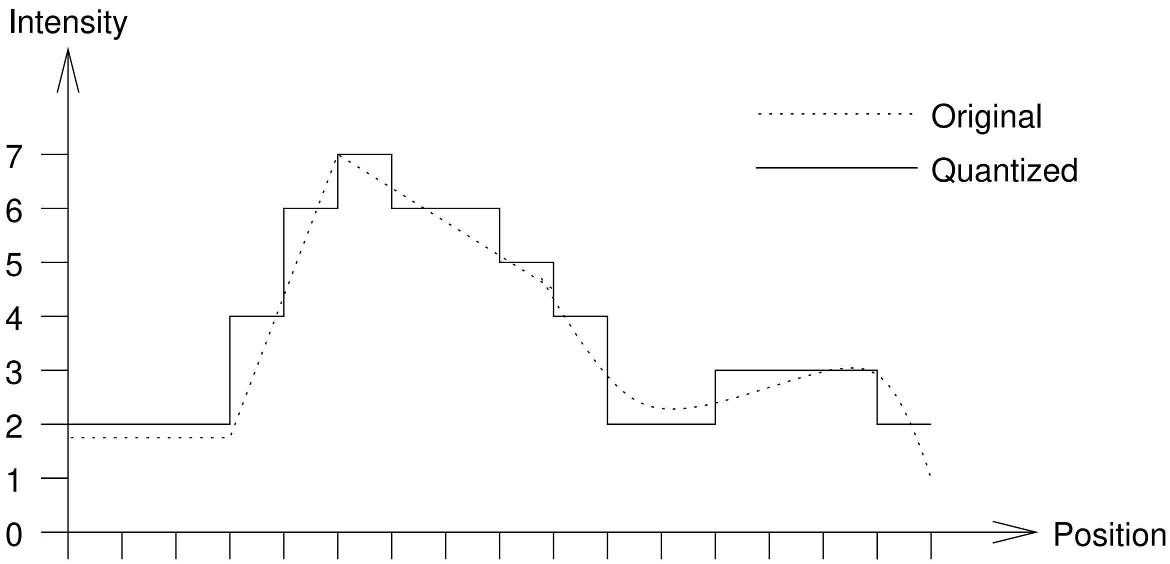

Figure 4.8: Effect of quantizing the signal intensity to 8 levels.

A situation where one may not wish to ignore quantization noise is when magnifying an image by interpolating pixel values. By default, the midpoint of the quantization region would be used as the true value of the pixel, but it is possible to choose the true value from anywhere within the original quantization interval. The most appropriate value could be decided by optimization of a suitable cost function, for example, to maximize the sharpness of the interpolated image.

4.2.7 A suitable image model

The aim of this chapter is to demonstrate a *simple* form of local segmentation applied to denoising greyscale photographic-type images. Equation 4.3 mathematically describes the process of going from an undigitized noise-free image, $\mathbf{f}$, to a noisy quantized version, $\mathbf{f}'_n$.

$$f'_n(x, y) = \lfloor f(x, y) + \mathcal{N}(0, \sigma^2) + 0.5 \rfloor \tag{4.3}$$

After quantization there is a final *clamping* step to ensure the pixel values fall within the legal range. Equation 4.4 describes the clamping process, which takes the noisy quantized image, $\mathbf{f}'_n$, and produces the final image, $\mathbf{f}'$, available to the image processor.

$$f'(x,y) = clamp\,[\;f_n(x,y)\;] = \begin{cases} 0 & \text{if} \quad f'_n(x,y) < 0 \\ Z-1 & \text{if} \quad f'_n(x,y) \geq Z \\ f'_n(x,y) & \text{otherwise} \end{cases} \tag{4.4}$$

To that end a *global* image model with the following properties will be assumed:

1. The image is composed of disjoint segments.
2. Segment boundaries coincide with pixel boundaries .
3. Each pixel may belong to one segment only.
4. A segment interior may be well approximated by a constant grey level, particularly at a local level.
5. Each pixel is corrupted by additive zero-mean Gaussian noise.
6. The noise variance is common to all pixels (and hence segments).
7. Quantization noise may be ignored.
8. Saturation effects from pixel clamping are to be ignored.

An image model assuming piece-wise constant greyscale segments with additive noise is probably the simplest, but still useful, image model to use. In Chapter 6, the extension of the principles outlined in this chapter to more complex image models will be discussed.

4.2.8 Image margins

When processing a whole image with dimensions $X \times Y$ in a local manner, each pixel $f(x,y)$ is processed in turn using only its immediate pixel neighbourhood. Difficulties arise when processing pixels inhabiting the margins of the image, as these pixels have incomplete neighbourhoods. Figure 4.9 illustrates this situation for $f(0, 0)$ when using a 3×3 window. The 5 pixels within the dotted window labelled ? are denoted as *missing*, and the remaining 4 are denoted as *available* pixels.

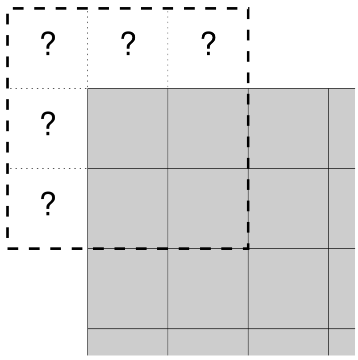

Figure 4.9: Incomplete 3×3 neighbourhoods when processing pixels at the image margins.

There are various approaches to handling this situation:

Constant fill In this scheme, missing pixels are replaced with a fixed constant value, most commonly zero or mid-grey.

Wrap around The image is treated as a torus, as if the upper and lower, and left and right, edges were spatially proximate. This is achieved by treating pixel coordinates in a modulo fashion $(x + X \bmod X, y + Y \bmod Y)$. Note that $x + X$ is used to handle negative coordinates.

Nearest neighbour Missing pixels take on values equal to their nearest available neighbour.

Average fill The missing pixels are set to the arithmetic mean of the available pixels.

Specialization The algorithm is specifically modified to handle the special cases where the neighbourhood contains fewer pixels than normal.

The "constant fill" and "wrap around" methods are clearly inferior because they invent pixel values bearing little relevance to the available pixels. The "specialization" method is desirable, but even when using a 3×3 window there are eight special cases to handle (four sides plus four corners), which can make it difficult to implement efficiently.

Figure 4.10 pictorially describes the "nearest neighbour" approach. This technique is simple to implement, and possesses the ability to preserve certain structure in the image. For example, if the available pixels near the margin form two separate populations, the extrapolated neighbourhood would too. The disadvantage is that some pixel values, such as the corners, are duplicated more than once, and hence over-represented in the extrapolated set. This can bias the results, especially if the available pixels are very noisy.

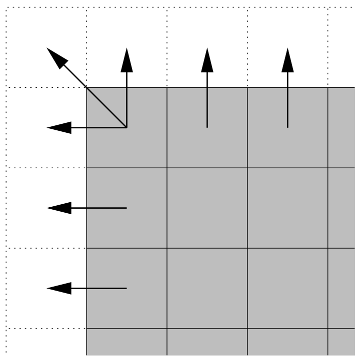

Figure 4.10: Missing pixels are easily replaced with their nearest neighbours.

The "average fill" method chooses values for the missing pixels which are closest, in a least squares sense, to the available pixels. This is desirable when the available pixels are homogeneous. However, when the available pixels are heterogeneous, this method would invent a third class of pixels all having the same value. It also requires more computation that the "nearest neighbour" approach.

In this thesis the "average fill" method will be used. The averaging process suggests it would be more robust under noisy conditions when compared to "nearest neighbour". It is hoped that it should have little effect on the overall results compared to "specialization", as the margin pixels comprise only a small proportion of the total number of pixels in the image.

Consider a $m \times m$ square window applied to an $X \times Y$ image. In Figure 4.11 the shaded area contains pixels which have an incomplete neighbourhood. The outer dotted line denotes the effective increase in image resolution when extrapolating missing pixels.

Equation 4.5 gives an expression for B, the ratio of missing pixels to the total number of pixels in the image. This is the proportion of pixels, which, when at the centre of the window, are forced to generate missing pixel values using one of the techniques just described.

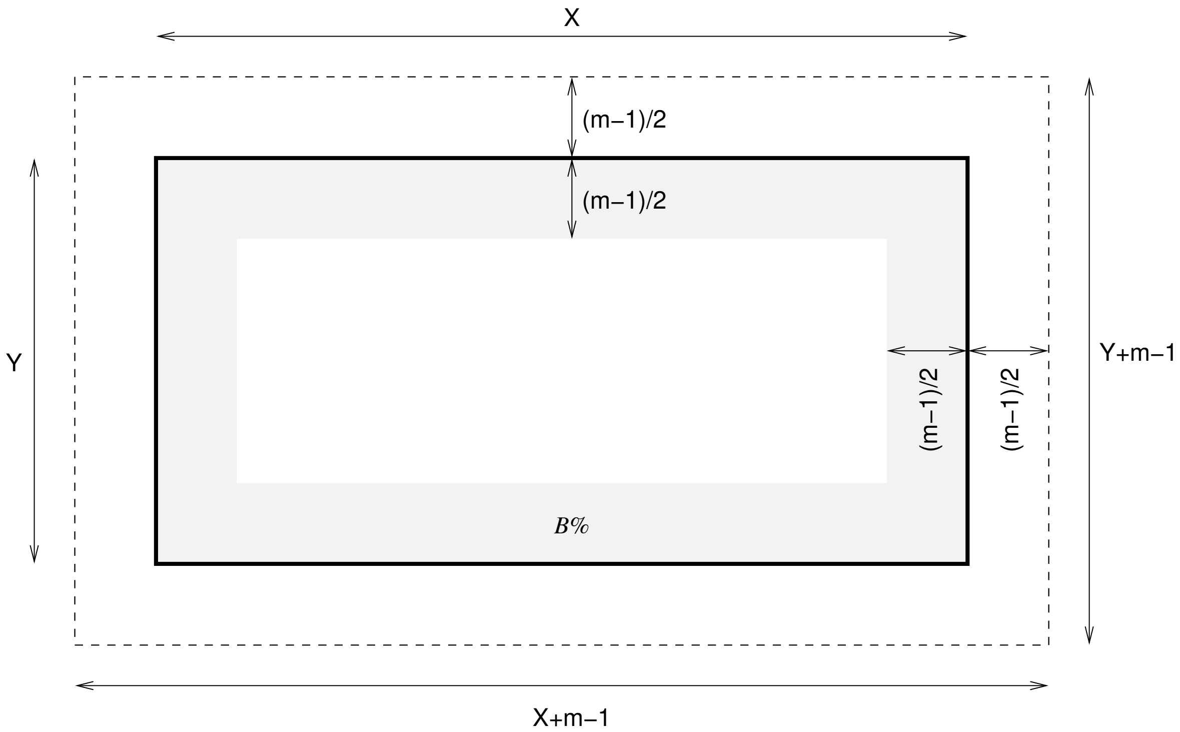

Figure 4.11: The pixels involved in border cases are shaded.

$$B = \frac{(m-1)(X+Y-m+1)}{XY} \qquad (m \text{ is an odd positive integer}) \tag{4.5}$$

Table 4.2 lists the proportion of affected pixels for various common combinations of image and neighbourhood dimensions. For the commonly used 3×3 window, about 1% of processing involves missing pixels. This is not large enough to be a serious concern, especially given the fact that the objects of interest are often positioned near the centre of the image, away from the margins.

4.3 Measuring image similarity

In most of the experiments performed in this thesis, an original image, $\mathbf{f}$, has controlled amounts of noise, $\mathbf{n}$, added to it to produce a noisy version, $\mathbf{f}'$. An example is given in Figure 4.12. A denoising algorithm produces an estimate, $\hat{\mathbf{f}}$, of the original image. In structure preserving filtering it is desired that only noise, and not image structure, be removed. Thus the aim is to produce $\hat{\mathbf{f}}$ as "close as possible" to $\mathbf{f}$.

m	X	Y	B
3	320	240	1.45%
3	512	512	0.78%
3	1280	960	0.36%
5	320	240	2.90%
5	512	512	1.56%
5	1280	960	0.73%
7	320	240	4.33%
7	512	512	2.33%
7	1280	960	1.09%

Table 4.2: Proportion of pixels at the margins for various mask and image sizes

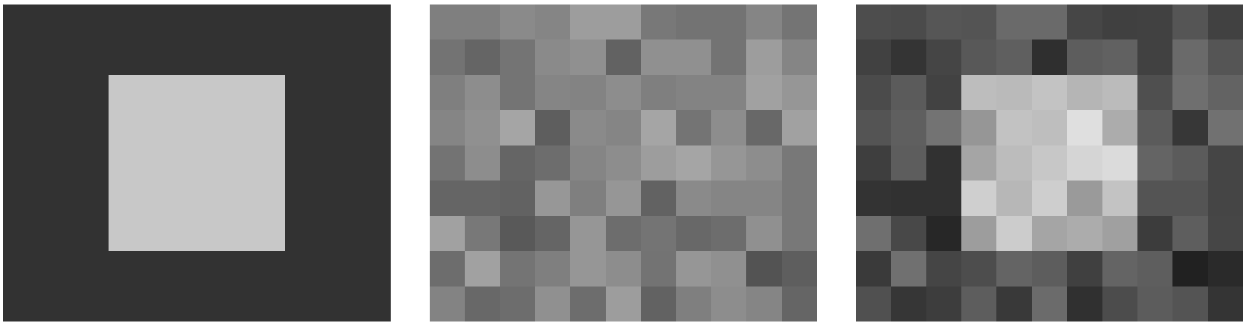

Figure 4.12: Noisy equals original plus noise: (a) original image, $\mathbf{f}$; (b) additive noise, $\mathbf{n}$; (c) noisy image, $\mathbf{f}'=\mathbf{f}+\mathbf{n}$

There are two different situations that may occur when measuring the quality of a denoised image: when the original image is available, and when it is unknown. The first case usually occurs in an experimental situation where a known image is artificially corrupted with noise. The original image is *ground truth*, to which any denoised images may be compared directly. The second case is the more realistic situation whereby a noisy image has been sourced, say remotely from a space telescope, and one wishes to denoise it before further processing. Here there is no ground truth with which to compare. In either case, there are subjective and objective techniques for assessing the quality of denoised images.

4.3.1 Visual inspection

Visual inspection is a subjective process whereby a human viewer assesses the quality of an image. In the case where ground truth is available, one or more assessors may perform a side by side comparison of the denoised and original images. The comparison is usually rated using predefined quality classes, such as “excellent”, “fine”, “passable”, “marginal”, “inferior” and “unusable” [GW92]. This type of experiment requires that the viewing conditions be strictly controlled. This includes factors like ambient lighting levels, the display hardware, and the position of the assessor relative to the displayed images.

Figure 4.13 shows samples of what an assessor may be asked to examine. For this example a reasonable assessment for the denoised image may be “passable”, as there is blotchiness in the foreground and background, and the top left corner of the bright square stands out as being too dark. Obviously though, what is “passable” to one may be “marginal” to another, and so on. However, most assessors will be consistent in their gradings, and a consensus rating can usually be determined.

Figure 4.13: Visual assessment using ground truth: (a) original image; (b) denoised image.

When ground truth is unavailable, it is still possible to perform a visual comparison by inspecting the *noisy* and denoised images. Figure 4.14 shows this situation for the example. Even without ground truth the assessor usually has some *a priori* beliefs on what features are present in the original image, and the denoised image can be examined relative to these beliefs. In effect, the assessor is determining the ground truth himself or herself. The human brain is quite good at identifying structure behind noise, so a skilled assessor could use a noisy image as a guide to what is expected in the denoised output. The qualitative descriptions described earlier can still be used.

4.3.2 Traditional quantitative measures

An objective measure for the quality of a denoised image would be very useful. One traditional measure for the closeness of two data sets is the *root mean squared error*, or

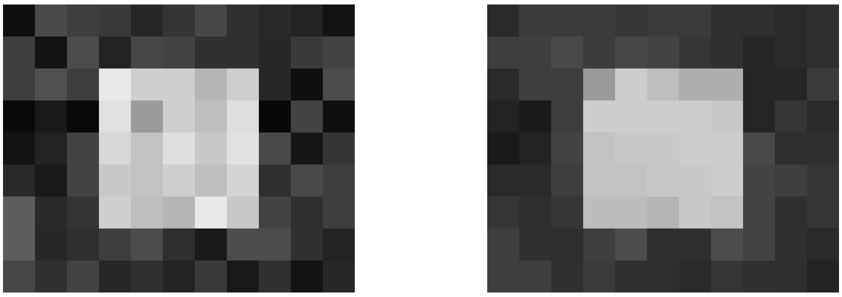

Figure 4.14: Visual assessment without ground truth: (a) noisy image; (b) denoised image.

RMSE [GW92]. When ground truth is available, the two data sets could be the denoised image, $\hat{\mathbf{f}}$, and the original image, $\mathbf{f}$. Equation 4.6 calculates the RMSE in this case.

$$RMSE = \sqrt{\frac{1}{XY}\sum_{y=0}^{Y-1}\sum_{x=0}^{X-1}\left[\hat{f}(x,y) - f(x,y)\right]^2} \tag{4.6}$$

The RMSE is proportional to the disparity between two images. In the case of two equivalent images, it is zero. For the case of additive zero-mean Gaussian noise, the RMSE between the noisy and original images is exactly equal to the noise standard deviation.

The *peak signal to noise ratio*, or PSNR, is derived from the RMSE, and is measured in decibels (dB) [RJ91]. This logarithmic measure is computed using Equation 4.7, where $Z - 1$ is the maximum possible pixel intensity.

$$PSNR = 20\log_{10}\left(\frac{Z-1}{RMSE}\right) \text{ dB} \tag{4.7}$$

RMSE and PSNR use the *square* of the pixel difference. This penalizes large errors very heavily. For example, a single pixel in error by 100 will have the same contribution as 10,000 pixels in error by 1. An alternative measure, which aims to alleviate this potential problem, is the *mean absolute error*, or MAE, calculated using Equation 4.8. The MAE penalizes errors by their magnitude, and is less likely, compared to RMSE, to be biased by occasional large errors.

$$MAE = \frac{1}{XY}\sum_{y=0}^{Y-1}\sum_{x=0}^{X-1}\left|f(x,y) - \hat{f}(x,y)\right| \tag{4.8}$$

The RMSE and MAE are useful in that they provide a number which can be compared objectively. Their drawback is that they do not take into account the spatial distribution of the pixel differences. Many small differences may be more tolerable than fewer larger errors, especially if those errors occur at "busy" locations in the image. In fact, this *perceptual masking* effect [TH94] is exploited by lossy image and audio compression algorithms. Large errors clumped together, or near the image margins, may be preferred the same errors spread around the image. There is evidence to suggest that RMSE may be well correlated to human observers' subjective opinions [MM99]. This fact, combined with its simple formula, has allowed RMSE to be used widely throughout the literature.

The *worst case absolute error*, or WCAE, is the magnitude of the single largest difference between two images. It provides a measure of a denoising algorithm's worst case performance. The calculation of the WCAE is given in Equation 4.9.

$$WCAE = \underset{(x,y)}{\text{argmax}} \left| f(x,y) - \hat{f}(x,y) \right| \tag{4.9}$$

In an experimental situation, artificial noise is added to an original image to create a noisy version thereof. Difficulties can occur if the original image already contains some noise. A good denoising algorithm is likely to remove both the original and artificial noise components, producing a denoised image which is less affected by noise than the original image itself. In this situation one can not expect the denoised image to be the same as the already noisy original. The original noise level is usually low compared to the artificial noise being added, and can be safely ignored. Images generated by synthetic means, such ray-tracing software [Ama87], can be made completely noiseless, not including quantization.

4.3.3 Difference images

A *difference image* has pixels representing the numerical difference between two images. If $[0, Z-1]$ is the possible range of intensity values in the two images being compared, then the range of the difference image will be $[-Z+1, Z-1]$. To view a difference image, the pixel values must be mapped and clamped back to the legal range $[0, Z-1]$. Equation 4.10 computes the difference image, $\mathbf{d}$, between two equal-sized images, $\mathbf{f}_1$ and $\mathbf{f}_2$. If the two images are similar, the difference image mean should be 0, so the $Z/2$ term centres 0 at the middle grey level. The clamping process has the side-effect of not being able to distinguish errors with a magnitude greater than $Z/2$, but these should be rare.

$$\mathbf{d} = clamp\left\lfloor \mathbf{f_1} - \mathbf{f_2} + \frac{Z}{2} \right\rfloor \tag{4.10}$$

Figure 4.15 shows how a difference image can be used to compare a denoised image to the original. The dark and light pixels in the difference image show where the denoising algorithm did a poor job of estimating the original pixels. The mid-grey pixels around the centre area represent where it did well. If the image were denoised perfectly, the difference image would contain exactly the noise which was added.

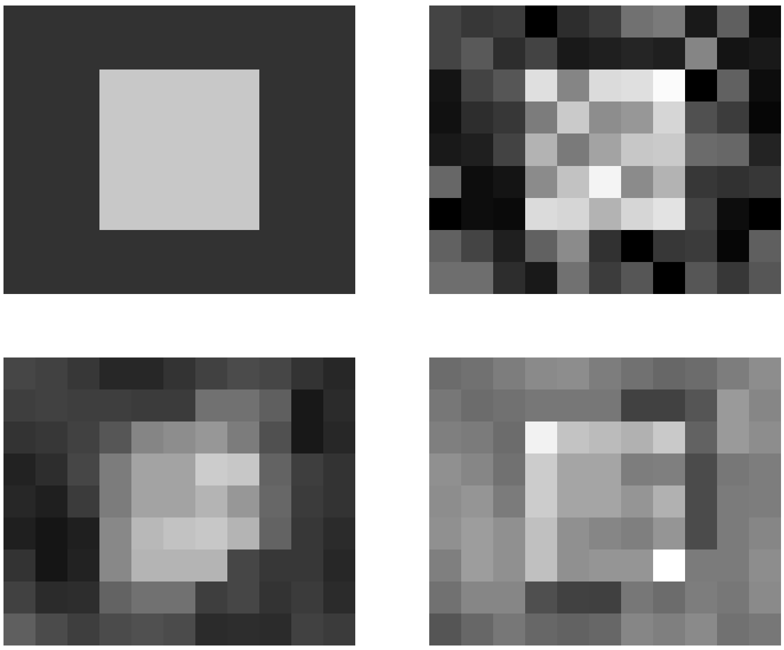

Figure 4.15: Qualitative measure of filter performance: (a) original image; (b) noisy $\sigma = 40$ image; (c) median filtered; (d) difference between original and denoised, mid-grey representing zero.

In the previous example ground truth was available, as Figure 4.15a was synthetically created. Usually, however, the noiseless image will be unavailable. The image processor must use only the noisy image to recover the original pixel information. In this situation, only the difference between the noisy and denoised version can be generated. Figure 4.16 gives an example of this situation. Although there is less apparent structure in the difference image, there are some clumps of very bright and dark pixels near the corners of the square. These errors could correspond to where the filter performed erratically, or to where there were very noisy pixels. Without the original image it is difficult to determine which it is.

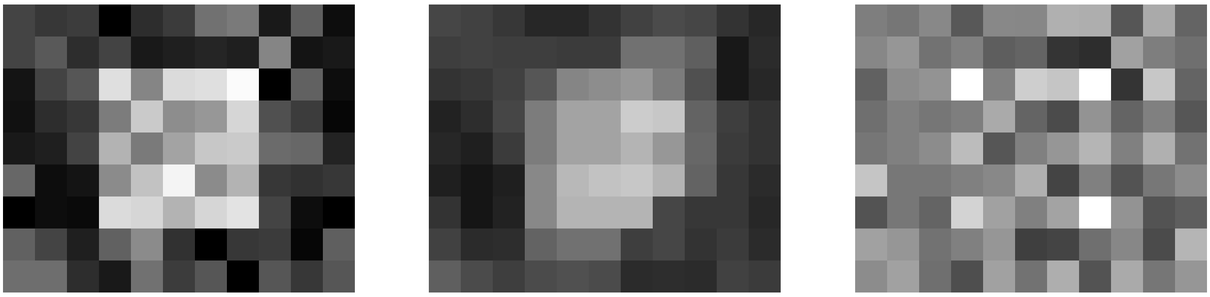

Figure 4.16: Qualitative measure of filter performance: (a) noisy image; (b) median filtered; (c) difference between noisy and denoised, mid-grey representing zero.

4.4 Test images

In this chapter, different denoising algorithms will be compared. It is laborious to provide results for a large set of images at each stage of the discussion. For this reason, one or more images from the small set introduced here will be used consistently throughout this chapter. The use of a larger set of image test set will be deferred until the final results are considered.

4.4.1 Square

The `square` image in Figure 4.17 has already been encountered. It is an 8 bit per pixel greyscale image of resolution 11×9. It consists of a 25 pixel square of intensity 200 atop a 74 pixel background of intensity 50. It will be often used for illustrative purposes, and for *subjectively* examining the effect of various techniques.

4.4.2 Lenna

The `lenna` image [len72] in Figure 4.18 has become a *de facto* standard test image throughout the image processing and image compression literature[2]. Its usefulness lies in the fact

[2] `lenna` is available from `ftp://nic.funet.fi/pub/graphics/misc/test-images/`

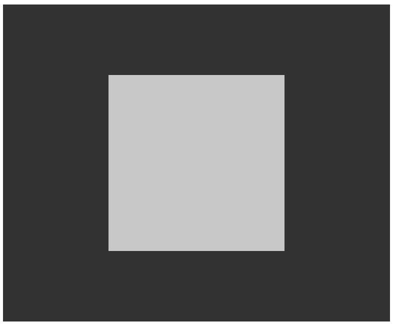

Figure 4.17: The 11×9 8 bpp `square` test image.

that it covers a wide range of image properties, such as flat regions, fine detail, varying edge profiles, occasional scanning errors, and the fact that so many authors produce results using it as a test image. One interesting feature is that the noise in `lenna` seems to be inversely proportional to the brightness, perhaps a legacy of having been scanned from a negative.

Figure 4.18: (a) the 512×512 8 bpp `lenna` image; (b) histogram.

4.4.3 Montage

Figure 4.19 shows a greyscale test image called `montage`, and its histogram. It has resolution 512×512 and uses 8 bits per pixel. The image consists of four quadrants. The top left is the middle 256×256 section of a smoothed version of the `lenna` image[3]. The bottom right is a right hand fragment of a German village street scene, which contains some small amounts of natural noise. The top right is a synthetically created image which is perfectly piece-wise constant. It covers a range of segment boundary shapes and intensity differences between adjacent segments. The bottom left is the same as the top right, except that the segments are piece-wise planar, covering a range of horizontal and vertical gradients.

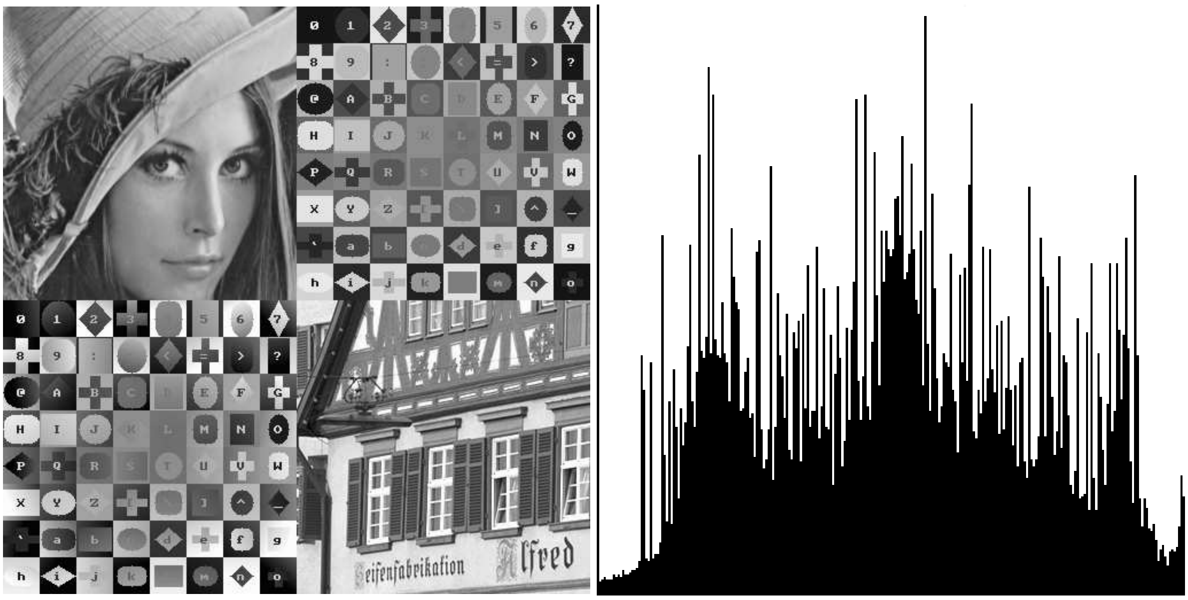

Figure 4.19: (a) the 512×512 8 bpp `montage` image; (b) histogram.

It is hoped that this image covers a wide range of image properties, while also being very low in noise. The low noise level is important for experiments in which synthetic noise will be added. The image has features such as constant, planar and textured regions, step and ramp-like edges, fine details, and homogeneous regions. The `montage` image will be used for measuring *objectively* the RMSE performance of various techniques.

4.5 A one segment model

Local segmentation is concerned with segmenting a small window on a much larger image. The pixel being processed, $f(x, y)$, is usually at the centre of the window. For the following

[3]This image was provided by Bernd Meyer, the author of the TMW algorithm [MT97]. It is available for download from `http://www.csse.monash.edu.au/~torsten/phd/`.

discussion a square 3×3 window containing $M = 9$ pixels will assumed, as shown in Figure 4.20. However, the techniques described may be applied to any window configuration and size. For brevity we will refer to the set of pixels from the window as the vector $\mathbf{p}$. The pixels are indexed in raster order, namely p_1 to p_M. The centre pixel $p_{\lfloor (M+1)/2 \rfloor}$ is the pixel being processed, and may also be denoted simply p, without the subscript.

$$\begin{array}{|c|c|c|}\hline f(x-1,y-1) & f(x,y-1) & f(x+1,y-1) \\ \hline f(x-1,y) & f(x,y) & f(x+1,y) \\ \hline f(x-1,y+1) & f(x,y+1) & f(x+1,y+1) \\ \hline \end{array} \iff \mathbf{p} \iff \begin{array}{|c|c|c|}\hline p_1 & p_2 & p_3 \\ \hline p_4 & p_5 & p_6 \\ \hline p_7 & p_8 & p_9 \\ \hline \end{array}$$

Figure 4.20: Equivalent naming conventions for pixels in the local window.

The simplest form of local segmentation is to do no segmentation at all, and to assume that the pixels in $\mathbf{p}$ are homogeneous. Because the local region is small compared to the whole image, a large proportion of local regions are expected to be homogeneous anyway. This is the assumption made by the linear filters with fixed weights described in Section 3.4.2.

Under a piece-wise constant image model, pixels, $\mathbf{p}$, from the interior of a global segment all have the same true value, denoted μ. Under a $\mathcal{N}(0, \sigma^2)$ noise model, these pixels will have noise, $\mathbf{n}$, added to them. The noisy pixels, $\mathbf{p}'$, have the properties shown in Equation 4.11.

$$\mathbf{p}' = \mathbf{p} + \mathbf{n} \quad \text{where} \quad p_i = \mu \quad \text{and} \quad n_i \sim \mathcal{N}(0, \sigma^2) \tag{4.11}$$

Equation 4.12 shows the noisy pixels to have the same expected value as they did prior to noise being added. However, the uncertainty, or variance, in their values has increased from 0 to σ^2. The noisy pixels may be considered to be distributed $\sim \mathcal{N}(\mu, \sigma^2)$.

$$\begin{array}{rclclcl} \mathrm{E}[\,p'_i\,] & = & \mathrm{E}[\,p_i\,] + \mathrm{E}[\,n_i\,] & = & \mu + 0 & = & \mu \\ \mathrm{Var}[\,p'_i\,] & = & 1^2 \cdot \mathrm{Var}[\,p_i\,] + 1^2 \cdot \mathrm{Var}[\,n_i\,] & = & 1 \cdot 0 + 1 \cdot \sigma^2 & = & \sigma^2 \end{array} \tag{4.12}$$

For this situation, the optimal least squares estimator, $\hat{\mu}$, for μ is the sample mean [MR74], computed using Equation 4.13. Assuming local homogeneity, the sample mean is also the best estimate for every pixel within the window. The variance, or "standard error" of the sample mean is σ^2/M. Thus the more pixels *from the same segment* that are averaged, the more accurate the estimate of their original value is expected to be.

$$\hat{p} = \hat{p}_i = \hat{\mu} = \frac{1}{M}\sum_{i=1}^{M} p'_i \qquad \mathrm{Var}[\,\hat{p}\,] = \frac{\sigma^2}{M} \tag{4.13}$$

Figure 4.21 shows the effect of this filter on the `square` image from Section 4.17. There is clear noise reduction in the centre of the square and in the background, but there is also obvious blurring at the boundary between them. The assumption of only one segment existing in the local window is equivalent to the box filter described in Section 3.4.2, which gives equal weight to each pixel in the window. As a result, it has all the same drawbacks as fixed linear filters. If filtering were repeated, further blurring would occur, eventually producing a completely homogeneous image. This common pixel value would be similar to the image mean, which for `square`, is $(25 \cdot 200 + 74 \cdot 50)/99 = 88$.

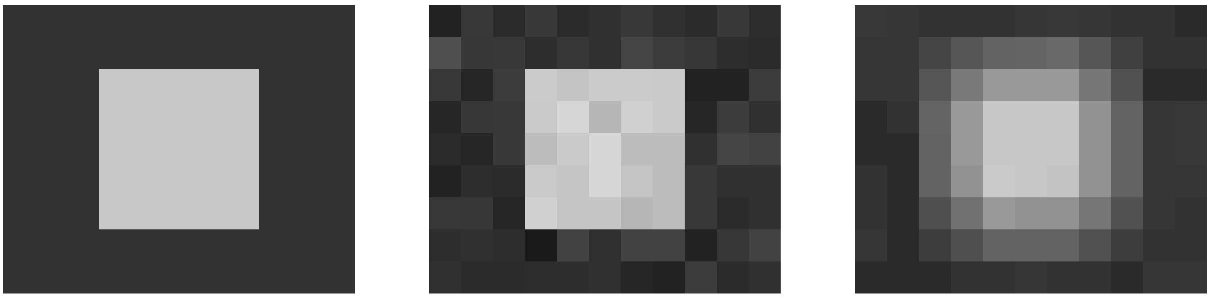

Figure 4.21: (a) original; (b) noisy $\sigma = 10$ version; (c) denoised using $3{\times}3$ averaging.

4.6 A two segment model

It may be thought that because the majority of pixels exist in homogeneous regions, that treating all windows as such would not overly affect results. Unfortunately, although heterogeneous windows are in the minority, they do contain a lot of visual information in the form of edges and texture. The human visual system is very sensitive to discontinuities, and any blurring or alteration to sharp edges would not go unnoticed [Ede99]. It is therefore important to preserve as much information in these regions as possible.

Reconsider the `square` image from Figure 4.21. Globally, it consists of only two segments, thus any local window must contain, at most, two segments. One solution for preserving edges is to first classify the pixels from the window into two segments. Only pixels belonging to the *same segment* could be used to calculate the denoised pixel values. This would prevent the mixing of pixels from different global segments, resulting in a sharper image.

The division of pixels into two segments is obviously a form of segmentation. The BTC technique, discussed in Section 3.3.2, successfully uses a two-class model for encoding 4×4 pixel blocks. In most variants of BTC a single threshold, T, is used to divide the pixels into two clusters [FNK94]. The representative values, $\hat{\mu}_1$ and $\hat{\mu}_2$, for each cluster are set equal to the average of the pixels within each cluster. This is shown in Equation 4.14, where m_1 and m_2 are the number of pixels in each cluster.

$$\hat{\mu}_1 = \frac{1}{m_1} \sum_{p'_i \leq T} p'_i \qquad \hat{\mu}_2 = \frac{1}{m_2} \sum_{p'_i > T} p'_i \tag{4.14}$$

The threshold, T, may be chosen in many ways. Many techniques, including the popular *Absolute Moment BTC* [LM84, CC94, MR95], simply use the mean of the pixels in the window, calculated using Equation 4.15.

$$T = \frac{1}{M} \sum_{i=1}^{M} p'_i \tag{4.15}$$

The mean measures the central tendency of the pixel values in the block. When the two clusters have unequal numbers of pixels, the mean is biased toward the centroid of the larger cluster. This pollutes the smaller cluster with pixels from the larger cluster, biasing both clusters' means. This effect is shown in Figure 4.22.

When the clusters are known to be normally distributed, it is better to choose T such that the overall mean squared error (MSE) is minimized, as shown in Equation 4.16. The optimal T can be found by exhaustive search of the $M - 1$ possible binarizing thresholds [DM79].

$$T = \underset{T \in (0, Z-1)}{\text{argmin}} \quad \sum_{p'_i \leq T} (p'_i - \hat{\mu}_1)^2 + \sum_{p'_i > T} (p'_i - \hat{\mu}_2)^2 \tag{4.16}$$

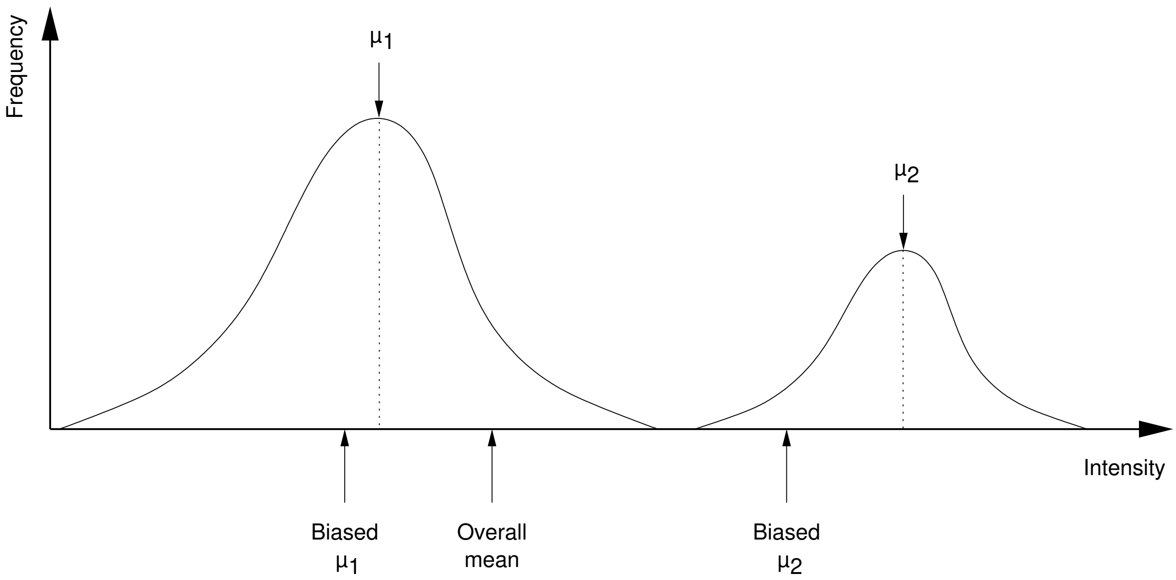

Figure 4.22: The mean is a poor threshold when the clusters have different populations.

An iterative technique for determining the MSE threshold in Equation 4.16 was proposed by Efrati *et al* [ELM91]. It uses the block mean as the initial threshold. The computed cluster means are then themselves averaged to produce a new threshold, and new cluster means computed. This continues until there is no significant change in the threshold. The algorithm is outlined in Listing 4.1, where ϵ quantifies the required accuracy. Usually pixel intensities have integer values, so the algorithm can terminate if the integer part of T is no longer changing.

1. Let $T = \frac{1}{M} \sum_{i=1}^{M} p'_i$.
2. Compute $\hat{\mu}_1$ and $\hat{\mu}_2$ as in Equation 4.14.
3. If $|\frac{1}{2}(\hat{\mu}_1 + \hat{\mu}_2) - T| < \epsilon$ then exit.
4. Let the new $T = \frac{1}{2}(\hat{\mu}_1 + \hat{\mu}_2)$ and go to Step 2.

Listing 4.1: An iterative method for choosing a binary threshold.

This algorithm produces means very similar to the levels produced by the Lloyd quantizer [Llo82], and is often referred to as Lloyd's algorithm. In their BTC survey paper, Fränti *et al* [FNK94] found Lloyd's algorithm to produce results identical to a true MSE optimizer in nearly all situations, all while using only 1.71 iterations on average. It was not made clear in which situations it failed — perhaps it was due to the mean being a poor initial threshold, resulting in convergence to a local, rather than global, optimum. Nevertheless, this method will be used for selecting binary thresholds in this chapter.

4.6.1 Application to denoising

Equation 4.17 describes how binary clustering may be used to denoise the pixels which were clustered. The denoised pixels, $\hat{p}_i$, are set equal to the mean of the cluster in which they belong. For this clustering algorithm, this corresponds to the cluster mean that each noisy pixel is closest in intensity to.

$$\hat{p}_i = \begin{cases} \hat{\mu}_1 & \text{if} \quad p'_i \leq T \\ \hat{\mu}_2 & \text{if} \quad p'_i > T \end{cases} \tag{4.17}$$

Figure 4.23 compares using 1-segment and 2-segment models for denoising `square`. Filtering under the assumption of two segments preserves the edges of the object much better. The thresholding procedure appears to have correctly classified the pixels, even in the presence of additive Gaussian noise. Within homogeneous regions the smoothing is a little worse, and is especially noticeable in the centre of the light square. This occurs because the pixels in each window are forcibly divided into two segments, even those which are homogeneous. Thus the variance within each local cluster is reduced only by a factor of m_1 or m_2, compared to $M = m_1 + m_2$ when the window is treated homogeneously. The possibility of applying the filter again to the denoised output for further smoothing is discussed in Section 4.8.

Figure 4.24 quantitatively compares the denoising performance of the 1-segment and 2-segment local models for the `montage` image. For each value of σ tested, artificial noise $\sim \mathcal{N}(0, \sigma^2)$ was added to `montage`, and the noisy pixels rounded to integers. The RMSE was measured between the original `montage` and the denoised output of each algorithm. The 2-segment model performs significantly better for all noise levels up to $\sigma = 27$. At the

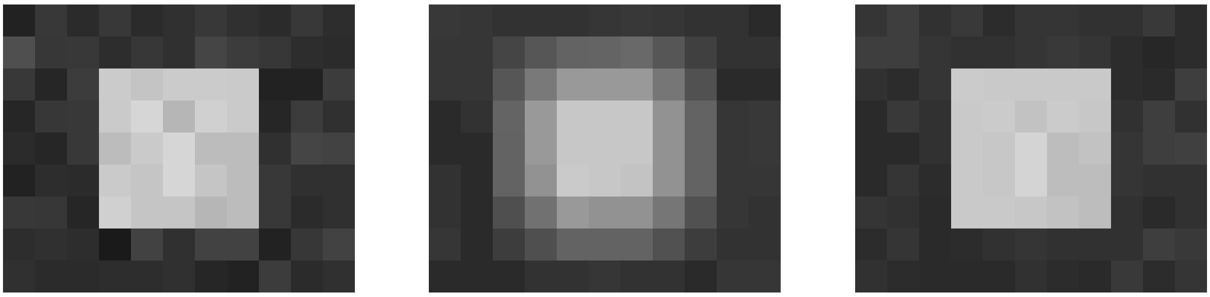

Figure 4.23: (a) noisy, $\sigma = 10$; (b) filtered with 1-segment model; (c) filtered with 2-segment model.

highest noise levels, the distinction between adjacent global segments becomes less clear, especially if the contrast between them is low. When the noise swamps the signal, inappropriate thresholds are more likely to be chosen.

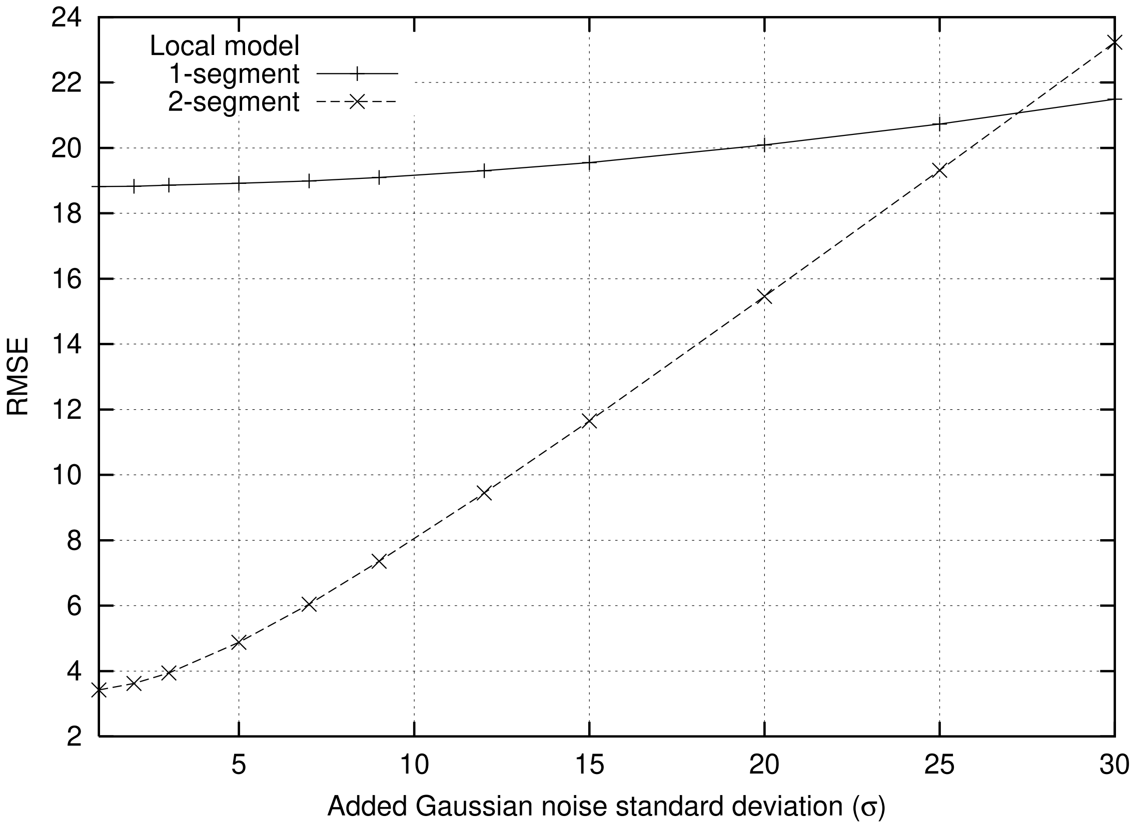

Figure 4.24: Comparison of 1-segment and 2-segment models for denoising `montage`.

4.7 Discriminating between two models

Modeling the local region as two segments helped to preserve edges and decrease the RMSE between the original and denoised images. However, within homogeneous regions denoising performance decreased. It would be desirable to be able to switch automatically between one and two segment models as appropriate. The general form of this problem is called *model order selection*, and many statistical and information-theoretic techniques have been developed to tackle it [WF87, Ris87, Pre89, BS94].

If k is the number of segments in the local window, the model selection problem can be framed as choosing either $k = 1$ or $k = 2$. Consider an image with a dark background of intensity μ_1 comprising $3/4$ of the total pixels, and a light foreground of intensity μ_2 comprising the remaining $1/4$ of the pixels. If this image were corrupted by additive noise $\sim \mathcal{N}(0, \sigma^2)$, its histogram would have the form of in Equation 4.18.

$$\frac{3}{4}\mathcal{N}(\mu_1, \sigma^2) + \frac{1}{4}\mathcal{N}(\mu_2, \sigma^2) \tag{4.18}$$

Figure 4.25a shows this histogram for $\mu_1 = 3$, $\mu_2 = 9$ and noise standard deviation $\sigma = 1$. It is clearly bimodal, because the two cluster means are 6σ apart. It was shown in Section 4.2.5 that 99.7% of normally distributed pixels will, on average, fall within 3σ of their mean. Figures 4.25b-d show the histogram's behaviour as the distance between the two means, $|\mu_1 - \mu_2|$, is decreased in multiples of the noise variance. At some point between 2σ and 3σ the histogram becomes unimodal, but not Gaussian, in nature. The exact point at which this occurs would also depend on the number of pixels in each cluster. The more unequal the blend, the more likely it is to appear unimodal.

From the previous observations it would be reasonable to suggest that a method for deciding between $k = 1$ and $k = 2$ should be based on determining whether the two clusters are well separated. If we restrict the criterion to use the difference between cluster means, the threshold should depend on the noise variance and the number of pixels in each cluster. Equation 4.19 gives a template for the proposed model order selection technique.

$$k = \begin{cases} 1 & \text{if} \quad |\hat{\mu}_1 - \hat{\mu}_2| \leq g(\sigma, m_1, m_2) \\ 2 & \text{otherwise} \end{cases} \tag{4.19}$$

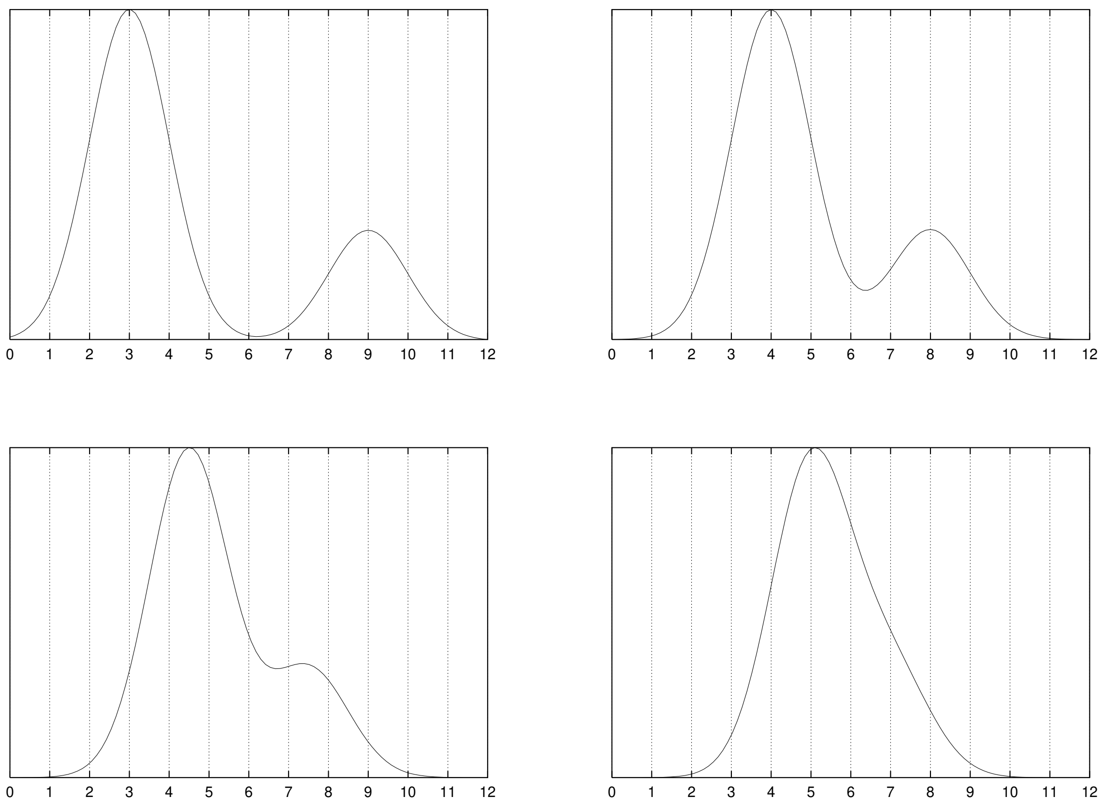

Figure 4.25: [top to bottom, left to right] A mixture of two normal distributions with common variance, with their means separated by (a) 6σ; (b) 4σ; (c) 3σ; (d) 2σ.

The two estimated cluster means, $\hat{\mu}_1$ and $\hat{\mu}_2$, are calculated using Equation 4.14. The hypothesis that there are two segments is rejected if the cluster means are too close together. The measure of closeness is decided by comparing the inter-cluster distance with the output of a threshold function, $g(\sigma, m_1, m_2)$. The threshold function may depend on the image noise variance and the number of pixels in each cluster. This should make the thresholding process adaptive to the image noise level.

4.7.1 A simple model selection criterion

Figure 4.25 showed that the two populations remained distinct up to a separation of around 4σ or so, corresponding to about a 95% confidence interval. There is no "exact" answer, as it varies for different cluster populations, but it is obvious that the magnitude of the noise variance is an important factor. The simplest, reasonable form for the separation threshold is given in Equation 4.20, where C is a positive constant.

$$g(\sigma, m_1, m_2) = C\sigma \tag{4.20}$$

This separation threshold is a form of Fisher's criterion (Equation 3.3) under the assumption of equal cluster variances. The two are related by the fact that $C = \sqrt{2\lambda}$ if $\sigma = \sigma_A = \sigma_B$. The problem is choosing an appropriate value for C.

Visual inspection

Consider the `square` image again, which has a background intensity of 50 and a foreground intensity of 200. Figures 4.26a shows a noisy $\sigma = 10$ version of this image. Figures 4.26b–f show the denoised output values of C equal to 1, 2, 3, 4 and 5 respectively. For each pixel, if $k = 1$ is chosen, the denoising filter uses the one segment model of Section 4.5. If $k = 2$ is chosen, the two segment model of Section 4.6 is applied. In Figure 4.26, the foreground and background are separated by 15σ, so no difficulties are expected. The denoised images for $C \geq 3$ are equivalent, having successfully smoothed nearly the whole image.

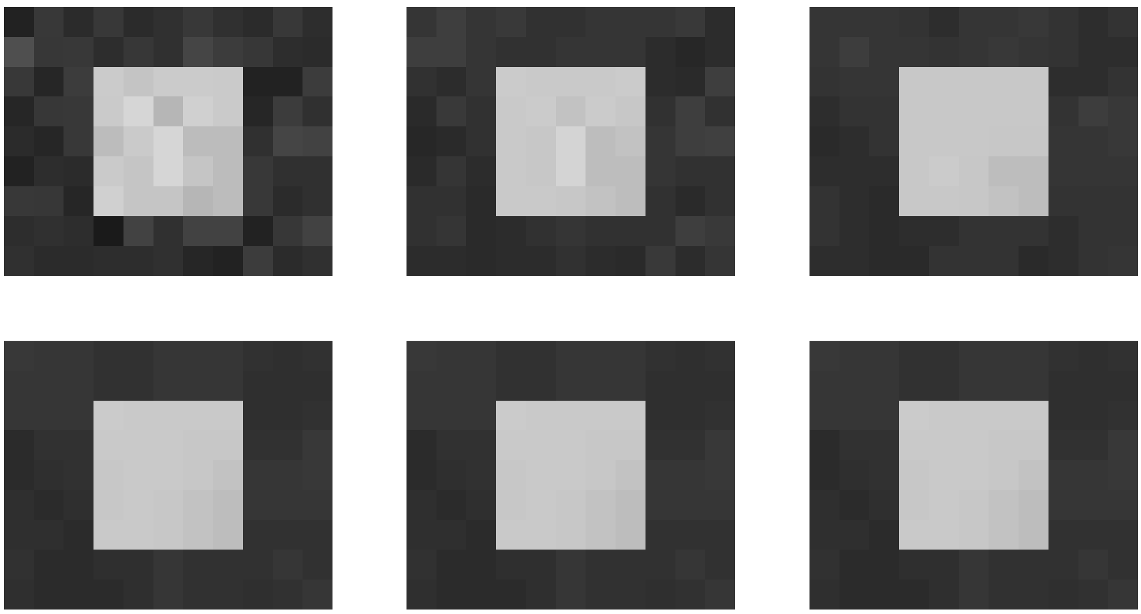

Figure 4.26: (a) noisy, $\sigma = 10$; (b)—(f) simple model selection using $C = 1 \ldots 5$.

Figure 4.27 provides the same results as Figure 4.26, except that the noise has been increased to $\sigma = 20$. The the two pixel populations are still well separated by 7.5σ. Once again, for $C \geq 3$, the maximum possible smoothing has occurred.

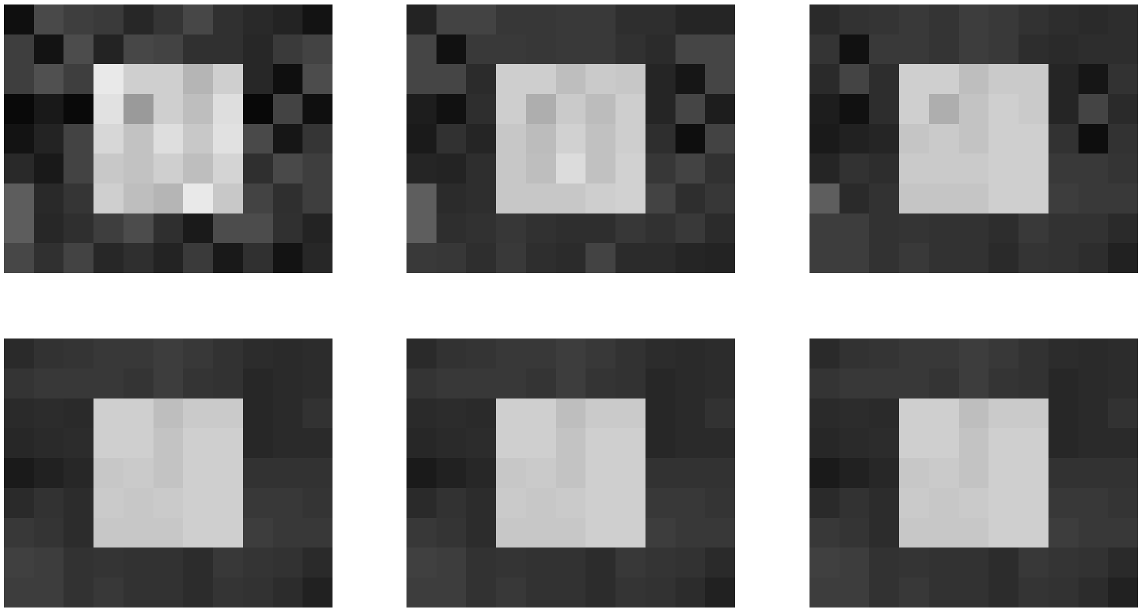

Figure 4.27: (a) noisy, $\sigma = 20$; (b)—(f) simple model selection using $C = 1 \dots 5$.

Figure 4.28 shows a much more visibly noisy image with $\sigma = 30$, corresponding to only a 5σ intensity difference between the two image segments. At this separation, there is a 2% overlap of the two populations. The best smoothing occurs when C equals 3 or 4. When $C = 5$, some blurring of segment boundaries is observed, meaning $k = 1$ was sometimes chosen incorrectly. When the global segment means are only 5σ apart, insisting on an equivalent cluster separation at the local level will sometimes be incorrect.

Finally, Figure 4.29 presents the very noisy case of $\sigma = 40$. The true object intensities are only 3.75σ apart, corresponding to a 10% overlap. Thus it is expected that about 10 of the 99 pixels will have intensities which could be mis-classified. The best denoising occurs when C equals 2 or 3, depending on what type of artifacts are preferred. As expected, for $C \geq 4$, the model selection criterion fails miserably, blurring all the edges around the square.

From this analysis it may be concluded that $C = 3$ is a good all-round choice for the `square` image when the noise has additive, zero-mean Gaussian characteristics. The disadvantage of the $C\sigma$ threshold is that it does not consider the number of pixels in each cluster. There is low confidence in the "average" of a cluster containing 1 pixel compared to its 8 pixel counterpart, because the standard error of the former is 8 times higher. A further problem with a fixed threshold is that two segments with an intensity difference of less than $C\sigma$ will probably never be detected.

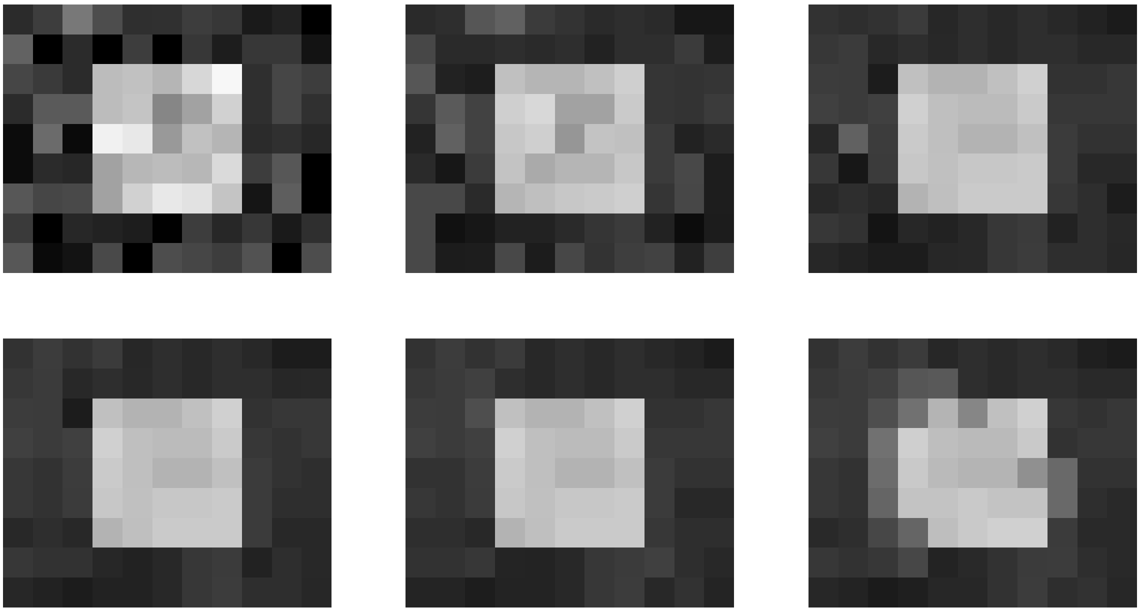

Figure 4.28: (a) noisy, $\sigma = 30$; (b)—(f) simple model selection using $C = 1 \ldots 5$.

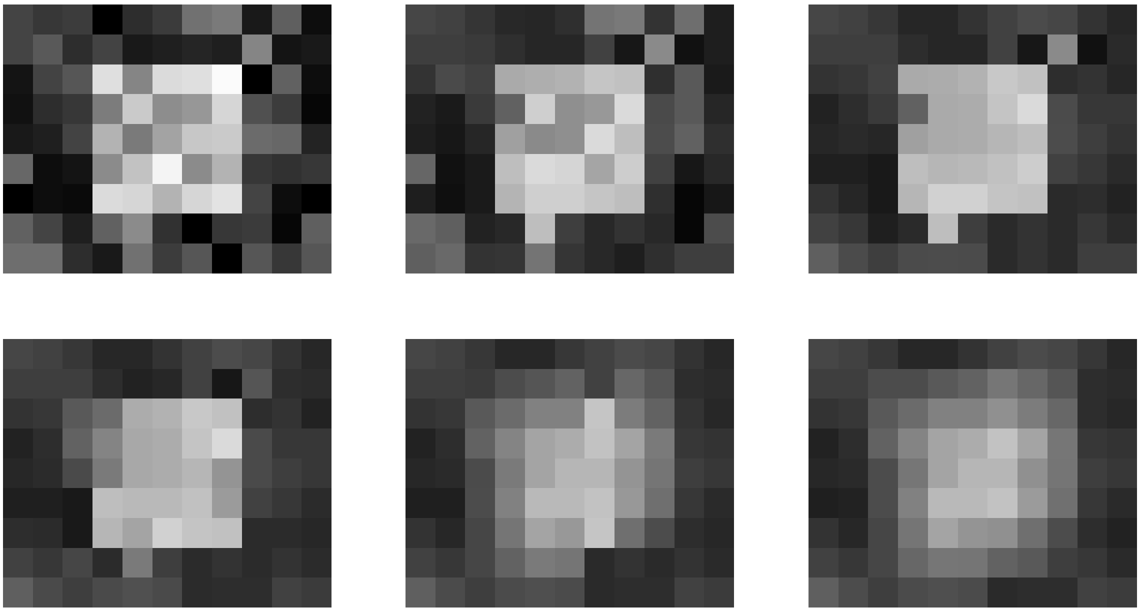

Figure 4.29: (a) noisy, $\sigma = 40$; (b)—(f) simple model selection using $C = 1 \ldots 5$.

Quantitative results

Figure 4.30 shows the RMSE between the denoised and original `montage` image for 6 integer values of C. One would conclude that $C = 3$ performs best on this image over all the noise levels. This supports the conclusion made in Section 4.7.1.

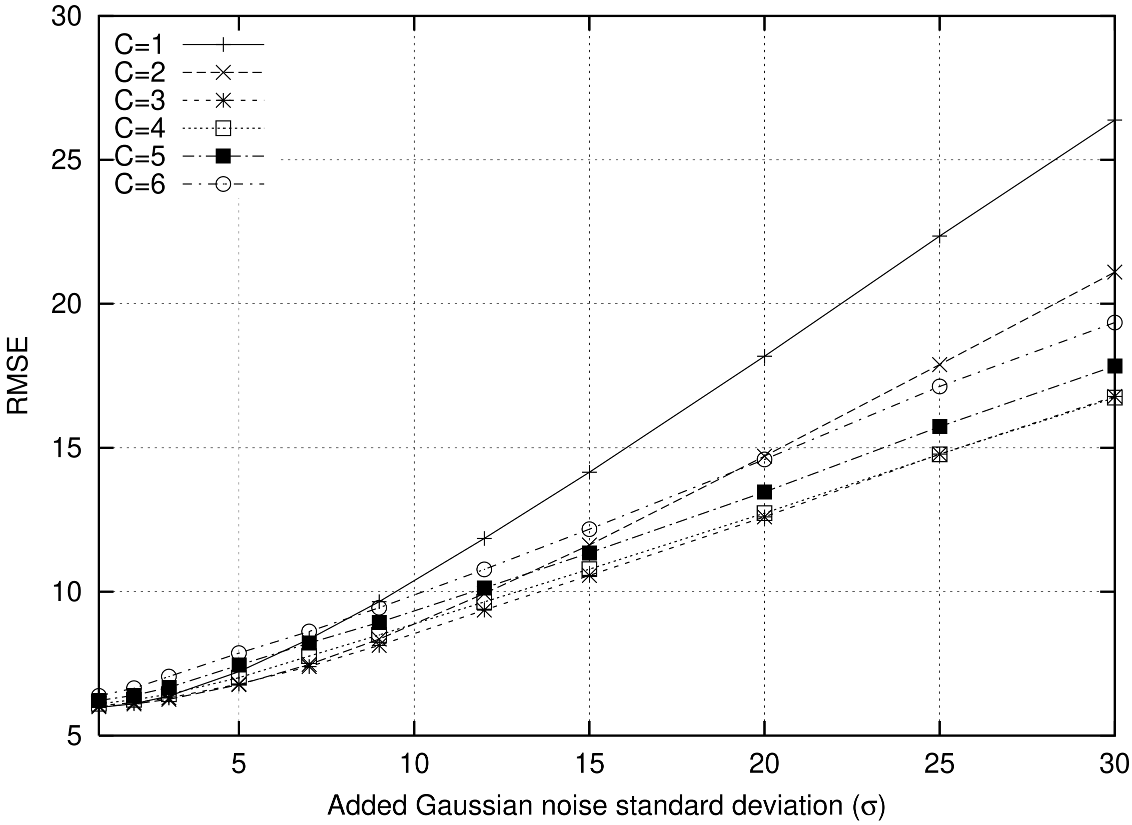

Figure 4.30: Effect of C on RMSE when denoising `montage` by switching between a 1-segment and 2-segment local model.

4.7.2 Student's t-test

The simple separation criterion, $g(\cdot) = C\sigma$, does not take into consideration the number of pixels in each cluster. As the number of pixels in a cluster decreases, we have less confidence that the computed cluster mean is a good estimate of the true segment mean. For the 3×3 case, the separation criterion for the situation where both clusters are about the same size, say $m_1 = 5$ and $m_2 = 4$, should be different to the skewed case when $m_1 = 1$ and $m_2 = 8$. In the latter, there is a much greater risk that the single pixel cluster would be mis-classified, as its mean was derived from a single noisy pixel.

The problem of deciding between $k = 1$ and $k = 2$ may be framed as traditional hypothesis testing of the equality of two population means [Gro88]. The *null hypothesis* states that the two class means are the same, and that only one class exists. The *alternative hypothesis* states that the means are not equal, and represent independent sub-populations. The application to model selection in local segmentation is shown in Table 4.3.

One segment	$k = 1$	$\iff$	$\mu_1 = \mu_2$	Null hypothesis
Two segments	$k = 2$	$\iff$	$\mu_1 \neq \mu_2$	Alternative hypothesis

Table 4.3: Using hypothesis testing for model selection in local segmentation.

If it is assumed *a priori* that the segments are piece-wise constant with additive Gaussian noise $\sim \mathcal{N}(0, \sigma^2)$, a statistical "$t$-test" can be used to determine the more likely hypothesis [Gro88]. In Equation 4.21, t is a random variable following Student's t distribution with $\nu = m_1 + m_2 - 2$ degrees of freedom.

$$t = \frac{\hat{\mu}_1 - \hat{\mu}_2}{\sigma\sqrt{\frac{1}{m_1} + \frac{1}{m_2}}} \tag{4.21}$$

If $|t|$ is less than a critical value, t_{crit}, the null hypothesis is accepted, and the local region is inferred to be homogeneous. If $|t| \geq t_{crit}$, the pixels are assumed to come from two populations. Equation 4.22 summarizes the t-test model selection criterion for local segmentation.

$$k = \begin{cases} 1 & \text{if} \quad |t| < t_{crit} \\ 2 & \text{if} \quad |t| \geq t_{crit} \end{cases} \tag{4.22}$$

The value of t_{crit} depends both on the degrees of freedom, ν, of t, and the percentage confidence one wishes to have in the inference. Because the same window is used for each pixel, ν is fixed at $m_1 + m_2 - 2 = M - 2$. To achieve $100(1 - 2\alpha)\%$ confidence, t_{crit} should equal $t_{\alpha,M-2}$. This may be calculated using numerical integration, or from pre-computed tables [PH66], an extract of which is given in Table 4.4. For the case of $\nu = \infty$, Student's t distribution becomes a normal distribution.

For a window of M pixels, and a predetermined value for α, the mean separation threshold function, $g(\sigma, m_1, m_2)$, may be written as follows:

$$|t| \quad < \quad t_{crit} \tag{4.23}$$

$$\frac{|\hat{\mu}_1 - \hat{\mu}_2|}{\sigma\sqrt{\frac{1}{m_1} + \frac{1}{m_2}}} \quad < \quad t_{crit} \tag{4.24}$$

Degrees of freedom ν	α .10	.025	.005
2	1.886	4.303	9.925
7	1.415	2.365	3.499
11	1.363	2.201	3.106
19	1.328	2.093	2.861
23	1.319	2.069	2.807
∞	1.282	1.960	2.576

Table 4.4: Various values of $t_{\alpha,\nu}$ used in t-testing.

$$|\hat{\mu}_1 - \hat{\mu}_2| < t_{crit}\,\sigma\sqrt{\frac{m_1+m_2}{m_1 m_2}} \tag{4.25}$$

$$\therefore \quad g(\sigma, m_1, m_2) = t_{crit}\,\sigma\sqrt{\frac{M}{m_1 m_2}} \tag{4.26}$$

Figure 4.31 compares four different values of α for denoising `montage` using the t-test criterion. The value of α is used to determine dynamically the mean separation threshold in Equation 4.26. A 3×3 filter is used, so $M = 9$, and the noise variance is assumed known. At very low noise levels, the value of α does not appear to affect results, but as σ is increased, the lowest value of $\alpha = 0.005$ does best. This is the highest of the confidence levels tried.

In this thesis a significance level of $\alpha = 0.005$, or 99% confidence, will be used. For 3×3 windows, $\nu = 7$, so $t_{crit} = t_{\alpha/2,M-2} = t_{0.005,7} = 3.499$, or approximately 3.5. In this case, the separation function reduces to Equation 4.27.

$$g(\sigma, m_1, m_2) = \frac{10.5\,\sigma}{\sqrt{m_1 m_2}} \tag{4.27}$$

The t-test threshold function has a similar form to the simple one in Section 4.7.1, except that C has been replaced by a term inversely related to the *geometric mean* of the two cluster populations. Figure 4.32 plots m_1 versus the effective number of σ separations required for $k = 2$ to be declared. At the extremes it has values between 3 and 4, but at the middle where the cluster populations are most symmetric is goes as low as 2.35. The t-test approach allows cluster means to be close together as long as the evidence, in the form of more accurate mean estimates, is strong enough. When the cluster counts are skewed, the criterion insists on a wider separation before accepting them.

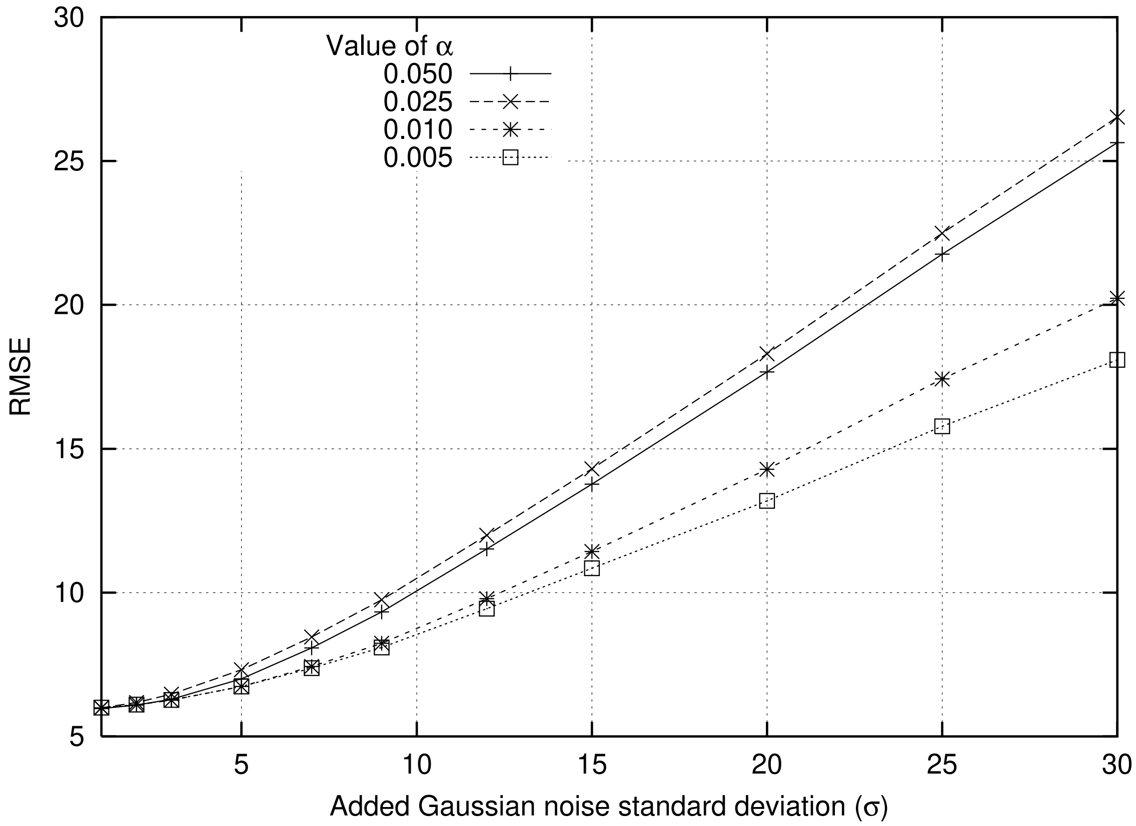

Figure 4.31: Comparison of α values for t-test filtering of `montage`.

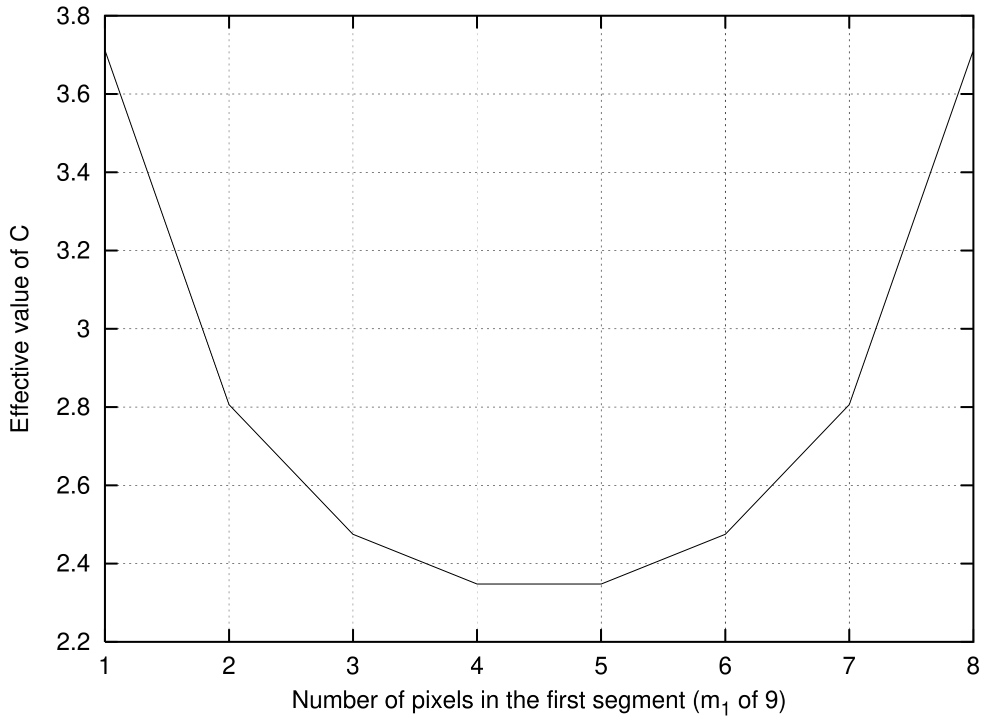

Figure 4.32: The effective C values used by the t-test criterion: $\nu = 7$ and $\alpha = 0.005$.

In an implementation, Equation 4.26 would not need to be recomputed for each pixel. For a given image, the values of t_{crit}, M and σ are constants. Because $m_2 = M - m_1$, the only degree of freedom within a window is m_1. Figure 4.32 is also symmetric around $M/2$, so there are only $(M-1)/2$ unique thresholds. These may be pre-computed and stored in a look-up table. This feature would be very useful in a hardware implementation.

Visual inspection

Figure 4.33 provides a visual comparison of the t-test and $C = 3$ model selection functions. The four rows correspond to added noise standard deviations of 10, 20, 30 and 40. The three columns contain the noisy image, the $C = 3$ denoised image, and the t-test denoised image.

When $\sigma = 10$, there is little to distinguish the two techniques. For $\sigma = 20$, the light squares have been filtered equally well, but the t-test has performed slightly worse on the background. For all cases other than $m_1 = 1$ or 9, the effective value of C determined by the t-test is less than 3. Therefore, on average, it will choose $k = 2$ more often than the simple criterion. If there is a very noisy pixel present in a homogeneous area, the t-test criterion is more likely to place that pixel in a segment of its own.

Interestingly, increasing the noise to $\sigma = 30$ removes any observable differences between the two methods. Normally, behaviour similar to that observed for $\sigma = 20$ is expected. But it seems in this case that, by random chance, no extreme pixel values were present in the noisy image. When $\sigma = 40$, the denoised outputs of both methods contain obvious artifacts. The filtering errors in the $C = 3$ output image are milder, but a little more frequent, than those observed in the t-test output.

Both techniques suffer at similar locations in the image. Consider a local window straddling the edge between the square and the background. If some of the noisy square's pixels are darker than the original background, or some of the noisy background are lighter than the original square, both algorithms will fail. The well-known fundamental drawback of thresholding algorithms is that they are unable to take spatial coherence into account. Unlike the human visual system, the local segmentation method described thus far is not sophisticated enough to recognize and extrapolate edge information.

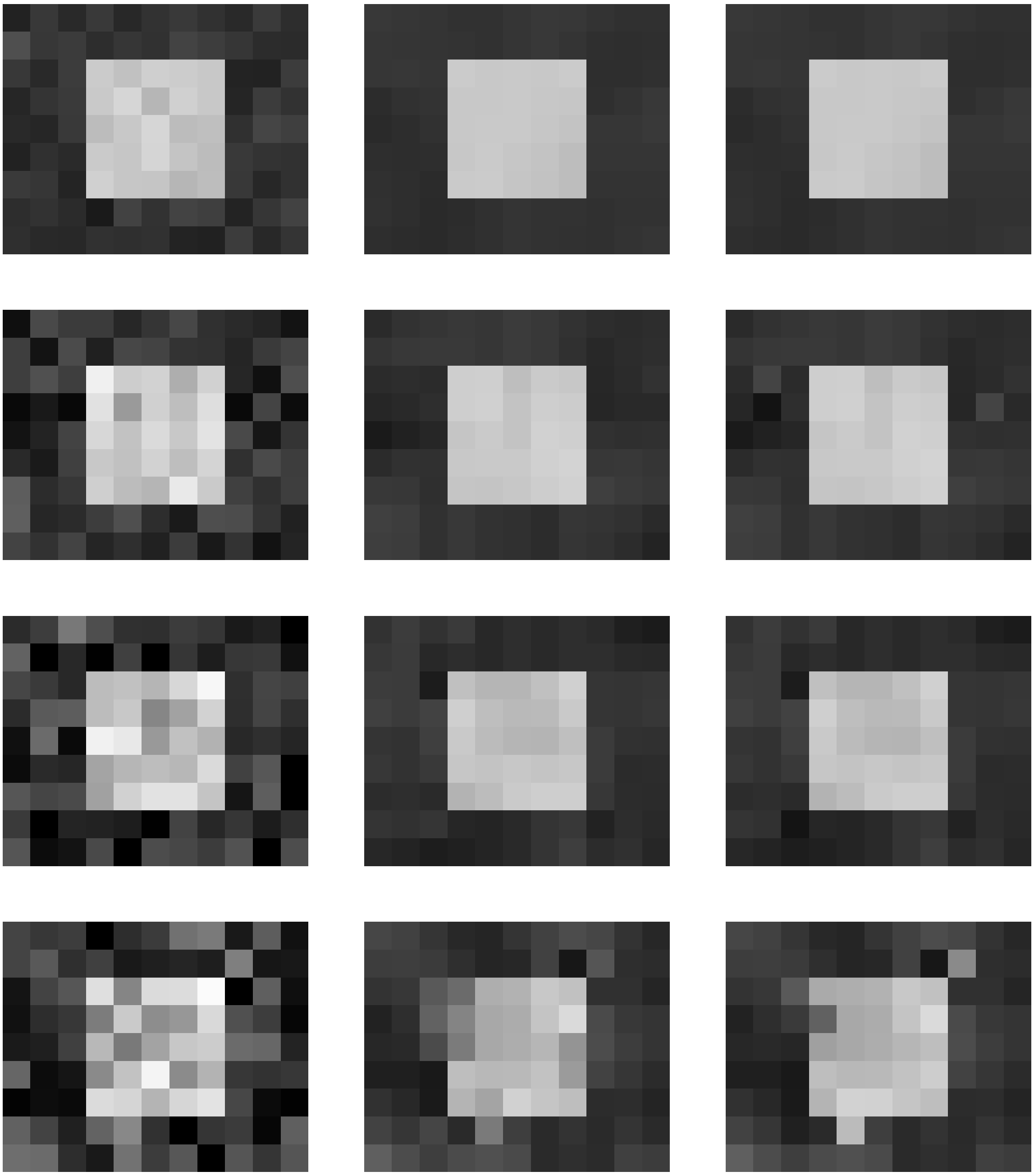

Figure 4.33: Visual comparison of two model selection functions. Columns are: noisy, $C = 3$ filtered, and t-test filtered images. Rows are: σ =10, 20, 30 and 40.

Quantitative results

Figure 4.34 shows that, for `montage`, the $C = 3$ and t-test criteria perform equivalently at low noise levels. For values of $\sigma \geq 12$, the simpler $C = 3$ criterion appears to do better. So far, only a $3{\times}3$ window has been considered. It must be noted that the t-test criterion varies more as the window size increases. Although not shown, I have performed further experiments which show that the t-test criterion still performs worse when the window size is increased to 21 and 25 pixels. This fact, in conjunction with the lack of compelling evidence in the qualitative results, recommends the use of the simple $C = 3$ model selection criterion over the more complicated t-test criterion.

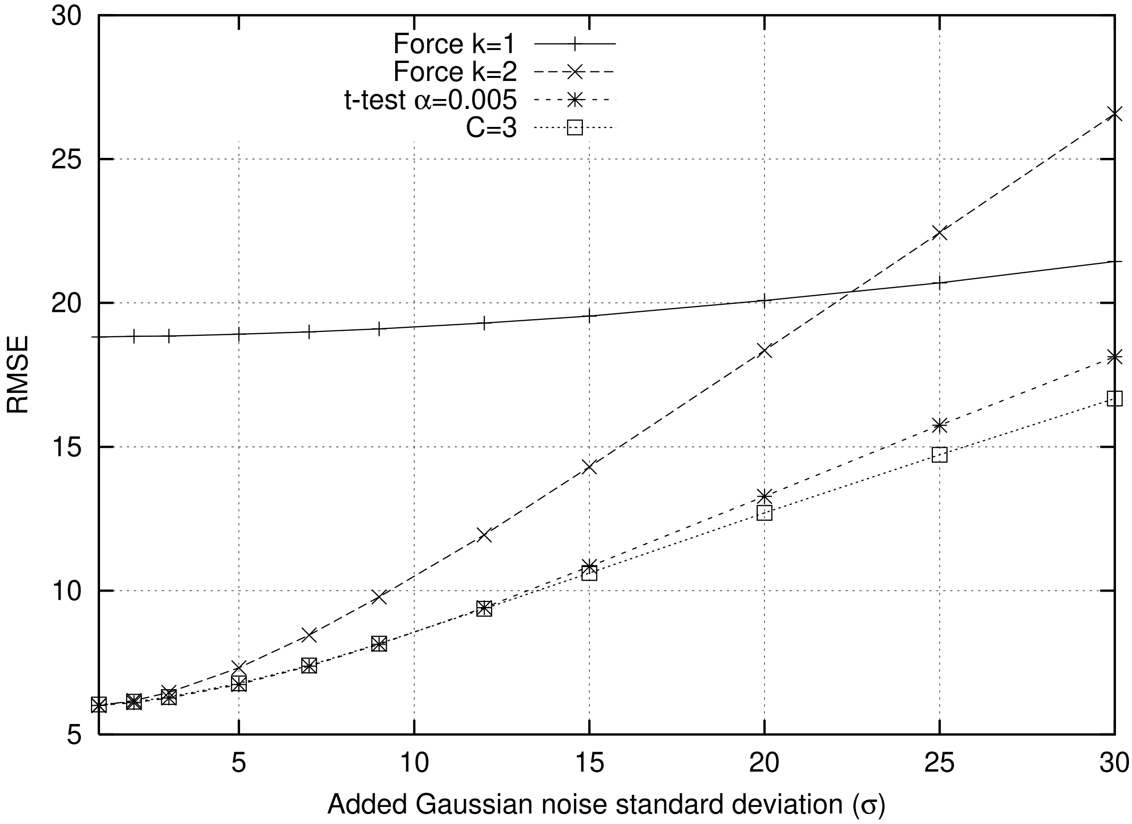

Figure 4.34: RMSE comparison of model selection criteria for `montage`.

4.8 Iterative reapplication of the filter

It is possible to take the output of a denoising algorithm and feed it back as input to the same algorithm, *ad infinitum*. Iteration has often been used as a way to improve the performance of denoising algorithms [Mas85], particularly anisotropic diffusion [PM90]. Equation 4.28 describes this process formally, where $\hat{\mathbf{f}}_t$ is the denoised image after the t^{th} iteration. Ideally, all pixel values are kept to floating point accuracy throughout. Rounding to integers after each iteration introduces feedback of quantization noise, which could cause unusual artifacts.

$$\hat{\mathbf{f}}_{t+1} = \mathrm{filter}(\hat{\mathbf{f}}_t) \qquad \text{where} \qquad \hat{\mathbf{f}}_0 = \mathbf{f}' \tag{4.28}$$

The effect on the final output as t increases depends wholly on the properties of the denoising algorithm. A *stable* algorithm is defined as one which, after a finite number of iterations t', produces no further changes in the output. Thus $\hat{\mathbf{f}}_{t'} \equiv \hat{\mathbf{f}}_\infty$, and is referred to as the *root image* [FCG85]. An example of an unstable algorithm is the median filter. Imagine using a local window of 5 pixels — the centre pixels and its four nearest neighbours — and handling the image margins by taking the median of the *available* pixels. Applying this median filter to the checkerboard image in Figure 4.35a results in the *inverted* checkerboard of Figure 4.35b. Reapplying for a second iteration produces Figure 4.35c, which is identical to the original image. In this case, the instability manifests itself as an oscillation between two images, the difference of which has the maximum possible RMSE.

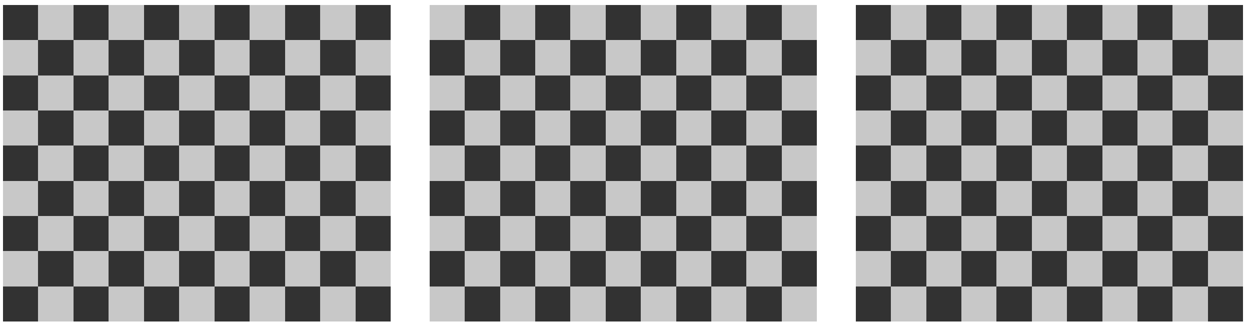

Figure 4.35: Oscillatory root images: (a) original image; (b) after one iteration; (c) after two iterations.

Ideally, the root image should resemble the original but without the noise. Many existing techniques produce a root image in which every pixel has the same value. For example, any fixed linear filter with more than one non-zero weight will, if iterated, eventually produce a homogeneous image. Even anisotropic diffusion [PM90], which is meant to be iterated, behaves in this manner if the default stopping function is used. The final common intensity will be roughly equal to the image mean. The exact value will vary with noise in the image, and whether pixel values are rounded back to integers after each iteration.

One would expect the local segmentation denoising algorithm to be stable and, in general, not to converge to homogeneity. This is because the explicit segmentation uses a hard cut-off to keep different groups of pixels separate. Pixels from different groups are unable to influence each other. Of course, this only works if the two groups of pixels differ enough to be detected by the local segmentation model selection criterion. This could be problematic if the noise level was poorly estimated, or if the intensity difference between two neighbouring segments was below the noise level.

Visual inspection

Figure 4.36 shows the result of iterating the local segmentation denoising algorithm of Section 4.7.1 100 times on the noiseless `square`. The simple model selection criterion with $C = 3$ is used, and σ was set to zero. The same value of σ was used for each iteration. The result of iterating a 3×3 median filter are also included for comparison purposes. It is highly desirable for a denoising algorithm behave as an identity filter when the input image is noiseless, and the local segmentation filter does that. The median filter truncates the corners, but otherwise keeps the object boundaries sharp.

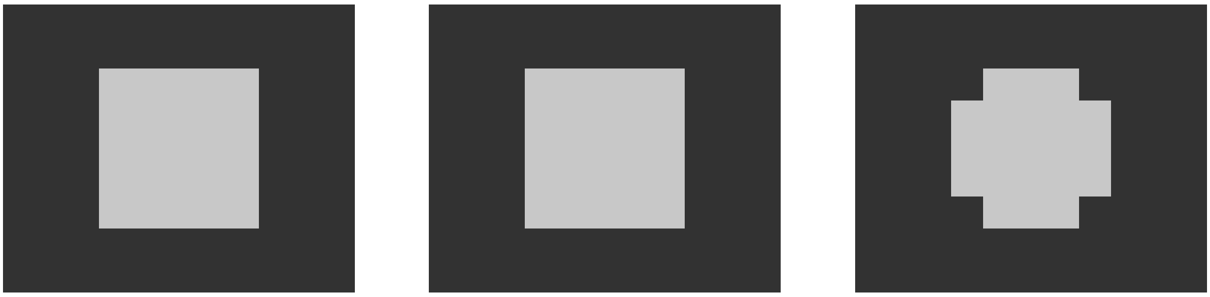

Figure 4.36: Iterating 100 times: (a) noiseless `square`; (b) 3×3 local segmentation assuming $\sigma = 0$; (c) 3×3 median filtered.

Figure 4.37 performs the same experiment, except that this time, noise $\sim \mathcal{N}(0, 30^2)$ has been added to `square`. Each iteration, the local segmentation filter assumed that $\sigma = 30$. The local segmentation filter proves to be both useful and stable. Its output only has two pixel values: 47 for the background and 192 for the foreground. The reason these are not equal to 50 and 200 as in the noiseless version could be due to three reasons: the noise not being distributed symmetrically within each segment, clamping of pixels to $[0, 255]$, and margin pixels being slightly over-represented due to their use in computing the extrapolated values when the filter window goes outside the legal image coordinates, as discussed in Section 4.2.8. Compared to Figure 4.28d, the result of only one iteration, the output is more visually pleasing. The median filter has also has a root image, albeit a disfigured one. The noise has caused the median to behave unusually.

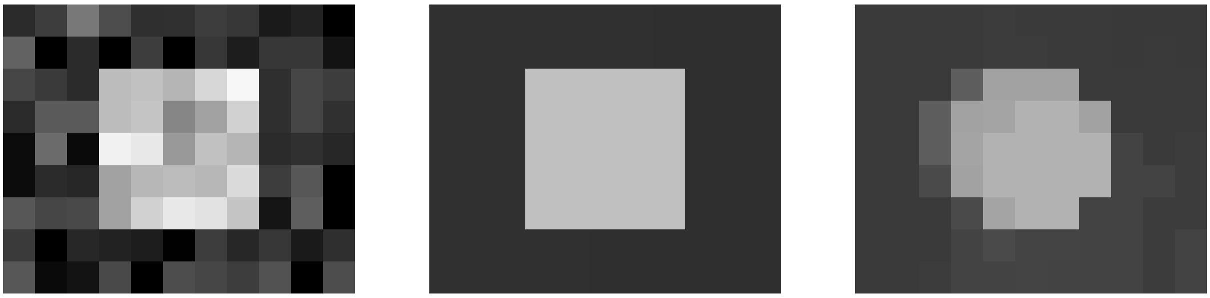

Figure 4.37: Iterating 100 times: (a) noisy $\sigma = 30$ `square`; (b) 3×3 local segmentation assuming $\sigma = 30$; (c) 3×3 median filtered.

As shown in Figure 4.38, the local segmentation filter fails when the noise is increased to $\sigma = 40$. Its denoised output only contains pixels of intensity 101, although the mean of the noiseless image is 88. This homogeneity is due to each of the original segments containing pixel values more suited to the other segment. Unfortunately, the clustering algorithm can not handle this situation. The result is "intensity leakage" between the segments. Given enough iterations, this leakage will spread throughout the image. The median filter has reached a better root image in this situation.

Figure 4.38: Iterating 100 times: (a) noisy $\sigma = 40$ `square`; (b) 3×3 local segmentation assuming $\sigma = 40$; (c) 3×3 median filtered.

Quantitative results

For `montage`, Figure 4.39 plots the RMSE between the original and denoised outputs as the number of iterations is increased. The local segmentation filter used $C = 3$ and was supplied with the true added noise level. In general, for a given level of added noise, the RMSE worsens as more iterations are performed. This is a little surprising given the qualitative results for `square` in Section 4.8. However, the assumption that the local region consists of two piece-wise constant segments is clearly false for many positions within `montage`. Any filtering errors after the first iteration would only be compounded by further iterations. Most natural images would have properties in common with `montage`.

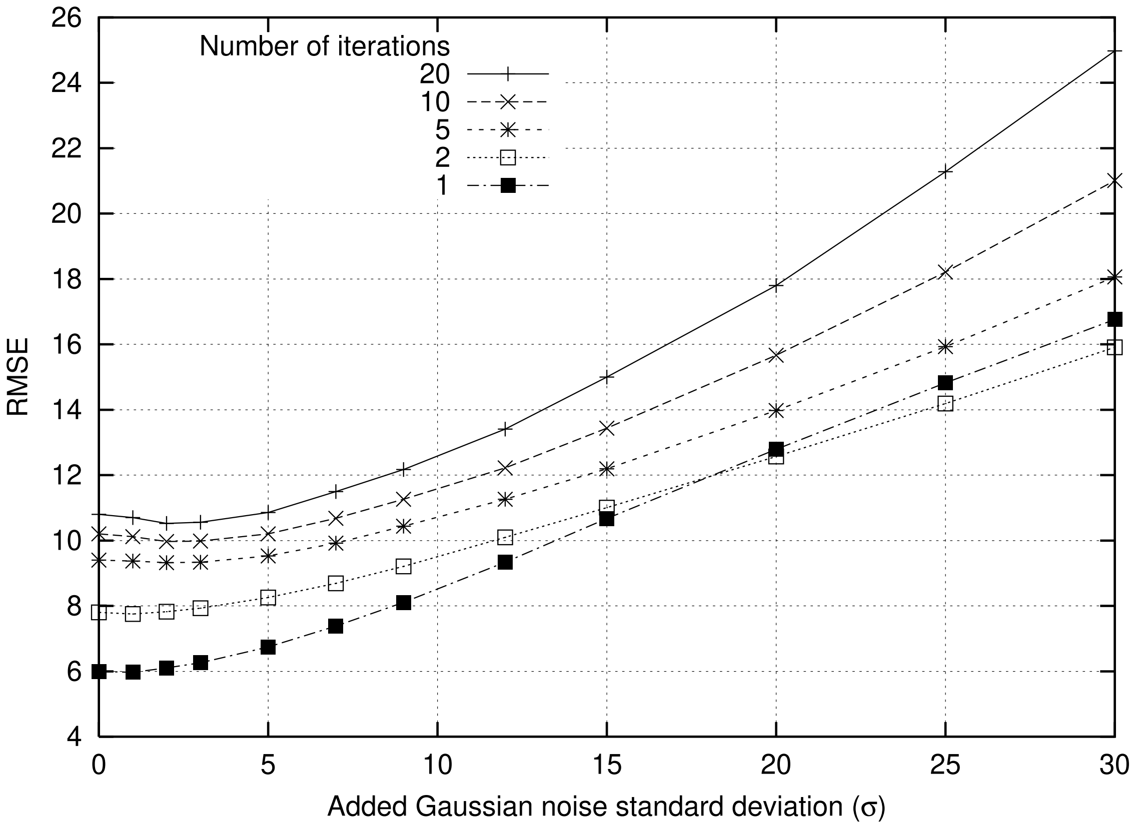

Figure 4.39: Effect of iteration on RMSE for denoising `montage`. The same value of σ is used for each iteration.

The exception to the overall trend is that 2 iterations becomes slightly more beneficial than 1 when $\sigma \geq 18$. This may just be the point at which the gain from further averaging outweighs the loss from degrading image structure further. Figure 4.40 shows the effect of iterations on a small portion of the noisy $\sigma = 20$ `montage` image, with the corresponding difference images underneath. Both visual inspection and RMSE agree that 2 iterations has produced a better output. The intensities of the text and background are more uniform and pleasing to the eye. For `montage`, $k = 1$ was chosen 80% of the time during the first iteration. This is approaching a pure averaging filter, which, in terms of RMSE, becomes a good choice when the noise level is high enough.

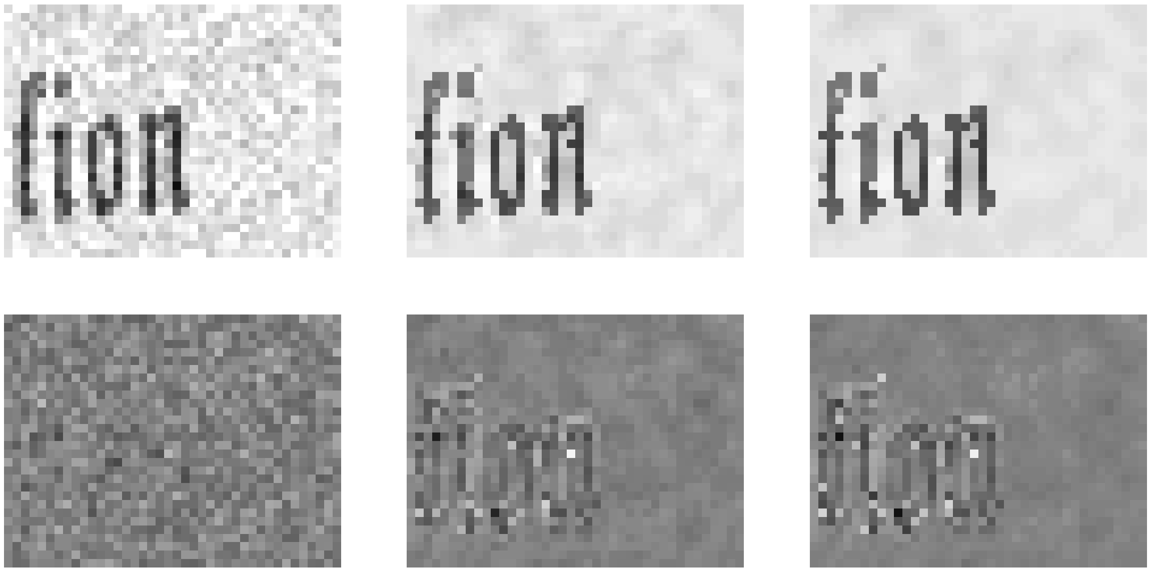

Figure 4.40: Bottom right hand corner of `montage`: (a) noisy $\sigma = 20$; (b) after 1 iteration; (c) after 2 iterations; (d)—(f) corresponding difference images relative to the noiseless original.

Conclusions

It is difficult to prove if a given denoising algorithm will be usefully stable in the general case. The previous experiments showed that for some simple cases, the local segmentation approach to denoising may be iterated successfully. When the noise is very high compared to the contrast between objects, pixel leakage unfortunately occurs, and multiple iterations could produce a homogeneous image. In Chapter 5, this problem will be tackled by using a segmentation algorithm which takes spatial information into consideration.

4.9 Rejecting poor local models

The local image model used thus far has assumed that pixels from the window are well modeled using either one or two segments of constant intensity. For non-synthetic images this assumption could be false for some windows. For example, the local region may consist of more than two segments, or it might not be piece-wise constant in nature. Figure 4.41a gives an example of a 3×3 block of pixels consisting of 3 segments with different constant intensities. If locally segmented under the assumption that $k = 2$, the resulting local approximation would have the pixel values given in Figure 4.41b.

$$\text{(a)} \begin{array}{|c|c|c|} \hline 20 & 40 & 60 \\ \hline 20 & 40 & 60 \\ \hline 40 & 40 & 40 \\ \hline \end{array} \quad \longrightarrow \quad \text{assuming } k = 2 \quad \longrightarrow \quad \text{(b)} \begin{array}{|c|c|c|} \hline 34 & 34 & 60 \\ \hline 34 & 34 & 60 \\ \hline 34 & 34 & 34 \\ \hline \end{array}$$

Figure 4.41: The effect of a two class model on a three class region: (a) original pixels; (b) segmented assuming $k = 2$.

The use of an incorrect model has forcibly merged two of the original segments. Its resulting mean falls between the those of original segments. Thus 7 of the 9 pixels, including the centre pixel, have a filtered value which is unreasonable given that the noise level is effectively zero. If one believes that the noise is additive in nature and distributed $\sim \mathcal{N}(0, \sigma^2)$, then confidence interval analysis states that 99.7% of pixels are expected to fall within 3σ of their original value. If a single filtered pixel value strays further than this, there is evidence to suggest that the local segmentation model is inappropriate.

The ability to diagnose whether the best fitting model actually fits the data well is very useful. All medical practitioners must take the Hippocratic Oath [HipBC], which has as one of its cardinal principles the following: *First of all, do no harm.* This principle may be applied to local segmentation when denoising images. If it appears that the best fitting model has produced an unreasonable local approximation to the original image, don't use it. Instead, use the original pixels as they were. It is probably better to leave them unmodified than to give them new, possibly noisier, values. Equation 4.29 describes the "do no harm" (DNH) principle. If the local approximation suggests changing *any* pixel's value more than $C\sigma$, then ignore it, and pass the pixels through unmodified.

$$\hat{\mathbf{p}} = \begin{cases} \mathbf{p}' & \text{if} \quad \exists i \; |\hat{p}_i - p'_i| > C\sigma \\ \hat{\mathbf{p}} & \text{otherwise} \end{cases} \tag{4.29}$$

For `montage`, Figure 4.42 shows the RMSE denoising performance of the 3×3 $C = 3$ local segmentation filter with and without DNH enabled. The algorithm was provided with the value of σ, as per all experiments so far. The DNH option provides an impressive improvement in RMSE for values of σ up to 20, after which it does only slightly worse. When σ is large, DNH is more likely to be invoked due to the dominating noise, rather than an inability to model the underlying image structure. It must be remembered that for the results presented here, the true noise variance was supplied to the local segmentation algorithm. In a real situation σ would have to be estimated from the noisy image. This issue will be discussed in Section 4.13.

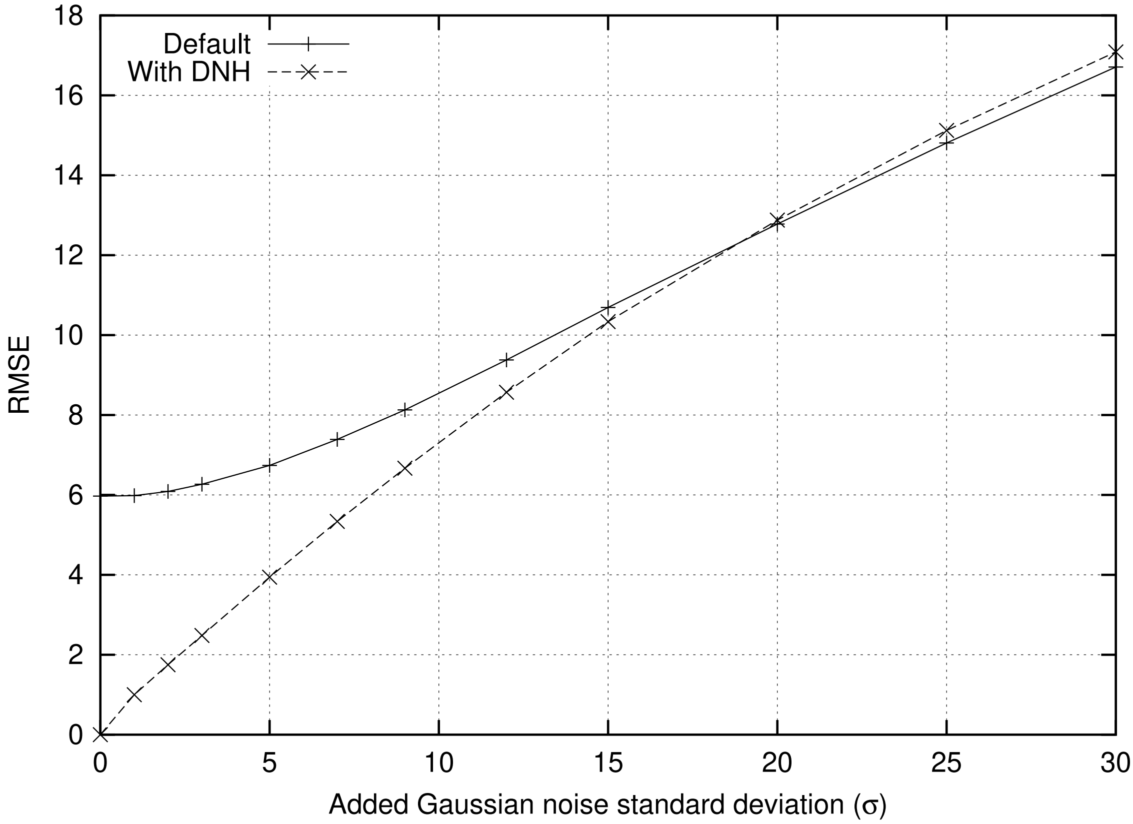

Figure 4.42: Effect of “do no harm” (DNH) on RMSE when denoising `montage`.

Figure 4.43 shows the effect that DNH has on the worst case absolute error (WCAE). As expected the WCAE performance is much better. Enabling DNH has the effect of limiting the worst case behaviour of the filter for each pixel. This is most useful in low noise environments, because it can help to avoid smoothing out fine image structure.

Figure 4.44a shows the top left quadrant of `montage` with added noise $\sim \mathcal{N}(0, 5^2)$. Figures 4.44b–c uses white to denote where $k = 1$ and $k = 2$ are chosen when DNH is not used. Figure 4.45 also uses white to show where $k = 1$, $k = 2$ and DNH are used when DNH is enabled. DNH seems to be used mostly on the broad ramp edges and busy hat feather regions, where a two segment model is probably insufficient for modeling the image structure. For this example, 73% of DNH invocations are due to rejecting a $k = 2$ model. It may seem unusual for a $k = 1$ model to be deemed better than a $k = 2$ one, but then be rejected because

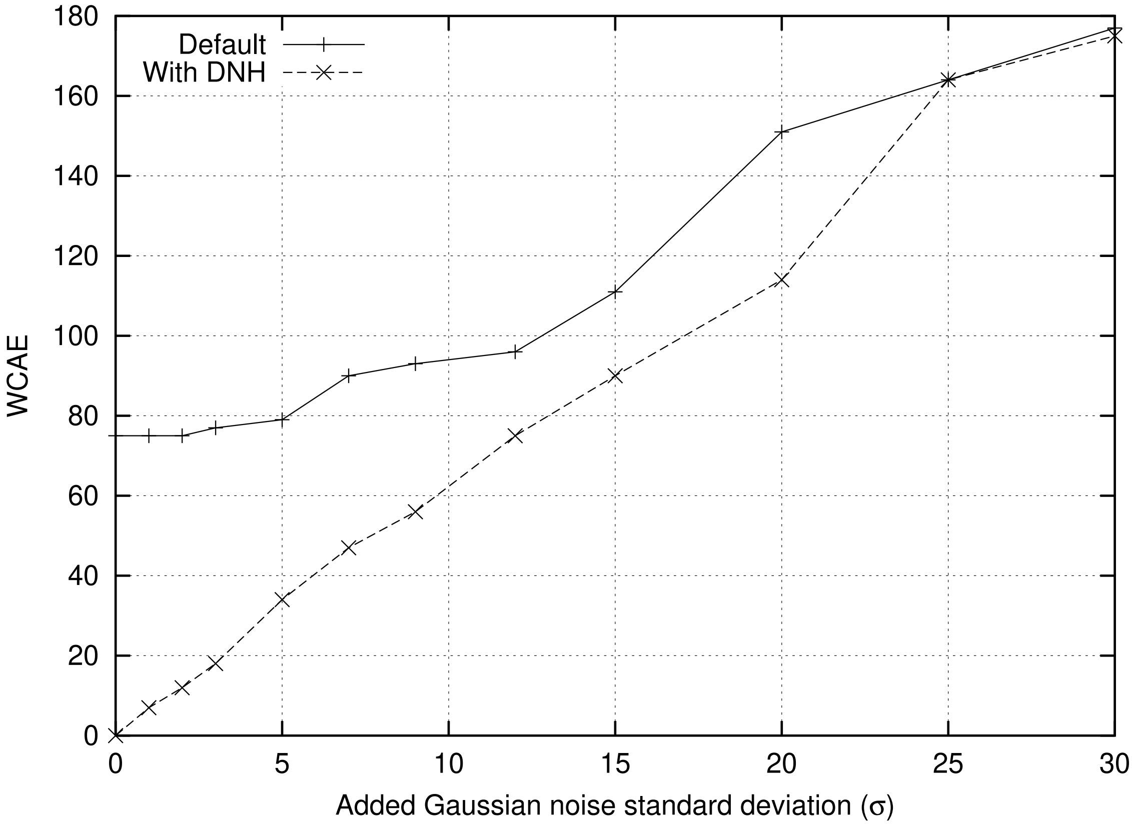

Figure 4.43: Effect of "do no harm" (DNH) on WCAE when denoising `montage`.

of DNH. This occurs because binary thresholding sometimes produces poor segmentations, and before DNH, $k = 1$ is a better overall choice.

Figure 4.44: DNH disabled: (a) original image; (b) $k = 1,$ 64%; (c) $k = 2,$ 36%.

The *do no harm* philosophy is not limited to local segmentation applications. It could be applied to any denoising algorithm to limit its worst case behaviour. All that is required is some measure of the natural level of variation within segments around the pixel being processed.

Figure 4.45: DNH enabled: (a) $k = 1$, 60%; (b) $k = 2$, 25%; (c) DNH, 15%.

For example, the simple box filter of Section 3.4.3 could invoke the DNH rule when the local average differs too much from any local pixel values. This filter would have two implicit local models: $k = 1$ and DNH. The result would be good smoothing in homogeneous regions, and either blurring or no smoothing elsewhere.

Equation 4.29 is just one possible DNH rule. Instead of rejecting the local approximation if just one pixel value would change too much, it could be relaxed to consider only the centre pixel. Here DNH would be invoked less often because it is not testing the fitness of the whole local segmentation model. For some windows the relaxed rule could improve denoising, but for others, such as the one in Figure 4.41, it would do more damage than good.

4.10 Different local windows

The examples and experiments so far have used a 3×3 window containing 9 pixels. The local segmentation approach is not restricted to this configuration. There are many other commonly used windows, five of which are shown in Figure 4.46. The 5 pixel one is used by anisotropic diffusion, while SUSAN utilizes the larger 37 pixel window.

As the number of pixels in the window increases, heterogeneous regions are less likely to be well modeled by only two segments. Doing so could introduce large errors into the output, regularly forcing DNH to reject filtering of the window. On the other hand, there would be greater smoothing in homogeneous regions, because more pixels are being averaged. Under the RMSE measure, a few large errors may have more negative influence than the

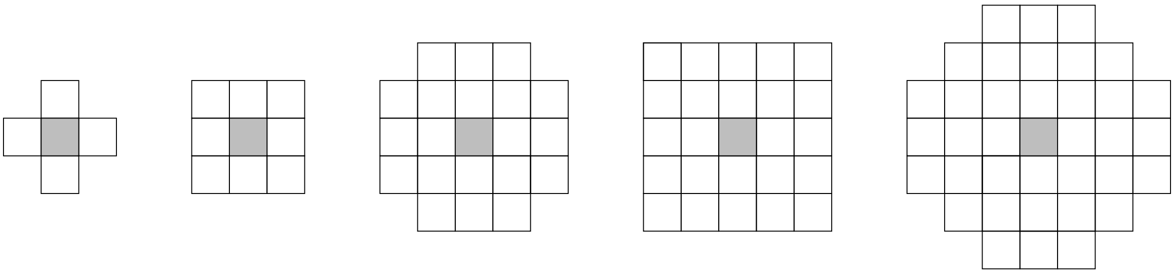

Figure 4.46: Common windows: (a) 5; (b) 9; (c) 21; (d) 25; (e) 37 pixels.

sum positive influence of many smaller errors due to better smoothing. The balance depends on the features of the image being denoised.

Figure 4.47 gives results for denoising `montage`, first with DNH disabled. The general trend suggests that smaller windows achieve better denoising with respect to RMSE. The 5 pixel window performs best until $\sigma = 15$, at which point the next smallest (9 pixel) window takes over. This is an interesting result. For the 3×3 window, the 4 corner pixels are roughly 41% further away from the centre pixel than the 4 side pixels. By argument of spatial coherence, the corner pixels are less likely to be related to the others. Segmenting fewer, more related pixels means that $k = 1$ should be chosen more often. A small window is less likely to cross segment boundaries than a larger one. The advantage of choosing $k = 2$ less frequently is that homogeneous regions are more likely to be well modeled. For very noisy images, $k = 1$ is chosen more often anyway, so a smaller window will give less smoothing. This is borne out by the 5 pixel window's RMSE increase as σ gets higher.

When DNH is enabled, the results change quite dramatically. Figure 4.48 shows how DNH limits the worst case behaviour of all the windows tested. The RMSE performances are essentially equivalent until $\sigma = 15$, whereafter the 9 pixel window takes over, just as in Figure 4.47. It appears that the 3×3 window strikes a good balance between compact locality, so that a two-segment model is sufficient, and having enough pixels for reasonable smoothing in homogeneous regions.

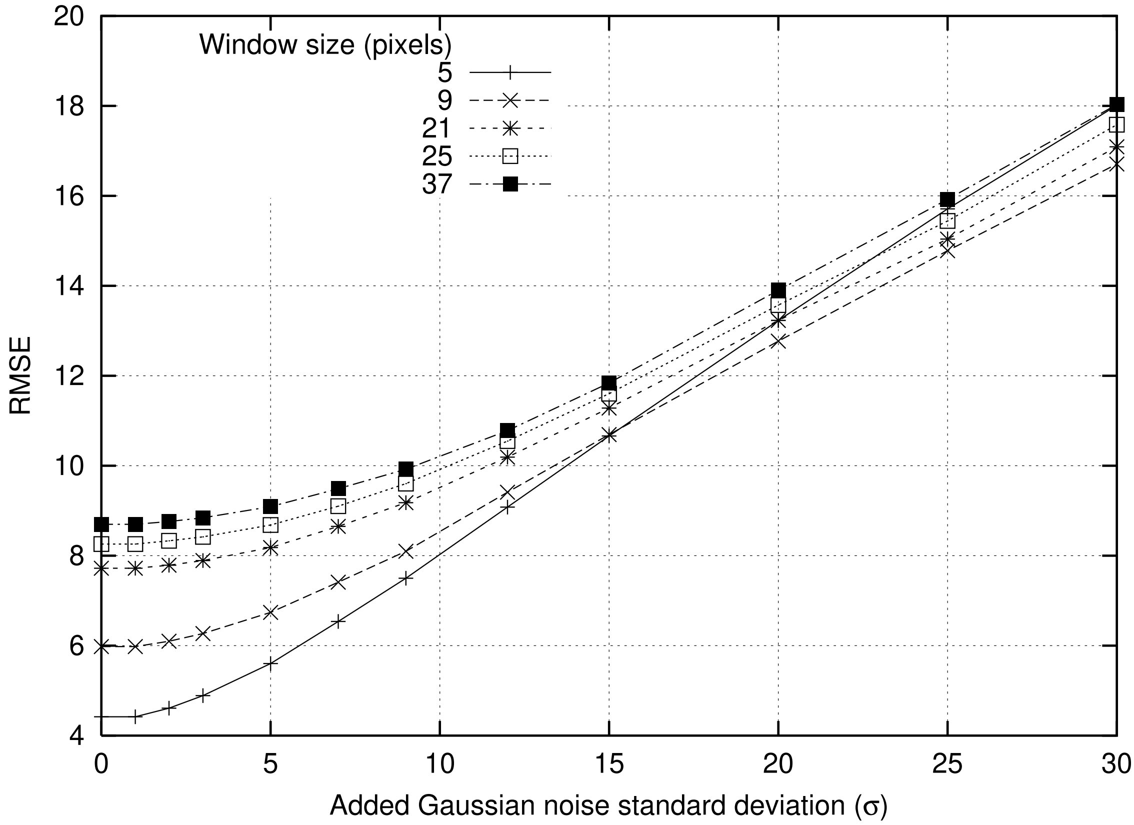

Figure 4.47: Effect of different windows for denoising `montage`, DNH disabled.

4.11 Multiple overlapping local approximations

The local segmentation principle states that the first step in processing a pixel should be to segment the local neighbourhood of that pixel. This is a democratic process, treating every pixel equally to infer the original values of *every* pixel in the neighbourhood, not just the one being processed. This is unlike many other local denoising techniques, such as SUSAN and GIWS, which use the centre pixel as a *reference pixel* for intensity comparisons.

Local segmentation provides a *local approximation* to the underlying image around each pixel. Because each pixel is processed independently, the local approximations *overlap* to some extent. If a window containing M pixels is used, it is easy to see that each pixel from the image participates in M separate local segmentations, at each of the M possible positions in the window. Figure 4.49 illustrates this graphically for of a $3{\times}3$ window, where $M = 9$.

Thus, for each noisy pixel, there are M different estimates of its true value. The estimates are not fully independent, as they share some pixel values in their derivation, so they can

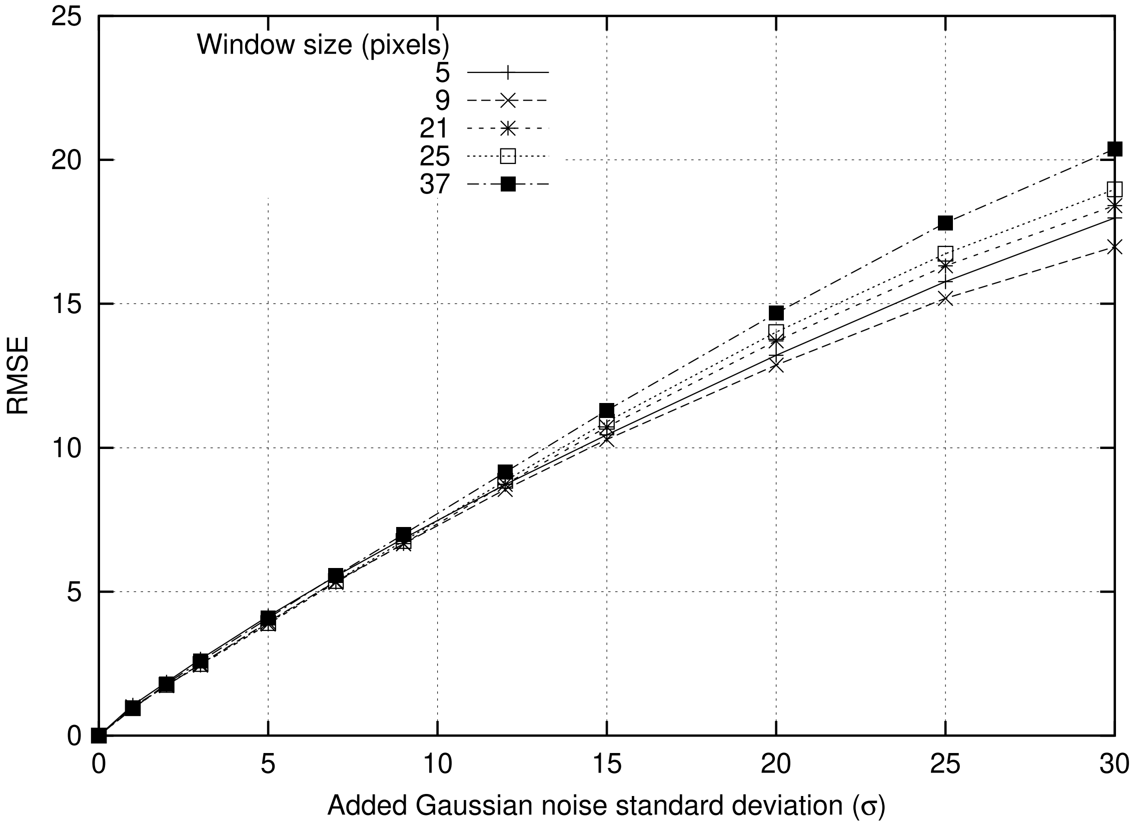

Figure 4.48: Effect of different windows for denoising `montage`, DNH enabled.

be considered semi-independent. What would be desirable is to combine these multiple overlapping estimates into a single *compound estimate*, rather than just using the centre estimate as in Equation 4.17. This should have three main benefits: a larger effective window size, further robustness, and improved denoising performance.

For square windows containing M pixels, the new effective window would contain exactly $(2\sqrt{M}-1)^2$ pixels, which is asymptotically a factor of four increase. For a 3×3 window the factor is $25/9 = 2.58$. The calculation of the compound estimate is likely to require little computational effort compared to performing local segmentation with a larger window. In homogeneous areas, up to four times as many pixels will contribute to the calculation of the mean, reducing its standard error by the same factor.

Let $\tilde{p}_1 \ldots \tilde{p}_M$ be the M overlapping denoised estimates for the *same pixel*. How should these be combined to produce a compound estimate $\hat{p}$? The simplest approach would be to take a convex linear combination of the estimates, shown in Equation 4.30, where w_j are the weights. The denominator term ensures the weights sum to unity.

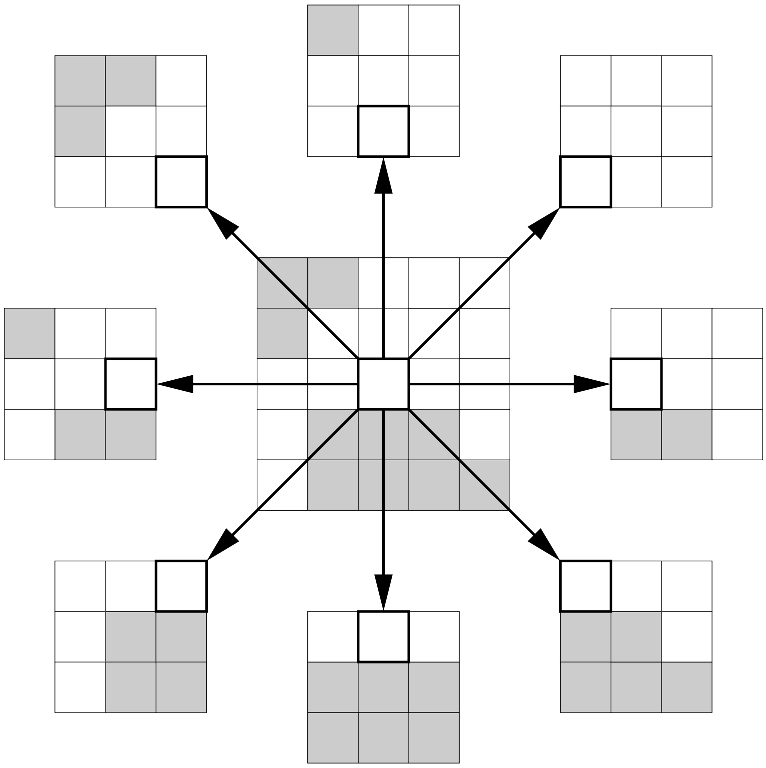

Figure 4.49: Each pixel participates in 9 local 3×3 windows.

$$\hat{p} = \frac{\sum_{j=1}^{M} w_j \, \tilde{p}_j}{\sum_{j=1}^{M} w_j} \tag{4.30}$$

4.11.1 Choosing the weights

There are various factors which could be used to influence the weight given to a particular estimate. Four possibilities are its position in the window from which it was computed, its standard error, the number of segments diagnosed in its original window, and the measured variance of its original segment.

Window position

Each denoised estimate $\tilde{p}_j$ is derived from a different local region. As shown in Figure 4.49, the same shaped window is used for each region, but the position of the pixel within the window differs per estimate. For a 3×3 window, the estimate appears in 9 different positions — at the four corners, the four sides, and once at the centre. At first consideration one may guess that the "centre" estimate should be given more weight than those from the sides and corners. However, the local segmentation process treats all pixels democratically, and the thresholding technique takes no spatial information into consideration. There appears to be little justification for treating them unequally.

This assertion will be tested by comparing a range of different weights. The weights should be chosen in a manner such that the four corners are treated equally, and likewise for the four edges. A simple model meeting these symmetry requirements is shown in Figure 4.50. Each weight is calculated by raising a constant, ρ, to the power of its distance from the centre of the window. In this case the Manhattan distance has been used. When $\rho = 0$, the model reverts to not producing a compound estimate at all. As ρ approaches 1, the estimates are treated more and more equally. Do not confuse this ρ with that used in image processing to measure the correlation between neighbouring pixels. Here the values being combined are all estimates of the *same pixel*.

$$\text{(a)} \begin{array}{|c|c|c|}\hline \rho^2 & \rho^1 & \rho^2 \\ \hline \rho^1 & \rho^0 & \rho^1 \\ \hline \rho^2 & \rho^1 & \rho^2 \\ \hline \end{array} \qquad \text{(b)} \begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline 0 & 1 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array} \qquad \text{(c)}\ \tfrac{1}{4} \begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 2 & 4 & 2 \\ \hline 1 & 2 & 1 \\ \hline \end{array} \qquad \text{(d)} \begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$

Figure 4.50: Weighting overlapping estimates: (a) any ρ; (b) $\rho = 0$; (c) $\rho = 0.5$; (d) $\rho = 1$.

Figure 4.51 compares four values of ρ in terms of RMSE for denoising `montage`. To observe the behaviour of overlapping averaging in the simplest possible environment, DNH is disabled. The WCAE results under for the same conditions is plotted in Figure 4.52. It is seen that any non-zero value of ρ improves both the RMSE and WCAE performance, compared to not combining overlapping estimates at all. As argued earlier, the best results are achieved when $\rho = 1$, which gives equal weight the overlapping estimates.

Figure 4.53 gives visual results for the `square` image after adding noise $\sim \mathcal{N}(0, 20^2)$ to it. The improved smoothing performance when $\rho = 1$ is clearly visible. In both cases the edges are reconstructed without any blurring.

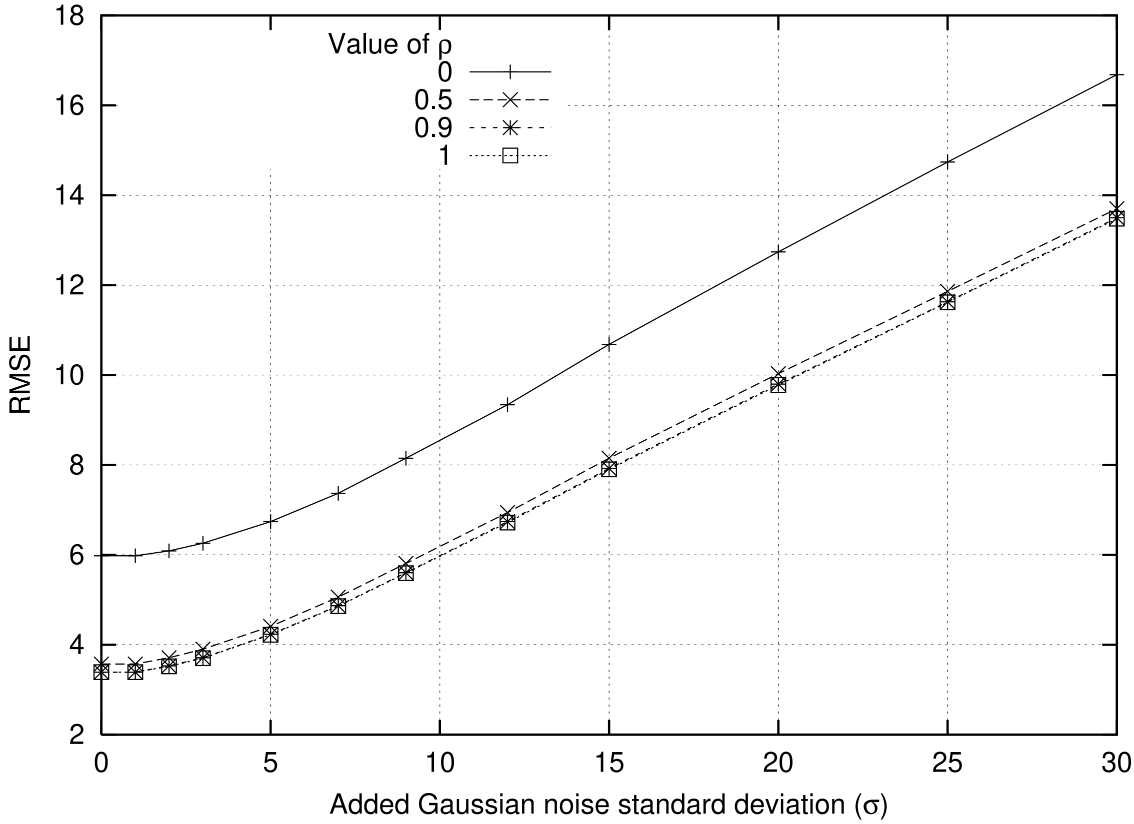

Figure 4.51: Effect of ρ on RMSE when denoising `montage`.

The standard error

A denoised estimate, $\tilde{p}_j$, takes its value as the mean of the cluster to which it belongs. It was shown in Section 4.5 that the variance, or *standard error*, of an estimated cluster mean is σ^2/m_j, where m_j is the number of pixels in the cluster. This variance may be used to measure the confidence in the estimate, and could contribute to its weight. A high standard error corresponds to a low confidence. As M and σ are assumed *a priori* known and constant, the confidence component could be expressed by the relationship $w_j \propto m_j$.

Figure 4.54 compares the RMSE of the confidence weighting, $w_j = m_j$, to equal weighting, $w_j = 1$. The weights were normalized for each pixel using Equation 4.30. Once again, DNH is disabled to limit the number of factors influencing the result. Somewhat surprisingly, confidence weighting performs worse than equal weighting up to $\sigma = 20$, at which point they converge. In homogeneous areas of the image, both approaches should behave equivalently, as $m_j = M$ for every estimate. The variation must occur in heterogeneous areas of the image.

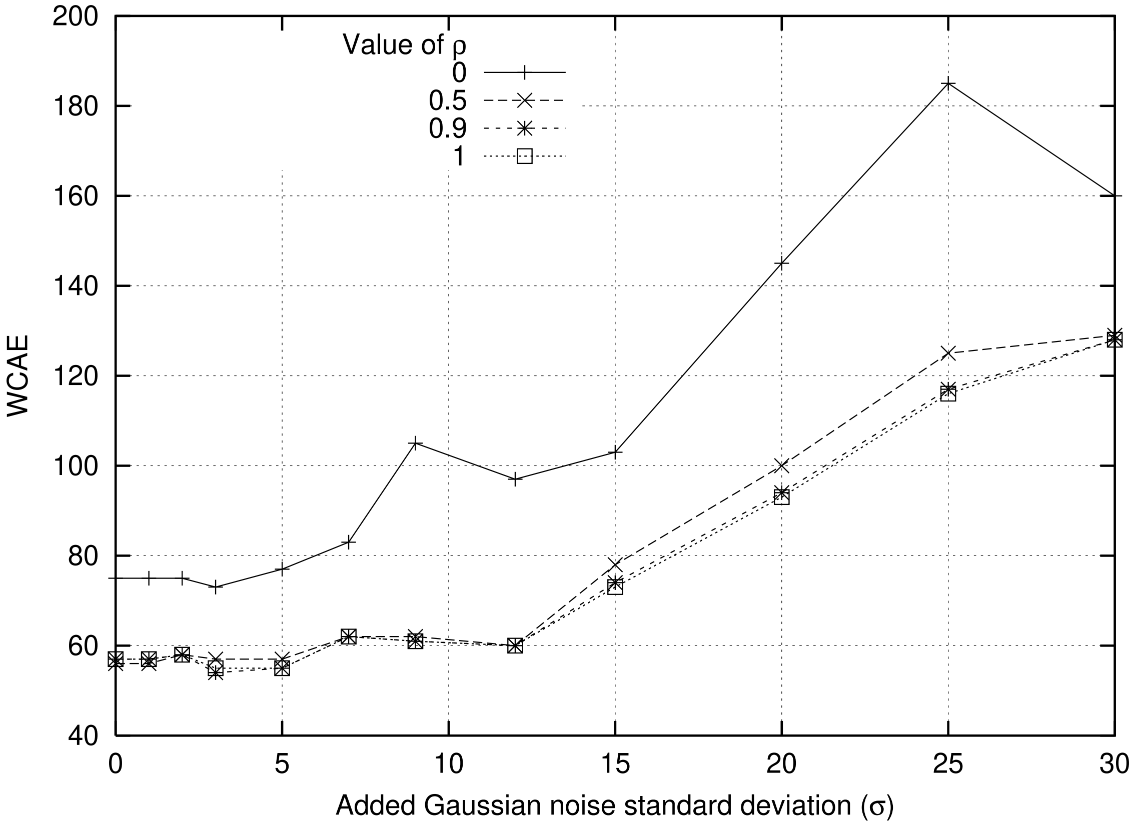

Figure 4.52: Effect of ρ on WCAE when denoising `montage`.

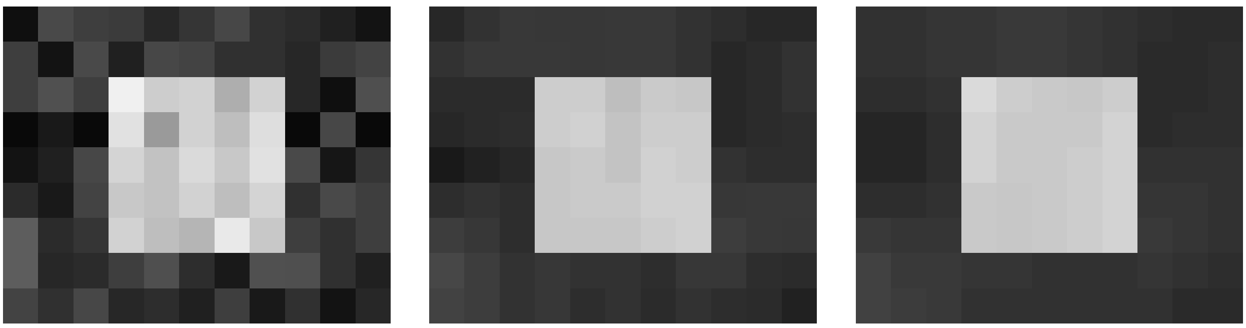

Figure 4.53: Denoising `square`: (a) noisy original, $\sigma = 20$; (b) output when $\rho = 0$; (c) output when $\rho = 1$.

In the vicinity of an edge, non-centre pixels from neighbouring homogeneous windows will receive more weight than centre pixels from heterogeneous windows closer to the edge. These estimates may be unreliable, so equal weighting (averaging) could be the best way to minimize the average error, and hence RMSE. It was found that enabling DNH option did not affect the pattern of performance observed in Figure 4.54.

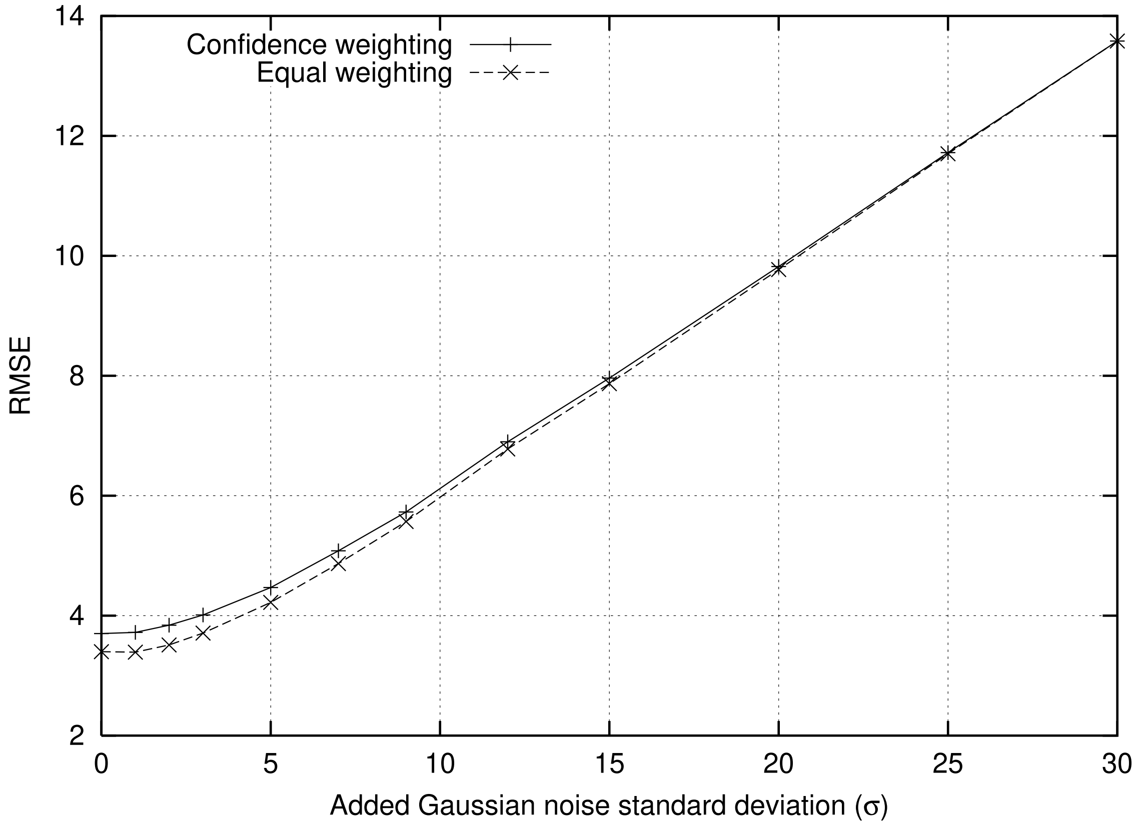

Figure 4.54: Effect of confidence weighting on RMSE when denoising `montage`.

Activity of the region

The *activity* of a window is considered low if it is homogeneous, and high if heterogeneous. The number of segments, k, determined to be present in the window may be used as a measure of activity. A high activity region is diagnosed when the window truly does consist of many segments, or when the assumed image model is unsuitable. One may wish to give preference to estimates from less active regions, to minimize potential modeling errors.

If k_j is the number of segments diagnosed for the window $\tilde{p}_j$ came from, the activity weight could be expressed as $w_j \propto 1/k_j$. The variables m_j and k_j are strongly correlated, because $m_j = M$ implies $k = 1$, and $m_j \neq M$ implies $k = 2$. To a large extent, the activity weighting is already covered by the confidence weighting described previously, so it is unlikely to improve results. Numerical evidence to support this is given in Figure 4.55, where the activity weighting provides no advantage over equal weighting.

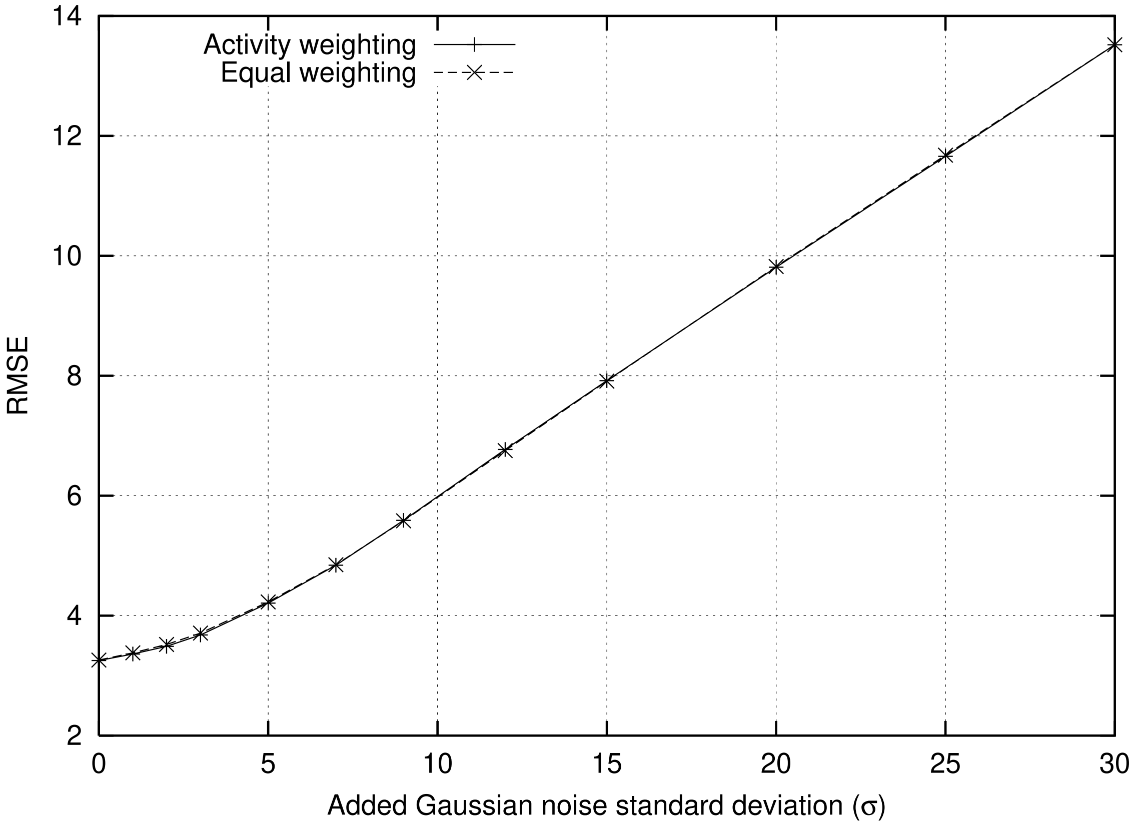

Figure 4.55: Effect of activity weighting on RMSE when denoising `montage`.

Measured class variance

Although the variance of each cluster is assumed to be σ^2, it is possible to also actually calculate the sample variance, s^2. If the image model is accurate, s and σ are expected, on average, to be very similar. A high value of s could be used to gauge reliability of a segment mean. But due to the small number of pixels in most clusters, a reliable estimate of the sample segment variance is difficult to obtain. In the case of $m_j = 1$, the sample variance is not even defined. For these reasons it will not be investigated further.

Summary

Of the four influences considered for weighting overlapping estimates, equal weighting, or simple averaging, was found to perform best in terms of RMSE. If overlapping is enabled, the compound estimate, $\hat{p}$, will be computed using Equation 4.31, where $\tilde{p}_1 \cdots \tilde{p}_M$ are the estimates being combined. Using a linear combination for calculating the compound estimate is only one possibility. Non-linear functions, like the median, could also be used, but further investigation of these possibilities is beyond the scope of this chapter.

$$\hat{p} = \frac{1}{M} \sum_{j=1}^{M} \tilde{p}_j \tag{4.31}$$

When DNH is triggered, the filtered pixel value for the window are set equal to the original pixel values — it is assumed that the best estimate of a pixel's true value is itself. In this thesis, an estimate, $\tilde{p}_i$, from a DNH "model" is included in the formation of the compound estimate. An alternative would be to consider DNH estimates as unreliable, and assign them zero weight. If the compound estimate could not be formed due to *all* estimates being unreliable, the original noisy pixel value could used instead.

4.12 Larger numbers of segments

It has been assumed so far that the local window consists of pixels from, at most, two segments. For most real images there will be occasions where this is not the case. This could be due to the presence of three or more segments, or two segments not being aligned to the pixel grid, as described in Section 4.2.3. In any case, results could improve if a larger number of segments were considered. If segments are allowed to be as small as one pixel, it is possible for M separate segments to occupy an M pixel window.

The obvious progression is to extend the thresholding method from Section 4.7 to work for more than two clusters. Local segmentation can be considered to have two main components. The first is the segmentation algorithm, which provides different candidate segmentations of the local region. In our case, each segmentation differs only in the number of clusters, k, but it could be possible to incorporate alternative segmentations. The second component is the model selection technique, which decides which of the candidate segmentations is the most appropriate. In our case, this would choose the appropriate value of k for the window.

4.12.1 Extending the clustering algorithm

In Section 4.6, Lloyd's iterative threshold selection method was used for binary thresholding. What is required is an equivalently good technique which works for up to M clusters. A generalized form of Lloyd's method is the k-means or H-means algorithm, referred to here as k-means. This algorithm was discussed in Section 3.2.1, and is repeated below in Listing 4.2. In Step 2, "closest" is usually taken to be the Euclidean norm between the two values. For greyscale pixels this simplifies to be the absolute intensity difference, but for higher dimensional data such as colour pixels, a full calculation would be necessary.

1. Choose k initial means (to be discussed in Section 4.12.3).
2. Assign each datum to the cluster with mean closest to its own value.
3. Recompute the cluster means using the new assignments.
4. If any pixels have switched clusters, go to Step 2.

Listing 4.2: The k-means algorithm for local segmentation.

The objective function of k-means is to minimize the sum of squared deviations of pixels from their cluster means. Each iteration of the k-means algorithm is guaranteed to improve the objective function, and the algorithm will always terminate. Unfortunately, the algorithm will stop at the nearest local minimum, which is not necessarily the global optimum [GG91, BB95]. The minimum reached depends wholly on the choice of initial cluster means [BF98]. Three techniques for choosing the initial means will be investigated in Section 4.12.3.

4.12.2 Extending the model order selection technique

So far, the decision between $k = 1$ and $k = 2$ has been decided by examining the separation of the two cluster means. The k-means algorithm can be used to segment pixels into an arbitrary number of clusters. The "mean separation criterion" can easily be extended to work with more than two clusters — each cluster mean is required to be far enough away from every other cluster mean. The highest value of k for which all the clusters are mutually well separated is deemed the optimal clustering. Equation 4.32 describes this formally, where k is the optimal number of clusters and $\hat{\mu}_i$ is the estimated mean for the i^{th} cluster.

$$\underset{k}{\operatorname{argmax}} \quad \underset{\substack{i=1\ldots k \\ j=1\ldots k \\ i\neq j}}{\forall i,j} \quad |\hat{\mu}_i - \hat{\mu}_j| \geq g(\sigma, m_1, m_2) \tag{4.32}$$

The k-means algorithm can easily be configured to generate cluster means in ascending numerical order, so only *adjacent* cluster means need to be tested for separation. Equation 4.33 shows a simplified version of Equation 4.32, where $\hat{\mu}_{(i)}$ is the i^{th} sorted cluster mean.

$$\underset{k}{\operatorname{argmax}} \quad \underset{i=1\ldots k-1}{\forall i} \quad |\hat{\mu}_{(i)} - \hat{\mu}_{(i+1)}| \geq g(\sigma, m_1, m_2) \tag{4.33}$$

4.12.3 Choosing the initial means

Inappropriate starting conditions can cause the k-means algorithm to converge to a local minimum. This problem can mostly be avoided in the one dimensional case (greyscale pixel values) by judicious choice of the initial cluster means.

Measurement space

One approach is to spread the k initial means evenly throughout the measurement space. Equation 4.34 describes this, where p'_{min} and p'_{max} are the lowest and highest pixel values being clustered, and $\bar{\mu}_i$ is the initial mean for cluster i.

$$\bar{\mu}_i = p'_{min} + (i-1) \cdot \frac{p'_{max} - p'_{min}}{k-1} \qquad \text{where} \qquad i = 1 \ldots k \tag{4.34}$$

This method assumes that the pixel values are uniformly distributed in intensity. If the pixel value distribution is highly skewed, some clusters may never be used. This is known as the *empty cluster* problem [HM01].

Rank space

An alternative assignment of initial means is given in Equation 4.35, where $p'_{(i)}$ is the i^{th} *sorted* pixel value. Instead of spreading the initial means uniformly by intensity, they are chosen uniformly by *rank*, where a pixel's rank is its sorted position. For example, in Equation 4.34 the pixels p'_{min} and p'_{max} are the same as $p'_{(1)}$ and $p'_{(M)}$ respectively.

$$\bar{\mu}_i = p'_{\left(\lfloor \frac{(i-1)M}{k-1} \rfloor\right)} \qquad \text{where} \qquad i = 1 \ldots k \tag{4.35}$$

The rank technique improves the chance that the initial clusters will have at least one member each, as the initial means are equal to existing pixel values. If the pixel data lacks variety, adjacent clusters may still be assigned the same initial mean. This would result in one cluster arbitrarily being chosen over another, leaving one empty after the first iteration.

Unique rank space

A simple modification can be made to the rank space method to avoid the problem of clusters sharing an initial mean. The M sorted pixels are first scanned to remove duplicates, leaving M' pixels. The rank space approach can then be applied to the M' unique pixels. This is described in Equation 4.36.

$$\bar{\mu}_i = p'_{\left(\lfloor \frac{(i-1)M'}{k-1} \rfloor\right)} \qquad \text{where} \qquad i = 1 \ldots k \qquad \text{and} \qquad k \le M' \le M \tag{4.36}$$

If $M' < k$, it is impossible to cluster the data into k distinct groups. Section 4.12.3 will discuss potential bounds on k in more detail.

Comparison of methods for choosing initial means

Figure 4.56 plots the RMSE denoising performance of the k-means-based denoising algorithm for the three different k-means initialization techniques. Enhanced features such as DNH and overlapping averaging are disabled. At low noise levels the "unique rank space" approach does best. From $\sigma = 4$ onwards there is little difference between the three methods.

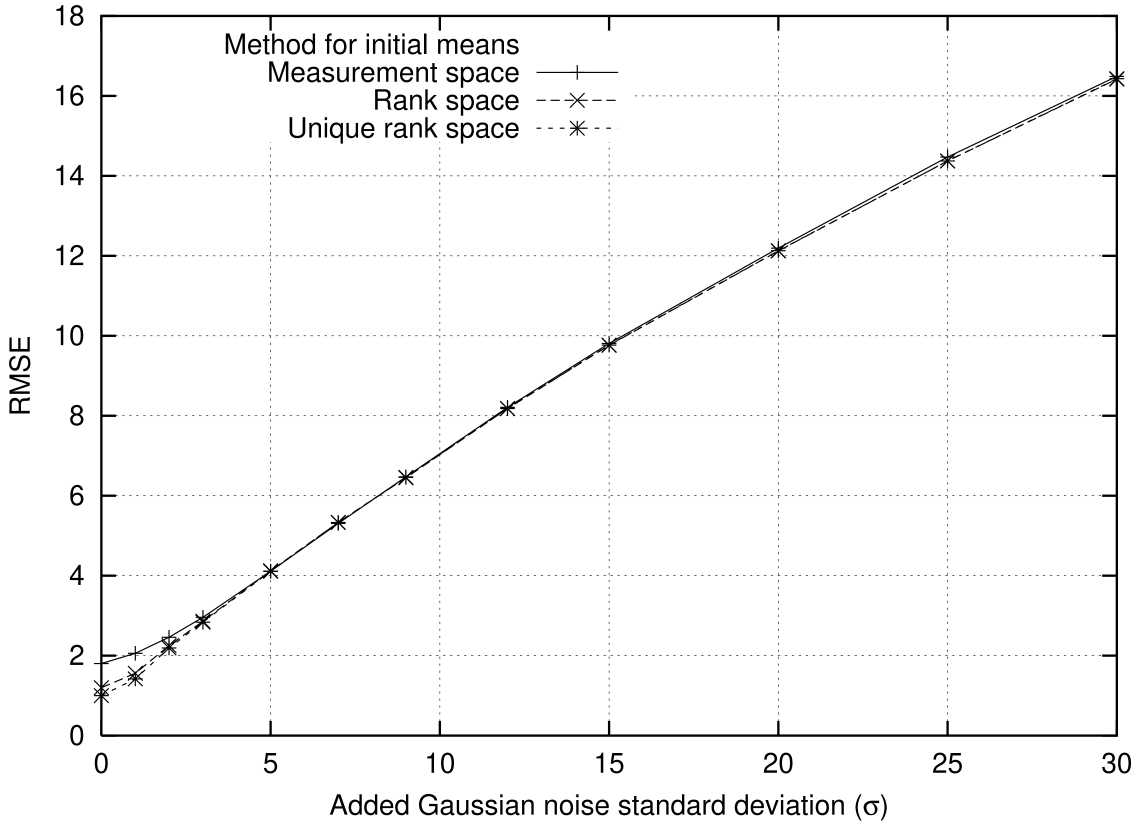

Figure 4.56: Comparison of three different methods for choosing the initial k means, in terms of RMSE denoising performance on `montage`.

Forcing an upper bound on k

Let K be the highest value of k to be tested. Higher values of k require more computation because there are more parameters to estimate and distributions to compare. Any heuristics or computable bounds on K for each pixel would reduce the amount of computation required by local segmentation.

Maximum upper bound If each of the M pixels has a distinct value distant enough from every other pixel's value, then the algorithm will infer M unique clusters. An absolute upper bound on K is therefore M.

Block variance If the window variance, $\mathrm{Var}[\,\mathbf{p}'\,]$, is less than the estimate of the noise variance, σ^2, then it is not possible for $k \geq 2$ to produce clusters far enough apart. In this situation $K = 1$.

Maximum possible fit The *range* of the pixel values, $(p'_{max} - p'_{min})$, also limits the maximum value of k achievable. Equation 4.37 calculates how many $C\sigma$-wide normal distributions could fit within the pixel range. If using the simple mean separation criterion, the same value of C should be used for each window. If using the t-test separation criterion, the maximum possible *effective* value of C must be chosen to ensure correctness in the extreme case.

$$K = \left\lceil \frac{p'_{max} - p'_{min}}{C\sigma} + 1 \right\rceil \tag{4.37}$$

Unique pixel values It is not possible to have more clusters than unique pixel values, because pixels with the same value are necessarily clustered together. If the pixels are already sorted, as required by the initial means method of Equation 4.36, the number of unique values may be obtained with a simple one-pass $\mathcal{O}(M)$ algorithm.

Limited by the user The user may wish to place an upper limit on K. They may have prior expectations regarding the number of segments in each region, or simply wish the implementation to run more quickly. This is particularly suited to large windows, because one would not reasonably expect to find 49 clusters in a 7×7 window.

4.12.4 Quantitative results for multiple classes

The local segmentation denoising algorithm could easily utilise the multi-class k-means segmentation algorithm in Listing 4.2, and the model selection criterion of Equation 4.33. The `montage` image contains many local areas having more than two segments, for example, the intersection of the four quadrants. One would expect the multi-class mode to better denoise these regions. The local segmentation denoising algorithm now has three main attributes:

1. Support for binary or "multi-class" local segment models.

2. An ability to average overlapping estimates or just use the central estimate.

3. A DNH option to minimize worst case behaviour filtering behaviour.

Figure 4.57 compares binary and multi-class thresholding in terms of RMSE for denoising `montage`. For the moment, both overlapping averaging and DNH are disabled. The RMSE is much better for the multi-class model, especially at lower noise levels.

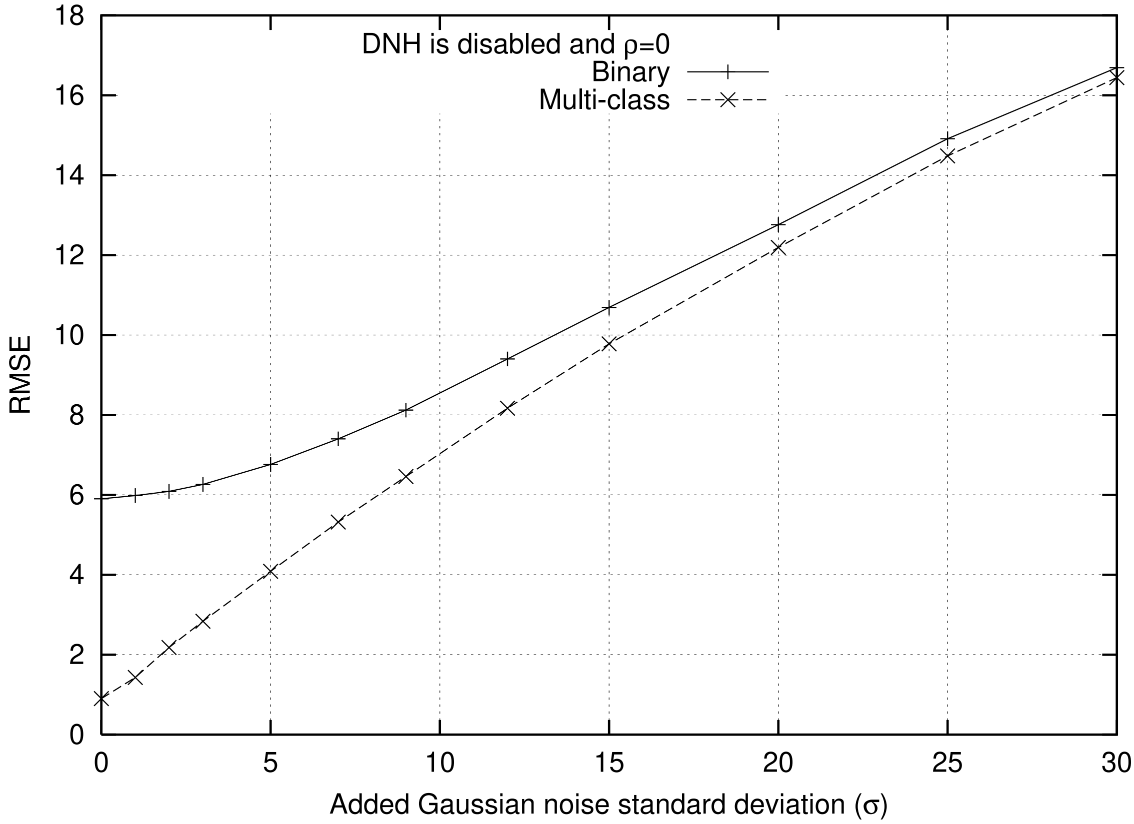

Figure 4.57: Comparison of binary and multi-class local segmentation for denoising `montage`. DNH is disabled and $\rho = 0$.

Figure 4.58 provides results for when overlapping averaging is enabled. Although the RMSE improves for both techniques, binary thresholding benefits the most. The gap between the two methods is only significant when $\sigma \leq 10$. This suggests that averaging estimates from 9 possibly incorrect binary models may produce a compound estimate which better reflects the underlying image than a single estimate alone.

In Section 4.9, "do no harm" was introduced as a way to limit the worst case performance of a denoising filter, and was found to dramatically improve RMSE performance at lower noise levels. Figure 4.59 compares the same binary and multi-class algorithms when DNH is enabled, but overlapping averaging is disabled. Surprisingly, binary thresholding does marginally better when $\sigma \leq 5$, while multi-class thresholding is better at high noise levels.

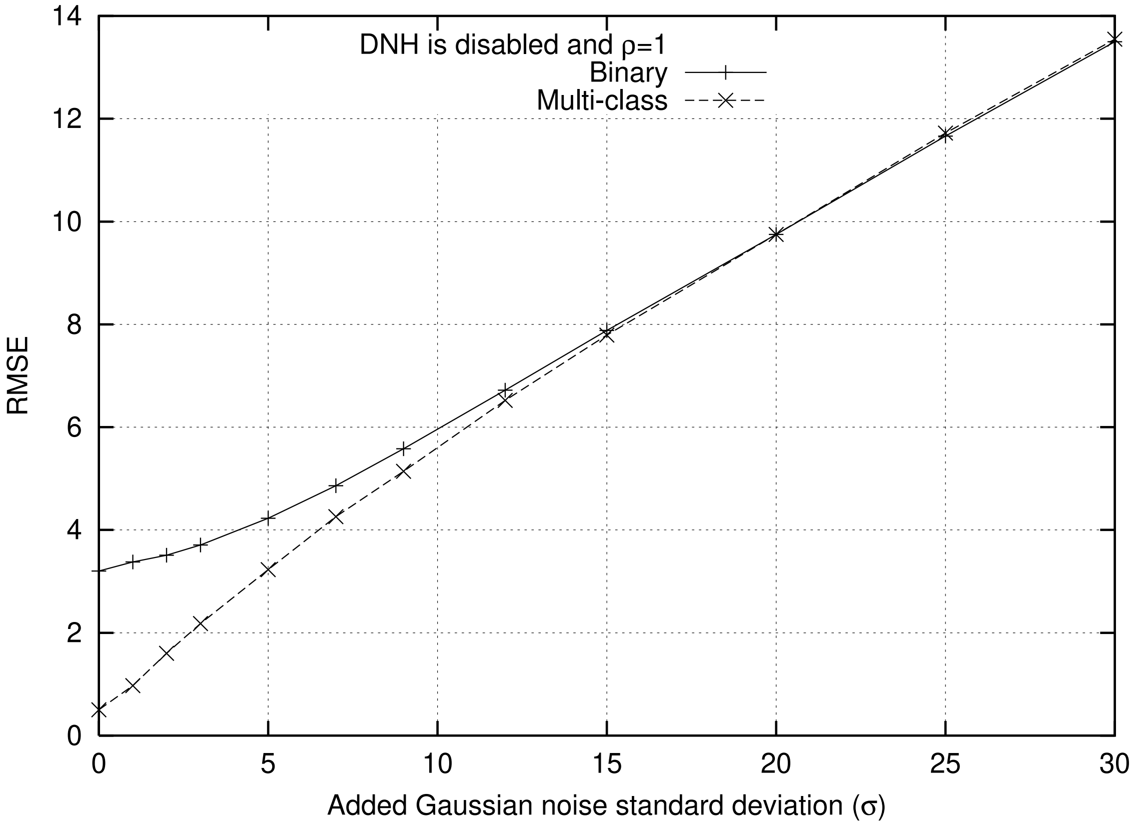

Figure 4.58: Comparison of binary and multi-class local segmentation for denoising `montage`. DNH is disabled and $\rho = 1$.

When the noise level is low, intricate image structure is more easily preserved. The multi-class algorithm will do its best to choose a high value of k to fit this structure, but will probably not capture all the detail. The binary model will also do its best, but with only two clusters it is unlikely to do as well as the multi-class fit. Thus the binary method is forced to invoke DNH, allowing fine detail to pass through unmodified. Retaining fine detail with a small amount of noise is more beneficial than removing both noise and some image structure.

Figure 4.60 again compares the two methods when both DNH and overlapping averaging are enabled. The RMSE metric is unable to distinguish the two methods to any significant level. Although not shown, my experiments show this behaviour is duplicated when larger windows, such as 5×5 and 7×7, are used instead.

These results show, at least for `montage`, that more complex multi-class modeling does not improve overall RMSE results when DNH and overlapping averaging are utilized. This behaviour could be exploited if the algorithm was implemented in hardware, or as embedded

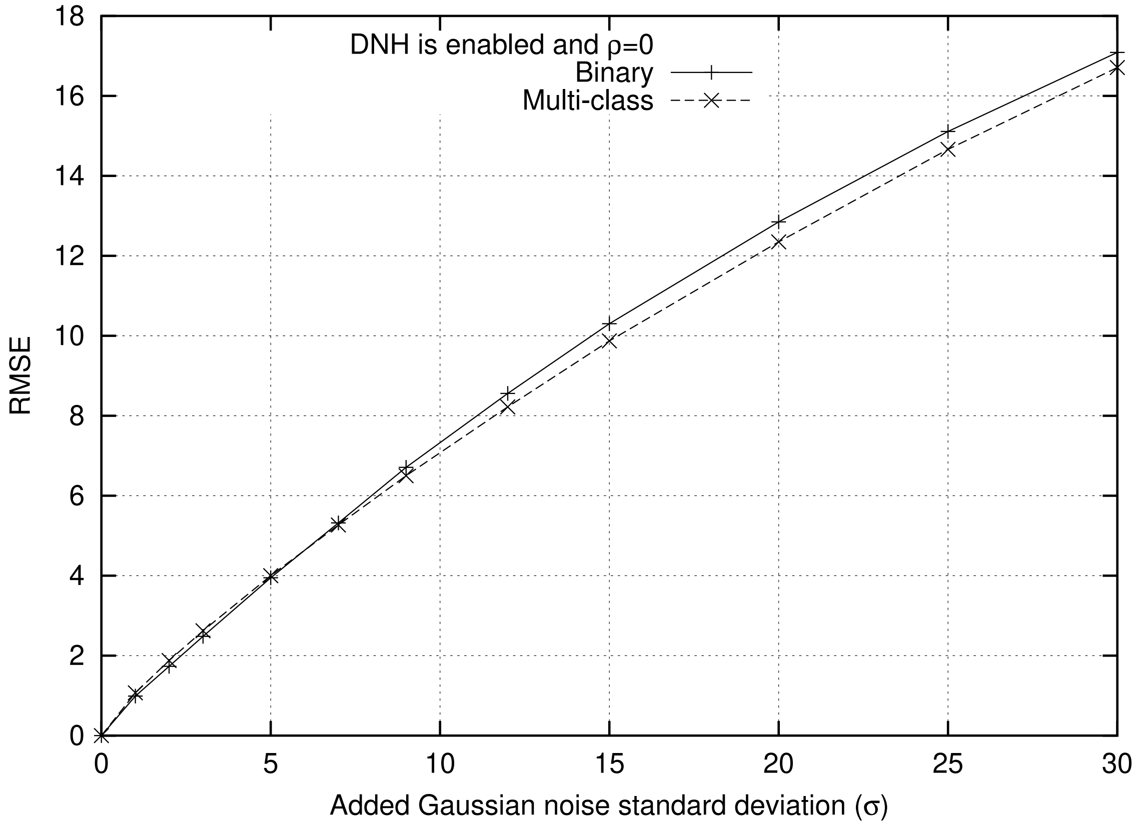

Figure 4.59: Comparison of $3{\times}3$ binary and multiclass local segmentation models on `montage`. DNH is enabled and $\rho = 0$.

software running on a system with relatively low processing and memory capabilities. The results also suggest that a large proportion of local windows are well modeled by two or fewer segments, or that DNH models complex regions better than a piece-wise constant multiclass model can. When a large number of segments are present, there are too few pixels with which to estimate the class parameters.

Consider a noisy $\sigma = 5$ version of `montage`. Figure 4.61 shows, using white, where the binary thresholding method used DNH, $k = 1$, and $k = 2$. Figure 4.62 shows the same for the multi-class method for values of k up to 8. The image for $k = 9$ is not included, as it was only used twice for the whole image. Table 4.5 lists the frequencies of model usage, DNH included, for the binary and multi-class methods for the same image.

The distribution of k for the multiclass method is monotonic decreasing. It did not use values of $k > 4$ very often in `montage`. Interestingly, $k = 9$ was chosen twice. This is equivalent to using DNH, because each pixel is given its own cluster. DNH is not used very often in the

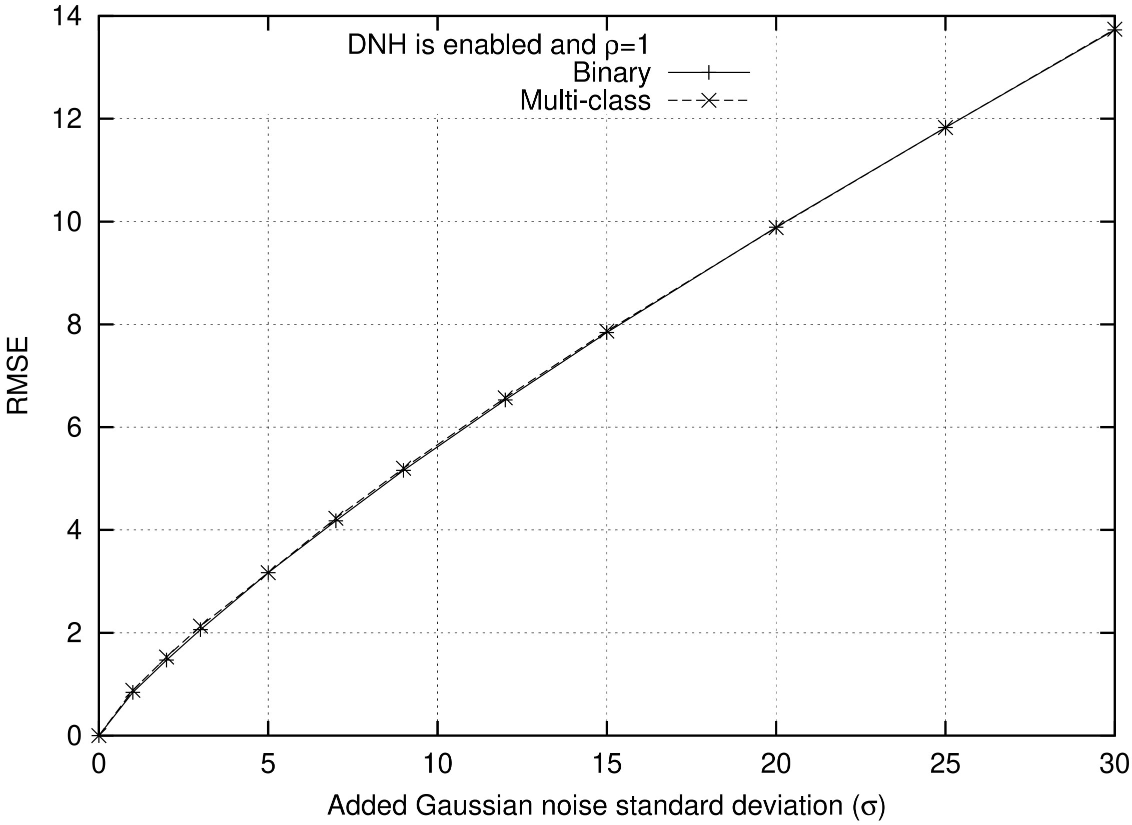

Figure 4.60: Comparison of $3{\times}3$ binary and multiclass local segmentation models on `montage`. DNH is enabled and $\rho = 1$.

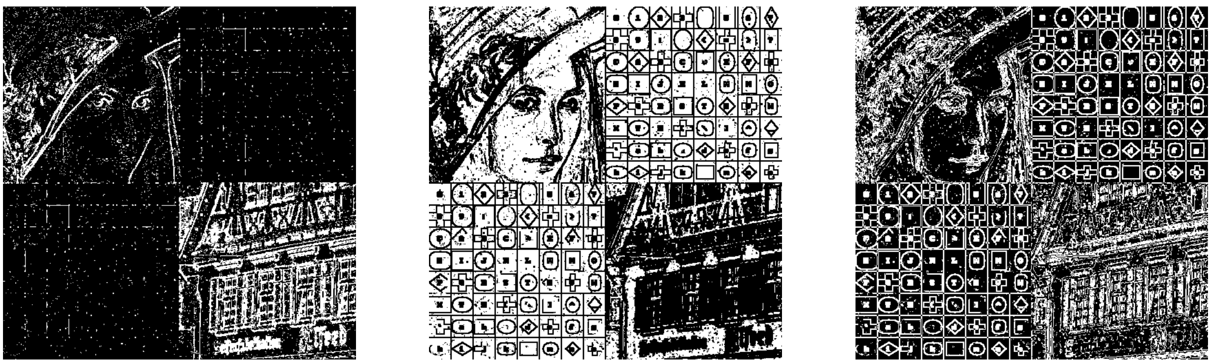

Figure 4.61: Binary local segmentation of noisy $\sigma = 5$ `montage`. White denotes: (a) DNH; (b) $k = 1$; (c) $k = 2$.

synthetic quadrants of `montage`, its invocation being more beneficial in the natural areas. When restricted to binary thresholding, the number of homogeneous regions remains the same. The 16% of windows previously diagnosed as $k \geq 3$ need to be reclassified. It seems that seems that 11% were allocated to DNH, while 5% were adequately handled by $k = 2$.

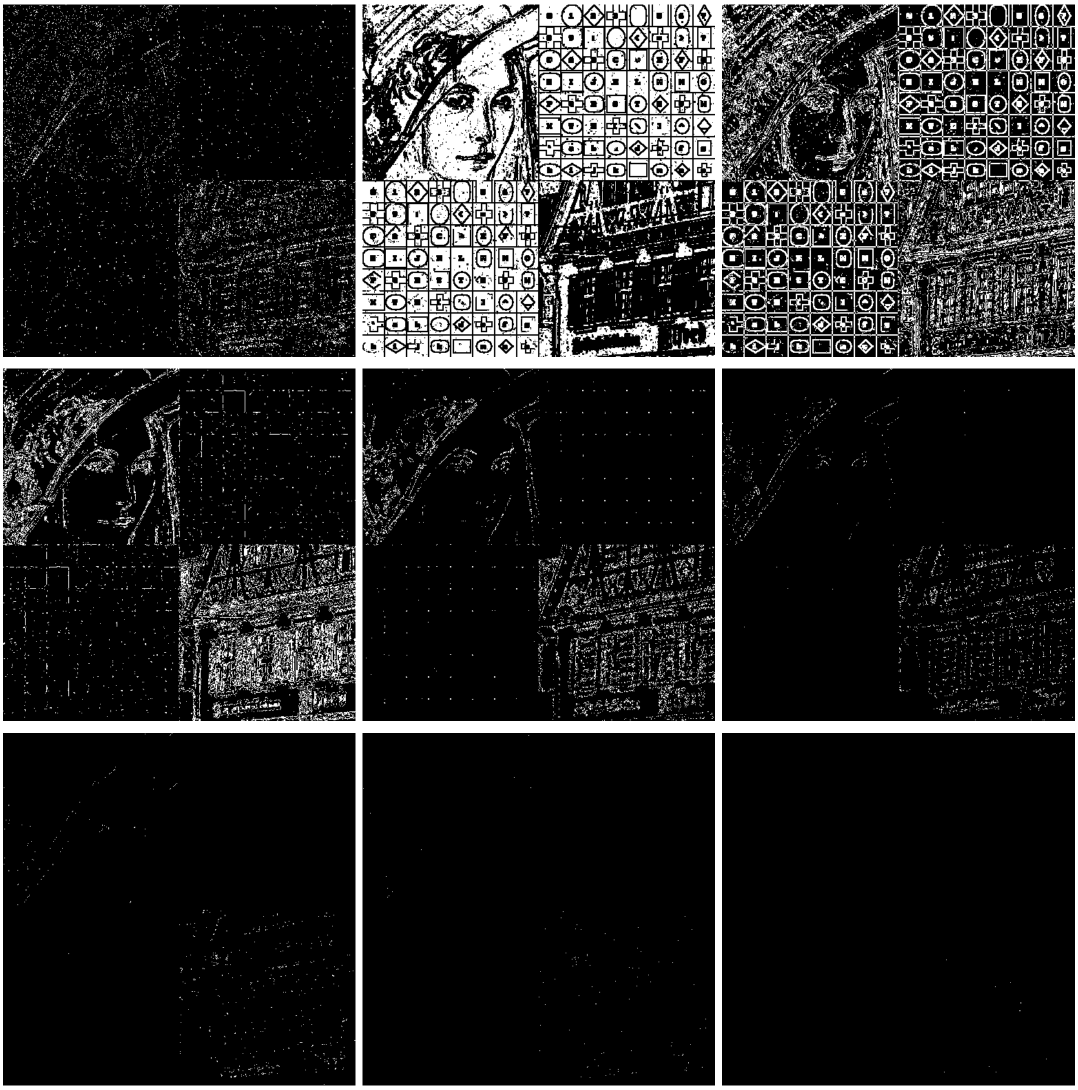

Figure 4.62: Multi-class local segmentation of noisy $\sigma = 5$ `montage`. White denotes: (a) DNH; (b)—(i) $k = 1$ to $k = 8$.

Model used	Multiclass $K = 9$	Binary $K = 2$
DNH	4%	15%
$k = 1$	52%	52%
$k = 2$	28%	33%
$k = 3$	10%	–
$k = 4$	4%	–
$k = 5$	1.1%	–
$k = 6$	0.2%	–
$k = 7$	0.03%	–
$k = 8$	11 pixels	–
$k = 9$	2 pixels	–

Table 4.5: Binary and multi-class model usage for noisy $\sigma = 5$ `montage`.

4.13 Estimation of the noise variance

In the experiments performed in this chapter, known amounts of synthetic noise are added to ground truth images. The various model selection criteria explored require that an estimate of the image noise variance, σ^2, be known *a priori*. Local segmentation uses the supplied noise variance to measure the natural level of variation of pixel values within segments. This helps discriminate between those features attributed to structure, and those attributed to noise.

Usually, the exact noise variance is unknown. It must be guessed by the user, or estimated from the noisy data itself. There are various algorithms in the literature for estimating the variance of additive noise in an image [Ols93]. The algorithms fall into two main categories:

1. Those which filter the noisy image first to remove structure and then estimate the variance from the residuals [Imm96, RLU99]
2. Those which ignore heterogeneous regions and use the remaining pixels to estimate the variance [Lee81b, Mas85, MJR90].

Those in the first category tend to overestimate the variance, as they are unable to completely remove structure. The second category tends to underestimate the variance, as avoiding heterogeneous regions would naturally bias them to less noisy regions.

Figure 4.63a shows `montage` with $\sigma = 5$ noise added to it. Figure 4.63b is the same image except that each pixel has been replaced by the unbiased standard deviation of its $3{\times}3$ local neighbourhood, itself included. As expected, the standard deviation is lowest in homogeneous regions, and highest in textured and edge regions. It is possible to use the distribution of local standard deviations to estimate the global noise variance, σ^2.

Figure 4.64 shows a histogram of all local standard deviations in `montage` having values from 0 to 100. If `montage` was completely homogeneous, the histogram would be normal with mean σ and variance $\sigma^2/9$. However, the existence of $3{\times}3$ heterogeneous regions in `montage` places many high variance entries in the right hand tail of the histogram [RLU99].

Figure 4.65 plots a smoothed version of the histogram for values up to 20. The smoothing was performed using a sliding window average of width 3. The smoothed histogram has a clear peak at around 5, which is equal to σ, the standard deviation of the added noise.

Figure 4.63: (a) noisy $\sigma = 5$ `montage`; (b) 3×3 local standard deviations.

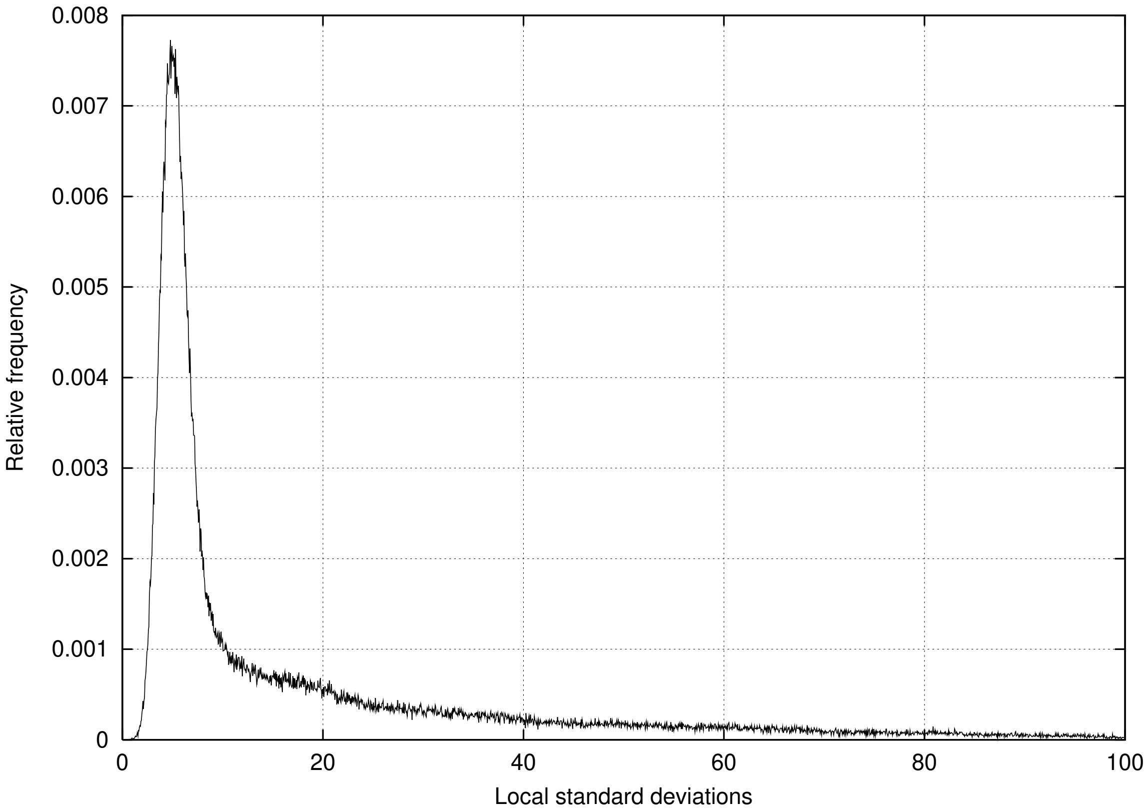

Figure 4.64: Histogram of local 3×3 standard deviations from Figure 4.63b.

Thus the mode of the local standard deviation distribution could be used as an estimate for σ [BS85]. This method belongs to the second category of noise estimation algorithms.

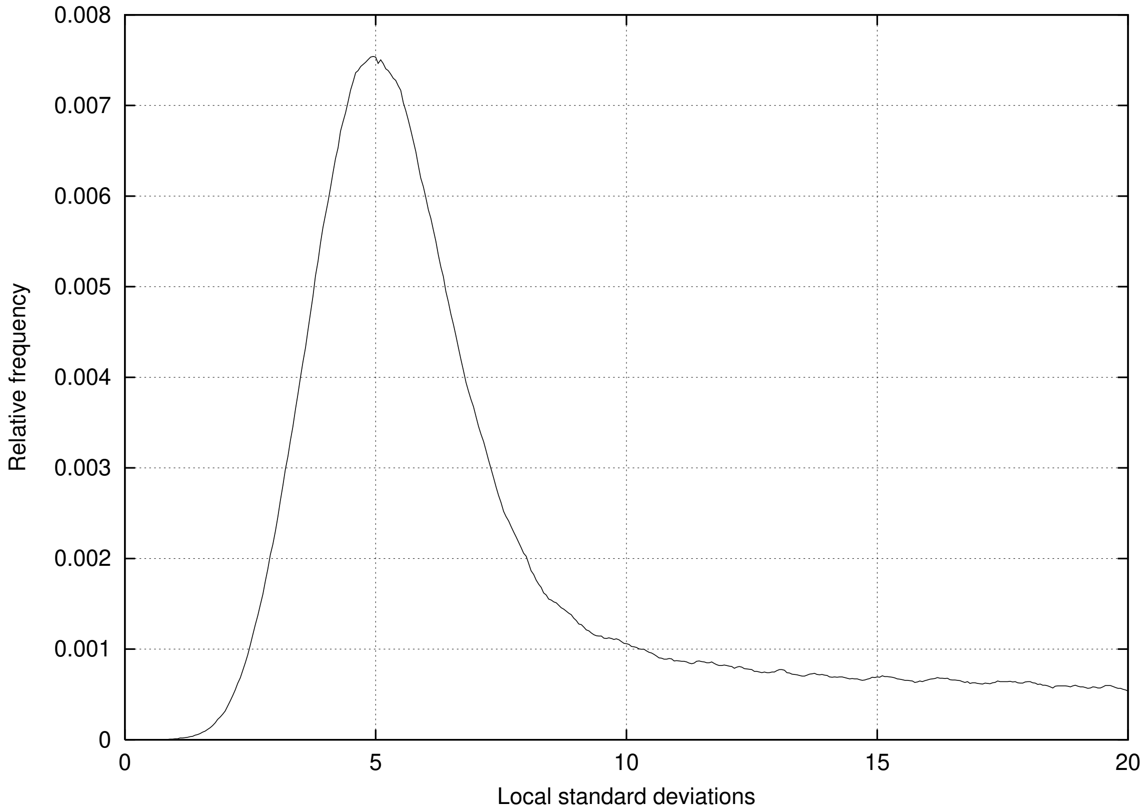

Figure 4.65: Smoothed version of main peak from Figure 4.64.

A noise estimation technique from the first category is Immerkær's method [Imm96], designed specifically for the case of zero mean additive Gaussian noise. To remove image structure it applies the $3{\times}3$ linear filter, N, in Equation 4.66, to the noisy image. This filter is the weighted difference of two Laplacian [GW92] filters, L_1 and L_2, which estimate the second derivative of the image signal. The effect of N is to reduce constant, planar and quadratic $3{\times}3$ facets to zero plus a linear combination of the noise. The effect of this filter on `lenna` is given in Figure 4.67.

$$L_1 = \begin{array}{|c|c|c|}\hline 0 & 1 & 0 \\ \hline 1 & -4 & 1 \\ \hline 0 & 1 & 0 \\ \hline\end{array} \qquad L_2 = \begin{array}{|c|c|c|}\hline 1 & 0 & 1 \\ \hline 0 & -4 & 0 \\ \hline 1 & 0 & 1 \\ \hline\end{array} \qquad N = L_2 - 2L_1 = \begin{array}{|c|c|c|}\hline 1 & -2 & 1 \\ \hline -2 & 4 & -2 \\ \hline 1 & -2 & 1 \\ \hline\end{array}$$

Figure 4.66: Filter masks used by Immerkær's noise variance estimation technique.

Once the image has been filtered to remove structure, the filtered pixel values can be used to compute the estimated noise variance, denoted $\hat{\sigma}^2$. Equation 4.38 describes the calculation,

Figure 4.67: (a) original `lenna` image; (b) after Immerkær structure suppression, with mid grey representing zero.

where $f(x,y)*N$ denotes the result of applying filter N at pixel $f(x,y)$. The formula does not use margin pixels because they do not have a full 3×3 neighbourhood.

$$\hat{\sigma}^2 = \frac{1}{36(X-2)(Y-2)} \sum_{x=1}^{X-2} \sum_{y=1}^{Y-2} (f(x,y)*N)^2 \tag{4.38}$$

By using the fact that, for a zero mean Gaussian random variable X, $\mathrm{E}[\,X^2\,] = \frac{\pi}{2}\mathrm{E}[\,|X|\,]$, Immerkær describes an alternative method for computing the estimated noise standard deviation. Rather than the sum of squared residuals, Equation 4.39 uses the scaled sum of absolute deviations. This formulation has two advantages: the summation requires no multiplications, and the absolute deviation is more robust to the presence of outliers [Hub81]. It would also be possible to use alternative robust methods, such as the median absolute deviation from the median [RL87], but these will not be investigated here.

$$\hat{\sigma} = \sqrt{\frac{\pi}{2}} \frac{1}{6(X-2)(Y-2)} \sum_{x=1}^{X-2} \sum_{y=1}^{Y-2} |f(x,y)*N| \tag{4.39}$$

Figure 4.68 compares the three noise estimation techniques for estimating the standard deviation of synthetic noise added to `montage`. As hypothesized earlier, the two Immerkær methods consistently overestimate σ, because they can not remove all the image structure. The mode method does very well, but slightly underestimates high values of σ.

Although `montage` is assumed noiseless, it was mentioned in Section 4.4 that the lower right quadrant does contain a small amount of noise, hence the robust Immerkær estimate may have some truth to it. If the noise level varies across an image, any global estimate will fall somewhere between the lowest and highest levels.

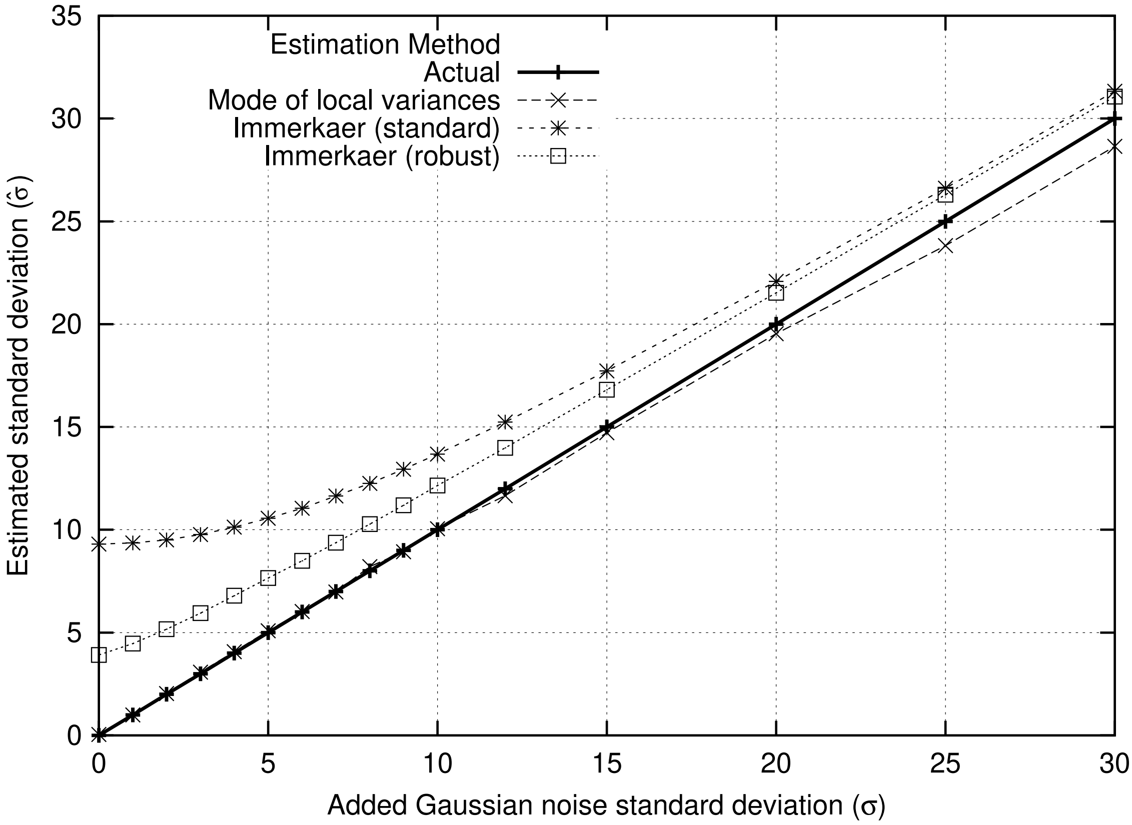

Figure 4.68: Different noise estimation algorithms for `montage`.

In real situations, the noise variance must be estimated from the noisy image without reference to any ground truth. Consider the familiar `lenna` image from Figure 4.18, which already contains noise. Let us assume that it is additive Gaussian noise with variance s^2. If synthetic noise of variance σ^2 were added to `lenna`, the actual noise variance would be a combination of the original and the synthetic, namely $\sigma^2 + s^2$. We can compare noise estimation techniques in the same manner as before on images already containing noise by correcting the "actual standard deviation" to include the pre-existing noise. Instead of comparing estimated noise levels to σ, the curve $\sqrt{\sigma^2 + s^2}$ can be used instead.

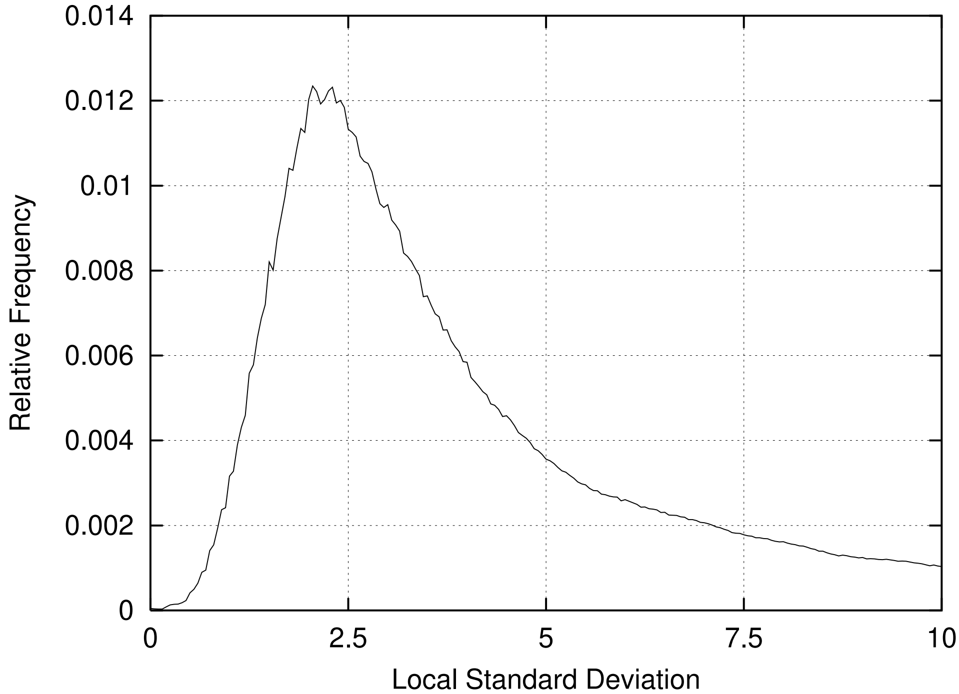

Figure 4.69: Smoothed histogram of local standard deviations for `lenna`.

A method for estimating the base noise level, s, is needed. Figure 4.69 plots a smoothed histogram of `lenna`'s local 3×3 standard deviations. The mode of this histogram estimates $s = 2.38$, while the robust Immerkær method estimates $s = 2.66$. Figure 4.70 compares noise estimation algorithms for `lenna`, using $s = (2.38 + 2.66)/2 = 2.52$ to correct the curves. All the methods now mostly follow the ground truth. The *standard* Immerkær method slightly overestimates at the lower end, but this is somewhat due to its estimate not contributing to the base noise level estimate, s. The mode method slightly underestimates at the higher end, just as it did for `montage`. At higher noise levels the assumption that all the heterogeneous variances exist in the far right of the histogram breaks down, causing the mode to be less accurate. It may be possible to estimate the mode by first fitting a spline to the data, and then analytically determining the peak. Overall, the deviations are minor, and any of the estimates would probably produce similar denoising results.

The specific noise estimation procedure used is only important with respect to the quality of the variance estimates it produces. The local segmentation process does not require modification if the noise estimation technique is varied. The noise estimator is a separate module, which may be replaced with a better technique if one becomes available.

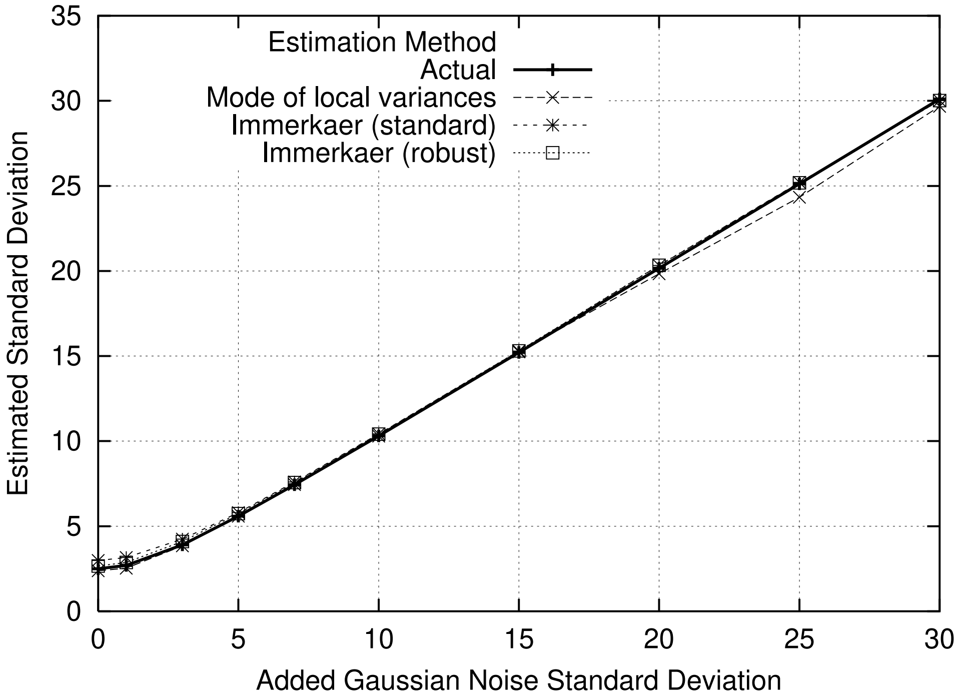

Figure 4.70: Different noise estimation algorithms for `lenna`, corrected for existing noise.

4.14 Comparison to other denoising algorithms

This chapter has developed a family of denoising algorithms based on the local segmentation principle. Various modifications to the basic technique were explored, many of which improved results both visually and in terms of RMSE. The best trade-off between efficiency and performance over a range of noise levels is determined to have the following attributes:

- Use of a local 3×3 window (Section 4.10)
- Model local regions as consisting of either 1 or 2 segments (Section 4.12)
- Use of Lloyd's algorithm for threshold selection local segmentation (Section 4.6)
- Simple model selection criterion using $C = 3$ (Section 4.7)
- Use of the "do no harm" (DNH) option to limit worst case behaviour (Section 4.9)
- Equal averaging of overlapping estimates, after DNH has been applied (Section 4.11)
- One iteration only (Section 4.8)
- Estimate noise variance with Immerkær's robust method (Section 4.13)

This variant shall hereafter be referred to as FUELS — "filtering using explicit local segmentation". FUELS will be compared to the four denoising algorithms described below, each of which was described in detail in Section 3.4.2. These algorithms were chosen because, like FUELS, they operate locally, are relatively efficient, and use similar image models.

WMED (Section 3.4.4)

WMED is the 3×3 centre weighted median with the centre pixel included three times. Although not as efficient as the mean in homogeneous regions, it can preserve fine lines and some edges. WMED may be considered a minimum level of performance that any structure preserving denoising algorithm should reach. It is probably better known for its resistance to impulse noise.

GIWS (Section 3.4.5)

Gradient inverse weighting smoothing, centre pixel not included, will be used. GIWS can be considered an accelerated single iteration of anisotropic diffusion.

SUSAN37 (Section 3.4.5)

The standard 37 pixel SUSAN filter is one of the best local denoising algorithms in the literature, and has a very efficient implementation. It requires a brightness threshold t, and Smith claims that $t = 20$ works well over all image types. I found that setting $t = 3\hat{\sigma}$ gave better results, where $\hat{\sigma}$ is the same estimated noise level that FUELS uses when it is not provided with one.

SUSAN9 (Section 3.4.5)

The SUSAN implementation also has a 9 pixel mode, which uses a 3×3 window. This was also included as it is closer in size to the window that FUELS uses. When something affects both SUSAN9 and SUSAN37, the generic term "SUSAN" is used.

4.14.1 The `montage` image

Supplied with noise level

Figure 4.71 plots the RMSE performance of FUELS, WMED, GIWS, SUSAN37 and SUSAN9 for `montage`. Just for this time, FUELS and SUSAN were provided with the true value of σ. FUELS clearly outperforms the others at all noise levels. SUSAN37 also did well until $\sigma \approx 12$, after which point SUSAN9 took over. SUSAN9 performed badly for very low noise levels. It is unable to assimilate enough pixels, and switches to being a 3×3 median filter. GIWS consistently tracks the SUSAN37 method, but 2 RMSE units higher.

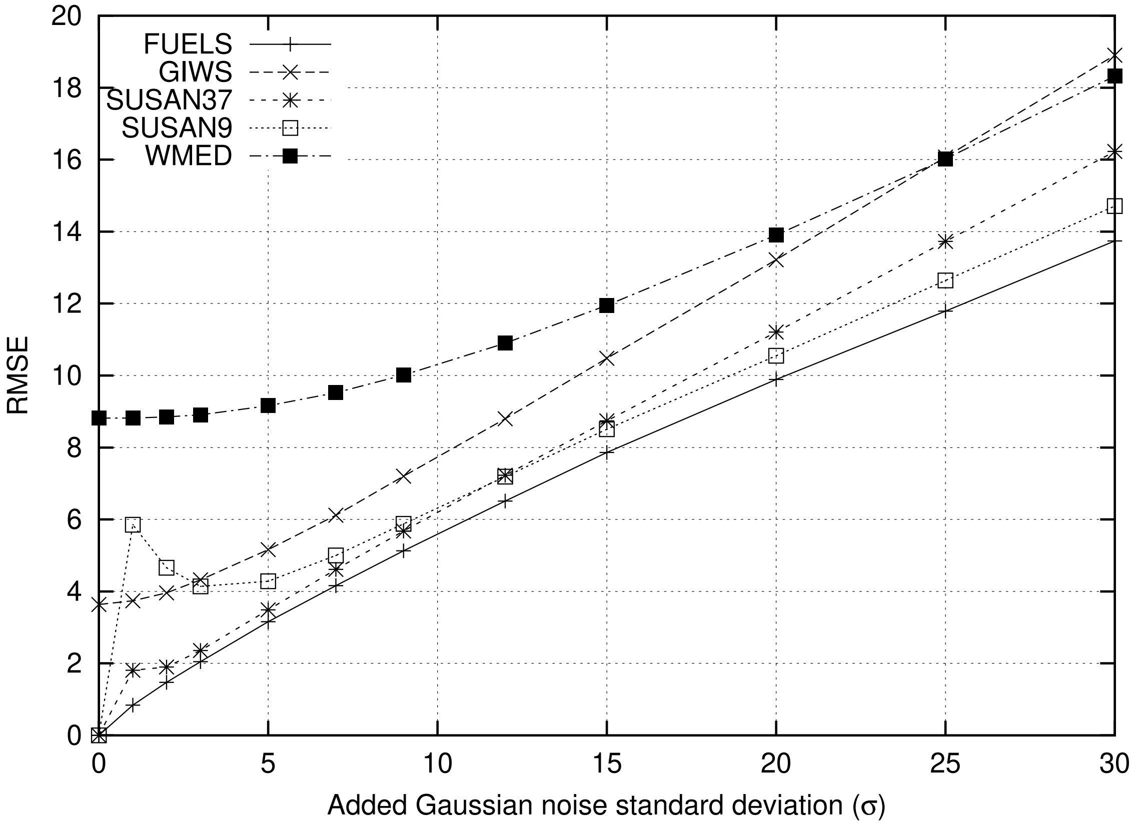

Figure 4.71: RMSE comparison for `montage`, true σ supplied.

Estimated noise level

Figure 4.72 repeats the experiment, except that this time FUELS and SUSAN utilise the estimated noise variance. Immerkær's noise estimation method tends to overestimate σ for `montage`. This is due to the large number of strong step edges, which are somewhat atypical for photographic type data. The results once again show FUELS to perform best, although SUSAN is within one RMSE unit at most points. It is not clear why SUSAN37 does better at low noise levels, and SUSAN9 at high levels. I would expect a larger mask to provide more smoothing at higher noise levels than a smaller one.

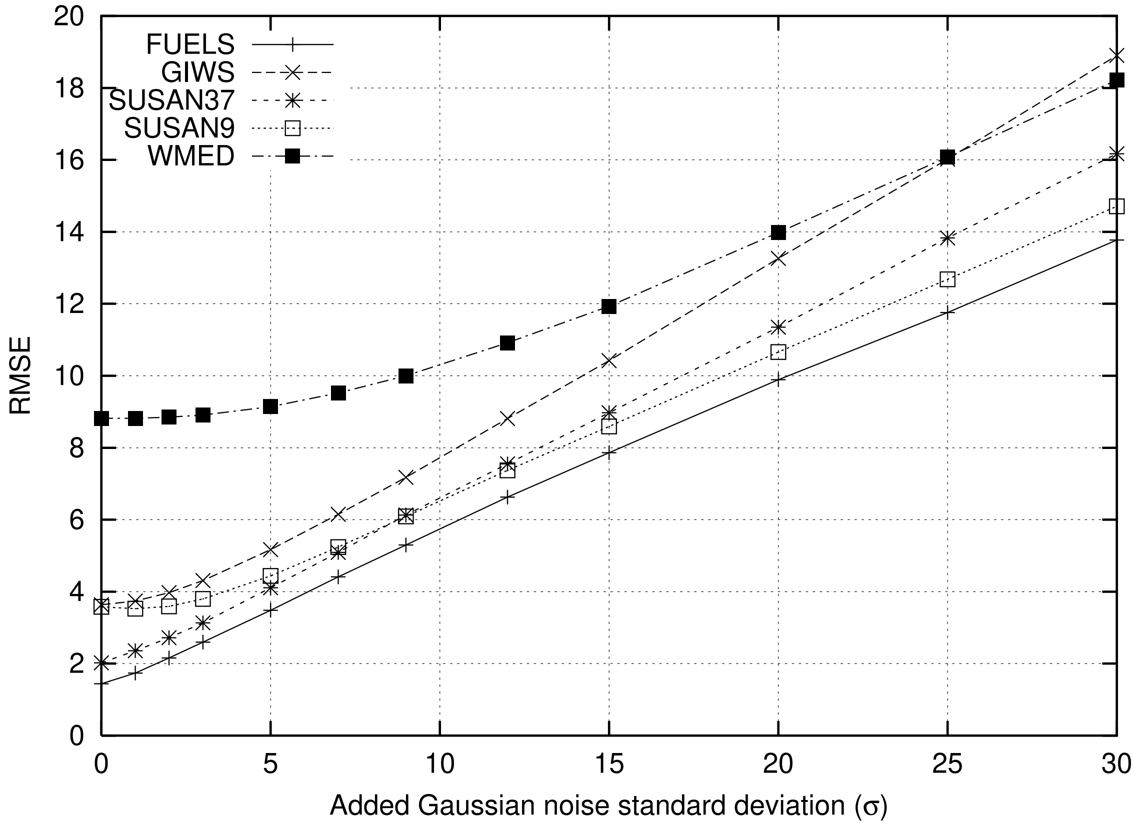

Figure 4.72: RMSE comparison for `montage`, with σ estimated.

The use of the estimated noise variance has narrowed the gap between those algorithms which exploit its knowledge, namely FUELS and SUSAN, and those which do not, namely GIWS and WMED. The poor performance of SUSAN9 at low noise levels has also gone. This must be due to the estimated noise variance being higher than the "true" variance. The resulting brightness threshold supplied to SUSAN is therefore higher, allowing it to assimilate more pixels.

The denoised output

Figure 4.73 compares SUSAN and FUELS in terms of structure preservation. The 96×96 image examined is from the centre of `montage` when $\sigma = 5$. Ideally the difference images should contain no structure. There are no obvious differences in the top left quadrant. In the top right quadrant both had difficulty with the oval shape, because it has a similar intensity to its background. SUSAN37 had trouble with the cross above it, whereas FUELS left little trace of it in the difference image. In the lower left quadrant FUELS seems to have smaller errors at segment boundaries, but SUSAN37 seems to have removed more noise within segments. This could be due it using a 37 pixel window, compared to FUELS' 9 pixels. The bottom right quadrant contains a lot of fine detail, and overall FUELS appears to have done better than SUSAN37 there. The roof and shutters are much more obvious in the SUSAN37 filtered difference image.

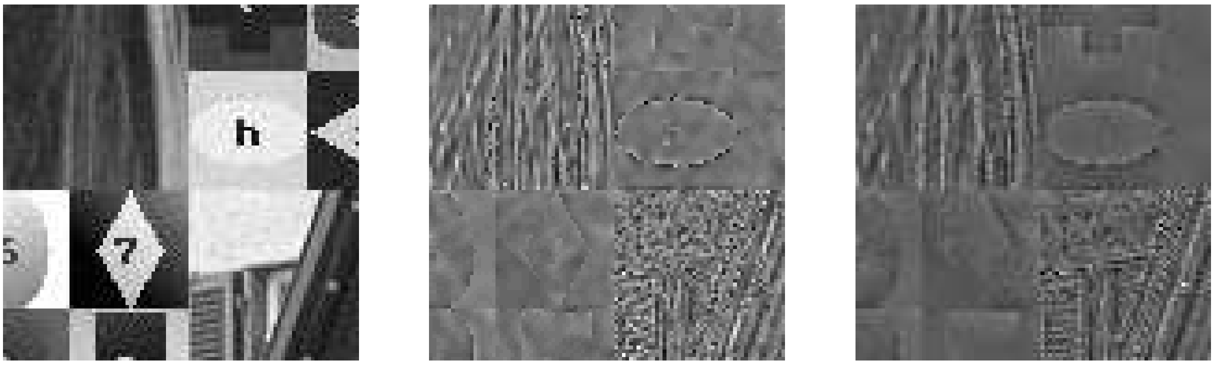

Figure 4.73: (a) noisy $\sigma = 5$ part of `montage`; (b) FUELS enhanced difference image, RMSE=3.48; (c) SUSAN37 enhanced difference image, RMSE=4.11.

Figure 4.74 plots the WCAE for `montage`. Both WMED and GIWS have a near constant WCAE, suggesting there is a pixel pattern in `montage` which they both consistently get wrong. FUELS' WCAE seems directly proportional to σ. This behaviour is desirable because it indicates that its mistakes are due to the random nature of the noise, rather than difficulty preserving any particular image structure. Overlapping averaging and DNH are mostly responsible for this positive feature of FUELS.

Both SUSANs have poor worst case performance at lower noise levels. Figure 4.75a shows a 40×30 part of `montage` without added noise, for which SUSAN is given a brightness threshold $t = 11.73$. Figures 4.75b–c show SUSAN37's output and the corresponding difference image. The two white spots in the difference image are large filtering errors, the worst of which is out by 150 intensity units. This behaviour can be explained by the fact that SUSAN switches into a special mode when all the neighbouring weights are too small. In this mode, SUSAN assumes the centre pixel is impulse noise, and instead uses the median of its eight immediate neighbours as its denoised estimate. The region of `montage` in Figure 4.75a is very busy, and the median obviously chose inappropriately.

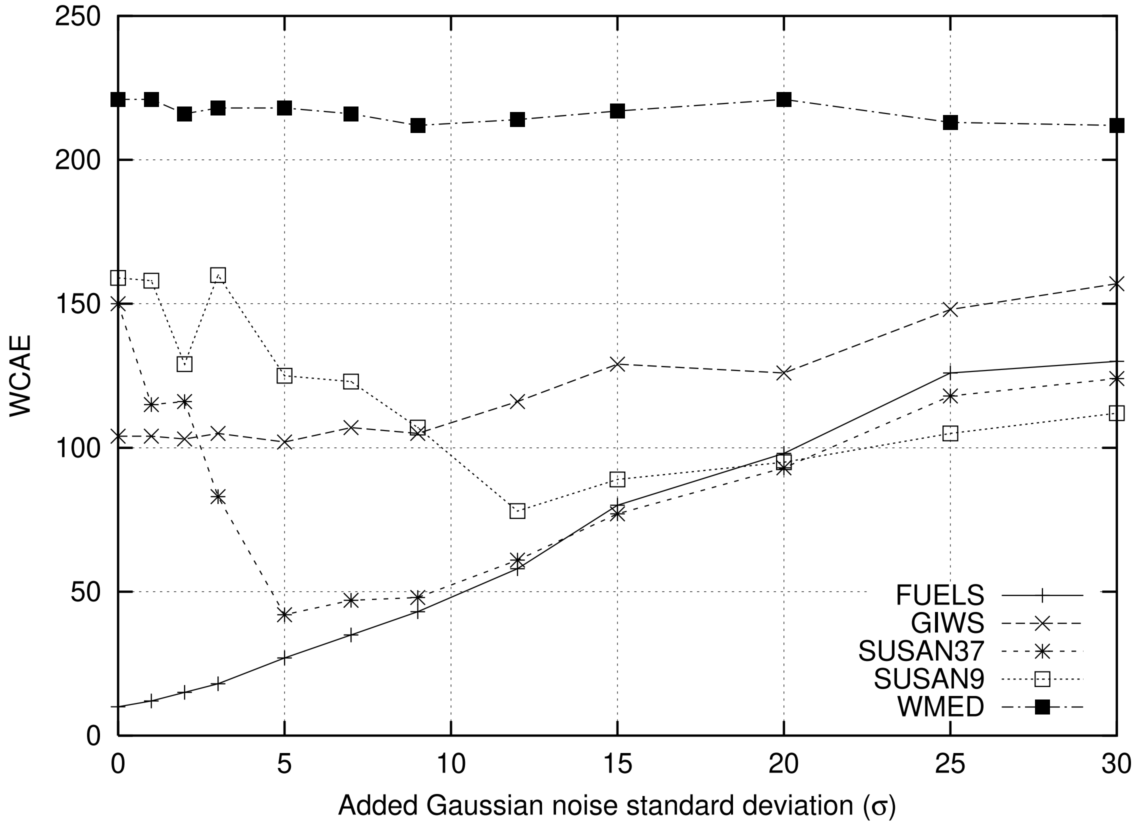

Figure 4.74: WCAE comparison for `montage` image, withσ estimated.

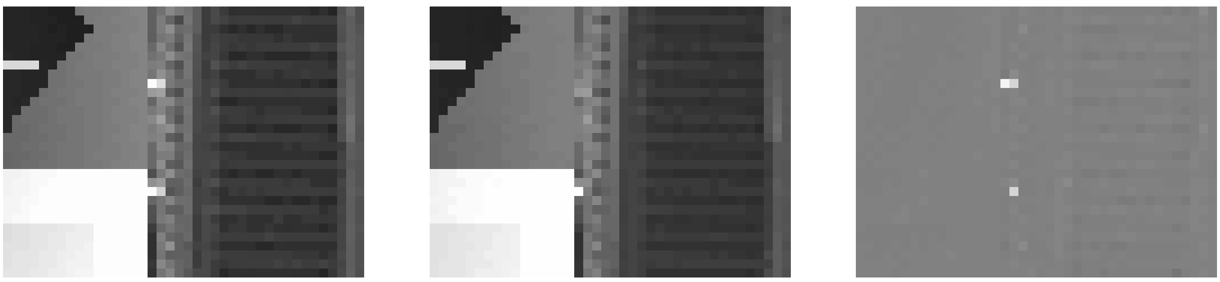

Figure 4.75: (a) part of `montage` without added noise; (b) SUSAN37 output using $t = 11.73$; (c) difference image.

4.14.2 The **`lenna`** image

The `lenna` image [len72] in Figure 4.18 is the single most popular test image used by the image processing community. Because `lenna` is not synthetic, it already contains some noise of unknown distribution. In Section 4.13, under the assumption of additive Gaussian noise, the natural noise standard deviation, s, was estimated as being somewhere between 2 and 3. Using the original `lenna` as ground truth is difficult, because any good denoising algorithm will remove both the original and synthetic noise. As the added noise level, σ, increases, the original noise is swamped and hence can mostly be ignored. However, when $\sigma \leq 2s$ or so, the original noise level will impact the RMSE results. In these cases visual inspection of the denoised and difference images would be necessary to fairly compare the quality of smoothing and structure preservation.

Figure 4.76 plots RMSE results for `lenna`. Although FUELS performs best at all noise levels, the two SUSAN variants are very close. In fact, when $\sigma \leq 9$, FUELS, SUSAN and GIWS perform within one RMSE unit of one another. Perhaps the common use of `lenna` as a test image has mildly biased algorithmic development toward those techniques which do well on `lenna` and other images with similar properties.

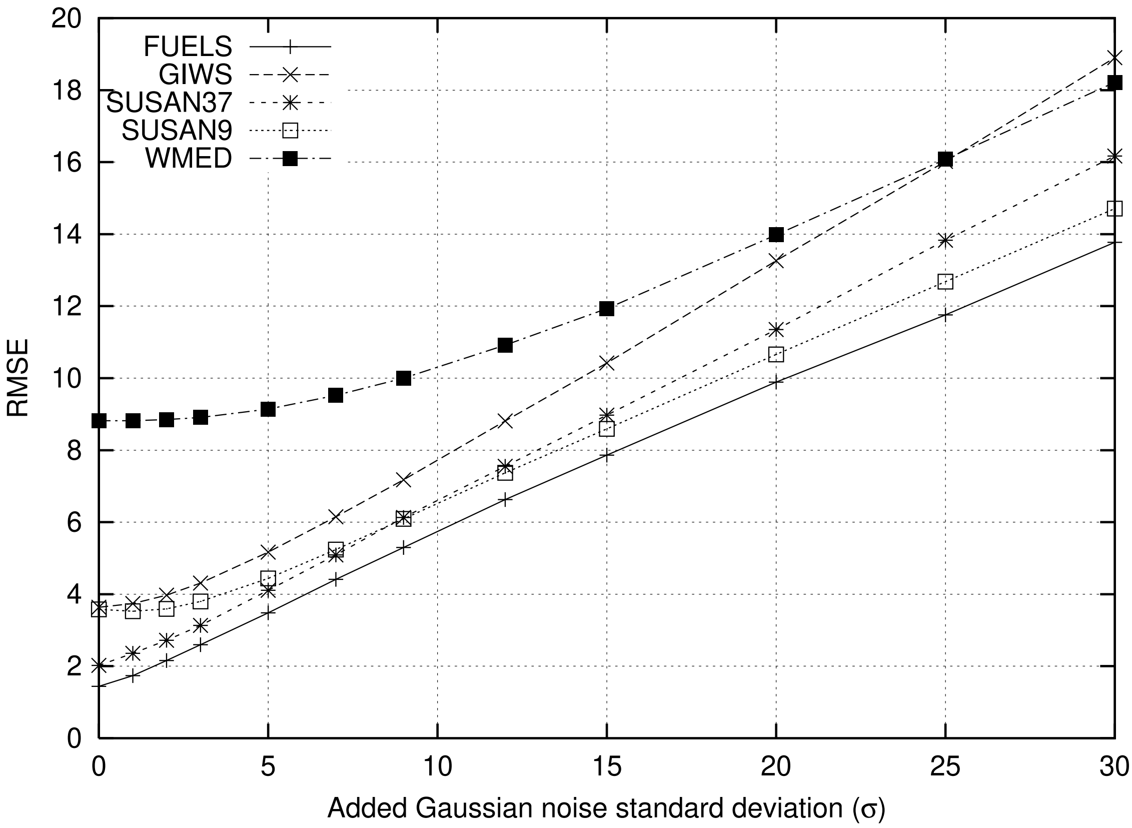

Figure 4.76: RMSE comparison for `lenna`, with σ estimated.

Figure 4.77 shows the WCAE when denoising `lenna`. All except FUELS have a near-constant WCAE. Interestingly, SUSAN37 improves its WCAE as the added noise increases from none to low amounts. This unusual behaviour is once again likely to be due to its fall-back median filter. Although the monotonic increase of FUELS' WCAE is desirable, it reaches levels higher than the other algorithms when $\sigma > 15$. FUELS has some large errors in the feathers of `lenna`'s hat, which is notoriously difficult to process. Modeling them as piece-wise constant segments is likely to be problematic, and invoking DNH in high noise environments will create errors proportional to the noise level.

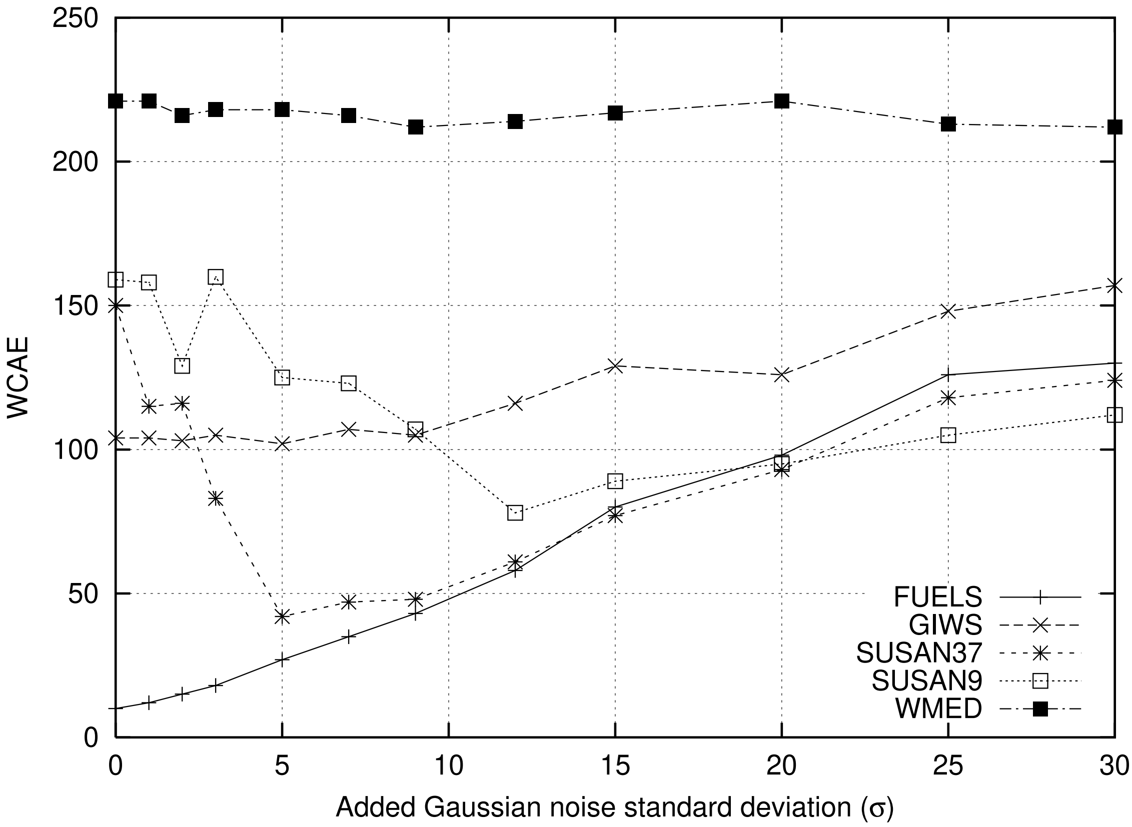

Figure 4.77: WCAE comparison for `lenna` image, with σ estimated.

4.14.3 The **`barb2`** image

Figure 4.78a shows `barb2`[4] from the JPEG test set [PM93]. It is larger than `lenna`, having resolution 720×576. A feature of images from the JPEG test set is that they only contain intensities in the range $[16, 235]$. The histogram of `barb2` in Figure 4.78b clearly shows the saturation effect caused by pixels being clamped to the reduced range. The image is interesting in that it contains large areas of straight edge patterns in the chair, clothes and books, but also contains clearly homogeneous background regions, including a dark, noisy

[4]Available from `http://www.csse.monash.edu.au/~torsten/phd/`

margin on the right hand side. This variation in features make barb2 useful for examining the structure preservation capabilities of denoising algorithms.

Figure 4.78: (a) the 720×576 barb2 image; (b) its histogram.

Figure 4.79 compares the denoising algorithms in terms of RMSE on barb2. Compared to lenna, the results are more diverse, but FUELS still does best for all values of σ. Another interesting fact is that SUSAN37 achieves lower RMSE results than SUSAN9 up until $\sigma = 7$, after which they swap. Intuition suggests to me that this should be the other way around, as a larger mask should allow better smoothing in the presence of more noise. The threshold chosen for SUSAN may not be appropriate for all image and noise level combinations.

When denoising a highly patterned image like barb2, one would expect most algorithms to produce some large errors. The WCAE graph in Figure 4.80 supports this hypothesis. SUSAN9 had particular trouble with the large number of edges in the image, again due to its fall-back median filter. FUELS has done very well for $\sigma \leq 15$, assisted by its DNH feature.

Figure 4.81 compares the denoised outputs of FUELS and SUSAN for a sub-image of barb2 without added noise. The estimated noise standard deviation was 4.2. The corresponding difference images are also included. These were enhanced by linearly stretching differences in the range $[-12, 12]$ to $[0, 255]$. Large differences were clamped to ± 12.

FUELS has left very little structure in the difference image. Its DNH option has obviously been used in the vest area, because most of the differences there are zero. SUSAN9's difference image exhibits little image structure too. The large differences occur in clumps, rather than spread evenly across the image. They are particularly noticeable on the vest. This could

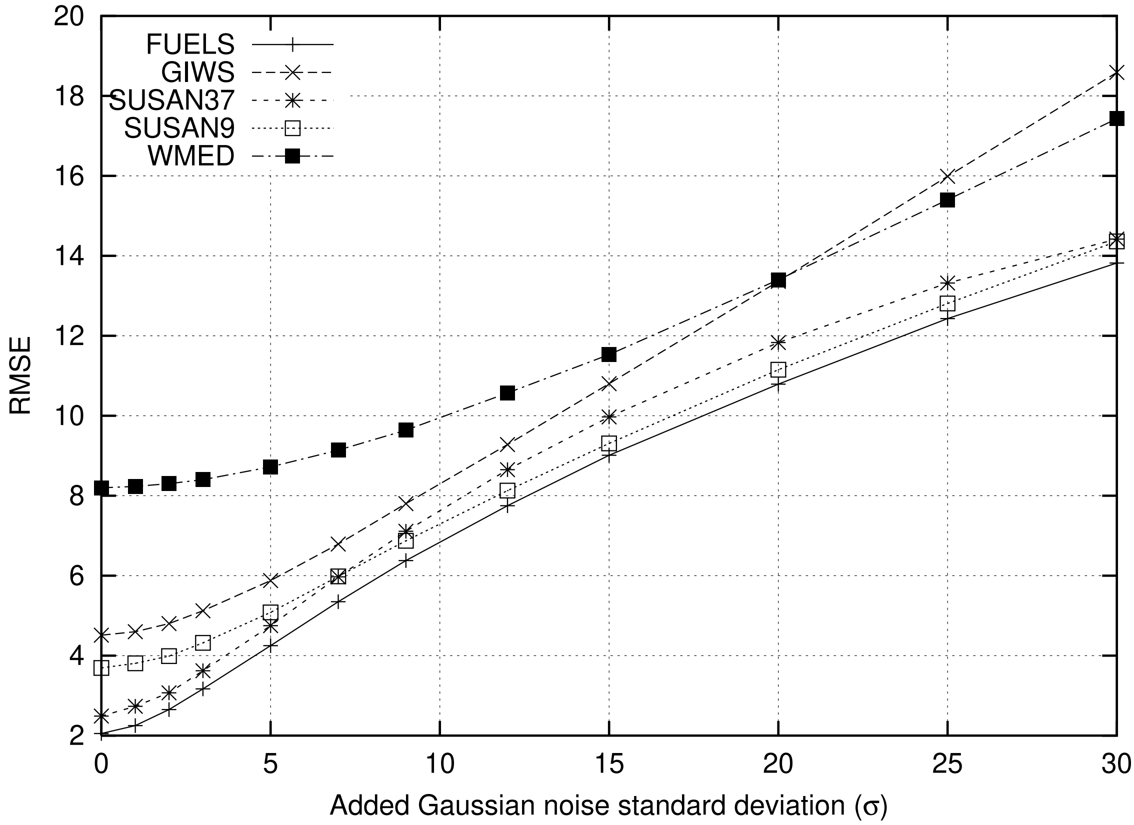

Figure 4.79: RMSE comparison for `barb2`, with σ estimated.

be due to the fall-back median filter, which is a poor choice for that type of image structure. The face and wicker chair from `barb2` are clearly noticeable in the SUSAN37 difference image. Although SUSAN37 produces smaller differences on average than SUSAN9, they are correlated with edges in `barb2`. This is probably due to the 37 pixel mask being more likely to encompass segment boundaries, and hence blur them slightly.

A histogram of the three difference images is given in Figure 4.82. FUELS' errors all occur in a tight distribution around zero. SUSAN37 has a slightly wider band, and one or two outlier errors at 50. SUSAN9's predisposition to larger errors is clearly indicated by the fat tails of its distribution.

4.15 Conclusions

It has been shown that the principles of local segmentation can be used to develop effective denoising algorithms. After many analyses, the FUELS algorithm for denoising greyscale

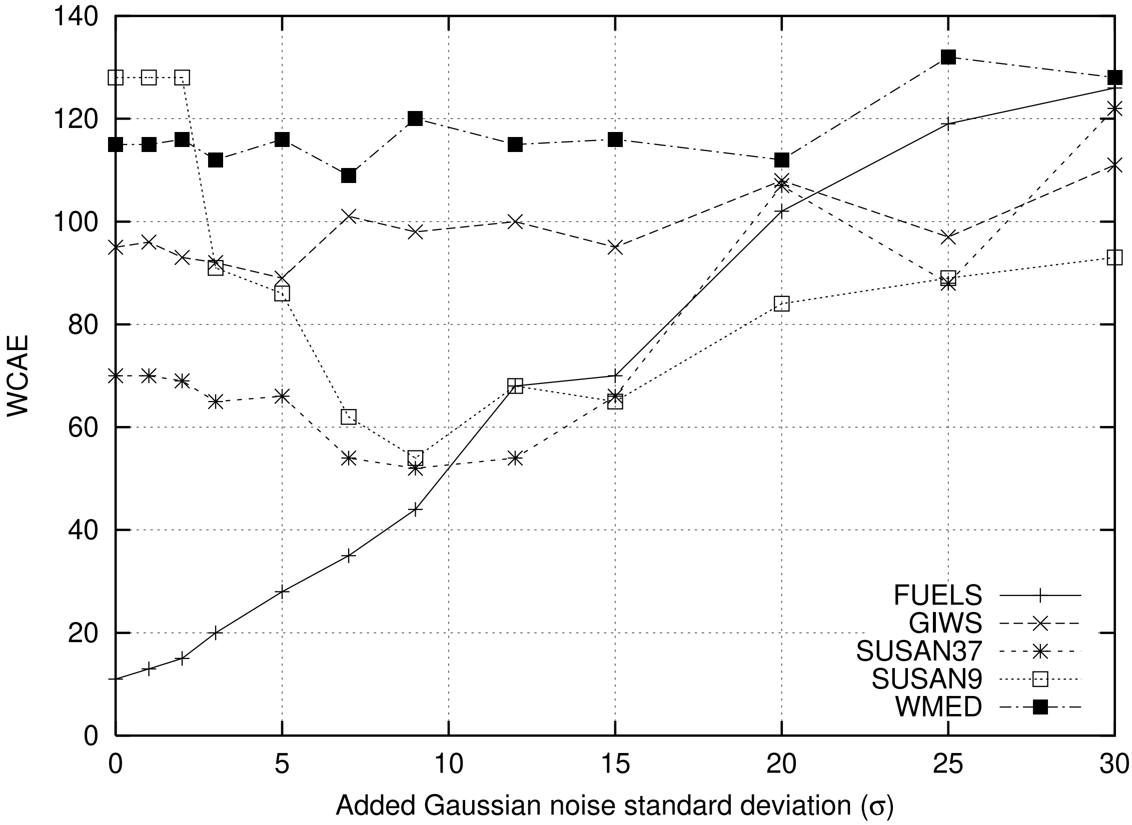

Figure 4.80: WCAE comparison for `barb2`, with σ estimated.

images contaminated by additive noise was presented. FUELS has an efficient implementation, and only requires one parameter, the level of noise in the image. This can be supplied by the user, or FUELS can determine it automatically. FUELS was shown to outperform existing methods, like SUSAN and GIWS, for a variety of images and noise levels.

Both quantitative and qualitative methods were used to compare FUELS to other methods. The RMSE was used to measure objectively the closeness of denoised images to the originals. FUELS consistently produced lower RMSE results than SUSAN, the next best performer. The WCAE was used to gauge the worst case performance of each algorithm. FUELS had the desirable attribute of having a WCAE proportional to the noise level in the image. The others, SUSAN included, tended to have constant or erratic WCAEs. To assess the structure preserving ability of each algorithm, difference images were used to highlight those areas of the image in which larger errors occurred. Although structure was apparent in all the difference images, FUELS' tended to contain the least.

The FUELS algorithm has various attributes which are responsible for its good performance.

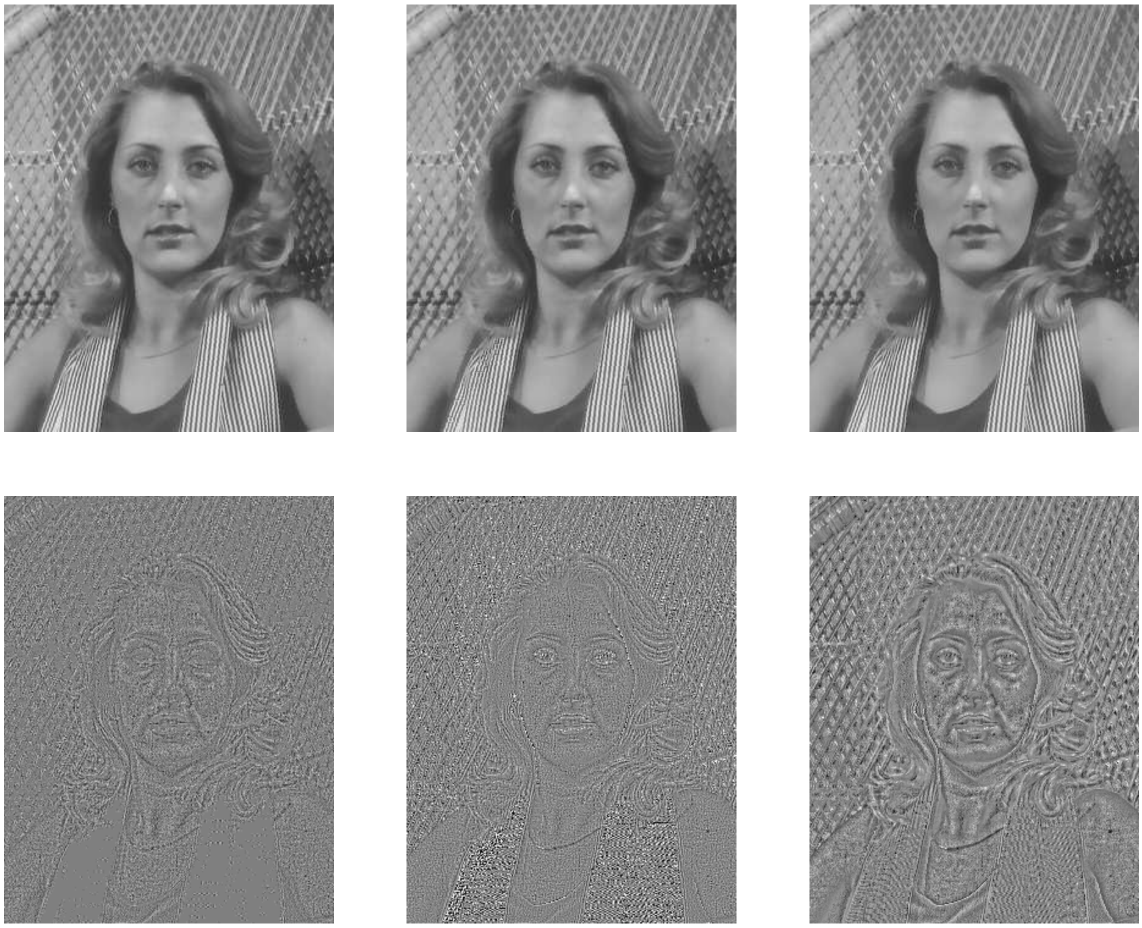

Figure 4.81: For barb2 with no added noise: (a)—(c) FUELS, SUSAN9 and SUSAN37 denoised output; (d)—(f) corresponding enhanced difference images.

Like SUSAN and GIWS, FUELS attempts to average only those pixels which, in some sense, belong together. FUELS achieves this by explicitly segmenting the whole local region, insisting that each pixel belong wholly to one segment. This contrasts with SUSAN and GIWS, which advocate a soft cut-off. A hard-cut off ensures that pixel values can not diffuse across segment boundaries to influence other, unrelated, pixels. It has the advantage of providing a local approximation to the underlying image, which is arrived at democratically, because each pixel contributes equally to the local segmentation. This is unlike SUSAN and GIWS, which assimilate pixels based on their relationship to the centre pixel in the window only.

FUELS acknowledges the requirement for denoising algorithms to have "magic" parameters to allow adaptation to different images. The most common type of parameter controls the distinction between noise and structure. GIWS implicitly has one in the $\frac{1}{2}$ term in the denominator of its weighting function, while SUSAN needs a brightness threshold to control

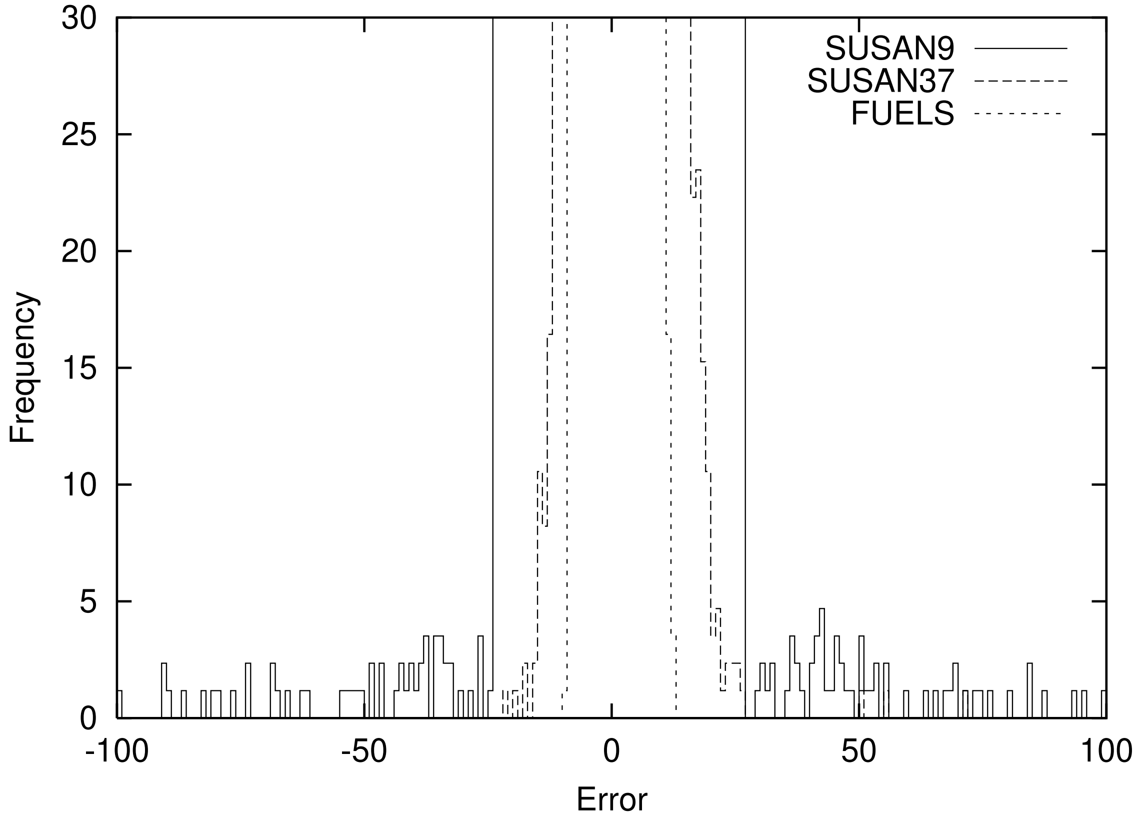

Figure 4.82: Partial histogram of FUELS and SUSAN denoising errors for `barb2`.

its pixel assimilation. It is not always clear how to choose these parameters appropriately. The use of local segmentation naturally links the image model to the segmentation algorithm used. FUELS was developed under the assumption of piece-wise constant segments corrupted by additive zero-mean Gaussian noise. This obviously suggests the noise standard deviation as a measure of the natural level of pixel variation within segments. FUELS has the advantage of automatically determining this parameter if it is unknown to the user.

Determining a local approximation to the underlying image at each pixel position has two advantages. Firstly, because each pixel participates in multiple overlapping local approximations, there exist multiple estimates of that pixel's true value. FUELS exploits this by averaging the overlapping estimates. This has the effect of increasing the effective window size by a factor of 2.8, without the overhead of segmenting more pixels. Secondly, the local approximation can be assessed, and rejected if necessary. Just because an algorithm has determined its "best" estimate for a pixel's value does not mean that estimate is of high quality. If any of the locally approximated pixel values differ too much from their original noisy values, FUELS refuses to accept them. Instead, the local approximation is set equal to the unfiltered pixel values. The acceptable intensity difference used is strongly related to the estimated image noise level. This “do no harm” philosophy significantly improves results at lower noise levels, and may be applied to any denoising algorithm, not just FUELS.

Examination of images in terms of local segmentation has lead to a better understanding of image processing on a small scale, particularly for the commonly used 3×3 configuration. At conversations and seminars, I have often heard the off-hand comment that “only 10% to 20% of images are edges”. Analysis of the distribution of k values chosen for images in the chapter suggest that only around 50–60% of pixels are locally homogeneous, 20–30% consist of two segments, and 10–20% tend to be difficult to model well. Perhaps the speakers were confusing edges with those pixels which are difficult to predict or model. The success of FUELS suggests that attempts to model these difficult blocks does not significantly improve denoising performance.

4.16 Related work

An early form of the work in Section 4.7.1 appeared in *Proceedings IAPR Conference on Pattern Recognition 1998* [ST98]. That paper outlined a structure preserving denoising algorithm which decided between $k = 1$ and $k = 2$, with $k = 2$ being chosen if the *range* of the pixels in the 3×3 neighbourhood was greater than 6σ. The noise level σ was estimated as an average of the local variances of homogeneous regions, with homogeneity being declared if the Sobel edge strength [Sob70] was less than 16. The $k = 2$ model was thresholded using the “dynamic mean” [CC94], and not iteratively determined. The concept of overlapping averaging was also introduced, albeit only with equal weights.

Chapter 5

Information Theoretic Local Segmentation

5.1 Introduction

In Chapter 4, the principle of local segmentation was introduced and applied to the problem of image denoising. Initially, two candidate segmentations were considered for modeling the local neighbourhood of each pixel being processed. The first candidate assumed the local region was homogeneous. The second candidate was generated by thresholding the pixels into two classes. A simple model selection criterion was then used to decide which of the two candidates best modeled the underlying image. The selection criterion was based on the level of noise in the image. It tried to choose a two-segment model only if the segment means were separated enough. Later, the concept was extended to allow for more than two candidate segmentations by using the k-means algorithm for multi-level thresholding. Up to M candidates were considered, where M was the number of pixels in the window.

Surprisingly, allowing the model selection process to consider models with more than two segments did not improve results significantly. There could be two main reasons for this. Firstly, it may be true, that, on a small scale, one is unlikely to encounter junctions between more than two segments. Secondly, the use of thresholding to group small amounts of data into a relatively large number of clusters may be inappropriate. Thresholding relies solely on pixel intensity for guiding segment membership. As the image noise level increases, spatial information may be required for successful segmentation. This could be used to better distinguish whether pixel variation is due to noise, or to underlying image structure.

Local segmentation is fundamentally concerned with choosing the best underlying local image model, from a set of candidate models. This is an example of the quite general problem of inductive inference, or model selection. Over the last few decades *Bayesian* techniques have proved to be the most reliable for model selection [RR98, BS94]. In this chapter, Wallace's *Minimum Message Length (MML)* criterion [WB68, WF87, WD00] for model selection is used to objectively compare different local segmentations. MML is an information theoretic criterion related to traditional Bayesianism, but extended to function better with models containing continuous parameters. MML evaluates models using their *message length*. A message is an efficient, unambiguous joint encoding of a model and the data. The model associated with the message of shortest overall length is deemed the best model for the data.

Image denoising will again be used as a test bed for exploring the potential of an information theoretic approach to local segmentation. It will be shown that the MML model selection criterion leads to better RMSE performance, especially at higher noise levels, by removing the minimum contrast difference that FUELS requires. Instead of being treated as a post-processing step, the "do no harm" heuristic will be shown to fit naturally into MML's information theoretic framework. A much larger set of candidate segmentations are considered, allowing spatial information to be exploited. With FUELS it unclear how to compare two different $k = 2$ models, but the MML criterion makes this straightforward. The MML denoising algorithm also learns all of its required parameters from the noisy image itself.

5.2 Statistical model selection

Imagine we have some set of measurements, D, from the real world, and a set of models, $\Theta = \{\theta_1, \theta_2, ...\}$, with which we attempt to explain these measurements. How do we assess the quality of and choose the best model $\hat{\theta}$ from Θ? Consider the familiar problem of polynomial regression. The measurement data consists of N coordinate pairs, $(x_i, y_i) \cdots (x_N, y_N)$. Equation 5.1 describes an n^{th} order polynomial regression model, with n a positive integer.

$$f_n(x) = \sum_{i=0}^{n} a_i x^i = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n \tag{5.1}$$

A particular model, θ, is fully determined by its polynomial order n, and its corresponding polynomial coefficients, a_0 to a_n. Because the coefficients are continuous variables, there is an infinite number of possible models, irrespective of the value of n. Because it is impossible to evaluate an unbounded number of models, the model space must be reduced intelligently.

The residuals, $r_1 \ldots r_N$, are the differences between the regression model and the actual measurements. Typically, the residuals are assumed normally distributed, namely $r_i \sim \mathcal{N}(0, \sigma^2)$. For a given model order n, the least-sum-of-least-squares algorithm [Nie94] can be used to estimate the optimal polynomial coefficients by minimizing $\sum_i r_i^2$. This reduces the set of models, Θ, to the family of least-squares polynomials parameterized by n. A model selection criterion only needs to choose the most appropriate value of n for the data.

Figure 5.1 gives an example of 11 data pairs. These pairs exhibit some positive correlation. Figure 5.2 shows three polynomial fits to these data pairs: $n = 0$ constant, $n = 1$ linear and $n = 2$ quadratic. The constant model does not fit the data very well. The linear and quadratic models are more difficult to judge. The linear model has the virtue of being simpler, but the quadratic appears to fit the end points more closely. An *objective* criterion is required to choose the best model automatically.

5.2.1 Maximum likelihood

The maximum likelihood (ML) approach is an objective model selection criterion. ML chooses the model which maximizes the probability of the data given the model — the so-called *likelihood* of the data. Equation 5.2 describes the ML criterion.

$$\hat{\theta}_{ML} = \underset{\theta \in \Theta}{\text{argmax}} \quad \Pr(D|\theta) \tag{5.2}$$

For the regression example, a model consists of the value of n, and $n + 1$ polynomial coefficients. The data consists of N y-coordinates in the form of residuals from the fitted model, namely $r_i = y_i - f_n(x_i)$. As mentioned in Section 5.2, the residuals are assumed normally

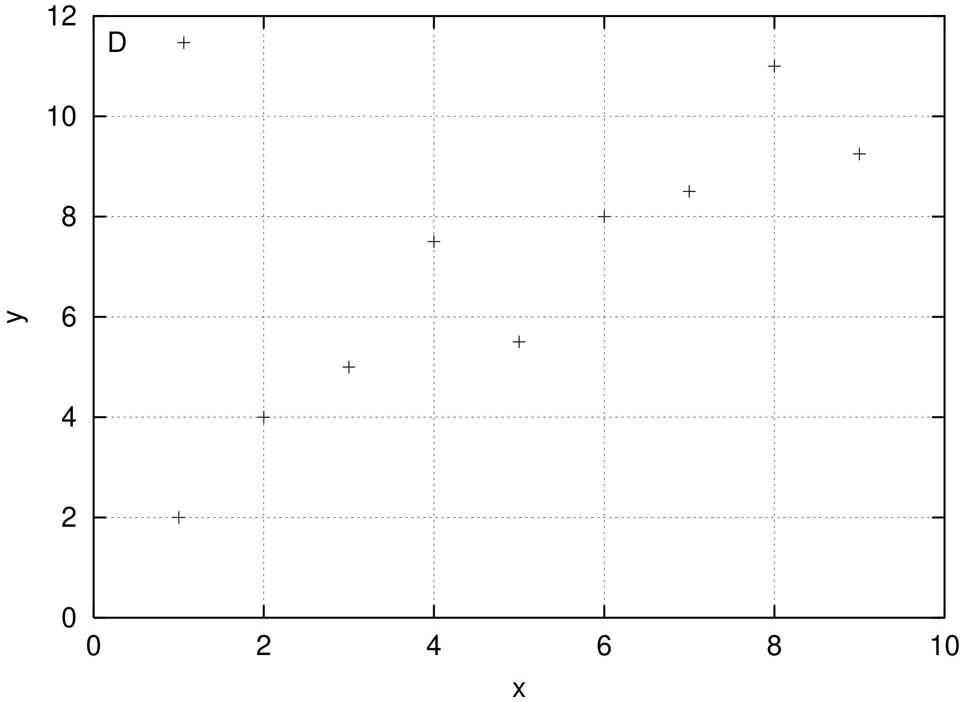

Figure 5.1: Scatter plot of 11 data points.

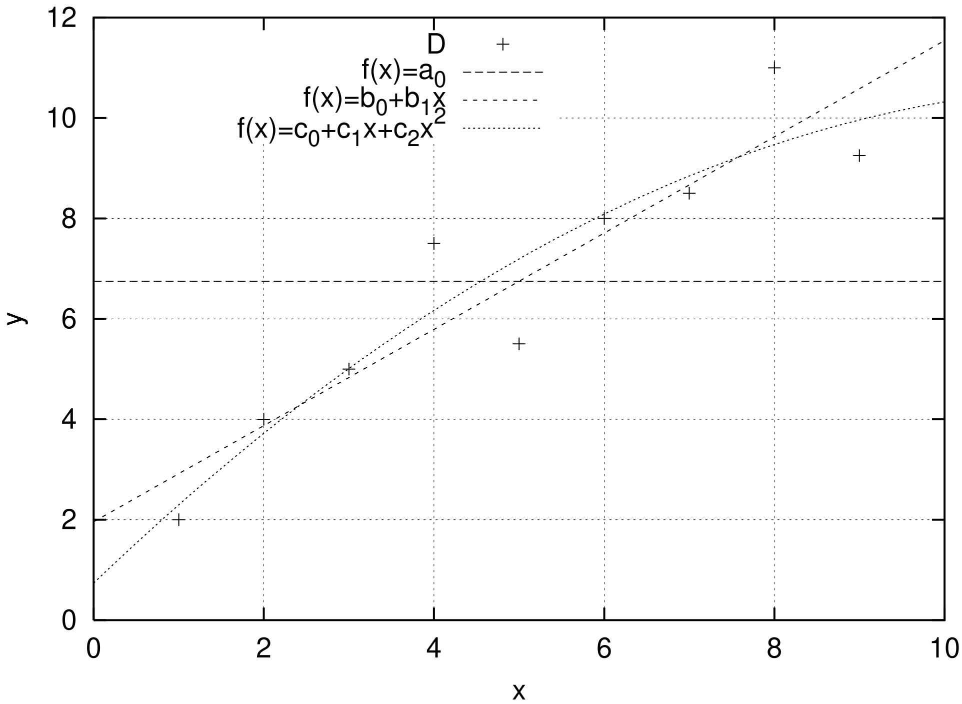

Figure 5.2: Three different polynomial fits to the 11 data points.

distributed. The variance of this distribution is calculated from the residuals [Nie94], and is assumed true and known, just like the x coordinates. The likelihood then is equal to the joint probability of the residuals, given the model. Traditionally the residuals are assumed independent, hence the joint probability becomes a simple product, shown in Equation 5.3.

$$\Pr(D|\theta) = \Pr(r_1, r_2, \ldots, r_N | f_n(\cdot), \sigma) = \prod_{i=1}^{N} \Pr(r_i | f_n(\cdot), \sigma) \tag{5.3}$$

Which model would the ML approach choose from the three in Figure 5.2? The higher the polynomial order, the better the fit to the data, and the higher the likelihood. Thus ML would choose the *most complex* model from Θ, which for Figure 5.2 is the quadratic polynomial ($n = 2$). Imagine a polynomial with the same number of coefficients as there were data points. This curve would pass through each and every point exactly, causing all residuals to vanish. In this situation the likelihood is unity, and ML would consider it the best fit for the data. This is despite the fact that the fitted polynomial would probably be contorted and non-intuitive, especially for values of x outside the range of the original data. Moreover, if polynomials with degree $n \geq N$ are allowed, the ML fit is not even unique. ML has a strong tendency to over-fit, as the complexity of the model is not taken into consideration.

5.2.2 Interchangeability of probability and code length

Often it is inconvenient to work with probabilities. By taking the negative logarithm of a probability, we obtain what is called a *code length* in Shannon's information theory [Sha48]. The MML criterion for model selection introduced in Section 5.1 constructs codes, or *messages*, for comparing different models. Code lengths are measured in *bits* when the logarithm is to base 2, and *nits* when natural logarithms are used. Where probabilities are multiplied, code lengths are added, and where probabilities are maximized, code lengths are minimized. Probabilities and code lengths are interchangeable using the rules in Table 5.1.

For example, reconsider the maximum likelihood formula in Equation 5.2. When phrased in terms of code lengths, ML chooses the model which minimizes the code length of the data, called the *negative log-likelihood*. This arrangement is shown in Equations 5.4 and 5.5.

Probability	$\iff$	Code length (bits)
$\Pr(x)$		$\mathcal{L}(x)$
$2^{-\mathcal{L}(x)}$		$-\log_2 \Pr(x)$
maximize		minimize
$\Pr(x)\Pr(y)$		$\mathcal{L}(x)+\mathcal{L}(y)$
$\Pr(x) \in (0,1)$		$\mathcal{L}(x) \in (0,\infty)$

Table 5.1: Interchangeable use of probability and code length

$$\begin{aligned} \hat{\theta}_{ML} &= \underset{\theta \in \Theta}{\operatorname{argmin}} \quad -\log \Pr(D|\theta) && (5.4) \\ &= \underset{\theta \in \Theta}{\operatorname{argmin}} \quad \mathcal{L}(D|\theta) && (5.5) \end{aligned}$$

5.2.3 Penalized likelihood

The maximum likelihood approach to model selection tends to over-fit the data, as it does not consider the model complexity. Human minds and science tend to prefer the simplest explanation of a data set which still has some useful predictive power [Tho18]. To incorporate this desire, many techniques have been devised which *penalize* the likelihood function to discourage over-fitting. Penalized likelihood criteria typically have the form of Equation 5.6, where $PT(\cdot)$ is a penalty term. The penalty term may depend on features of the data and model. For example, the amount of data, or the number of model parameters.

$$\hat{\theta}_{PML} = \underset{\theta \in \Theta}{\operatorname{argmin}} \quad \mathcal{L}(D|\theta) + PT(\theta, D) \tag{5.6}$$

Over the years many penalized likelihood techniques have been developed for particular domains [Kuh00]. The most enduring approach has been the Akaike Information Criterion, or AIC [Aka92, Boz83]. AIC is approximately equivalent to minimizing the expected Kullback-Leibler distance [KL51, DDF$^+$98], and has been applied to various problems such as Markov chains and segmenting time series data. Penalized likelihoods methods have been mostly superseded by other techniques, discussed later.

The local segmentation model selection criteria of Chapter 4 may be interpreted as a penalized likelihood technique. The k-means clustering algorithm attempts to find the maximum likelihood clustering for the pixel data, under the assumption of Gaussian clusters. The mean separation criterion may be considered the penalty term. When the means are not separated enough, the penalty term becomes infinite, and hence that model can not be chosen.

5.2.4 Bayesianism

Equation 5.7 shows Bayes' formula for the probability of a model given the data, usually referred to as the *posterior probability* of the model. As $\Pr(D)$ is constant with respect to different models for the same data, it may be disregarded. The resulting posterior probability is proportional to the product of the model probability and the probability of the data with respect to that model, the latter being the familiar likelihood.

$$\Pr(\theta|D) = \frac{\Pr(\theta \;\&\; D)}{\Pr(D)} = \frac{\Pr(\theta) \times \Pr(D|\theta)}{\Pr(D)} \propto \Pr(\theta) \times \Pr(D|\theta) \tag{5.7}$$

The model probability, $\Pr(\theta)$, is usually referred to as the *prior probability* of the model. The prior defines a probability distribution over all possible models and their parameters, and must be supplied by the user. It allows the incorporation of relevant background knowledge *before* the data is observed. Ignorance can be expressed using a *uniform prior*, which gives equal probability to all models. Thus the prior term may be ignored, and using Bayes' rule becomes equivalent to using the maximum-likelihood method of Section 5.2.1.

The resulting posterior is a probability distribution over all models. If any of the model parameters are continuous, the posterior will be a density, and any specific model will have zero posterior probability attached to it [BO95]. Some Bayesians will insist that, for inference problems, the posterior distribution is a sufficient final result — they believe it is unsound to select a specific model from it [OB94]. However, some applications require either a single "best" model, or at most a few likely ones.

Collapsing the posterior distribution results in a single best model, called a *point estimate*. Common point estimates are the mode, mean, or median of the posterior distribution, each corresponding to different loss functions [OB94]. The mode of the posterior is most commonly used, and is called the *maximum a posteriori* (MAP) estimate. The MAP estimate picks the model which makes the data most likely, and is shown in Equation 5.8.

$$\hat{\theta}_{MAP} = \underset{\theta \in \Theta}{\text{argmax}} \quad \Pr(\theta)\Pr(D|\theta) \tag{5.8}$$

The main problem with the MAP estimate is that, although it picks the peak of the posterior distribution, this peak may not have much probability mass associated with it. An alternate lower peak may have much more posterior probability mass surrounding it, and could be a better overall estimate. Figure 5.3 gives an example of this situation for the case of a model with a single continuous parameter. When the model consists only of discrete parameters, the MAP estimate does not suffer from the same difficulties, because neighbouring (discrete) points in model space are not necessarily related.

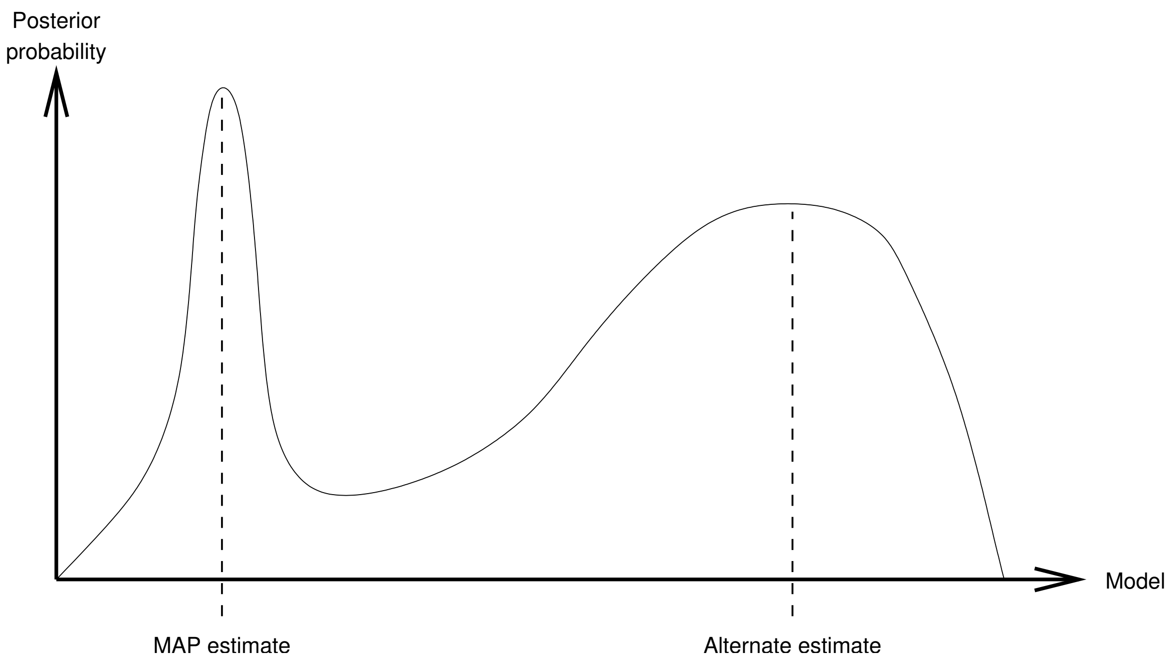

Figure 5.3: The Bayesian MAP estimate may not always coincide with the largest peaked zone of probability mass.

5.2.5 MDL : Minimum Description Length

Model selection has moved toward information-based interpretations of Bayes's rule. Rissanen's original Minimum Description Length (MDL) criterion [Ris78] is given in Equation 5.9, where $\mathcal{L}(D|\theta)$ is the negative log-likelihood of the data, N_θ is the number of model parameters, and N_D is the number of data items. For the regression example in Section 5.2, $N_\theta = n+1$, the number of polynomial coefficients, and $N_D = N$, the number of data points.

$$\hat{\theta}_{MDL} = \underset{\theta \in \Theta}{\text{argmin}} \quad \log_2^* N_\theta + \frac{N_\theta}{2} \log_2 N_D + \mathcal{L}(D|\theta) \tag{5.9}$$

The $\log_2^* x$ function is pronounced "log star". It estimates the length in bits for encoding an arbitrary integer, $x \in [1, \infty]$. Its computation is outlined in Equation 5.10, where 2.865 is a normalization constant which ensures decodability. The first $\log$ term encodes the value of the integer. To decode the value, the decoder needs to know how many bits were used to encode it. This is provided by the double-$\log$ term. The triple-$\log$ term encodes the number of bits used for the double-$\log$ term, and so on. Eventually a point is reached where encoder and decoder have mutual understanding. The log star term is usually small compared to the overall code length, so is sometimes ignored when applying Equation 5.9 [BO95].

$$\log_2^* x \approx 2.865 + \log_2 x + \log_2(\log_2 x) + \log_2(\log_2(\log_2 x)) + \cdots \quad \text{bits} \tag{5.10}$$

This MDL criterion may be interpreted as a penalized likelihood or Bayesian method, with the "model order" costing $\log_2^* N_\theta$ bits, and each model parameter costing $\log_2 \sqrt{N_D}$ bits. The model parameters are transmitted more accurately only if justified by more data. MDL has been developed over the years [Ris87, Ris00], but each formulation has one or more of the following drawbacks: not being invariant under transformation of the data or model parameters, poor performance with small amounts of data, an inability to specify useful prior knowledge about the data, and a focus on selecting a model class rather than a particular model [WF87, BO95]. In local segmentation, a model class would be $k = 1$ or $k = 2$, but the model class does not specify any particular parameter values, such as which segment each pixel belongs to, or the segment means.

5.2.6 MML : Minimum Message Length

Wallace's MML criterion [WB68, BW72, WF87, WD00] is an information theoretic criterion for model selection whose origins even pre-date AIC and MDL. Good introductions to MML are found in Oliver *et al* [OH94, OB94, BO95]. MML proposes that, for each model being considered, a *two part message* containing a description of the model and the data given that model be constructed. The message having the shortest overall message length is considered to contain the best model for the data. This is shown in Equation 5.11.

$$\hat{\theta}_{MML} = \underset{\theta \in \Theta}{\operatorname{argmin}} \quad \mathcal{L}(\theta) + \mathcal{L}(D|\theta) \tag{5.11}$$

Each message must be constructed in such a way that it can be decoded by a receiver given only the same prior knowledge. That is, the message must be a *lossless encoding of the data*. If the data and model parameters are all *discrete*, MML is equivalent to the Bayesian MAP estimate in Equation 5.8, due to the equivalence of probability and code length.

The fundamental difference to Bayes' rule occurs when the model consists of one or more *continuous* parameters. To transmit a continuous value with a finite number of bits, it must be *quantized*. MML provides a framework to optimally quantize the prior density. The quantization may be interpreted as modifying the shape of the prior density such that peaked regions with little probability mass are flattened out, and high mass regions are somewhat boosted [Far99]. The posterior mode in the MML situation may therefore be different to the Bayesian MAP estimate when the models have continuous parameters.

Quantizing model parameters introduces two complications. Firstly, the decoder does not usually know which quantization levels were used. Secondly, data usually needs to be encoded with respect to *specific* parameter values, not parameter ranges. The first issue is resolved by including some bits for encoding the quantization bin sizes, the so-called *accuracy of parameter values* (AOPVs) [OH94]. The second problem is handled by computing an *expected message length* rather than an absolute message length. The expectation is computed by averaging all the message lengths that would result from using all possible parameter values within the quantization regions.

MML is essentially a Bayesian method at heart. By converting Bayes' formula in Equation 5.7 to code lengths, the MML formula in Equation 5.11 is arrived at. MML and Bayesianism both advocate the incorporation of prior beliefs via $\Pr(\theta)$. This is unlike MDL, which attempts to do away with priors altogether [BO95]. An inference produced by MML has the advantage of being invariant under linear and non-linear transforms of the parameter space. MML also works well for small amounts of data [BO95, WD00]. When the amount of data is very large, the length of the second part of the message dominates, and MML gracefully reverts to maximum likelihood, just as Bayes' rule and MDL do.

5.3 Describing segmentations

Segmenting pixels into a fixed number of classes results in two sets of information: class labels denoting which class each pixel belongs to, and one or more parameters describing the properties of each class . Consider segmentation of the noisy pixels, $\mathbf{p}'$, in Figure 5.4. If a threshold of 60 is used, the first class has $m_1 = 6$ members averaging $\mu_1 = 22$, and the second class $m_2 = 3$ members averaging $\mu_2 = 84$. If the classes are numbered from 1, then $\mathbf{c}$ contains the class labels for each pixel. The resulting local approximation, $\hat{\mathbf{p}}$, is computed by replacing each pixel with the mean of the segment it belongs to.

$$\mathbf{p}' = \begin{array}{|c|c|c|}\hline 20 & 22 & 84 \\ \hline 24 & 21 & 81 \\ \hline 23 & 22 & 87 \\ \hline \end{array} \qquad \begin{array}{ll} \mu_1 = 22 & m_1 = 6 \\ \mu_2 = 84 & m_2 = 3 \end{array} \qquad \mathbf{c} = \begin{array}{|c|c|c|}\hline 1 & 1 & 2 \\ \hline 1 & 1 & 2 \\ \hline 1 & 1 & 2 \\ \hline \end{array} \qquad \hat{\mathbf{p}} = \begin{array}{|c|c|c|}\hline 22 & 22 & 84 \\ \hline 22 & 22 & 84 \\ \hline 22 & 22 & 84 \\ \hline \end{array}$$

Figure 5.4: Local segmentation of a 3×3 window into two classes.

The *segment map*, $\mathbf{c}$, consists of M labels — one for each pixel taking part in the segmentation. If K is the maximum number of segments allowed, then each label could take on a value from 1 to K. By imposing a canonical ordering on the elements, the segment map, $\mathbf{c}$, can be treated as a 1-D vector, with elements indexed as c_i. Figure 5.5 illustrates the use of a canonical raster ordering for labels within a segment map.

$$\mathbf{c} = \begin{array}{|c|c|c|}\hline c_1 & c_2 & c_3 \\ \hline c_4 & c_5 & c_6 \\ \hline c_7 & c_8 & c_9 \\ \hline \end{array} \qquad c_i \in \{1, 2, \ldots, K\}$$

Figure 5.5: Raster order convention for labels within a segment map.

5.3.1 Possible segmentations

If there are M pixels, each of which may belong to one of K segments, then there are K^M distinct ways to assign pixels to segments. That is, there are K^M possible values for $\mathbf{c}$. Consider the local segmentation of a 2×2 window into 1 or 2 segments. In this case $M = 4$ and $K = 2$, so there are $2^4 = 16$ possible segment maps. Figure 5.6 shows all of them, with light and dark squares denoting $c_i = 1$ and $c_i = 2$ respectively.

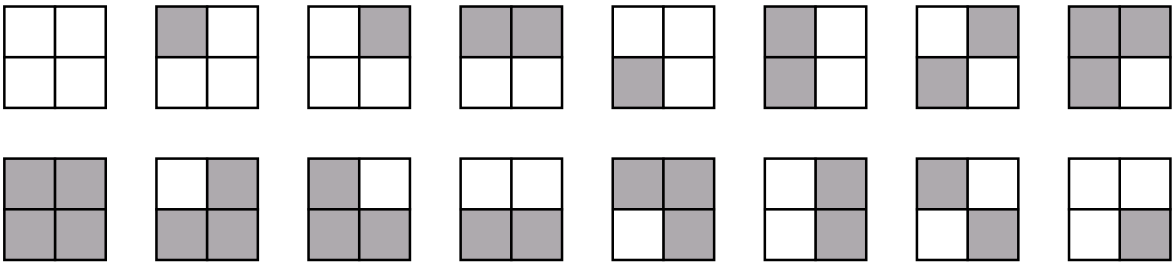

Figure 5.6: The 16 possible 2×2 binary segment maps.

5.3.2 Canonical segmentations

Figure 5.6 has the feature that the second row is an inversion of the first row, the role of labels having been interchanged. The labels themselves are not crucial — what is important is the uniqueness of the segment map pattern. Figure 5.7 shows the 8 *canonical* segmentations after the inverted duplicates are removed. When $K = 2$, the number of canonical segmentations is always 2^{M-1}. Note also that the bottom right hand pixel is always in the same "dark" segment. Because the labels are interchangeable, one degree of freedom is removed, so only $M - 1$ labels need to be determined relative to a predetermined one.

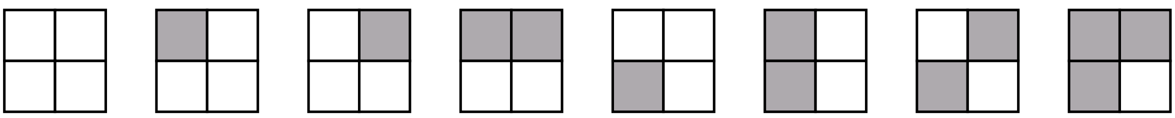

Figure 5.7: The 8 canonical 2×2 binary segment maps.

Listing 5.1 describes a recursive algorithm which, given M and K, generates only canonical segmentations. The algorithm fixes the class label for the first pixel as "1". It then does a

```
// Global constants
// M : number of pixels
// K : maximum class label ie. number of segments

// Parameters to 'generate' function
// [c] : current segment map vector so far
// m : number of labels in [c] so far
// k : largest label used so far

// Return parameter of the 'generate' function
// list : list of canonical segmentations ([c1],[c2],[c3],...)

proc generate( [c], m, k, list )
  if (m > M) then
    list.insert( [c] )   // we are done with this vector
  else
    for i := 1 to min(k+1, K)
      generate( [c].prepend(i), m+1, max(i,k), list )
    endfor
  endif
endproc

proc main
  list = ()                        // empty list
  generate( [1], 1, 1, list )      // generate all canonical K-class M-tuples
endproc
```

Listing 5.1: Algorithm to generate canonical segmentations only.

depth first traversal to determine the next label in **c**. The only labels considered at each step are those up to and including the largest label used so far, plus the next one along, but always restricted to the range $[1, K]$. This ensures that no duplicates are generated.

5.3.3 Valid segmentations

For the binary segment maps generated so far, no specific requirements on spatial structure have been imposed. One may wish to insist that the labels for a given segment be spatially connected. In Section 2.6 the concept of pixel connectedness was introduced. For the example of 2×2 windows, one could insist that the pixels within a segment be 4-connected. Figure 5.8 shows all the *valid* binary segment maps having, at most, two sets of labels.

Valid segmentations may be generated by generating all canonical segmentations with List-

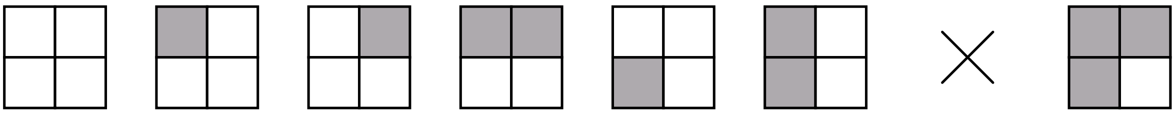

Figure 5.8: The 7 valid 4-connected $2{\times}2$ binary segment maps.

ing 5.1, and removing those which fail a spatial-connectedness test. The algorithm for this test is relatively simple, and is not included in this discussion. Table 5.2 shows that as M and K increase, the proportion of canonical segmentations among all possible segmentations decreases, as does the proportion of valid segmentations within the canonical set.

Pixels in window (M)	Max. number segments (K)	Number of segment maps		
		Possible	Canonical	Valid
4	2	16	8	7
9	2	512	256	133
9	3	19,683	3,281	915
9	4	262,144	11,051	2,411

Table 5.2: Number of possible segmentations for various window configurations.

5.3.4 Segmentation parameters

Given the noisy data, $\mathbf{p}'$, a segment map, $\mathbf{c}$, and K, it is a simple matter to compute the segment parameters. For the case of additive zero-mean Gaussian noise, the mean is optimal, but this could be modified for different types of noise. Figure 5.9 gives a sample $3{\times}3$ block of pixels and two candidate segment maps: the homogeneous $\mathbf{c}_1$, and the heterogeneous $\mathbf{c}_2$.

$$\mathbf{p}' = \begin{array}{|c|c|c|}\hline 20 & 22 & 84 \\ \hline 24 & 21 & 81 \\ \hline 23 & 22 & 87 \\ \hline \end{array} \qquad \mathbf{c}_1 = \begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array} \qquad \mathbf{c}_2 = \begin{array}{|c|c|c|}\hline 1 & 1 & 2 \\ \hline 1 & 1 & 2 \\ \hline 1 & 1 & 2 \\ \hline \end{array} \qquad K = 2$$

Figure 5.9: Noisy pixels and two candidate segment maps.

The number of segments The number of segments is equal to the number of different labels used in the segment map, $\mathbf{c}$, and shall be referred to as $K(\mathbf{c})$. Because $\mathbf{c}_1$ is homogeneous, $K(\mathbf{c}_1) = 1$. For the heterogeneous segment map, $K(\mathbf{c}_2) = 2$.

The segment populations The population, m_i, for segment i is equal to the frequency of pixels in the segment map having label i. Segment map $\mathbf{c}_1$ only has one population, so $m_1 = M = 9$, whereas $\mathbf{c}_2$ has two segments, so $m_1 = 6$ and $m_2 = 3$.

Representative values for each segment In this thesis, the noise is assumed to be additive zero-mean Gaussian, for which the arithmetic mean is an efficient estimator of the true population mean. Each segment has one mean, calculated using Equation 5.12. For $\mathbf{c}_1$, $\mu = 42.7$, while for $\mathbf{c}_2$, $\mu_1 = 22$ and $\mu_2 = 84$.

$$\mu_j = \frac{1}{m_j} \sum_{\forall i \; c_i = j} p'_i \tag{5.12}$$

5.4 Using MML for local segmentation

The FUELS algorithm developed in Chapter 4 has two main characteristics which could be improved. The first is that only two candidate clusterings are considered for each local region. This limitation will be tackled later in Section 5.14. The second drawback is that FUELS' model selection criterion depends solely on the numerical intensity difference between segment means. Specifically, they must be at least $C\sigma$ units apart, where σ is the image noise level, and $C = 3$. Thus FUELS is unable to distinguish two segments having a contrast difference below 3σ grey levels.

A different model selection criterion is required to handle adjacent segments of low contrast. In Section 5.2.6, the MML approach to model selection was described. MML evaluates a model using the length of a two part message which describes jointly the model and data. The first part of the message describes the model and its parameters. The second part of the message describes the data with respect to the prefixed model. Each message must be an unambiguous and efficient encoding of the data it describes. The model associated with the message of overall shortest length, measured in bits, is the preferred model for the data.

5.4.1 A message format

Figure 5.10 proposes a message format for encoding noisy pixels under a local segmentation model. The first part of the message is the model, θ, which contains all the information required to construct a local approximation to the underlying pixel values. The segment map, $\mathbf{c}$, states which segment each pixel belongs to. The segment map is followed by the means for each segment. The number of means depends solely on the number of unique segments in $\mathbf{c}$. This value is denoted $K(\mathbf{c})$, and may be derived from the segment map, $\mathbf{c}$.

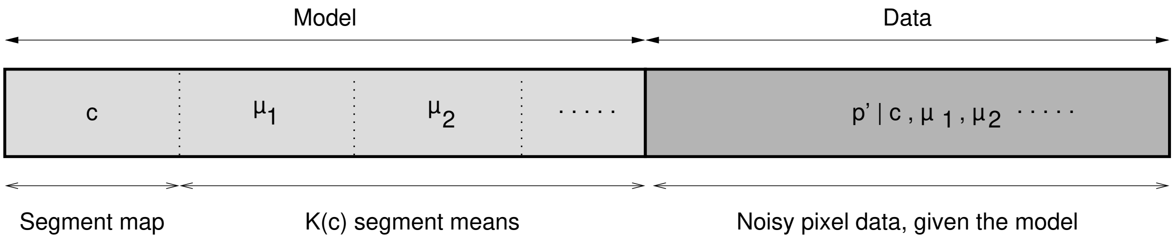

Figure 5.10: The proposed local segmentation message format.

Equation 5.13 states that the message length of the model part is equal to the sum of the lengths of each component it contains. This is only true if each component of the message is assumed independent. The details of encoding each component depend on our prior expectations for the pixel values. This will be discussed soon in Section 5.4.2.

$$\mathcal{L}(\theta) = \mathcal{L}(\mathbf{c}) + \sum_{i=1}^{K(\mathbf{c})} \mathcal{L}(\mu_i \mid \mathbf{c}) \tag{5.13}$$

The second part of the message encodes the data with respect to the model preceding it. In our case the data consists of noisy pixels, $p'_1 \cdots p'_M$. Pixels from a greyscale image are usually quantized to one of Z integers. If encoded independently under the assumption of uniformity, each pixel value requires at most $\log_2 Z$ bits, but relative to a model (segment map and means), a more efficient encoding is possible.

It was shown in Section 4.2.6 that a quantized pixel value, z, could be considered to have had a true value anywhere in the range $[z - 0.5, z + 0.5)$. In Chapter 4, it was assumed that clean images were corrupted by additive Gaussian noise $\sim \mathcal{N}(0, \sigma^2)$, with σ assumed constant across the image. Thus it is assumed that pixels within a segment are normally distributed around the segment mean. Equation 5.14 calculates the length of the data component (the negative log-likelihood) given the model parameters.

$$\mathcal{L}(D \mid \theta) = -\sum_{j=1}^{M} \log \int_{x=p'_j-0.5}^{x=p'_j+0.5} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu_{\mathbf{c_j}})^2}{2\sigma^2}} dx \quad \text{bits} \tag{5.14}$$

5.4.2 A uniform prior

Our prior expectation regarding the types of local segmentation structure present in an image directly impacts on the encoding of the model parameters. Like FUELS, the model domain is restricted to 3×3 windows ($M = 9$) consisting of a maximum of two segments ($K = 2$). In Section 5.3.2, it was shown that there are only 2^{M-1} canonical binary segment maps. For the moment, let us assume that each segment map is equally likely. Equation 5.15 shows that $\mathbf{c}$ may therefore be encoded with $M - 1$ bits. Once $\mathbf{c}$ is known, the number of segments, $K(\mathbf{c})$, and the number of pixels belonging to each segment are implicitly known.

$$\mathcal{L}(\mathbf{c}) = \log_2 2^{M-1} = M - 1 \quad \text{bits} \tag{5.15}$$

How many bits should be used for each segment mean? An optimal number will be discussed in Section 5.7. For the moment a useful approximation is considered. We already know that a pixel value can be encoded with at most $\log_2 Z$ bits. We also know that a denoised pixel is usually rounded back to an integer at the final stage of processing. Thus a reasonable number of bits for a segment mean could also be $\log_2 Z$ bits. Equation 5.16 gives an expression for the overall length of the model component of the message under this assumption. Note that because $M - 1$ is constant for all models, it could be ignored.

$$\mathcal{L}(\theta) = M - 1 + K(\mathbf{c}) \log_2 Z \quad \text{bits} \tag{5.16}$$

5.4.3 A worked example

In this section an example of the MML model selection criterion will be given. Figure 5.11 shows the noisy 3×3 window which will be used. The global noise standard deviation is

$$\mathbf{p}' = \begin{array}{|c|c|c|}\hline 15 & 14 & 15 \\ \hline 13 & 17 & 12 \\ \hline 16 & 11 & 10 \\ \hline \end{array} \qquad \sigma = 1.5 \ \text{(assumed)}$$

Figure 5.11: A 3×3 window of noisy pixels.

assumed to be 1.5. Two candidate models will be considered, coinciding with those which would have been generated by FUELS.

Figure 5.12 shows the first candidate model. It is the simplest one possible — a homogeneous segment map and a single mean. This model is 16 bits long — 8 bits for the segment map and 8 bits for the average pixel value.

$$\mathbf{c} = \begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array} \qquad \mu = 13.78 \quad m = 9 \qquad \begin{array}{l} \mathcal{L}(\theta) = 8 + 1 \times \log_2 256 = 16 \text{ bits} \\ \mathcal{L}(D \mid \theta) = 24.93 \text{ bits} \\ \mathcal{L}(\theta \ \& \ D) = 40.93 \text{ bits} \end{array}$$

Figure 5.12: Candidate model 1 and its message length calculation.

Figure 5.12 shows the second candidate model. It is the result of iteratively thresholding the noisy pixel values, just as FUELS does. The resulting heterogeneous segment map fits the pixels well. Its model length is 8 bits longer than the first candidate's, as two means must be encoded. However the data encoding length is much shorter. This is due to the pixels being closer in value to their segment means.

$$\mathbf{c} = \begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 1 & 1 & 2 \\ \hline 1 & 2 & 2 \\ \hline \end{array} \qquad \begin{array}{l} \mu_1 = 15 \quad m_1 = 6 \\ \mu_2 = 11 \quad m_2 = 3 \end{array} \qquad \begin{array}{l} \mathcal{L}(\theta) = 8 + 2 \times \log_2 256 = 24 \text{ bits} \\ \mathcal{L}(D \mid \theta) = 14.23 \text{ bits} \\ \mathcal{L}(\theta \ \& \ D) = 38.23 \text{ bits} \end{array}$$

Figure 5.13: Candidate model 2 and its message length calculation.

Under the MML criterion, two candidate models are compared using their overall two-part message lengths. Figure 5.14 illustrates the comparison using the two example models just described. Candidate 2 has the shorter overall message length, and therefore Figure 5.13 is the preferred model for the noisy pixels in Figure 5.11.

Using FUELS' model selection criterion, Candidate 2 would be rejected because its two means are too similar: $|15 - 11| \leq 3 \times 1.5$. Unlike FUELS, the MML criterion is not limited

Candidate 1	θ: 16 bits	D: 24.93 bits
Candidate 2	θ: 24 bits	D: 14.23 bits

Figure 5.14: Comparing two message lengths.

to a fixed minimum contrast difference. In this example, MML made the better inference. It should be noted that the t-test threshold from Equation 4.27, not used by FUELS, is equal to 3.71 for this example. In this case the t-test and MML would concur.

5.4.4 Posterior probability

The overall length of a message in MML is related to the Bayesian posterior probability. The interchangeability of code length and probability ensure this fact. Equation 5.17 describes how to compute the posterior probability of a model, $\mathrm{post}(\hat{\theta})$. The denominator normalizes the probabilities over the set of models Θ considered.

$$\mathrm{post}(\hat{\theta}) = \frac{2^{-\mathcal{L}(\hat{\theta}\ \&\ D)}}{\sum_{\theta \in \Theta} 2^{-\mathcal{L}(\theta\ \&\ D)}} \tag{5.17}$$

For the example in Section 5.4.3, Candidates 1 and 2 have posterior probabilities 0.13 and 0.87 respectively. Here the "best" model is over six times as likely as its nearest competitor. This is good evidence for the existence of two local segments, but not compelling evidence. These posterior probabilities may be interpreted as saying that, in 13% of cases like this, the less likely candidate may actually be the correct one.

A corollary of Equation 5.17 is that the *difference* in message length of any two models may be used to measure the *relative* posterior probability of the two models. For example, Candidate 2 had a message length 2.7 bits shorter than Candidate 1. In terms of posterior probability, the Candidate 2 is $2^{2.7} = 6.5$ times as probable as Candidate 2. This is a useful shorthand for assessing the relative merit of two competing models.

5.5 Application to denoising

The FUELS denoising algorithm generates two candidate local segmentation models. It chooses what it considers to be the better one, and uses the resulting local approximation to estimate the underlying noiseless pixel values. It is a simple matter to replace the existing FUELS model selection criterion with the MML one described in Section 5.4. Let the resulting denoising algorithm be called "Pseudo-MML". This name has been chosen to avoid confusion when MML is applied differently later.

Figure 5.15 compares the RMSE performance of FUELS and Pseudo-MML for denoising `montage`. The true value of σ was given to both algorithms. Overlapping averaging and DNH were disabled, as the primary goal is to observe the effect of changing the model selection criterion. Pseudo-MML only considered the same two candidate models as FUELS.

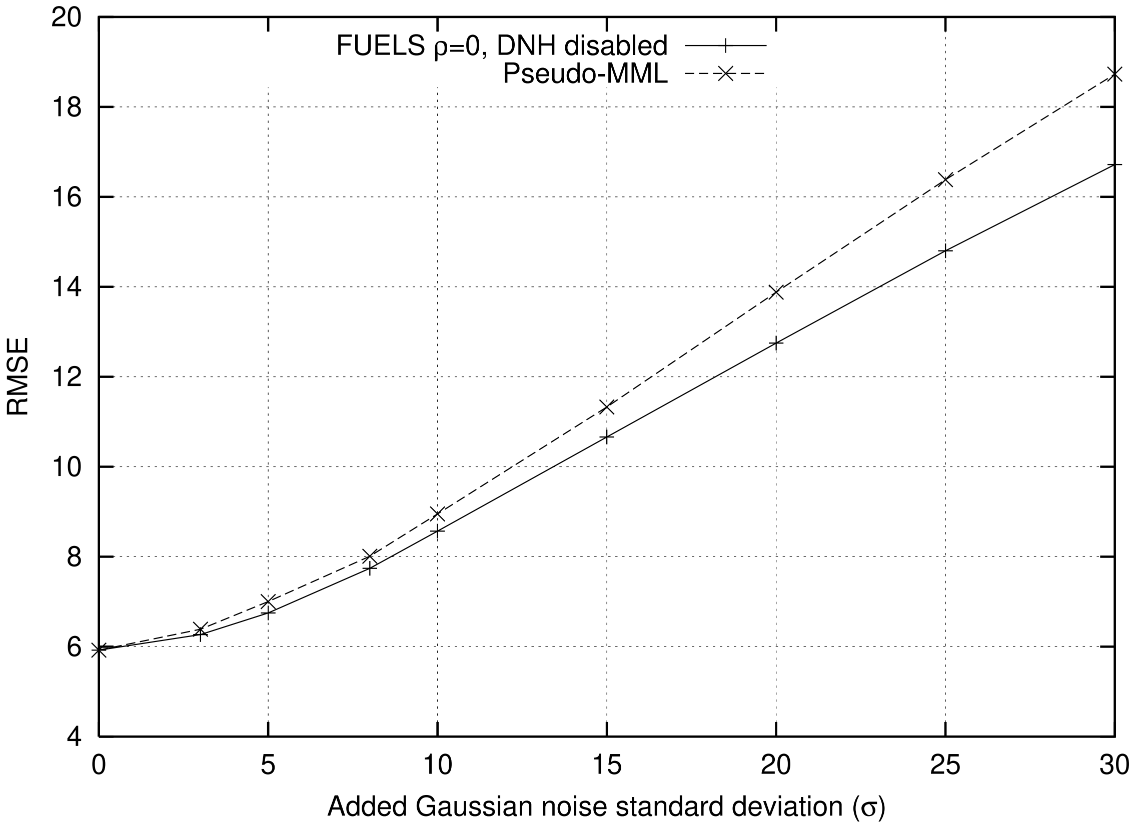

Figure 5.15: RMSE comparison for denoising `montage`, true σ supplied.

The results show FUELS to outperform Pseudo-MML at all noise levels, with the discrepancy increasing along with the noise level. Figure 5.16 shows where the two algorithms differed in their model selections. Pseudo-MML is much more likely to choose $k = 2$ than FUELS. One advantage of this is that it was able to identify the low contrast oval edge around the "h". However, it also chose $k = 2$ in obviously homogeneous regions.

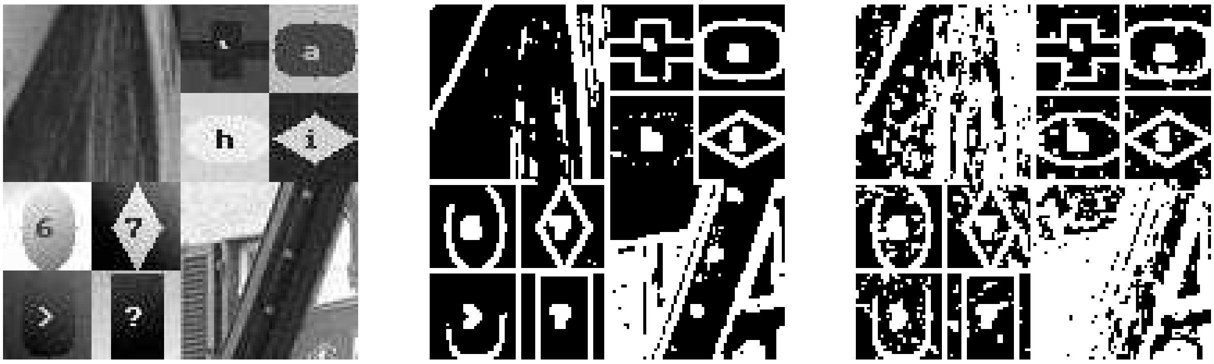

Figure 5.16: (a) noisy $\sigma = 5$ middle section of `montage`; (b) FUELS model selection; (c) Pseudo-MML model selection. Black denotes $k = 1$ and white $k = 2$.

5.6 A better segment map prior

In Section 5.5, it was shown that the Pseudo-MML model selection criterion has a tendency to over-fit. It chose $k = 2$ more often than it should have, resulting in less smoothing and poorer RMSE results compared to FUELS. Although it only considers two models, Pseudo-MML uses the same number of bits for encoding any segment map encountered. This corresponds to a uniform prior probability distribution for $\mathbf{c}$.

The only way to modify Pseudo-MML's behaviour is to change the prior. The analysis of FUELS in Chapter 4 showed that, for typical images, around 50% of 3×3 regions could be considered homogeneous. Thus it would be more realistic to assign a higher prior probability to the single homogeneous segment map, and share the remaining probability equally among the remaining heterogeneous segment maps. This arrangement is shown in Equation 5.18. Note that 0.5 is an arbitrary choice, and its optimization will be considered in later sections.

$$\Pr(\mathbf{c}) = \begin{cases} 0.5 & \text{if} \quad K(\mathbf{c}) = 1 \\ \frac{0.5}{2^{M-1}-1} & \text{if} \quad K(\mathbf{c}) = 2 \end{cases} \tag{5.18}$$

Equation 5.19 gives a new expression for the overall message length using this prior. In terms of encoding $\mathbf{c}$, the new prior corresponds to using 1 bit for homogeneous cases, and nearly 9 bits for each heterogeneous case. Although not implemented as such, this could be interpreted as a two step process. First, a single bit states whether $k = 1$ or $k = 2$. If $k = 1$, nothing else needs to be sent. If $k = 2$, another $\log_2 255 = 7.99$ bits are used to state the which of the $k = 2$ segment maps is relevant. It should be noted that among the various heterogeneous segment maps, some are probably more likely to occur than others. This issue will be discussed later in Section 5.13.

$$\mathcal{L}(\theta \,\&\, D) = -\log_2 \Pr(\mathbf{c}) + K(\mathbf{c}) \log_2 Z + \mathcal{L}(D \mid \theta) \quad \text{bits} \tag{5.19}$$

5.6.1 Results

Figure 5.17 is the same as Figure 5.15 except that there is an extra entry for Pseudo-MML using the non-uniform prior of Equation 5.18. The more suitable prior has resulted in Pseudo-MML's performance reaching that of FUELS.

Figure 5.18 shows where the three algorithms graphed in Figure 5.17 differed in their model selections for the middle section of `montage`. The non-uniform prior improved Pseudo-MML's model order selections, because most of the spurious $k = 2$ decisions are gone. It was still able to discern most of the low contrast oval edge around the "`h`", along with the horizontal boundary below the oval. This example illustrates the importance of choosing a suitable prior. It is especially important when the window is small, because the model part of the message contributes significantly to the overall message length.

5.7 Optimal quantization of segment means

Under the local segmentation framework, each segment mean is used as the location parameter for a Gaussian distribution. The spread parameter for the distributions is assumed common and equal to the global noise standard deviation, σ. The noisy pixel values from each segment are encoded with respect to the appropriate distribution, illustrated in Figure 5.19.

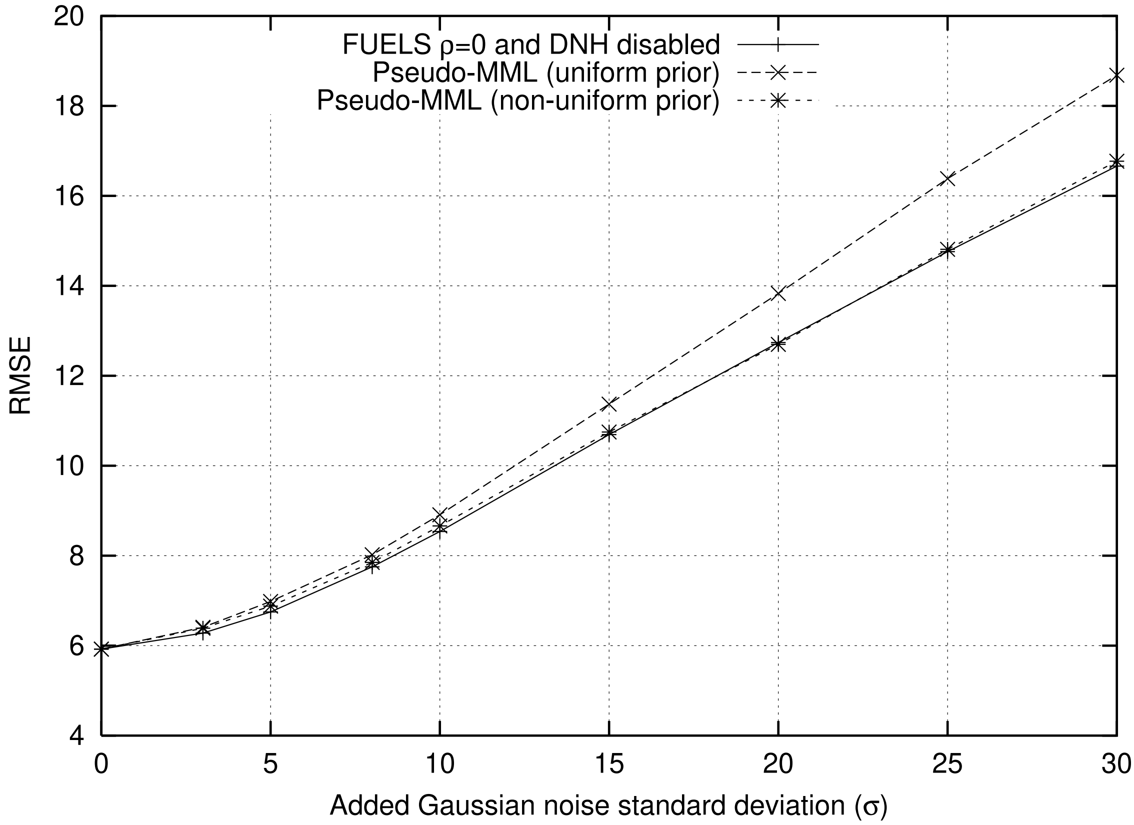

Figure 5.17: Non-uniform prior: RMSE comparison for denoising `montage`, true σ supplied.

Figure 5.18: Model selection comparison when $\sigma = 5$, black denotes $k = 1$ and white $k = 2$: (a) FUELS; (b) Pseudo-MML; (c) Pseudo-MML with a non-uniform prior.

In Section 5.4.2 it was arbitrarily decided that $\log_2 Z$ bits was a reasonable accuracy for encoding a segment mean. If the segment means were encoded using fewer bits, the model part of the two part message would be shorter. Correspondingly, the data part should expand, as the segment means are less precise. There should exist a quantization level for each segment mean which optimally trades off the length decrease of the model part with the

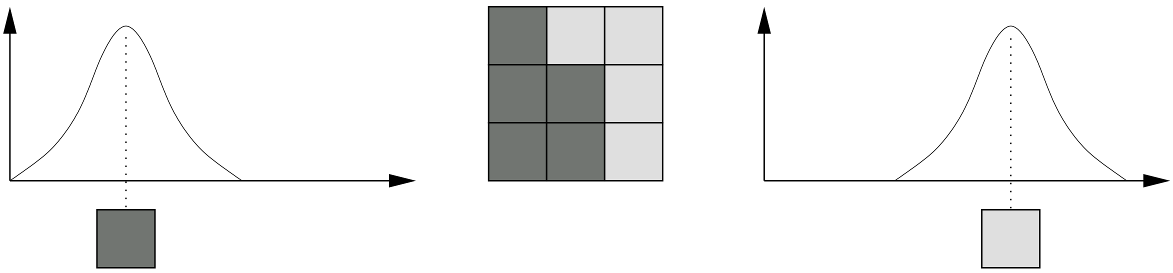

Figure 5.19: Pixels are encoded relative to their segment mean.

length increase of the data part.

Much of the MML literature is concerned with optimally quantizing the parameter space for various standard statistical models [WD00]. Under certain uniformity assumptions [OH94], the optimal bin size, $AOPV_\mu$, for the sample mean of a normal distribution is given in Equation 5.20, where m is the number of data points used to compute the sample mean.

$$AOPV_\mu = \sigma\sqrt{\frac{12}{m}} \tag{5.20}$$

As the variance increases, the accuracy with which we are willing to encode the mean decreases. This is because the distribution has a wider spread, so its exact location is not as important. As the amount of data increases, a higher accuracy is warranted. Equation 5.21 describes the number of bits required to encode an optimally quantized segment mean, where Z is the intensity range of the image and σ is the standard deviation of the noise.

$$\mathcal{L}(\mu) = \log_2 \frac{Z}{AOPV_\mu} = \log_2 \left(\frac{Z}{\sigma}\sqrt{\frac{m}{12}}\right) \quad \text{bits} \tag{5.21}$$

A segment mean encoded with this number of bits is still decodable. Both σ and Z are assumed to be *a priori* known, and m may be determined from $\mathbf{c}$, the first model parameter encoded. Figure 5.20 compares the number of bits needed to encode a segment mean using the two methods described. The original method uses exactly 8 bits, regardless of the noise level. The optimal quantization method uses much fewer bits, and the function grows slowly.

Quantization of segment means causes one difficulty. Each pixel is no longer encoded with respect to a fully accurate segment mean. Rather, a quantized range of possible segment

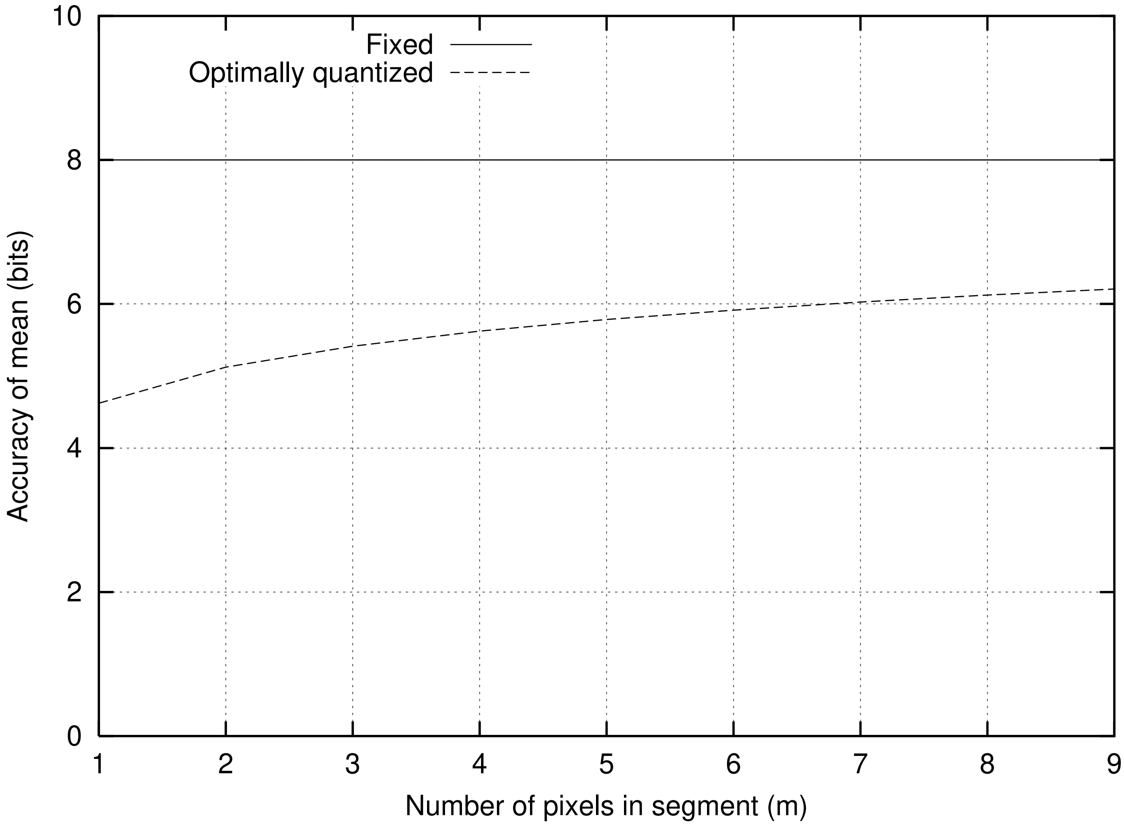

Figure 5.20: Different encodings for a segment means when $\sigma = 3$ and $Z = 256$.

means has been communicated. Which mean from the range should be used? Rather than taking an arbitrary point estimate, such as the midpoint, MML prefers to compute the *expected message length*. That is, the average of all message lengths that would result if every possible point estimate in the quantization range was tested. It is often assumed that each value in the quantization range is equally likely, corresponding to a locally flat prior density in the range. A convenient closed form approximation to the expected message length can sometimes be derived [WF87].

For the purpose of this thesis, a point estimate equal to the sample mean will be used. This simplifies the computation and reduces the running time, without hopefully causing too much variation in message length. The resulting expression for the approximate expected message length is in Equation 5.22.

$$\mathrm{E}[\,\mathcal{L}(\theta \;\&\; D)\,] \approx -\log_2 \Pr(\mathbf{c}) + \sum_{i=1}^{K(\mathbf{c})} \log_2 \left(\frac{Z}{\sigma} \sqrt{\frac{m_i}{12}} \right) + \mathcal{L}(D \mid \theta) \quad \text{bits} \tag{5.22}$$

5.7.1 Results

Figure 5.21 provides RMSE results for denoising `montage`. The algorithm using quantized means is referred to as "MML" to distinguish it from the earlier "Pseudo-MML". Both algorithms use the non-uniform prior of Equation 5.18 to encode the segment map. DNH and overlapping are still disabled, and the true value of σ is provided.

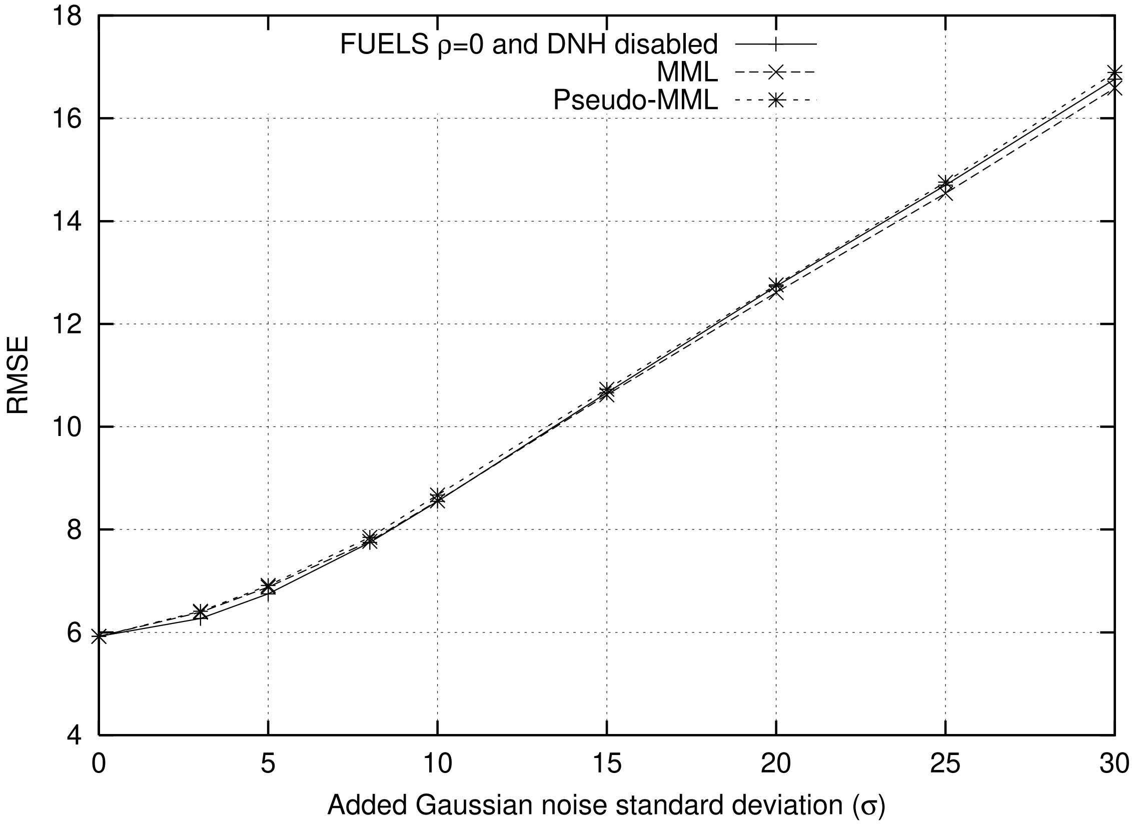

Figure 5.21: RMSE comparison for denoising `montage`, true σ supplied.

The RMSE performance of the three algorithms is mostly indistinguishable. FUELS does marginally better at low noise levels, while MML does slightly better when the image is very noisy. Figure 5.22 compares the value of k Pseudo-MML and MML chose for each pixel when $\sigma = 5$. Their model selections are very similar, so Figure 5.22c highlights the differences. Mid-grey indicates that both algorithms chose the same value of k, white that MML chose a higher order model, and black that MML chose a lower order model.

Interestingly, if there was a difference, it was always that MML chose $k = 2$ compared to Pseudo-MML's $k = 1$. The quantization of segment means has reduced the cost of $k = 2$

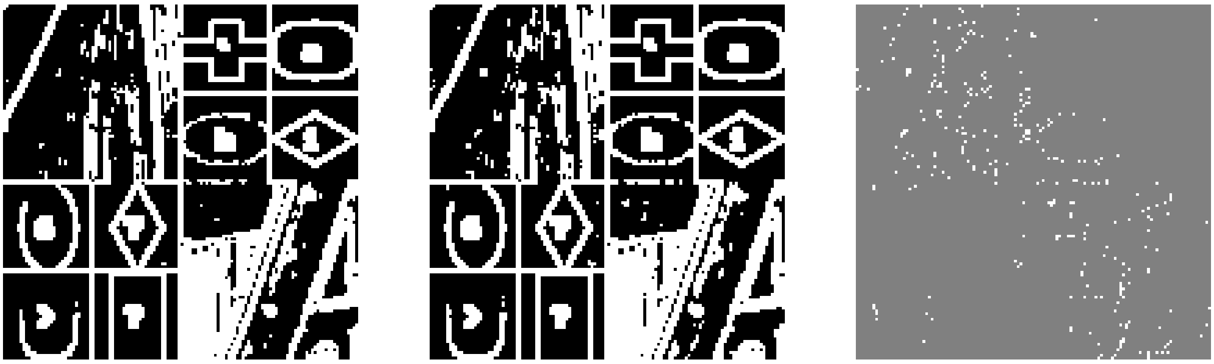

Figure 5.22: Model order selection when $\sigma = 5$, black denotes $k = 1$, white $k = 2$: (a) Pseudo-MML; (b) MML; (c) difference: white shows where MML chose $k = 2$ over Pseudo-MML's $k = 1$.

models enough for them to be selected more often. In fact, the message lengths calculated by the MML denoising algorithm are always shorter than those of Pseudo-MML. For a given segmentation, both algorithms use the same number of bits for the segment map, but MML uses fewer bits for the means. Because both use the true mean as the point estimate, the pixels are encoded in the same number of bits too. Thus MML's messages are always shorter.

5.8 Posterior blending of models

In Section 5.4.4 it was described how each model's posterior probability could be calculated from its message length. The posterior probability may be interpreted as a confidence in each model's ability to approximate the underlying image. This type of information is not really available when using FUEL's model order selection criterion.

The MML criterion chooses the model with the overall shortest message length, corresponding to the highest posterior probability. This is a good thing to do when one wishes to infer the best complete model for the window. If one is only interested in the true pixel values, and not the most appropriate segment map and segment means, then the best thing to do is to *blend over all models* weighted by their posterior probability. Parameters of non-interest should be "integrated out" [BS94].

In the MML and Pseudo-MML algorithms, the local approximation suggested by the most probable model alone was used for denoising. It is possible to produce a blended local approximation, $\hat{\mathbf{p}}_{blend}$, which is a linear combination of the local approximations from each model considered. The weight given to each model is exactly equal to its posterior probability. Equation 5.23 describes its calculation, where $\mathrm{post}(\theta)$ is the posterior probability of model θ, and $\hat{\mathbf{p}}_\theta$ is the local approximation associated with model θ.

$$\hat{\mathbf{p}}_{blend} = \sum_{\theta \in \Theta} \mathrm{post}(\theta)\ \hat{\mathbf{p}}_\theta \tag{5.23}$$

The blended local approximation may alternatively be considered those pixel values inferred by a composite model, θ_{blend}. This is shown in Equation 5.24. Combining the predictions from various "experts" is a popular technique in computer science, particularly in image compression [STM97, MT97].

$$\theta_{blend} = \sum_{\theta \in \Theta} \mathrm{post}(\theta)\ \theta \quad \longrightarrow \quad \hat{\mathbf{p}}_{blend} \equiv \hat{\mathbf{p}}_{\theta_{blend}} \tag{5.24}$$

Recall the two example candidate models from Section 5.4.3. Figure 5.23 applies Equation 5.23 to blend the local approximations associated with the two candidates. In this case, rounding back to integers would cause single best local approximation to equal the posterior blended one. However, the denoised pixels could be retained at a higher accuracy if further processing was to be done.

$$\hat{\mathbf{p}}_{blend} = 0.13 \times \begin{array}{|c|c|c|}\hline 13.8 & 13.8 & 13.8 \\ \hline 13.8 & 13.8 & 13.8 \\ \hline 13.8 & 13.8 & 13.8 \\ \hline\end{array} + 0.87 \times \begin{array}{|c|c|c|}\hline 15.0 & 15.0 & 15.0 \\ \hline 15.0 & 15.0 & 11.0 \\ \hline 15.0 & 11.0 & 11.0 \\ \hline\end{array} = \begin{array}{|c|c|c|}\hline 14.8 & 14.8 & 14.8 \\ \hline 14.8 & 14.8 & 11.4 \\ \hline 14.8 & 11.4 & 11.4 \\ \hline\end{array}$$

Figure 5.23: Posterior blending of the two example models from Section 5.4.3.

5.8.1 Results

Figure 5.24 compares the RMSE performance of the posterior blended version of MML to the MML algorithm which simply uses the most probable model. For $\sigma \geq 8$, posterior blending improves the RMSE performance of the MML algorithm. It may also be possible to blend models in the FUELS algorithm, but a metric would have to be invented to quantify the suitability of each candidate model, whereas this metric comes naturally for MML.

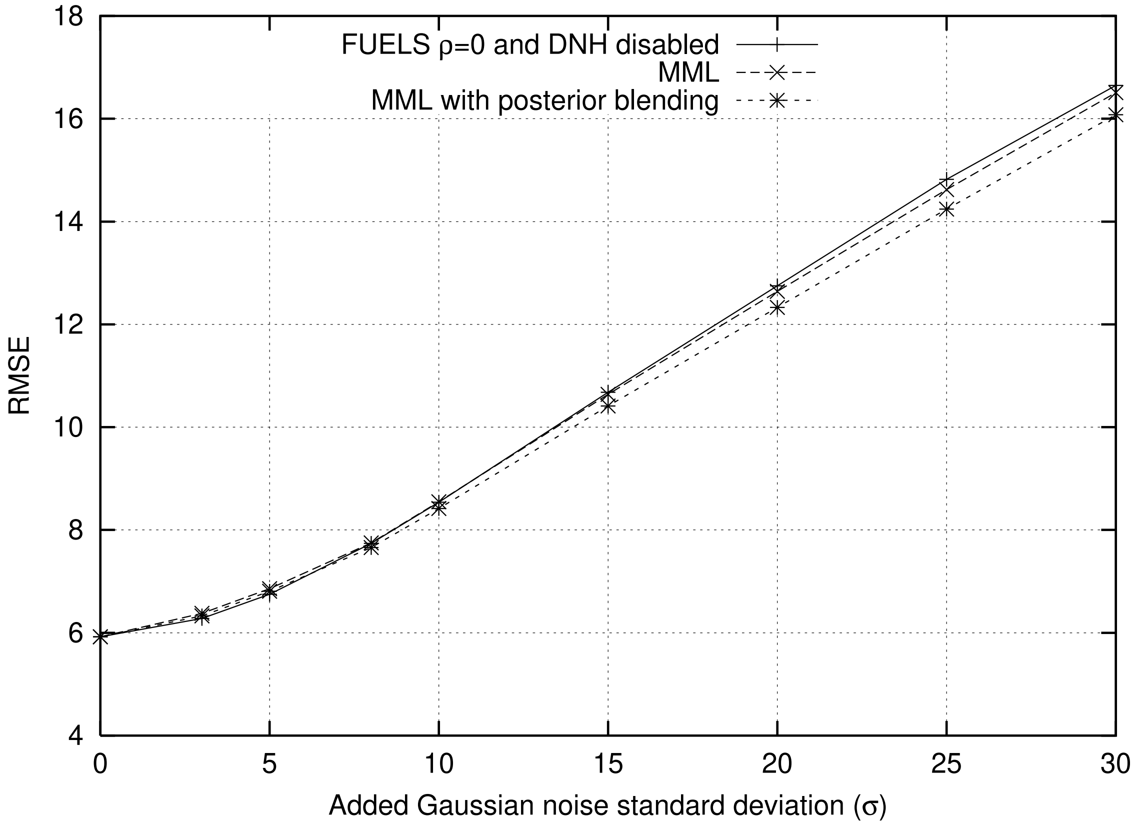

Figure 5.24: RMSE comparison for denoising `montage`, with true σ supplied.

The ability to blend over multiple models is very useful when there is no clear mode (peak) in the posterior distribution. In this case, the *posterior mean* has been used. This is related to the use of a squared error loss function [OB94], which also happens to be what the RMSE metric uses. Combining multiple local approximations together tends to produce a better overall local approximation. In the case where one model has nearly all the posterior probability associated with it, the algorithm behaves just like the non-blended version.

5.9 Incorporating "Do No Harm"

5.9.1 The FUELS approach to DNH

In Section 4.9 the "do no harm" (DNH) idea was applied to FUELS. DNH has the effect of limiting the worst case behaviour of a denoising algorithm. In FUELS, if any pixel value in the optimal local approximation differs from its noisy value by more than 3σ, the local approximation is rejected, and the noisy pixels are passed through unmodified. It is a simple matter to incorporate this post processing to the posterior blended pixel values produced by the MML denoising algorithm.

The results so far have shown FUELS and MML to perform similarly. Figure 5.25 compares their RMSE performance when both utilise FUELS-style DNH. As expected, their performances are much better at lower noise levels when DNH is enabled. When $\sigma \geq 8$, MML consistently beats FUELS, but the gap is not significant. Although not shown, their WCAEs are identical at all points.

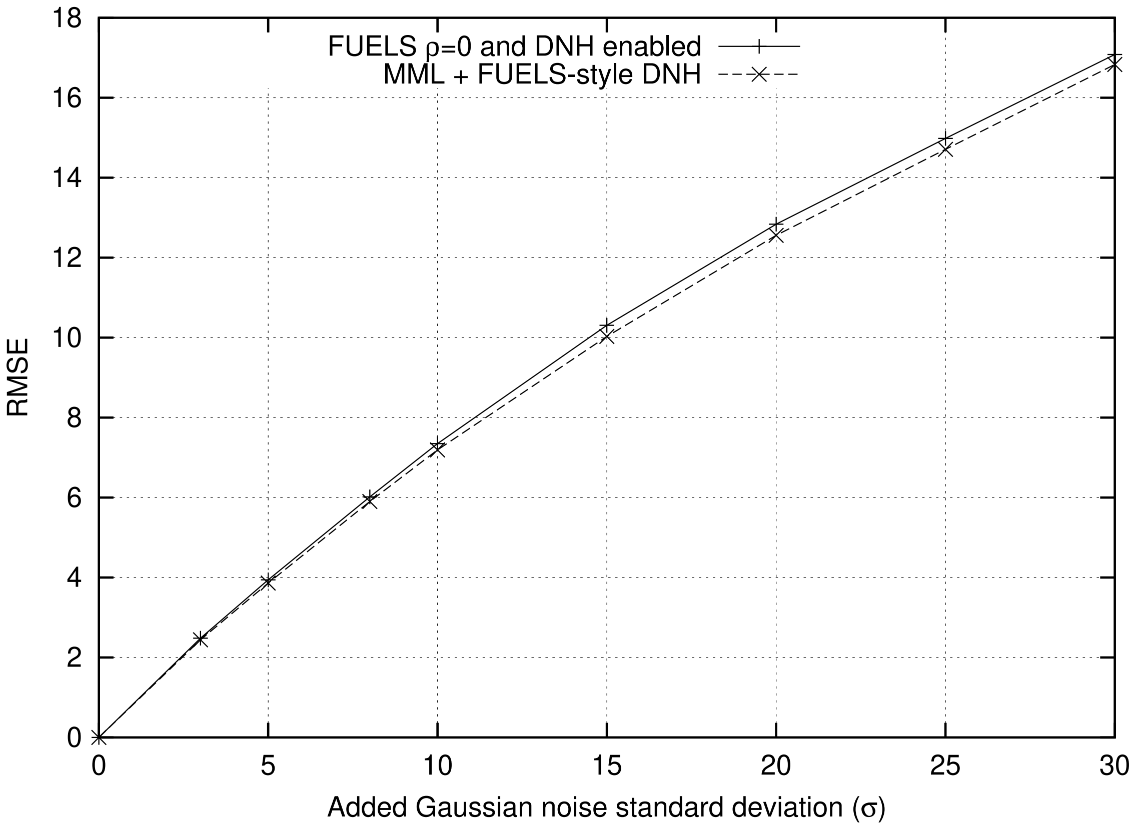

Figure 5.25: RMSE comparison for denoising `montage`, true σ supplied.

5.9.2 Information based DNH

In FUELS, the $C\sigma$ threshold was common to both the DNH and model selection components. It may be beneficial for the MML algorithm to use an information theoretic approach to DNH which fits in more appropriately with its model selection criterion and posterior blending. When DNH is invoked by FUELS, the noisy pixel values are passed through unaltered. Under MML, this corresponds to encoding pixels as they are, without respect to any particular model. A *raw encoding* of M noisy pixel values requires, at most, $M \log_2 Z$ bits. For example, a raw encoding of a 3×3 window of 8 bpp greyscale pixels would need 72 bits.

The raw encoding may be interpreted as a *null model*. The null model could be placed into the existing pool of segment-based candidate models, and be judged alongside them. This is similar to the t-test method in Section 4.7.2, where a null and alternative hypothesis are compared. The null model could actually have the shortest two-part message length, causing it to be selected as the “best” model. This is like FUELS where DNH is invoked when all other models appear unreliable. When posterior blending is used, the null model becomes part of blend, with its local approximation being equal to the original pixel values.

For this idea to work correctly, the null model and the standard segment-based models must be compared fairly. Each model needs to be prefixed by a binary event stating whether a standard, or null, model is to follow. Figure 5.26 illustrates this arrangement, where “DNH” denotes the use of the null model.

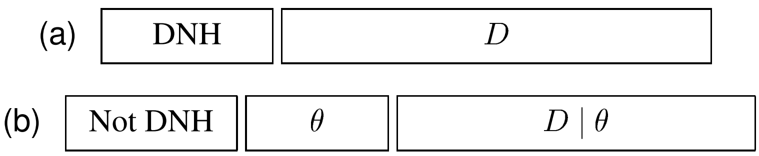

Figure 5.26: Incorporating DNH: (a) null model encodes the data as is; (b) standard model.

Let $\Pr(DNH)$ denote the prior probability associated with the prefix for the null model. For example, if $\Pr(DNH) = 0.1$ the null model would have a prefix of length $(-\log_2 0.1) = 3.32$ bits. The prefix for a standard, segment-based model would be only $(-\log_2 0.9) = 0.1$ bits, because according to the prior, they will be used more often.

Figure 5.27 plots the proportion of pixels for which FUELS-style DNH is invoked when denoising `montage`. The two curves are effectively plotting suitable values of $\Pr(DNH)$ to use at different noise levels. Both curves have a hyperbolic shape, approximately following Equation 5.25. When $\sigma = 0$, the prefix has zero length for the null model and infinite length for any standard model. If this formulation was used, DNH would be invoked for every pixel

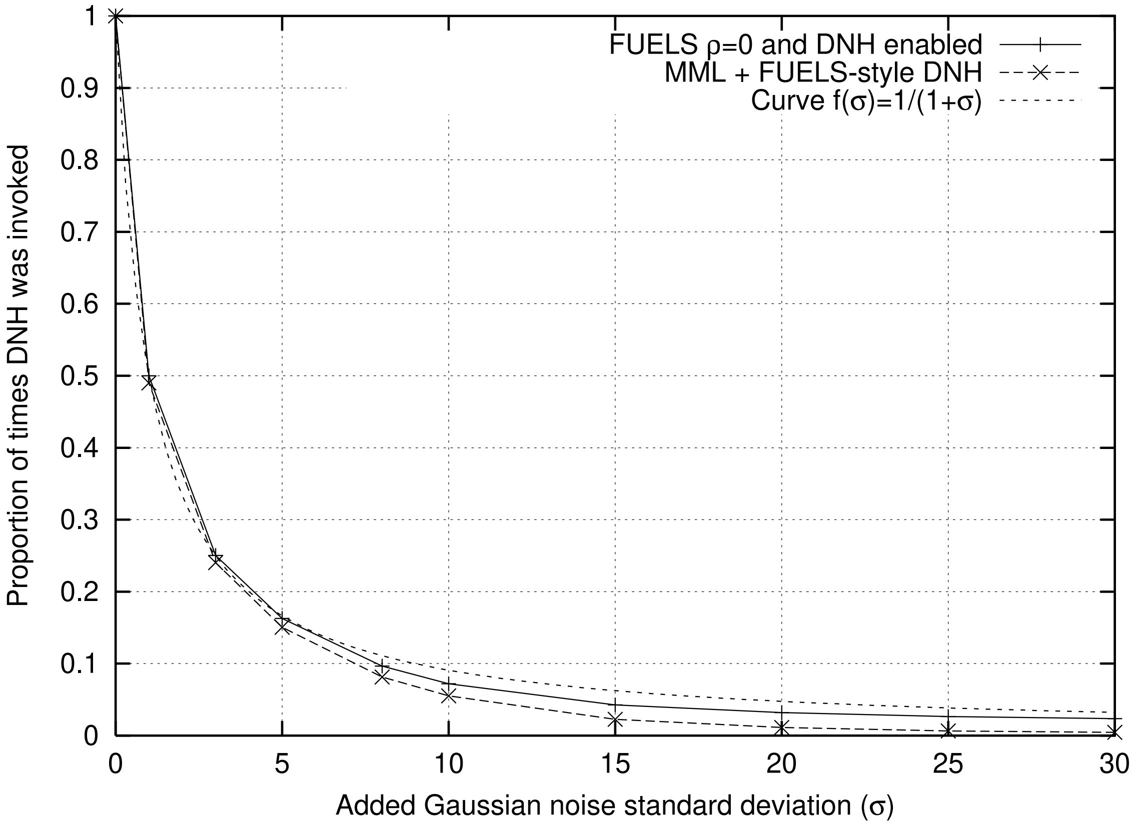

Figure 5.27: FUELS-style DNH usage when denoising `montage`, true σ supplied.

when $\sigma = 0$. This makes sense because if the noise level is zero, then all pixels are noiseless and should remain unaltered.

$$\Pr(DNH) = \frac{1}{1+\sigma} \quad \longleftrightarrow \quad \mathcal{L}(DNH) = \log_2(1+\sigma) \quad \text{bits} \tag{5.25}$$

Figure 5.28 compares three denoising algorithms: FUELS using its DNH, MML using FUELS-style DNH, and MML using the new information based DNH, with $\Pr(DNH) = 1/(1+\sigma)$. For $\sigma > 5$, MML with its information based DNH consistently outperforms the others in terms of RMSE when denoising `montage`. Although not shown, similar behaviour was observed for other images.

The worst case absolute error (WCAE) results are provided in Figure 5.29. FUELS and MML with FUELS-style DNH have the same WCAE for all noise levels tested. When $\sigma \geq 10$, the information-based DNH approach consistently has a lower WCAE. This is interesting because the MML DNH approach makes no explicit attempt to restrict the maximum

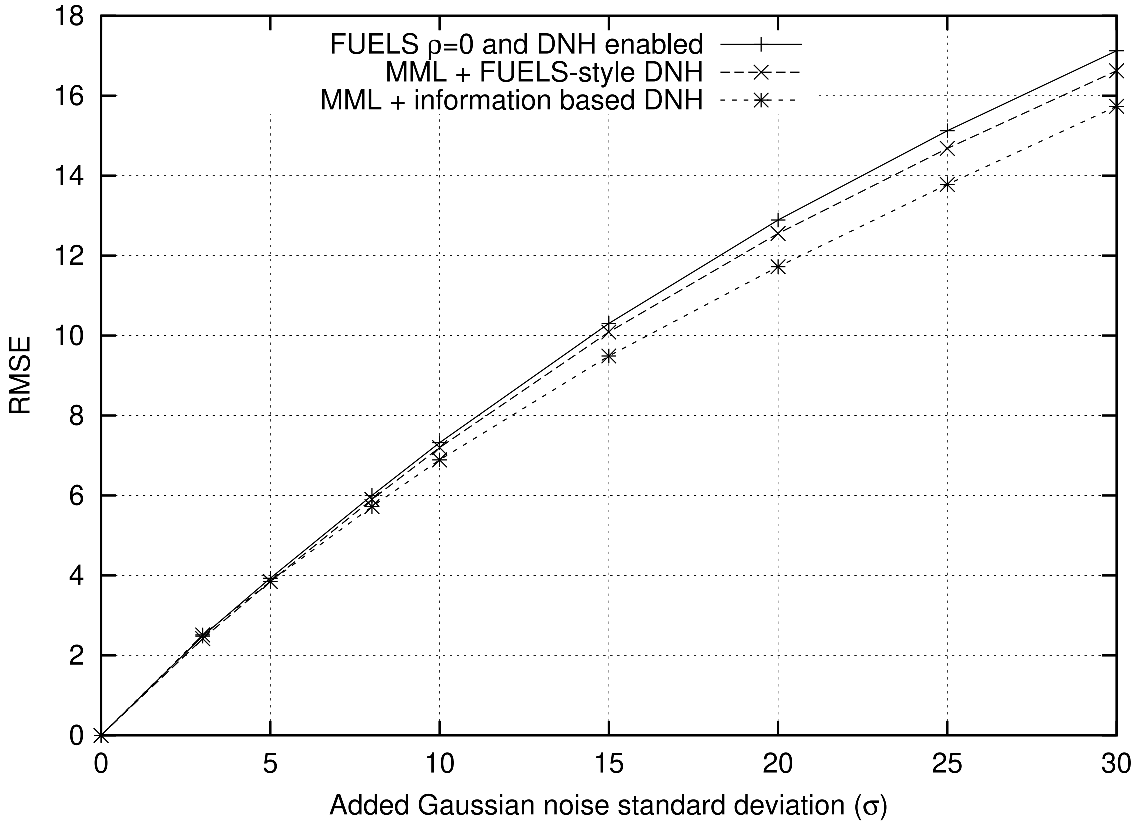

Figure 5.28: RMSE comparison for denoising `montage`, true σ supplied.

allowed change in pixel intensity during the denoising step. Of course, large alterations in intensity are unlikely to fit within the Gaussian segment models well, causing those models to have low posterior probability.

DNH ensures that pixels do not change by more than 3σ relative to the *noisy image* only. However, the WCAE is taken between the denoised image and the *ground truth* image. Consider a noiseless image corrupted by additive noise with standard deviation σ. There will always be a small fraction of pixels whose values change by more than 3σ. In the worst case, a pixel could change by $Z - 1$ intensity units. DNH operates relative to the noisy image, so it is still possible for the WCAE to be very large.

Figure 5.30 plots the actual relative frequencies of DNH usage for the three algorithms tested. MML invokes DNH less often when it uses the information based DNH than it does with FUELS-style DNH. There is a trade-off between invoking DNH too often, whereby no denoising occurs, or not invoking it, whereby there is a risk of producing a poor local approximation which detrimentally affects the RMSE. It seems that the information based DNH is

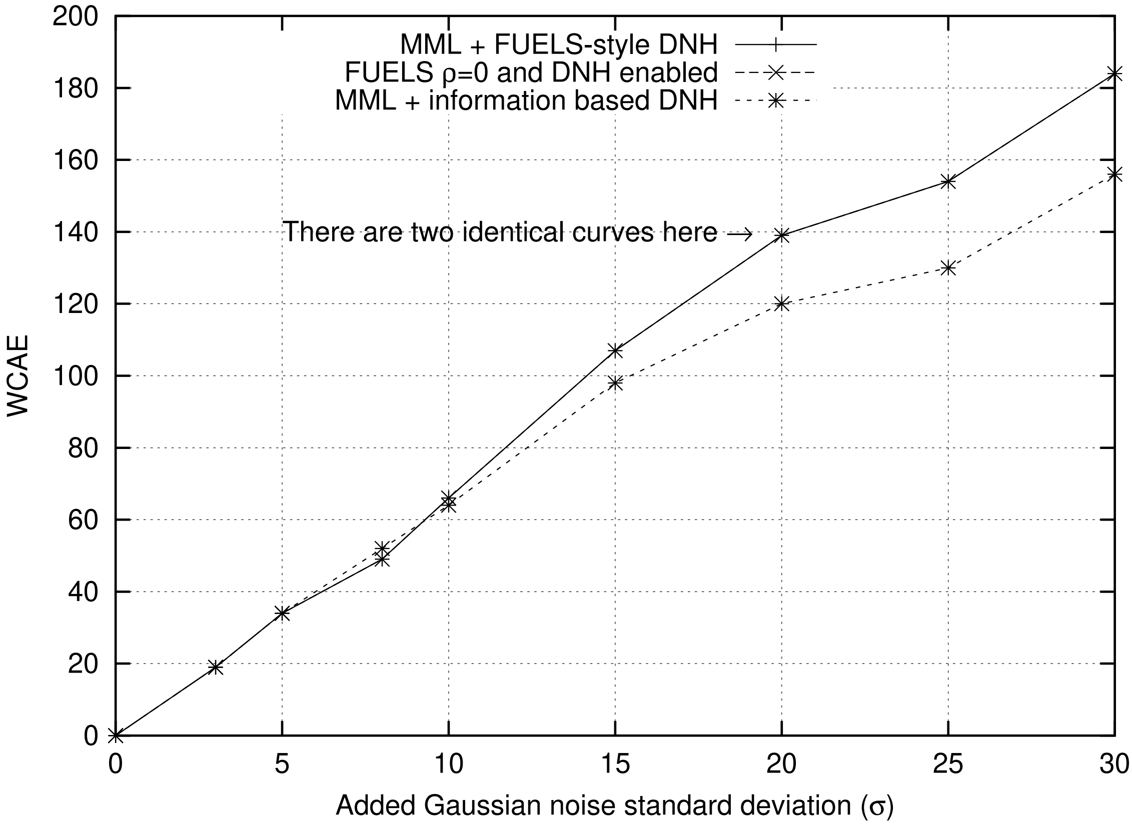

Figure 5.29: WCAE comparison for denoising `montage`, true σ supplied.

better at making this trade-off. Although it was decided in a sensible manner, the 3σ threshold used by FUELS-style DNH may not be optimal. It is possible that a higher value, such as 4σ, could improve FUELS' results.

5.10 Combining overlapping estimates

In Section 4.11 the idea of combining estimates from overlapping windows was introduced. For FUELS, various schemes for linearly combining estimates were tested. It was found that an equal weighting ($\rho = 1$) gave the best RMSE results. This same concept may be applied to the MML algorithm with ease.

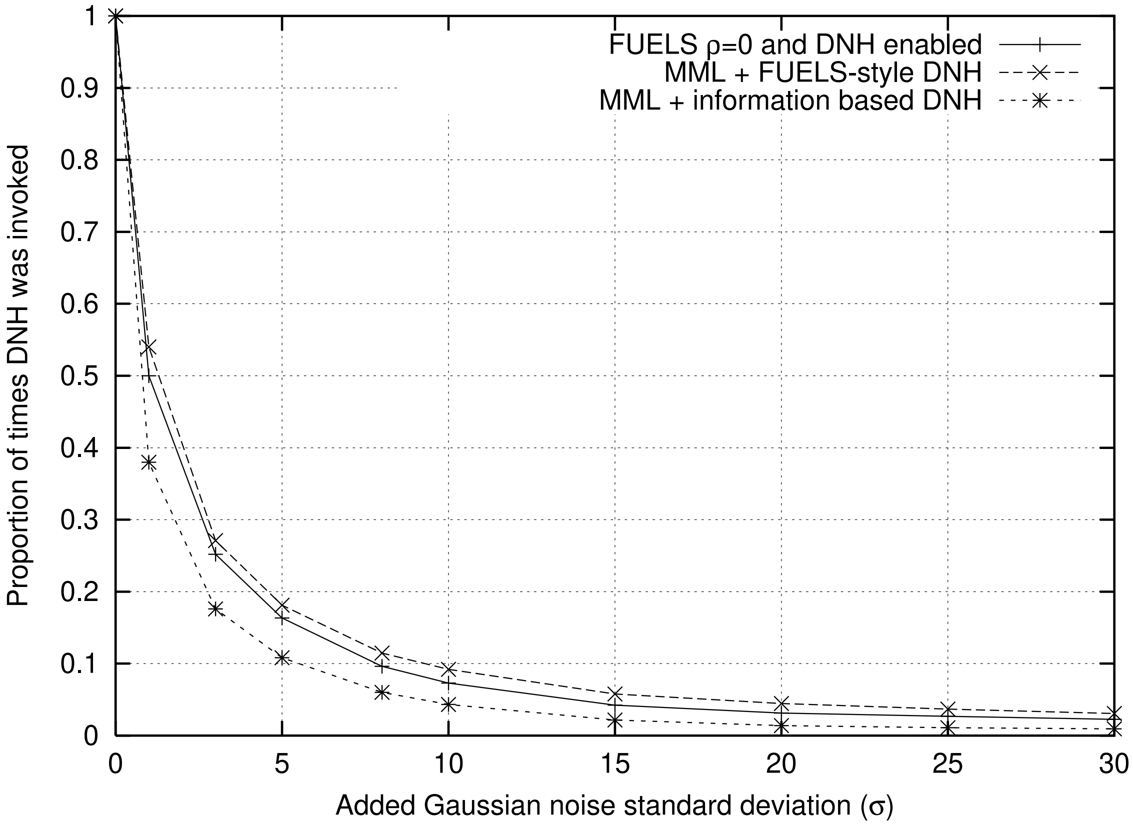

Figure 5.30: How often DNH is invoked for `montage`, true σ supplied.

5.10.1 Equal weighting

Figure 5.31 compares the RMSE performance of four algorithms on the `montage` image: FUELS with $\rho = 0$ and $\rho = 1$, and MML with $\rho = 0$ and $\rho = 1$. The appropriate DNH option is used for all instances. As expected, overlapping averaging significantly improves the RMSE results for both FUELS and MML at all noise levels. When $\rho = 1$, there is little to no difference between the two. It appears FUELS is able to benefit more from the overlapping averaging than MML. This may be due to MML having already an advantage through its application of posterior blending of models at the same pixel position.

5.10.2 Message length weightings

Overlapping averaging combines estimates of the same pixel from different models. For FUELS in Section 4.11, the combination was restricted to a linear one. Equation 5.26 describes

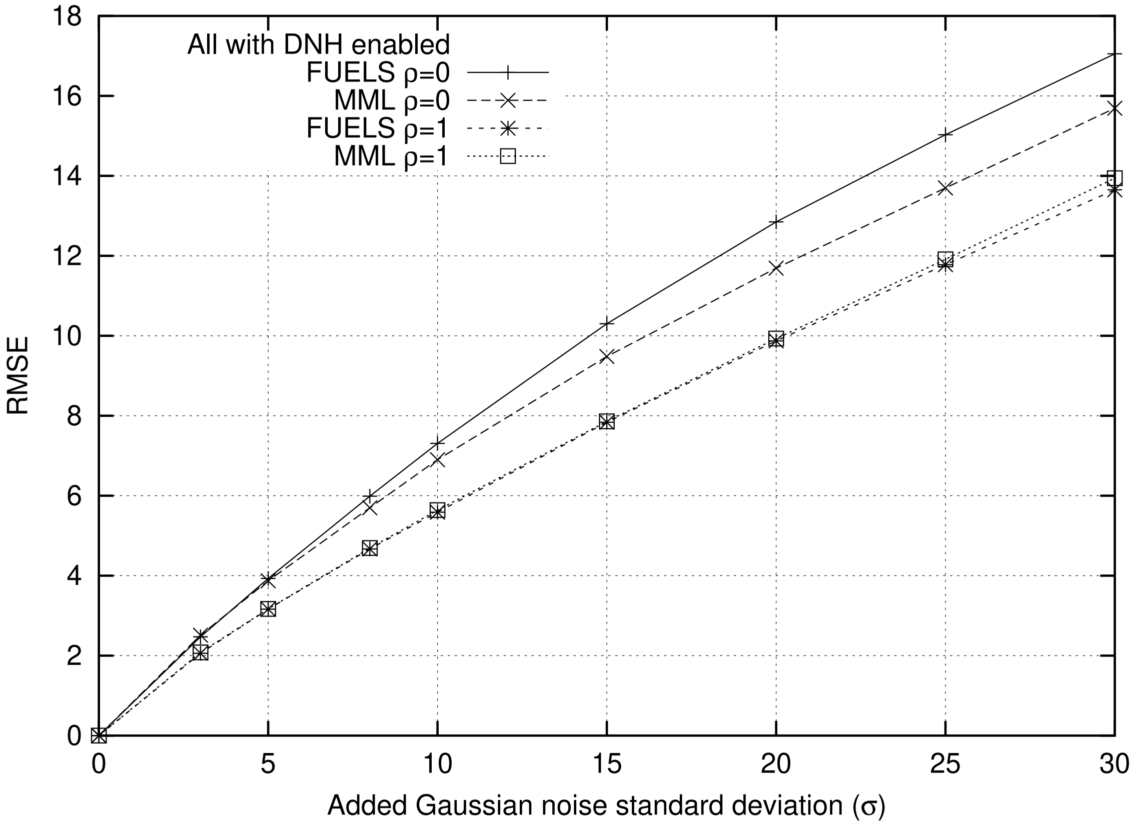

Figure 5.31: RMSE comparison for denoising `montage`, true σ supplied.

this again, where $\tilde{p}_1 \dots \tilde{p}_M$ are the denoised estimates for the same pixel position from each of the M overlapping models, $\hat{p}$ is the final denoised value, and w_j are the weights.

$$\hat{p} = \frac{\sum_{j=1}^{M} w_j \cdot \tilde{p}_j}{\sum_{j=1}^{M} w_j} \tag{5.26}$$

For FUELS, various schemes for choosing the weights were assessed. Each scheme tried to use some attribute of each model to modify the weights. The attributes attempted to capture the quality, or *goodness of fit*, of a model.

Model and data posterior

In an MML framework, the message length of a model is a natural measure of its goodness of fit. Equation 5.27 bases the weights on the unnormalized posterior probability of the model and data. This is done using the message length associated with the model and data from which the estimate originally came. This attempts to give more weight to those estimates which came from models which perhaps better explained their local region.

$$w_j = 2^{-\mathcal{L}(\theta)-\mathcal{L}(\mathbf{p}' \mid \theta)} \qquad \text{with} \qquad \tilde{p}_j \in \mathbf{p}' \tag{5.27}$$

Model and pixel posterior

An alternative to using the full message length is to use only the length of the model and pixel in question. It could be argued that we are only interested in how well the local approximation modeled the particular pixel being combined. Equation 5.28 describes this formulation.

$$w_j = 2^{-\mathcal{L}(\theta)-\mathcal{L}(\tilde{p}_j \mid \theta)} \tag{5.28}$$

Pixel posterior

One final potential weighting scheme is given in Equation 5.29. The weight is proportional to the posterior probability of the pixel alone. This attempts to measure how well the noisy pixel fitted into the model it came from. That model was either the MAP or posterior blended model for the window the pixel originated from. It is expected that this weight would behave similarly to a weight based on the squared error: $w_j = (p' - \hat{p})^{-2}$. This is because a pixel message length is based on a Gaussian distribution.

$$w_j = 2^{-\mathcal{L}(\tilde{p}_j \mid \theta)} \tag{5.29}$$

Results

Figure 5.32 compares the RMSE performance of four weighting schemes for `montage`: $\rho = 1$, and the three probabilistic ones just described. None of the variable weighting schemes perform better than equal weighting for any values of σ. The same results were found to occur for the `lenna` image. For these reasons, schemes other than equal weighting will not be explored further.

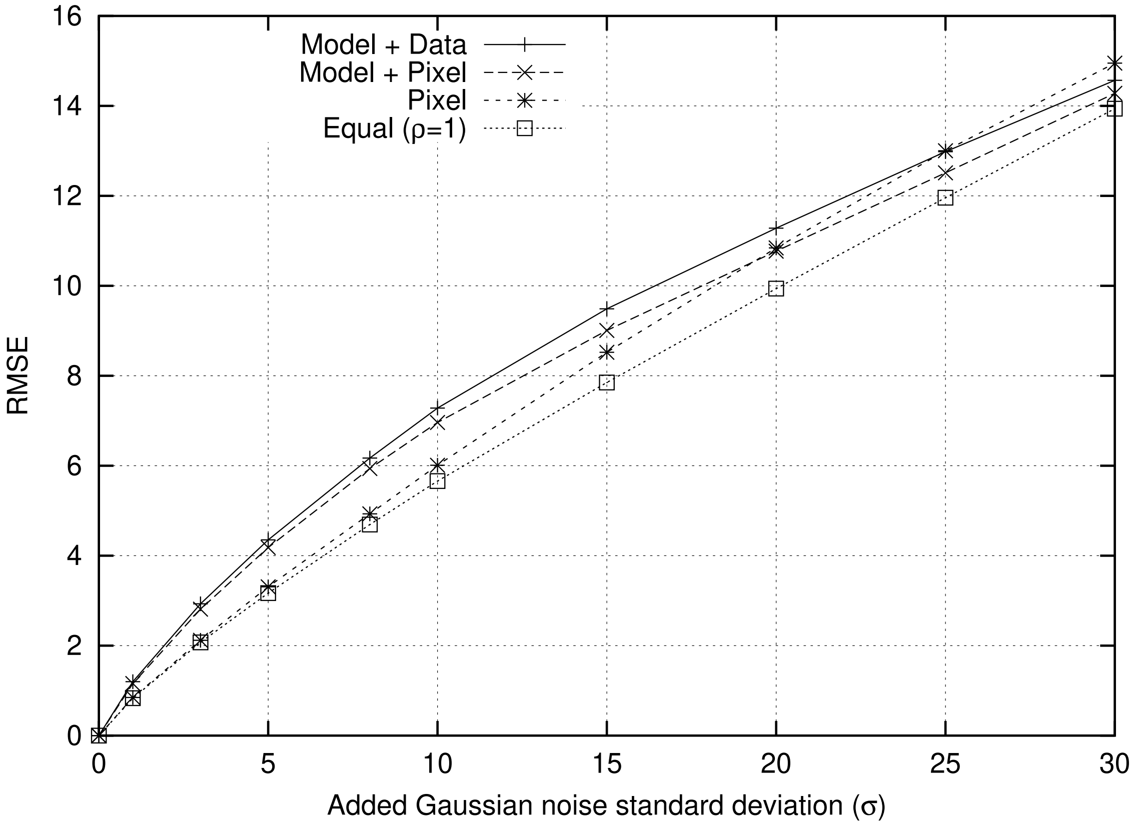

Figure 5.32: RMSE comparison for denoising `montage`, true σ supplied.

5.11 Estimating the noise level

The analyses performed so far in this chapter have assumed that the true value of σ, the standard deviation of the synthetic noise, is known. In real situations, the noise is not synthetic and its level is unknown. It typically has to be estimated from the data. In Section 4.13, the FUELS algorithm was adapted to use the robust Immerkær noise variance estimation algorithm [Imm96]. MML could use this estimate too.

Figure 5.33 compares the FUELS and MML algorithms when both use Immerkær's estimated value for σ, rather than the true value. The main variation occurs for low σ, where the noise level is over-estimated. This causes DNH to be invoked less often, and the RMSE results to be higher. Either way, there is very little difference between FUELS and MML.

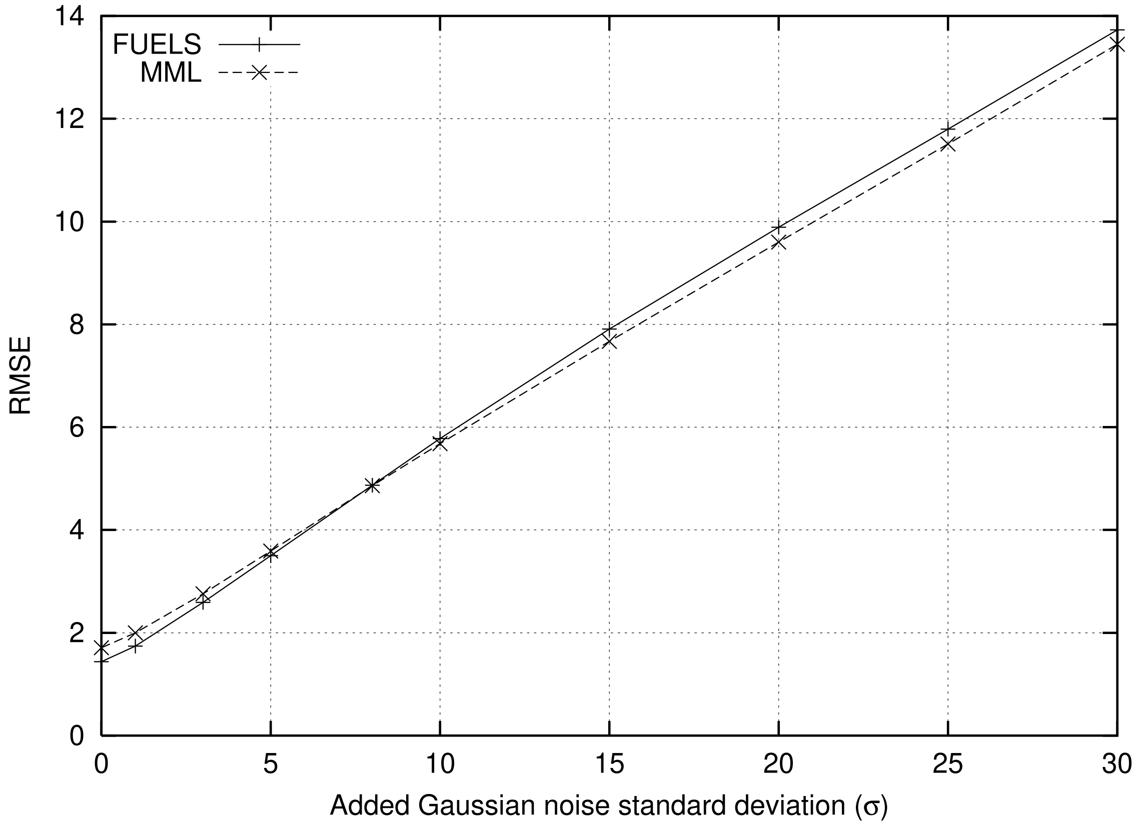

Figure 5.33: RMSE comparison for denoising `montage`, with σ estimated.

5.12 Evolution of the message length

Each modification to the MML denoising algorithm has so far improved its RMSE performance. This suggests that, *on average*, the models generated are becoming more reliable. Under the MML framework, models which have shorter two-part messages are assumed to better explain the data they are modeling. Figure 5.34 plots the *average message length* over the whole `montage` image for the various MML-based algorithms explored so far. The estimated noise level is used in all cases.

As expected, the average message length is mostly an increasing function of the noise level. For Pseudo-MML, switching to a more suitable segment map prior reduces the average message length by about 4 bits. The move from Pseudo-MML to MML via the use of quantized segment means provides a similar gain. The inclusion of a DNH candidate model again improves the average message length for the MML algorithm at low noise levels.

For the algorithms without DNH, the average message length is observed to slightly increase,

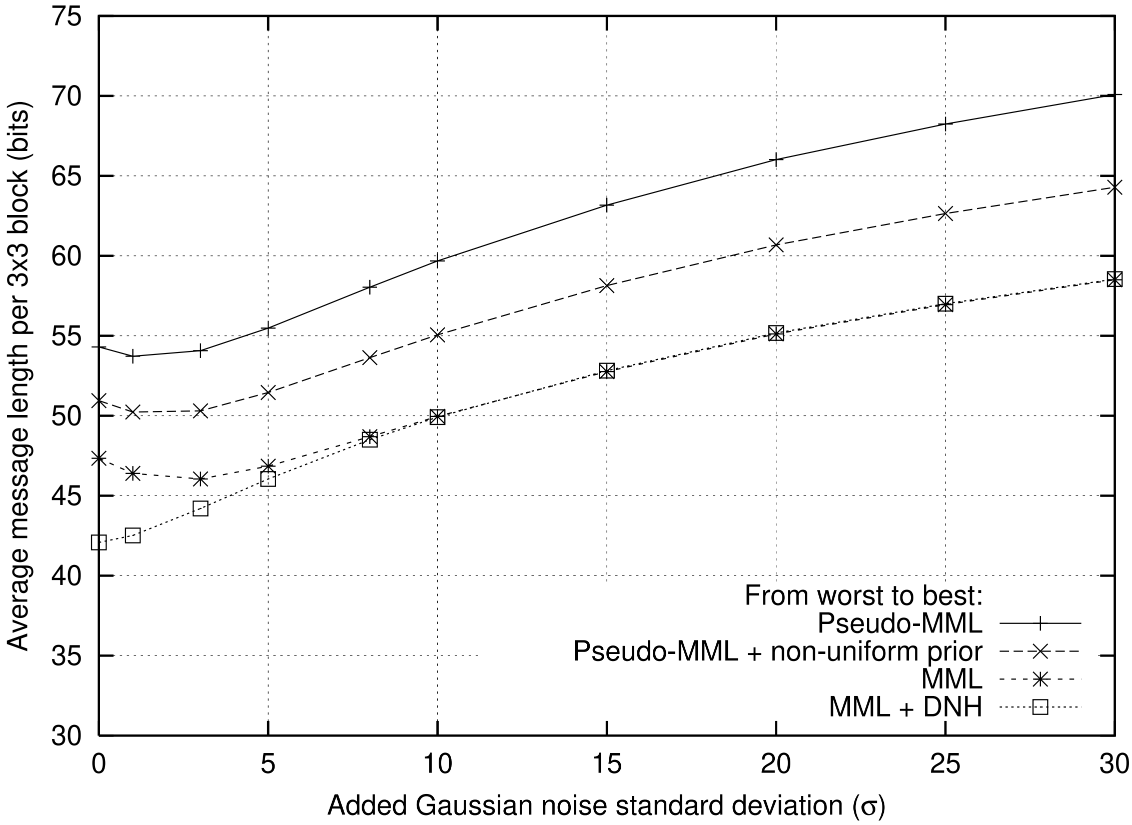

Figure 5.34: Evolution of the average two-part message length for the MML denoising algorithm, with σ estimated from the noisy `montage` image.

rather than decrease, when $\sigma \leq 2$. Immerkær's method overestimates σ when it is very low. This could increase the relative cost of two segment models, causing fine structure to be corrupted. Usually this structure is swamped by noise, but when there is no noise it becomes significant. The expression in Equation 5.21 for the optimal quantization of sample means begins to break down when $\sigma \ll Z$. This is due to an assumption of uniformity within quantization regions [OH94]. Also, the implementation of numerical integration for calculating the likelihood may not be accurate for very small intervals[1].

5.13 Learning from the data

In this chapter, a new approach to local segmentation model selection has been developed and applied to the problem of image denoising. Using FUELS as a basis, each component

[1]The standard `erf()` and `erfc()` functions from the GNU C maths library v2.2 were used.

was slowly modified or replaced to use ideas and techniques from information theory. This helped to remove arbitrary constants from the algorithm, and allow a wider range of image structure to be potentially discovered. The results for the new MML denoising algorithm are not significantly better than FUELS. Using MML improves the WCAE at higher noise levels, but only mildly improves RMSE. It is possible that only slight improvements are possible because FUELS is already quite good — Chapter 4 has already shown that FUELS outperforms rival denoising algorithms.

The performance of MML is dependent on the prior used. When denoising, the prior consists of σ, $\Pr(DNH)$, and $\Pr(\mathbf{c})$. The prior determines the length of the model part of each message being compared. The prior is determined before processing of the image begins, and is not modified during processing at all. It could be possible to vary the priors on a per block basis. For example, the prior probability of each segment map could be dependent on the posterior probability of segment maps from nearby pixels already processed. The aim would be to exploit edge coherence. However, in this chapter the same priors will be re-used for processing each pixel in the image.

Under the MML framework, models are assessed using their respective message lengths. The overall message length (the posterior) is computed as the length of the model (determined by the prior) plus the length of the data given the model (the negative log-likelihood). If the amount of data is large, the data part will dominate and eventually swamp the effect of the prior. If the amount of data is small, the prior exerts a large influence on the posterior [Pre89], as illustrated in Figure 5.35. In fact, once the amount of data becomes large enough, the model component may be ignored, and MML reduces to maximum likelihood.

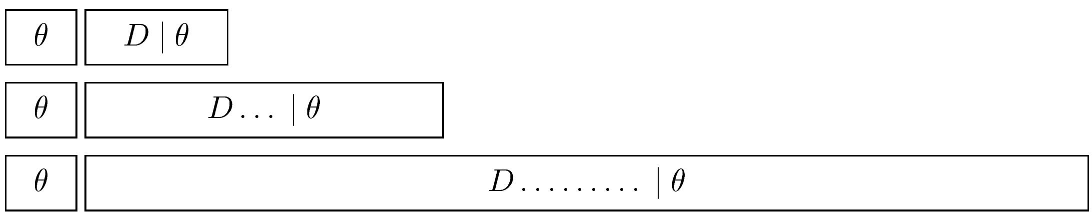

Figure 5.35: As the amount of data increases, the model part becomes less significant.

Because local image processing algorithms only use a small number of pixels, and different images vary in their properties, a technique for dynamically "learning" the prior from the noisy image is desirable. This may be achieved by iterating the algorithm. After processing each pixel using the global prior (common to each local window), a local posterior is obtained. The local posteriors can be accumulated to form a global posterior when the whole image has been processed. The global posterior is simply a probability distribution over possible models, formed from observation of the image at all its localities. It captures the particular essence of an image, briefly summarizing its local segmentation features. This makes it ideal for use as a new global prior for re-processing the same image from scratch. Iteration should continue until the priors converge.

The iteration process has an analogy with the human visual system [FH96, Ede99]. Imagine visiting a new friend's house and pulling a photo album from their bookshelf. Before you open it, you have no idea of what the first photo will contain. This could be considered a state of total ignorance, expressed by the uniform priors already used in Section 5.4.2. After opening the album to the first picture, your visual system scans it to form an initial impression. The information gained from this first scan may be considered a posterior, which is then used as a revised prior for a second scan. This process continues until we have recognized or made sense of the picture, as shown in Figure 5.36.

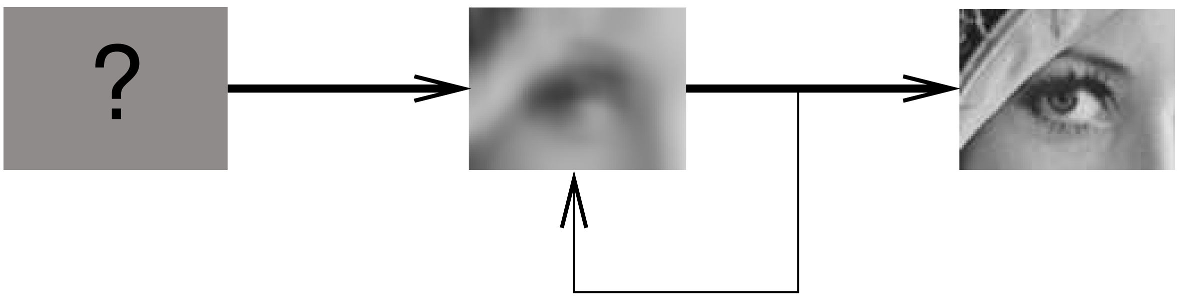

Figure 5.36: Pictorial representation of the iterative approach to image understanding.

A data driven prior incorporates image dependent information which is shared by all the candidate local models. Because this information is common to all models, it does not need to be incorporated into the message length. Its code length is the same for every model.

5.13.1 Learning a prior for the segment maps

The prior probability distribution for the segment maps, $\Pr(\mathbf{c})$, is important, because low level image structure is the heart of local segmentation. Two priors have been considered so far. The uniform prior assumed all segment maps to be equally likely. In Section 5.6, a non-uniform prior biased toward $k = 1$ was introduced. This prior improved the results significantly, because it coincided more accurately with the expectation about which segment maps would be common in `montage`.

Figure 5.37 shows four segment maps. Currently, the prior probability for the first homogeneous block is 0.5. In fact, that is expected to vary for different images. The other three segment maps have the same prior probability $0.5/(2^{M-1} - 1) \approx 0.00196$. However, the 255 heterogeneous segment maps are not expected to occur with equal frequency. Common edge patterns would probably occur more often than, say, the random looking segment map in Figure 5.37d. A more informative segment map prior should improve denoising results.

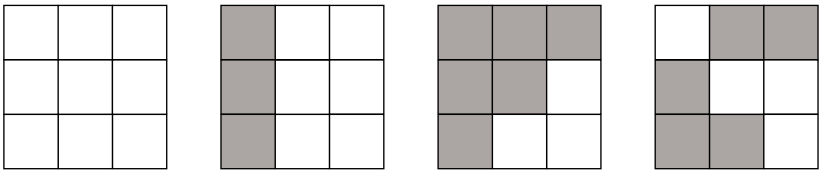

Figure 5.37: Potential segment maps: (a) popular $k = 1$; (b–c) common edge patterns; (d) would rarely occur.

The segment map prior may be considered a multistate probability distribution, $\Pr(\mathbf{c})$, which sums to unity over all possible segment maps. This is a *global prior* which is re-used for each window processed. After processing each pixel (x, y), a *local posterior* probability distribution over models considered is obtained. Let it be denoted $\mathrm{post}(x, y, \theta)$, where θ refers to a specific model from Θ, the set of candidate models.

Each local posterior may be accumulated to form a *global posterior*, denoted Post($\mathbf{c}$), using Equation 5.30. The global posterior probability for a specific segment map is the normalized sum of local posterior probabilities for models using that particular segment map. For example, imagine an image with four pixels, and that only one candidate model per pixel used a homogeneous segment map. If that model had local posterior probabilities 0.9, 0.7, 0.1, and 0.3 for each pixel respectively, then the global posterior for the homogeneous segment map would be $(0.9 + 0.7 + 0.1 + 0.3)/4 = 0.5$.

$$\mathrm{Post}(\mathbf{c}) = \frac{1}{XY} \sum_{y=0}^{Y-1} \sum_{x=0}^{X-1} \sum_{\theta \in \Theta} \mathrm{post}(x, y, \theta) \quad \text{where} \quad \mathbf{c} \in \theta \tag{5.30}$$

Equation 5.31 describes how the global posterior may be used as a new global prior for the image. This iterative process begins with the same noisy image each time, but uses an improved, *data driven prior*. This is applicable to any components of the prior distribution.

$$\Pr(\mathbf{c})_{t+1} = \text{Post}(\mathbf{c})_t \quad \text{and} \quad \Pr(\mathbf{c})_0 \quad \text{is chosen sensibly.} \tag{5.31}$$

This iterative process is equivalent to using the E.M. algorithm discussed in Section 3.2.1. The E.M. algorithm is guaranteed to converge, but not necessarily to a global optimum. The best way to encourage a good optimum is to use reasonable starting conditions. The two priors for **c** used so far are both reasonable, but the uniform prior would probably be a better choice when applying the algorithm to images with unknown properties, as it has no bias.

Results

Figure 5.38 compares the MML denoising algorithm using 1 iteration and a non-uniform segment map prior, to that using 6 iterations to learn a segment map prior. For `montage`, a small improvement in RMSE is observed for $\sigma \geq 15$. This suggests that iteration has made a useful difference to the segment map priors. This could allow the algorithm to identify structure behind the noise better, because common structural segment maps are cheaper to encode. The original prior set all $k = 2$ segment maps to be equally likely, making it difficult to give preference to one over the other when the noise level was high. Another possibility is that the algorithm has learned to choose $k = 1$ more often, which at higher noise levels will probably give more smoothing without the concern of damaging high contrast edges.

Figure 5.39 shows the 15 most probable segment maps that the new MML algorithm learned from `montage`. Note the distinct lack of random-looking patterns. The top row coincides with the many boundaries between squares in the two artificial quadrants in `montage`.

5.13.2 Learning the prior for DNH models

The prior probability $\Pr(DNH)$ is used to encode the prefix which distinguishes between the null model and standard models. In Section 5.9, the quite reasonable prior, $\Pr(DNH) =$

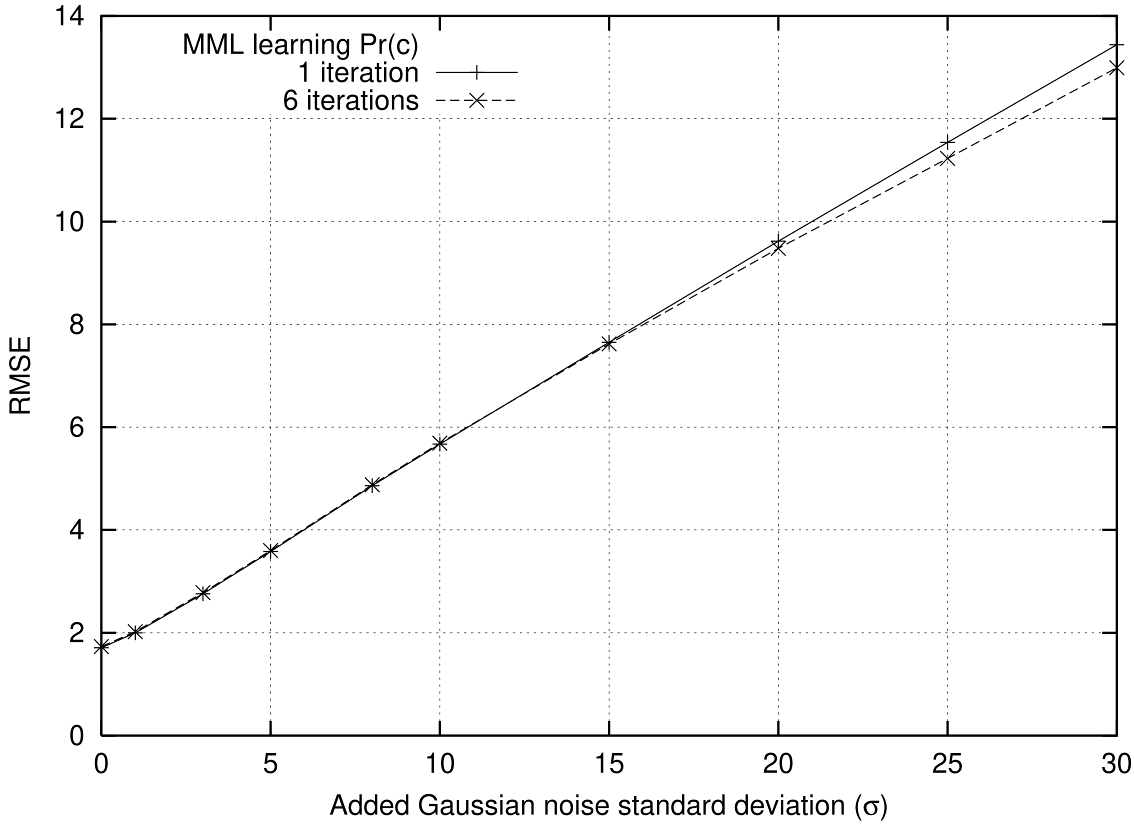

Figure 5.38: Learning $\Pr(\mathbf{c})$: RMSE comparison for `montage`, with σ estimated.

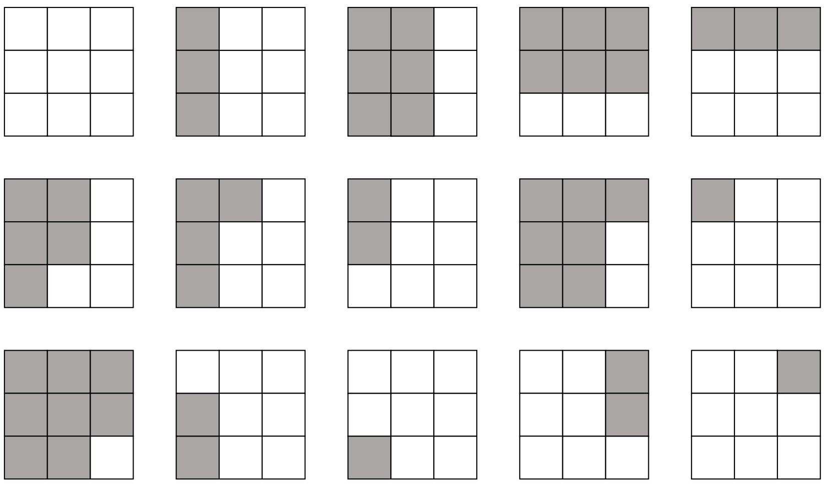

Figure 5.39: The 15 most popular (canonical) segment maps for `montage` when $\sigma = 0$.

$1/(1 + \sigma)$, was used. However, it should be possible to use the data driven prior method to learn automatically a probability suited to the noisy image being processed. This could be done in the same way as the segment maps were learned. Instead of accumulating the local posterior probabilities from models using a particular segment map, those posteriors for the null models are accumulated. This global posterior probability is exactly equal to the best value of $\Pr(DNH)$ to use for the next iteration.

Results

Figure 5.40 compares the RMSE performance of the non-iterated, one-pass MML algorithm to one which attempts to learn only $\Pr(DNH)$ over 6 iterations. The prior for the segment map was not learned. The results indicate that learning $\Pr(DNH)$ makes little difference to the algorithm's performance. The same result occurred when an initial uniform prior, namely $\Pr(DNH) = 0.5$, was used. This suggests that the existing static prior is a good one.

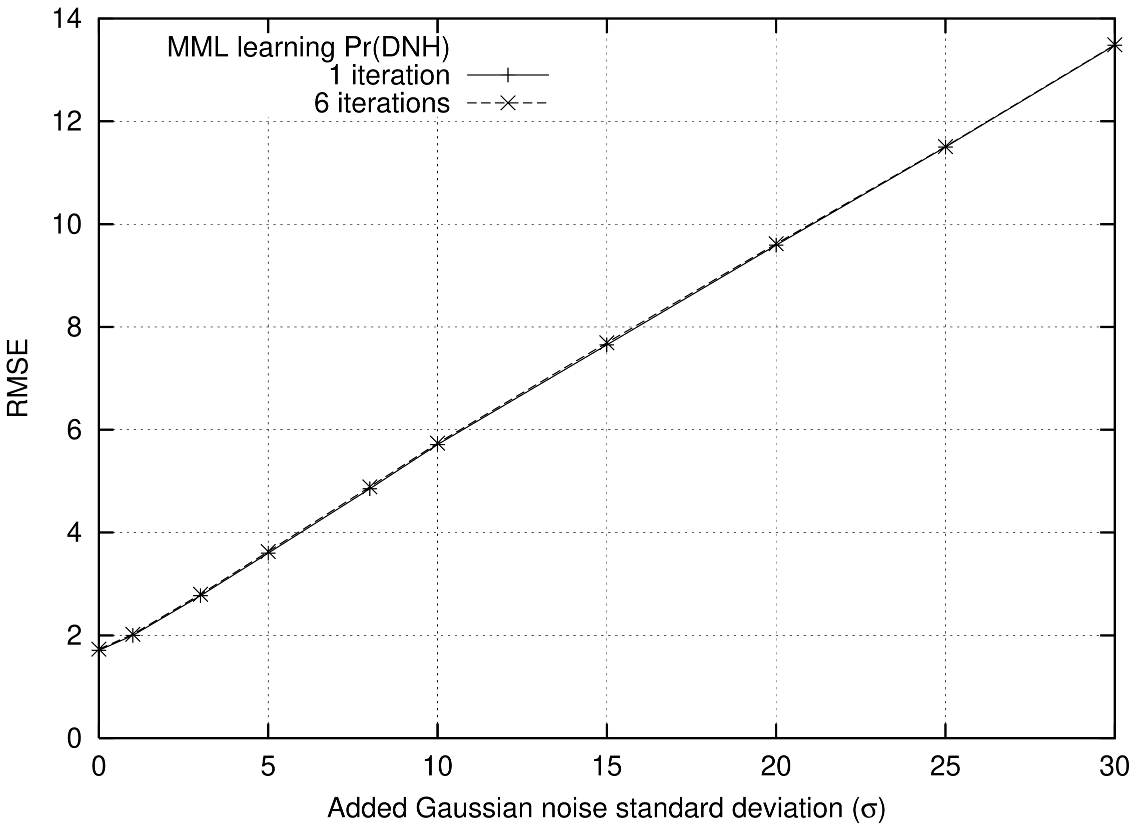

Figure 5.40: Learning $\Pr(DNH)$: RMSE comparison for `montage`, with σ estimated.

Figure 5.41 shows proportion of times that DNH was chosen as the best model by the two algorithms. Iterated MML learns to use DNH less often than its static counterpart, but this makes little difference to the RMSE results. A possible explanation is that Figure 5.41 only plots how often the null model was the "best" model. Because there are only three candidate models, the "best" model could have posterior probability as low as 0.34 or so. After posterior blending, the influence of the "best" model may actually be quite low compared to the sum influence of other candidates.

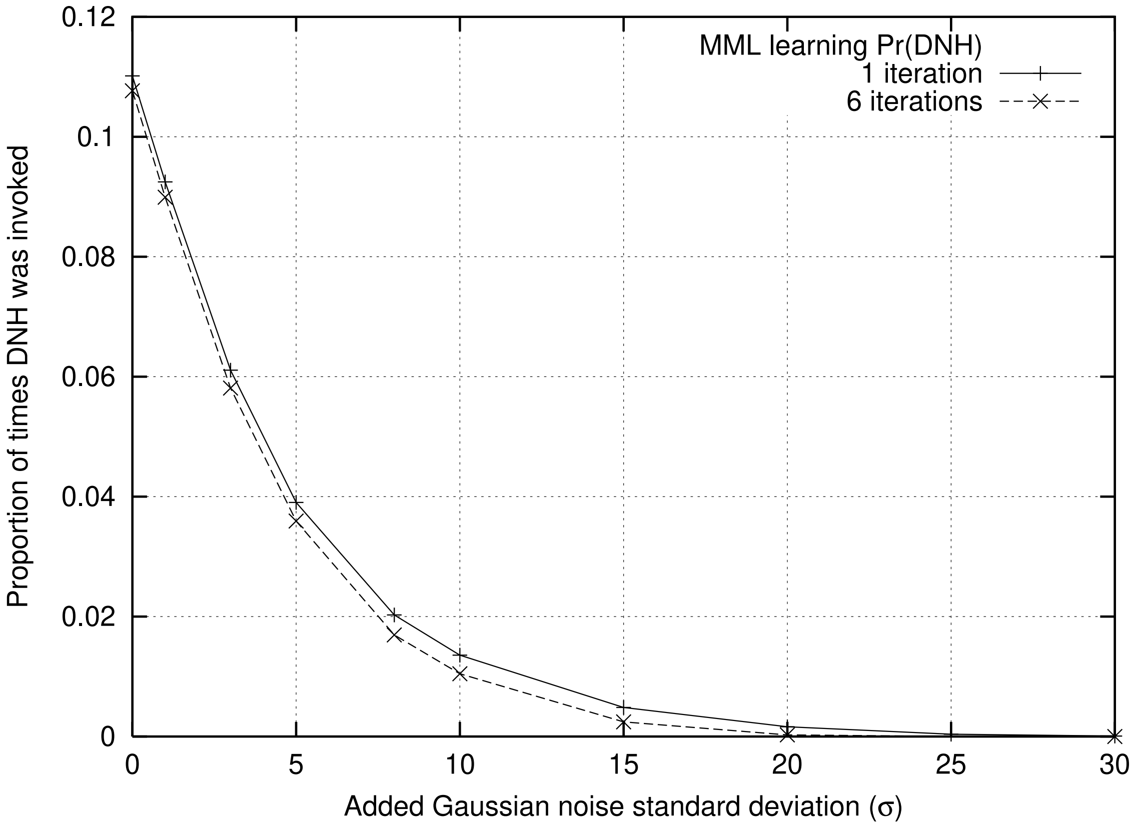

Figure 5.41: Comparison of DNH invocation for `montage`, with σ estimated.

5.14 Incorporating more models

Learning a non-uniform prior for the segment maps allows some spatial information to be incorporated into the MML denoising algorithm. This typically manifests itself at higher noise levels. In most circumstances, when σ is high, FUELS has little choice but to choose $k = 1$. The MML algorithm may choose the alternative $k = 2$ thresholded model if its message length is short enough. However, it is still hampered from having to choose from a pool of only 3 candidate models: $k = 1$, one thresholded $k = 2$, and DNH.

Under the MML framework it is a simple matter to incorporate more candidate models. For example, all valid segmentations (Section 5.3.3) could be tested, or a few extra $k = 2$ segmentations produced using different thresholds could be added. This thesis will take the simplest route and consider all 256 canonical 3×3 binary segment maps. The examination of all possible segment maps increases the model pool from 3 to 257 candidates, including DNH. The posterior distribution should be more varied, and the data driven segment map prior should more accurately reflect the local image structures. However, this is at the expense of nearly a one hundred fold increase in computation.

Figure 5.42 compares FUELS with two versions of the MML algorithm. Let "MML-2" be the familiar version which uses 2 candidate local segmentations plus DNH, and let "MML-256" be the new version which includes all 256 models with unique segment maps. Both algorithms are run for 6 iterations to learn $\Pr(DNH)$ and $\Pr(\mathbf{c})$. When $\sigma \leq 5$, there is no noticeable difference between MML-2 and MML-256. For images with little noise, thresholding adequately segments the window. As the noise increases, the RMSE performance of MML-256 improves, until $\sigma = 30$, where it worsens again.

Figure 5.43 plots the proportion of times that the candidate with the shortest message length was $k = 2$. It is obvious from the graph that considering more 2-segment models per local region significantly increases the chance of choosing a specific 2-segment model as the best model. This supports the argument that thresholding is insufficient when the noise level is high. Although not shown, the increase in $k = 2$ is mostly at the expense of $k = 1$. DNH usage of the MML algorithms remained relatively unchanged.

5.15 Improving the running time

The MML-256 algorithm has a superior ability to discern the local structure of a noisy image. This power comes at the cost of higher computational requirements. On a serial computer, the running time per pixel is proportional to the number of candidate models considered. The parameter estimation and message length calculations for each model are independent of each other — a parallel implementation could compute them simultaneously. The comparison of the resulting message lengths could be done serially or using a recursive binary algorithm. This is probably insignificant compared to the model calculation anyway.

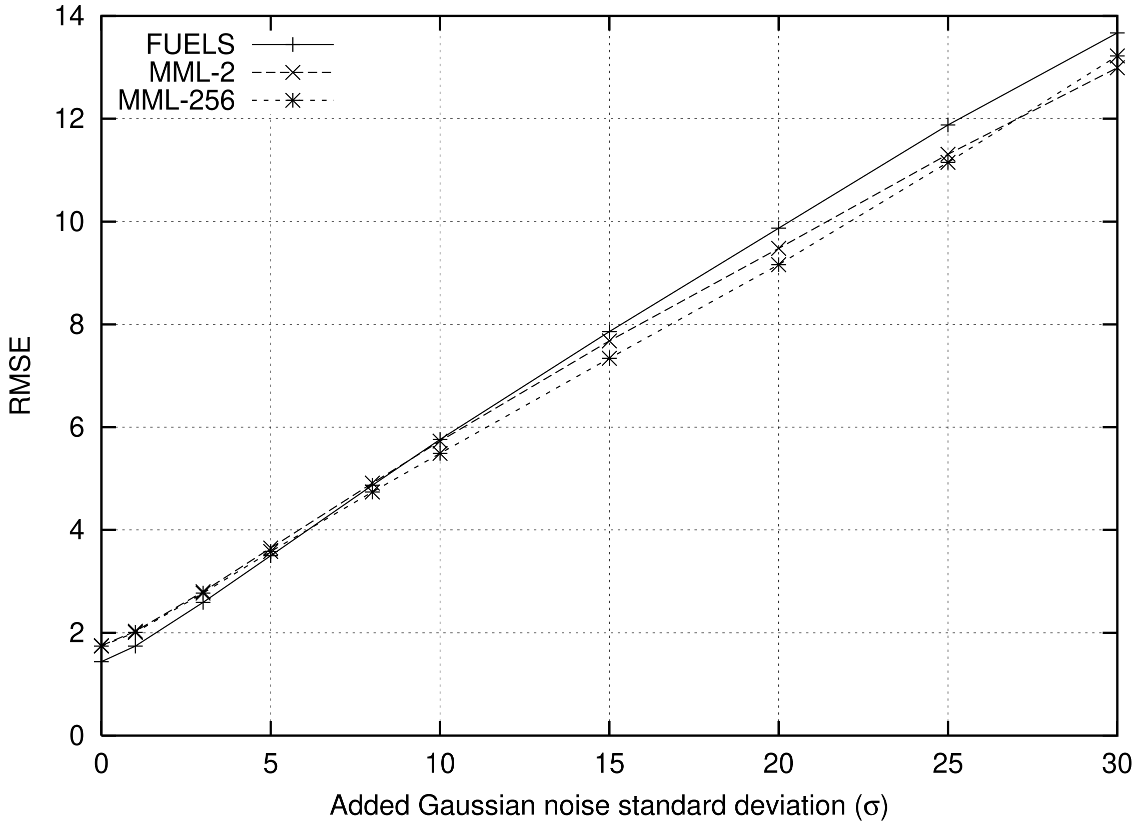

Figure 5.42: More models: RMSE comparison for denoising `montage`, with σ estimated.

If restricted to a serial computer, there is still a simple modification that can reduce the number of models which need to be considered for each window. Let σ^2 be the assumed image noise variance, and $s^2 = \text{Var}[\,\mathbf{p}'\,]$ the sample variance of the pixels in the window. If $s < \sigma$, the window is likely to be homogeneous, as the local variance is lower than the global noise variance. If it happens not to be homogeneous, a thresholded model would probably be sufficient. Thus if $s < \sigma$, MML-256 can change its mode of operation to that of MML-2. Only 3, rather than 257, candidate models need to be considered. The unconsidered models are assigned zero posterior probability. Relative to the remaining 3, the other 254 should have long message lengths anyway. Let this variant of MML-256 be called "MML-256/2".

Figure 5.44 compares the RMSE performance of the 3 MML variants, each run for 6 iterations. For $\sigma \leq 15$, MML-256 and MML-256/2 perform equivalently. As the noise in `montage` increases, the faster MML-256/2 implementation actually outperforms MML-256. Similar behaviour was observed for other images. Although not shown, the DNH usage of MML-256 and MML-256/2 is nearly identical.

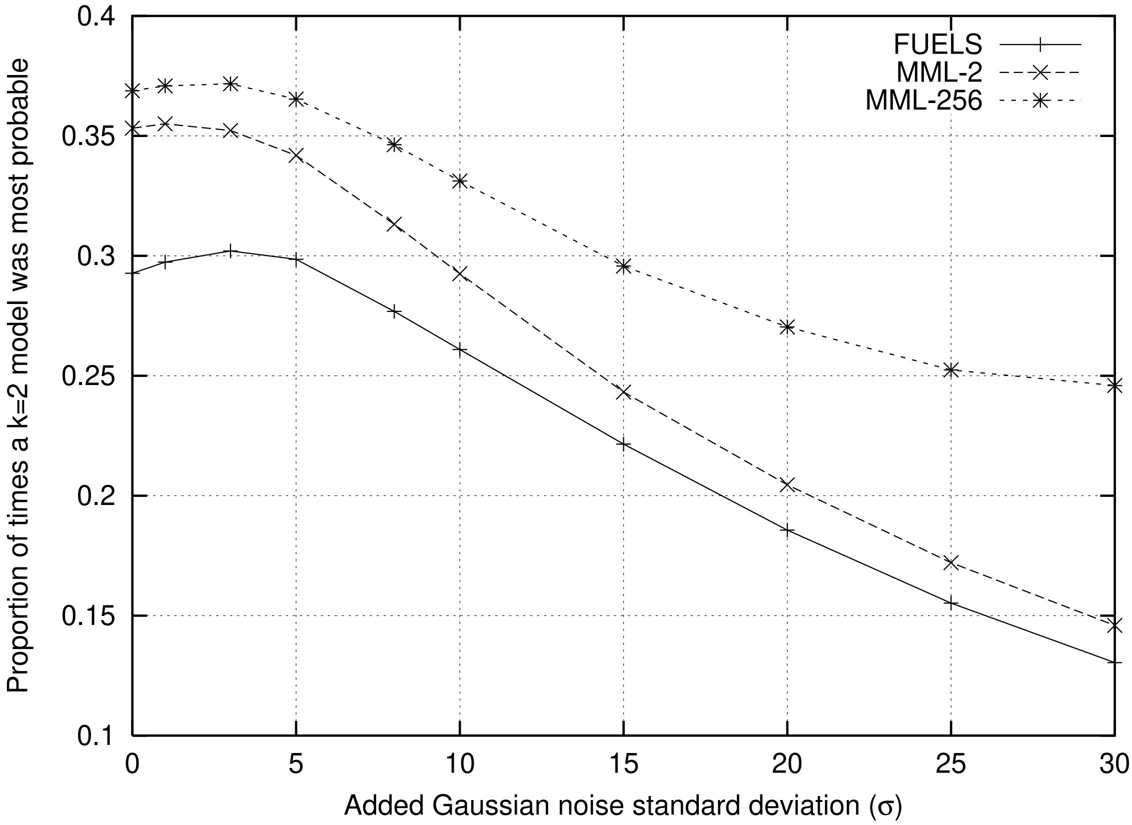

Figure 5.43: Proportion of times a $k = 2$ model was most probable in `montage`.

Figure 5.45 show the proportion of two-segment regions diagnosed by the algorithms. MML-256/2 is less likely than MML-256 to choose higher order models, especially as σ increases. Switching to MML-2 mode forces MML-256/2 to choose $k = 1$ more often. When the noise is very high, averaging all pixels in the window pays off better in terms of RMSE, even if it removes slightly more image structure. This does suggest, however, that the posterior probabilities for the various two-segment models are overstated. This could be true as the message length calculations do not strictly account for the quantization of segment means, giving slightly shorter messages than warranted by the data.

The reason for introducing MML-256/2 was to decrease the running time. Figure 5.46 compares the average processing time per pixel used by each algorithm[2]. FUELS is very fast relative to the MML variants, and barely registers on the graph. MML-2 is about 60 times slower than FUELS. Somewhat surprising is that MML-256 is only about 43 times slower than MML-2, despite having to evaluate 86 times as many models. The positive result is

[2]The experiments were run on a 1.2GHz AMD Athlon with 512MB RAM running Linux kernel 2.4.16.

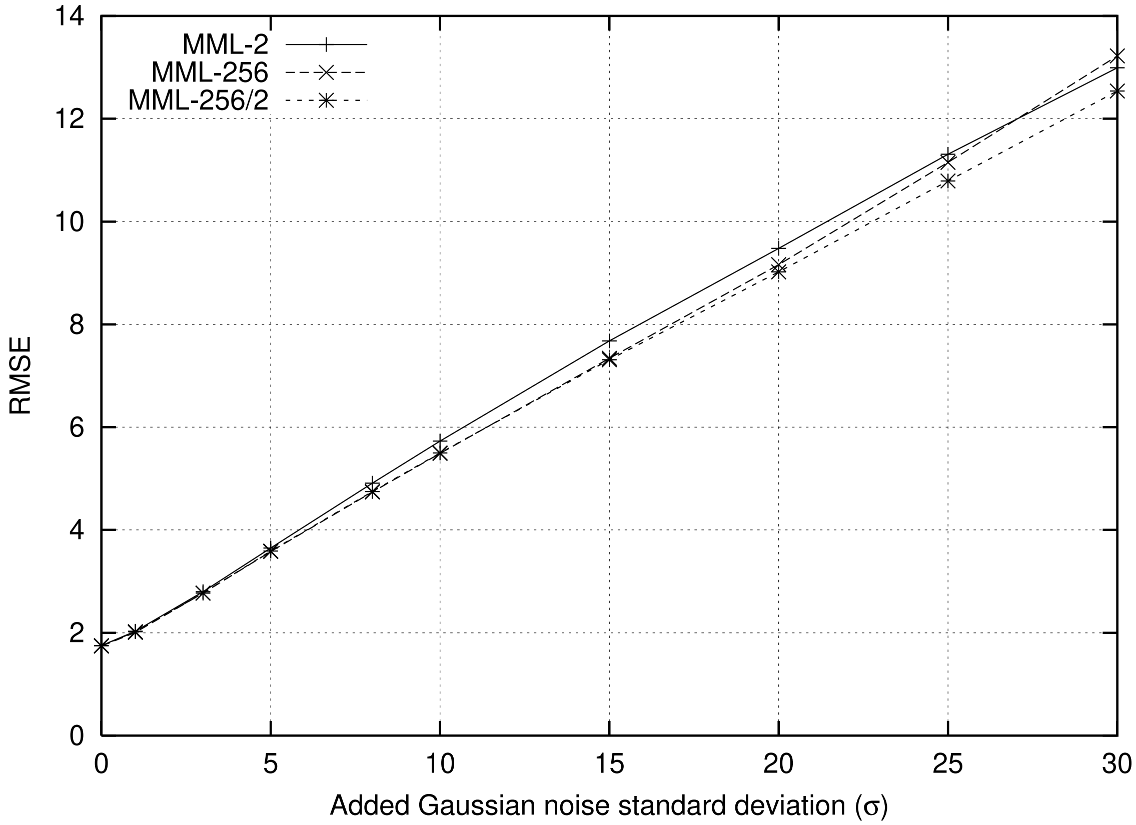

Figure 5.44: RMSE comparison for denoising `montage`, with σ estimated.

that MML-256/2 requires, on average, 60% of the computation that MML-256 does, and achieves better RMSE results at higher noise levels.

5.16 Results

This chapter has so far shown that an MML-based local segmentation criterion improves the denoising performance of the FUELS algorithm. This is a direct benefit of improved modeling of images at a local scale. Two useful MML variants have been identified. MML-2 considers 3 models in total, and is a logical extension to FUELS. MML-256/2 increases the pool of candidate models considered to 257, in an effort to push the structure recognition capabilities a little further. Together with FUELS, they represent the common trade-off between denoising ability and processing speed. In this section MML-2, MML-2/256, FUELS and SUSAN will be compared using three test images.

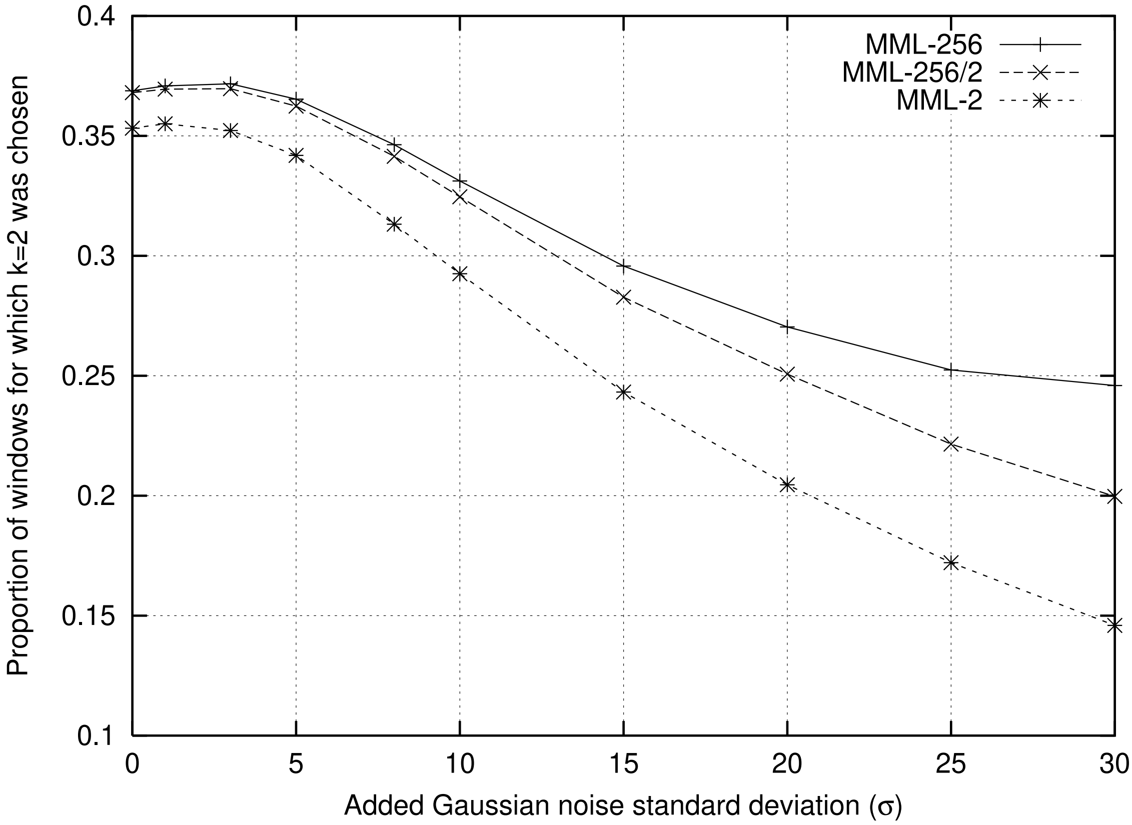

Figure 5.45: Proportion of times that $k = 2$ was deemed best when denoising `montage`.

5.16.1 The `lenna` image

Figure 5.47 provides RMSE results for denoising the original `lenna` image[3]. The MML algorithms used 6 iterations each to learn the segment map and DNH prior probabilities. Both FUELS and MML used overlapping averaging. SUSAN9 is the 3×3 variant of the SUSAN algorithm, chosen over SUSAN37 based on the results in Chapter 4. All of the algorithms utilized the estimated noise variance as described in Section 4.14. The RMSE was calculated using the original `lenna` as ground truth, despite it already containing noise.

SUSAN9 appears to perform worst for `lenna` in terms of RMSE. When $\sigma < 5$, the two MML variants are beaten by FUELS, but as the noise increases, they outperform FUELS. The poor performance of MML relative to FUELS at low noise levels is an interesting problem. That fact is that the ground truth `lenna` is already noisy. The MML algorithms could actually be producing better noiseless estimates of the "true", but unknown, `lenna` image.

[3]Available from `ftp://nic.funet.fi/pub/graphics/misc/test-images/`

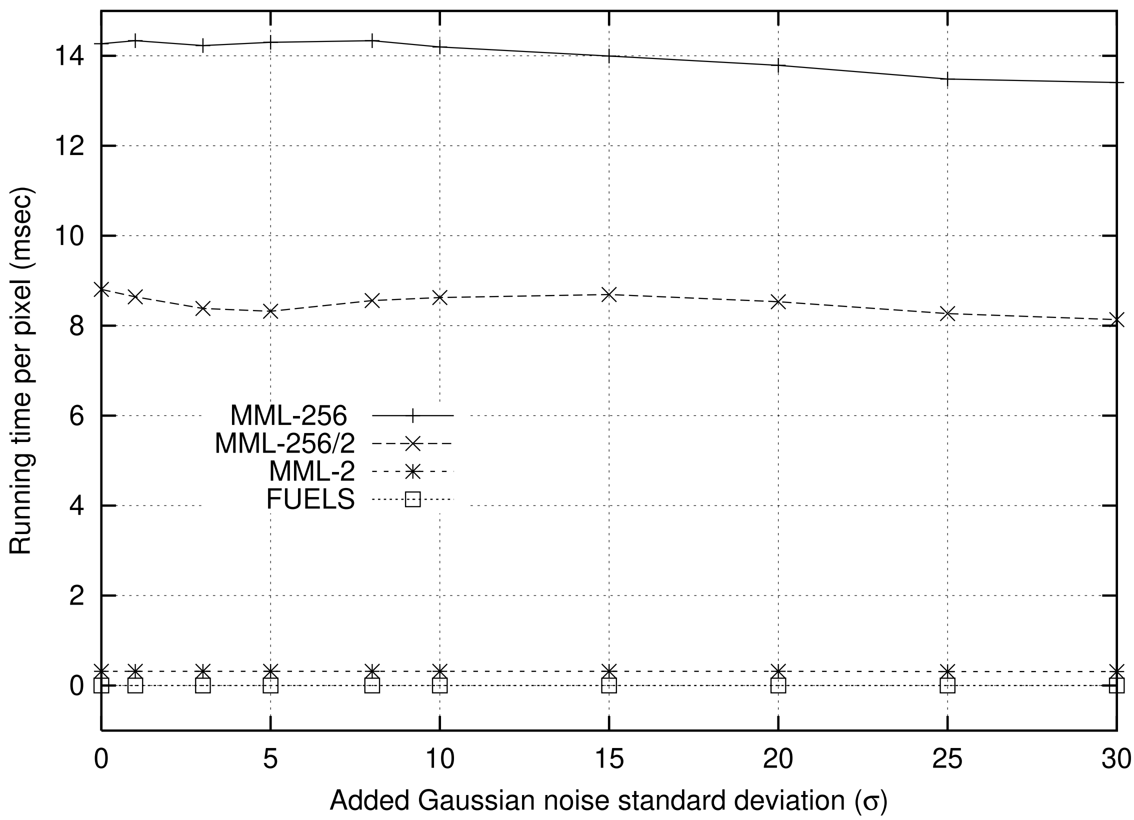

Figure 5.46: Comparison of algorithm running times for denoising `montage`.

Interestingly, MML-2 beats MML-256/2 when $\sigma \geq 20$. Although not shown, the same result occurs using the mean absolute error (MAE) metric. This unexpected behaviour was not observed with the `montage` test image in Section 5.15.

Figure 5.48 compares the enhanced difference images, relative to the original `lenna`, for the four algorithms on a 64×64 sub-image taken from the centre of `lenna`. FUELS and MML-256/2 have similar difference images, with some larger errors scattered under the eye. MML-2's output does not exhibit these same errors. This is because MML-2 inferred $k = 1$ at those points, whereas the others chose DNH or $k = 2$ (not shown). The SUSAN9 difference image does have less apparent structure in it, but its homogeneous regions are more blotchy. This means it had difficulty assimilating large numbers of pixels, resulting in poor smoothing and a higher RMSE result.

Figure 5.49 shows the worst case absolute error (WCAE) profile of the four denoising algorithms. As expected, SUSAN9 suffers at low noise levels, as it has no DNH-like capabilities. The most interesting feature is that the WCAE of MML-256/2 nearly doubles after $\sigma = 20$,

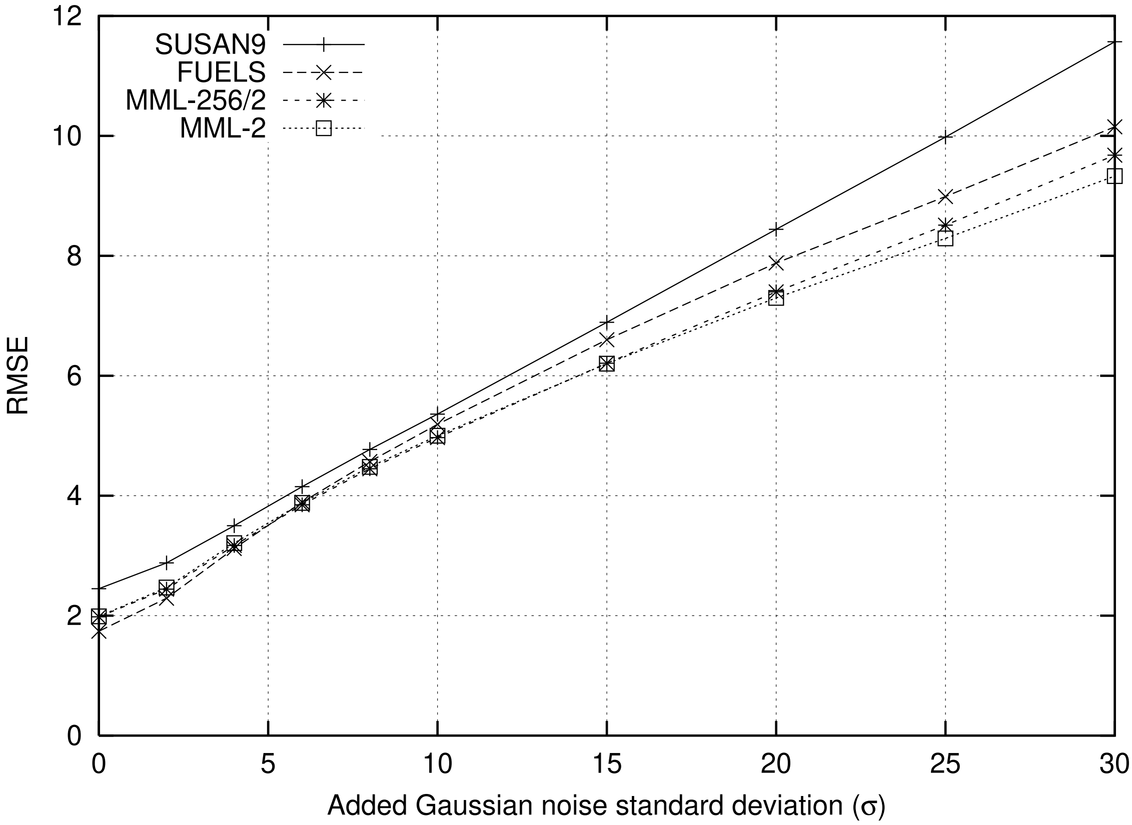

Figure 5.47: RMSE comparison for denoising `lenna`, with σ estimated.

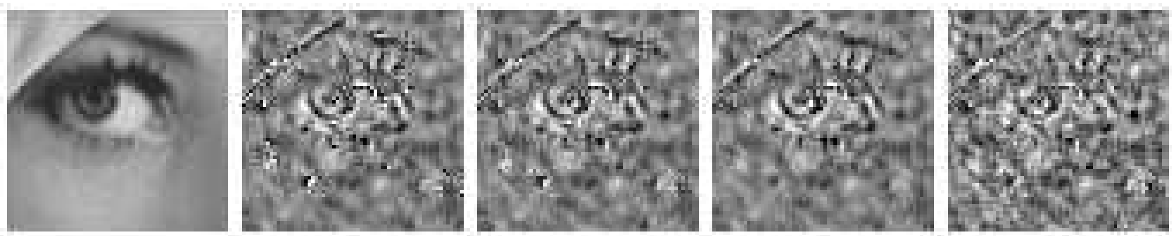

Figure 5.48: Enhanced difference images for `lenna` when $\sigma = 25$: (a) ground truth; (b) FUELS; (c) MML-256/2; (d) MML-2; (e) SUSAN9.

whereas MML-2 increases gracefully. This also helps to explain the earlier RMSE anomaly, because large errors contribute significantly to the RMSE value. This suggests that some two-segment models were made inappropriately probable by MML-256/2 when the noise level was high. If an image's properties vary spatially, a single global segment prior could produce this type of behaviour.

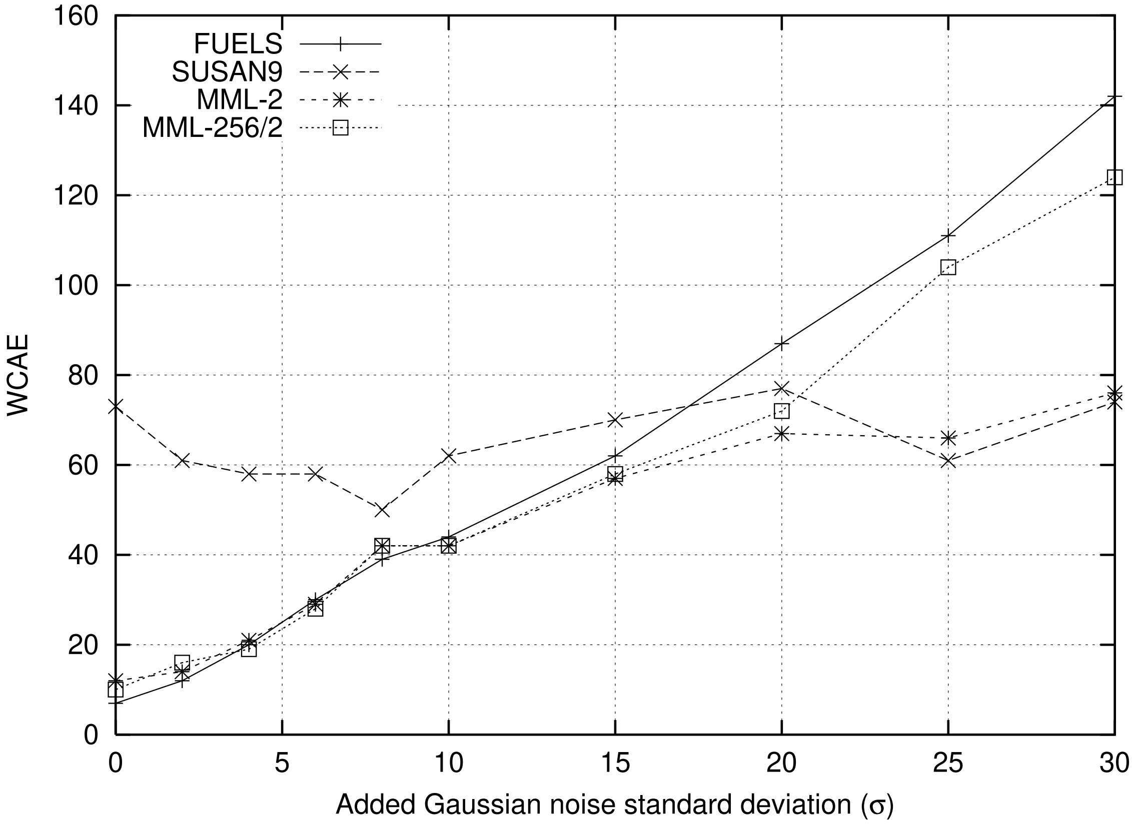

Figure 5.49: WCAE comparison for denoising `lenna`.

5.16.2 The **barb2** image

Figure 5.50 plots the RMSE for denoising the larger and more textured `barb2` image[4], originally introduced in Section 4.14.3. As usual, FUELS beats MML-2 and MML-256/2 at lower noise levels, but as σ increases, the disparity between the three is more apparent. MML-256/2 proves that a higher level of image modeling can improve RMSE results to a significant level. When $\sigma \geq 10$, it beats FUELS and SUSAN by up to 1.8 RMSE units.

The enhanced difference images when $\sigma = 25$ are given in Figure 5.51. They are from a 32×32 sub-image from the centre of `barb2`. These difference images have similar properties to those in Figure 5.48 for `lenna`. The errors in MML-2's difference image are lower in magnitude, particularly the small circular cluster on the left hand side. This cluster has occurred because the noisy image (not shown) had some very noisy pixels clumped together.

The WCAE results for `barb2` are given in Figure 5.52. The deficiency of SUSAN9 at low

[4] Available from `http://www.csse.monash.edu.au/~torsten/phd/images/`

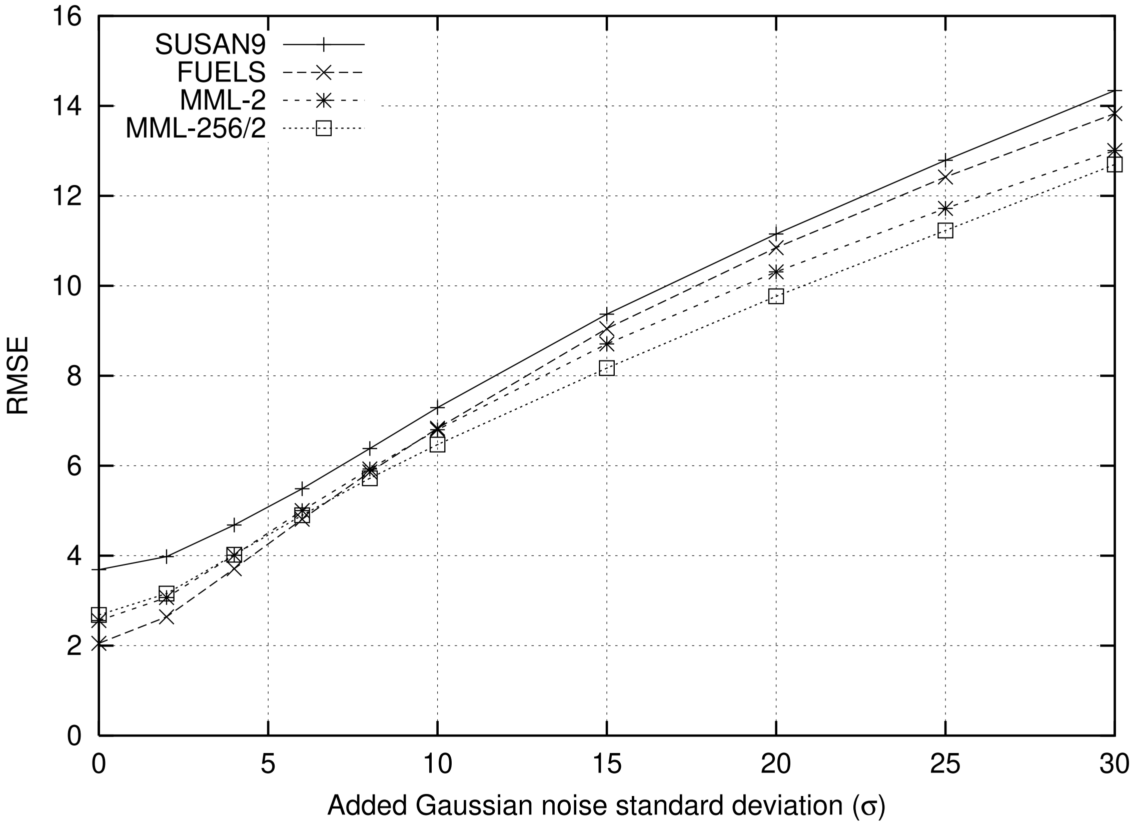

Figure 5.50: RMSE comparison for denoising `barb2`, with σ estimated.

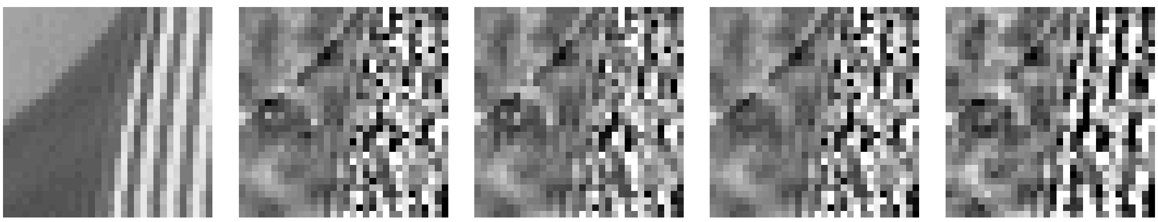

Figure 5.51: Enhanced difference images for `barb2` when $\sigma = 25$: (a) ground truth; (b) FUELS; (c) MML-256/2; (d) MML-2; (e) SUSAN9.

noise levels is again obvious, but at higher noise levels it does better than the other techniques. SUSAN9's behaviour, partially dependent on the threshold, seems to be ambitious for low σ, but successful for high σ. MML-256/2 has a lower WCAE for most high values of σ, which is much better than its behaviour observed for `lenna`. This is perhaps due to the distribution of segment maps being more uniform across the image, in particular the large number of crisp edges on the wicker chair and clothing.

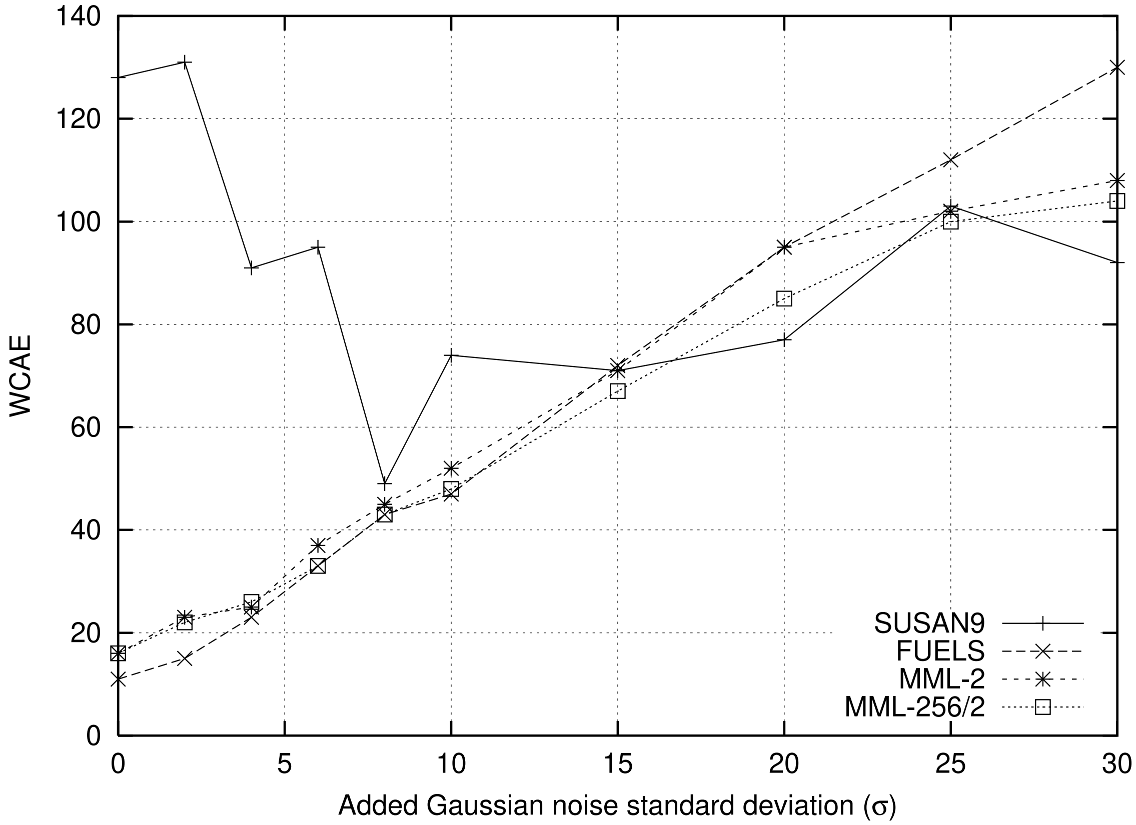

Figure 5.52: WCAE comparison for denoising `barb2`.

5.16.3 The `camera` image

The `camera` image[5] is shown in Figure 5.53. It has dimensions 256×256, which is smaller than most test images used so far. It consists of a large number of homogeneous regions. Although the man's coat appears uniformly dark, it hides a lot of low contrast structure.

Figure 5.54 compares the RMSE denoising performance on `camera`. Unlike for `lenna` and `barb2`, FUELS and the MML variants have very similar results, while SUSAN9 did relatively worse. MML-2 and MML-256/2 are indistinguishable. The large number of homogeneous regions makes it difficult for MML-256/2 to stand out, as it gains only from considering more two-segment models. The fact that `camera` contains fewer pixels also means that the MML algorithms have less data with which to re-estimate the priors.

The difference images for a homogeneous, possibly mildly planar, 32×32 sub-image of `camera` are given in Figure 5.55. FUELS' limited model selection criterion has obviously

[5] Available from `ftp://nic.funet.fi/pub/graphics/misc/test-images/`

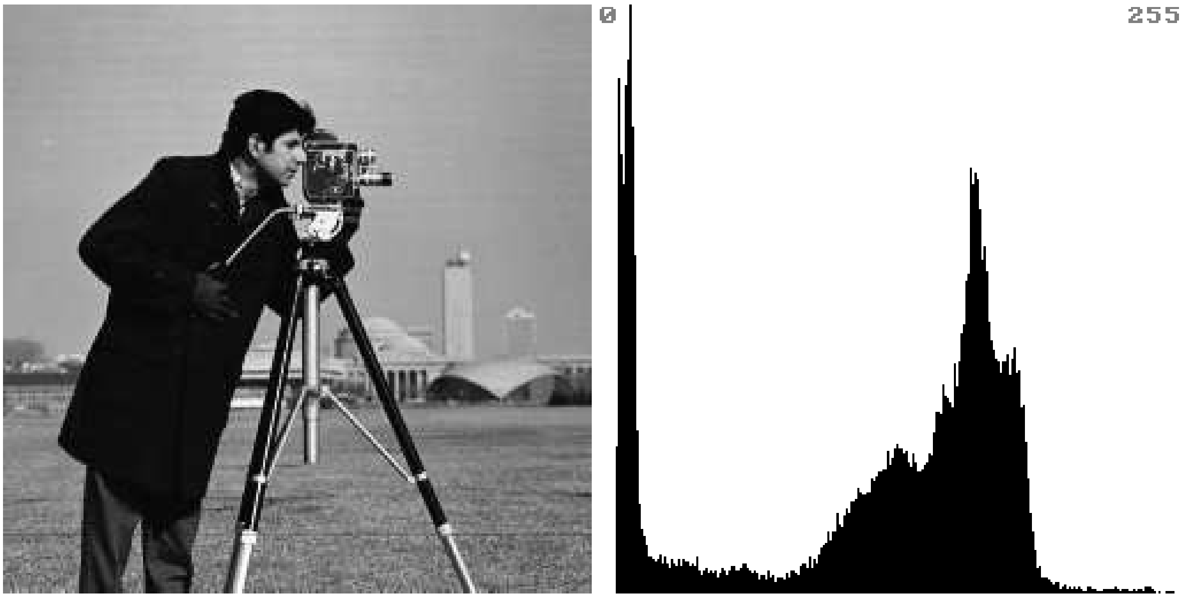

Figure 5.53: The 256×256 8 bit `camera` image and its histogram.

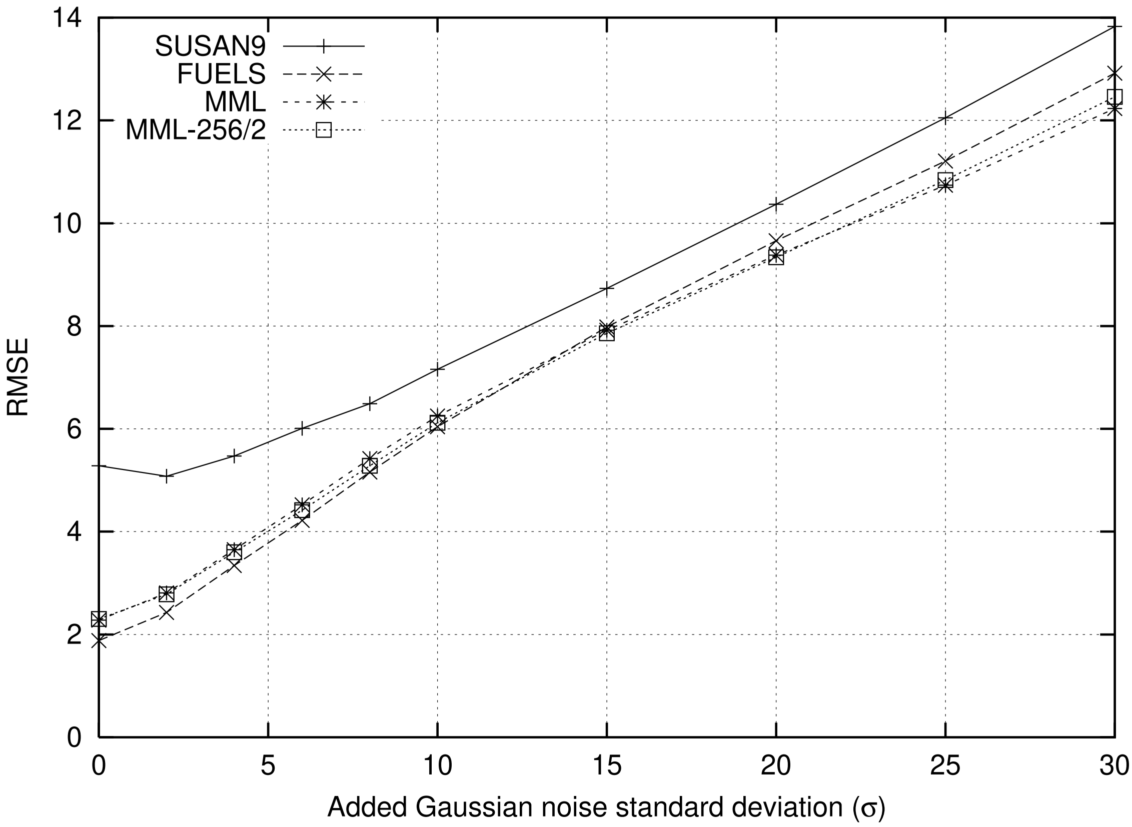

Figure 5.54: RMSE comparison for `camera` image, with σ estimated.

failed on a few pixels, but this is understandable given that $\sigma = 15$. MML-256/2 also had a little trouble with the same points, suggesting that it may be incorrectly favouring two-segment models. Recall from Section 5.7 that the *approximate* expected message length is calculated for each candidate. This approximation may be mildly biased toward 2-segment models. MML-2 had no difficulty ignoring the "correctly" diagnosed glitch in the original image. It is less likely to find an (un)suitable two-segment model, as it only considers one.

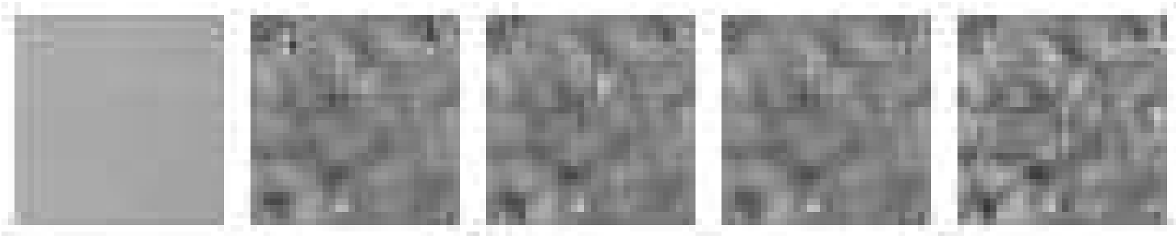

Figure 5.55: Enhanced difference images for `camera` when $\sigma = 15$: (a) ground truth; (b) FUELS; (c) MML-256/2; (d) MML-2; (e) SUSAN9.

5.17 Conclusions

In this chapter, the MML methodology was applied to local segmentation. The simple mean separation criterion used by FUELS was replaced by a message length comparison between candidate models. Each message was a concise, decodable encoding of the noisy pixels. The message with the shortest length was deemed to contain the most appropriate model for the local region. By using MML, the arbitrary constant, C, that FUELS required, was eliminated. This introduced the possibility of diagnosing the presence of two segments with close, but distinct intensities. FUELS was incapable of this in very noisy images.

MML made it straightforward to incorporate alternative models into the pool of candidates being considered. All that was required was that the models' messages be decodable. The "do no harm" (DNH) concept was first introduced by FUELS to reduce its worst case denoising performance. It was a simple matter to incorporate a DNH model into the MML framework. The DNH candidate simply encoded the pixel values as they were, without respect to any segment map or means. This information theoretic approach to DNH was found to improve RMSE results compared to FUELS' simpler method.

MML chooses a single best model from the pool of candidates, the so-called MAP model. The message length of a candidate model may be interpreted as a posterior probability for that model. Posterior probabilities provide an objective measure of the confidence in each model. It was shown that if one is only interested in the resulting local approximation, and not the details of the model, then blending over all models weighted by their posterior probability was a good thing to do. This alters results when other candidate models have message lengths close to, but just short of, that of the MAP model. Posterior blending was found to improve RMSE performance, especially for very noisy images.

The ease with which a DNH model was incorporated paved the way for the addition of more segment based models. For a 3×3 window, there exists only 256 canonical binary segment maps, including the homogeneous one. It was a simple matter to enumerate all these models and add them to the pool. Along with DNH, this brought the total number of models considered per pixel to 257. Doing this effectively incorporated spatial information into the local segmentation, as segment membership was no longer restricted by pixel intensity. The RMSE results improved, but at the expense of a large increase in running time. The running time was halved by switching to only 3 candidate models if the local variance was low enough to warrant a more limited search.

Using a large pool of candidate models in conjunction with posterior blending may lead to a more accurate local approximation. Small improvements in the accuracy of the local approximation may not be noticeable because the final approximated pixel values are rounded to their nearest integers. There may be applications where one is willing to pay for a more accurate local approximation. For example, examination of small features where there is little contrast may be important in medical imaging applications.

Although an image may use the full range of intensities, a sub-image may only use a small range. If the pixel values in the sub-image were known to higher accuracy, an enhanced version may be more accurate than if the original integer pixels were used. There was not enough time left to pursue this line of investigation. In future work, a photo-realistic ground truth image could be generated synthetically. The pixel depth of this image could be far greater than 8 bits. Varying amounts of noise could be added to it, and the pixel values of the resulting image rounded to 8 bit precision. A denoising algorithm which produces floating point pixel estimates, such as FUELS or MML-256/2, could then be compared to the more accurate ground truth image. This would determine whether more accurate local approximations contain any further meaningful information.

When segmenting 3×3 regions, only a small amount of pixel data is available, and the model component of a two part MML message becomes significant. This places demands on efficiently encoding the model, demands which may be less important for larger windows. The length of the model part is directly related to the prior probability distributions used. The prior for the segment maps could be directly controlled. It was shown that changing it to a more meaningful non-uniform distribution improved results. The logical extension to this was to learn the prior from the noisy image itself. The idea of data-driven priors was applied to the MML denoising algorithm, something FUELS was incapable of doing. It was found that the priors converged in about 6 iterations. The RMSE results were further improved using this technique.

The result of this chapter is two useful techniques for local segmentation. "MML" considers 3 candidate models, and is the logical extension to FUELS. "MML-256/2" extends this even further to consider 257 models in most cases, but only 3 models when a full search is deemed unnecessary. The subjective analysis also showed the MML methods to produce better local approximations, on average, than FUELS and SUSAN. However, this small improvement only came with a lot of computational and developmental effort, suggesting that FUELS and SUSAN are already very good. The MML-256/2 algorithm could potentially be improved by deriving a more accurate approximation for the expected message length. In general, the use of MML-256/2 is only warranted when a proper analysis is vital, the noise level is very high, or the best possible local approximation is required.

5.18 Related work

An early form of the work in this chapter appeared in *Proceedings of the International Picture Coding Symposium 1997* [ST97]. In that paper, 11 candidate segmentations of the 3×3 window were considered: 3 homogeneous models using the mean, median and midpoint as the reconstructed value, and 8 binary models using the 8 possible thresholds to generate segment means. These models were constructed differently to those described in Section 5.4.

The first event stated whether $k = 1$ or $k = 2$, with both events assumed equally likely. If $k = 1$ the representative value for the block was assumed uniform on $[0, Z-1]$. If $k = 2$, the next event was how many pixels, m, were in the cluster with the lower mean, assumed uniform on $[1, 8]$. This was followed by a segment map describing the spatial arrangement of the cluster labels. Because m was already known, there were only $\binom{9}{m}$ possible bitmaps, which were assumed equally likely. The first cluster mean, μ_1, was assumed uniform on $[0, Z-1]$. Because the second mean was known to be higher than the first, a uniform distribution on $[\mu_1 + 1, Z - 1]$ could be used. Finally, the residuals between the local approximation and the noisy data were encoded. These were treated differently than in the MML method of this chapter. Two *discrete* residual distributions were kept, one for use by $k = 1$ models, and one by $k = 2$ models. No numerical integration was required as all segment parameters were quantized to their nearest integers.

Only the centre pixel of the most probable resulting local approximation was used for denoising. The ideas of overlapping estimates and "do no harm" had not been developed yet. The algorithm was also iterated such that the posterior probabilities were fed back to be priors, just as MML does in Section 5.13. The idea of posterior blending was not yet developed, and the global posteriors were only updated from the most probable model at each point, rather than a weighted contribution from all models.

Chapter 6

Extensions and Further Applications of Local Segmentation

6.1 Introduction

In Chapters 4 and 5 the principle of local segmentation was applied successfully to removing additive noise from greyscale images. The relative performance of different ideas and parameter settings were evaluated and compared using the RMSE criterion and difference images. Image denoising was naturally suited to demonstrating some advantages of examining images from a local segmentation perspective. However, one should not be led into thinking that its usefulness ends there.

Local segmentation may be considered a core component of many other image processing applications. A local segmentation analysis provides a way to split an image into its signal and noise components. The signal may then be enhanced further without interference from the noise. It provides a snapshot of the structural features at each point in an image, identifying which pixels belong together and where boundaries occur. Figure 6.1 espouses the application of local segmentation as a first step in low level image processing, from which many other tasks may be derived.

This chapter will illustrate how local segmentation could be applied to pixel classification, edge detection, pixel interpolation and image compression. It will also examine alternative

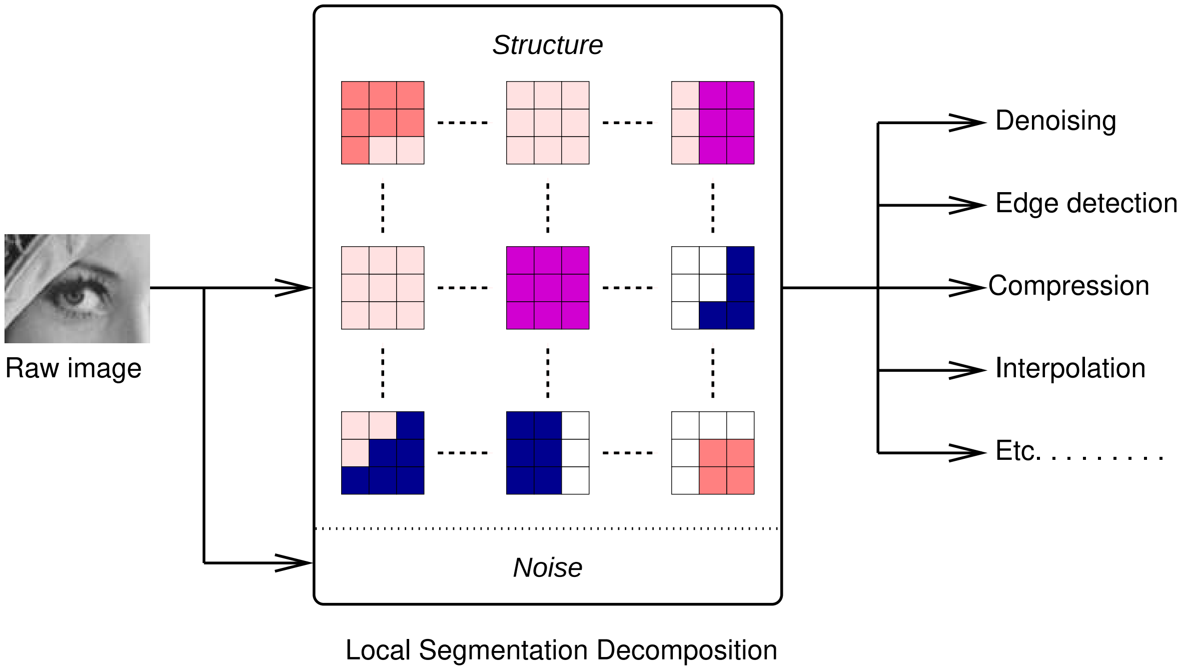

Figure 6.1: Local segmentation decomposition as a core image processing component.

image models, noise models, and segmentation algorithms for use in local segmentation. The extension to different data types will also be considered. Most of the ideas will only be presented in brief, as a full study is beyond the scope of this thesis. However, some topics do include preliminary implementations and results.

6.2 Different noise models

The FUELS and MML techniques described in this thesis have focused on the removal of additive zero-mean Gaussian noise, independently and identically distributed for each pixel. In this section it will be shown how different noise models, such as impulse and multiplicative noise, may be incorporated into the local segmentation framework.

6.2.1 Impulse noise

Recall Section 2.5, which described the nature of ergodic impulse noise. Pixels in an impulse affected image have probability q of being corrupted. This corruption takes the form of a pixel being completely replaced by a random pixel value. This differs from additive noise where some information about the original pixel value is retained. The random replacement pixel is usually assumed to be drawn uniformly from the full intensity range $[0, Z-1]$. A pixel which has been affected by impulse noise is known as a *corrupt pixel*.

There is a small chance that a corrupt pixel will have an intensity very similar, or equal, to its original value. However, most of the time its value will stand out from its neighbours. The median filter in Section 3.4.4 is sometimes well suited for filtering impulse noise. In homogeneous regions, the median filter can cope with up to 50% of the pixels being corrupt and still produce a reasonable denoised value. In heterogeneous regions it is less well behaved.

In terms of local segmentation, a corrupt pixel may be considered a very small segment having an intensity sufficiently different from its neighbours. In the simplest case, this segment would consist of one (corrupt) pixel and be surrounded by other larger segments. It is possible for corrupt pixels to be adjacent, resulting in small, spatially connected groups of corrupt pixels. If an adjacent pair of corrupt pixels is similar enough in intensity, it could even be (mis)interpreted as a valid two pixel segment.

Let the *minimum acceptable segment size* (MASS) be the smallest sized *local* segment that should be interpreted as structure rather than noise. Recall the 3×3 median filter from Section 3.4.4, which was unable to filter the centre pixel correctly when the intersection of its segment and the windows contained fewer than 5 pixels. It could be considered as having a MASS of 5. The centre weighted median from Section 3.4.4 is able to adjust its effective MASS by varying the centre weight, c. When $c = 2$, the MASS reduced to 4.

Ideally, the MASS should depend on q, the fraction of corrupted pixels in the image. The MASS could be a global parameter to the local segmentation process, in the same way that σ is for additive noise. It could be supplied by the user, or somehow estimated from the image itself. Equation 6.1 gives an expression for the probability that a given pixel, p, will be surrounded (in an 8-connected sense) by exactly n corrupt pixels, where q is the per pixel probability of impulse noise.

$$\Pr(p \text{ has exactly } n \text{ corrupt neighbours}) = \binom{8}{n} q^n (1-q)^{8-n} \tag{6.1}$$

The probability of a given pixel having no corrupt neighbours is $(1-q)^8$. The probability of that pixel being itself corrupt is q. Therefore the probability of a randomly selected pixel being isolated and corrupt is $q(1-q)^8$. When $q = 10\%$, this is only 4.3% of pixels, meaning 5.7% of corrupt pixels should have one or more corrupt neighbours. This corresponds to the majority of impulse noise occurring in clumps rather than isolation, with pairs being most common. Fortunately, these pairs are not necessarily similar in intensity and could be diagnosed as two separate single pixel segments.

The multi-class version of FUELS from Section 4.12 could be adapted to additionally remove impulse noise. For each pixel, the optimal local segmentation could be determined, using the estimated additive noise level, σ, as before. If the centre pixel's segment consists of fewer than MASS pixels, it could be considered an impulse segment. The pixels from an impulse segment need to be assigned new values. The mean of the largest segment not containing the centre pixel could be used. For a 3×3 window, that segment will always contain some pixels neighbouring the centre pixel. However, for larger windows this would not necessarily be the case. A more advanced version of this idea could take that into consideration when choosing replacement pixel values.

Results

Figure 6.2 illustrates the application of this idea to the middle section of `montage` contaminated by 5% impulse noise. The multi-class FUELS algorithm was modified to handle both additive and impulse noise as described earlier. Both DNH and overlapping averaging were disabled because they would only complicate the implementation. Results for MASS=2 and MASS=3 are included. The outputs of the 3×3 median (effective MASS=5) and centre 3-weighted median filters (MASS=4) are also present for comparison. The image margins should be ignored, as the method used for handling "missing pixels" is not entirely appropriate when impulse noise is present in the image.

When MASS=2, single pixel segments will be obliterated, while larger segments will remain intact. In Figure 6.2 it can be seen that the remaining noise mainly consists of pairs of pixels which are similar in intensity. But this means two-pixel segments, such as the dot on the '`i`', remain intact. When MASS=3, the modified FUELS manages to remove most of the

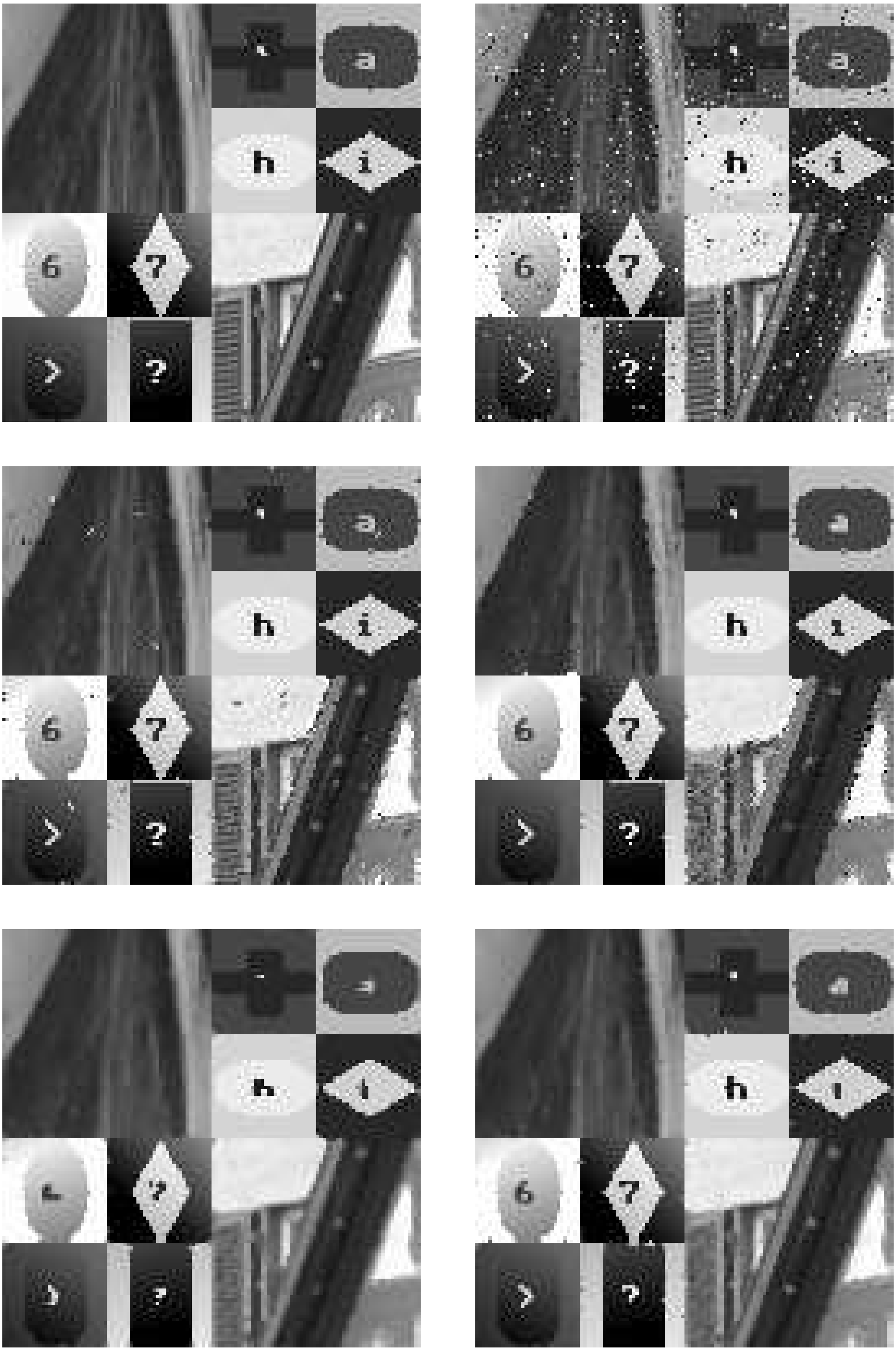

Figure 6.2: (a) clean image; (b) with $q = 0.05$ impulse noise; (c) multi-class FUELS, MASS=2; (d) MASS=3; (e) median; (f) weighted median.

noise that remained when MASS=2. But this comes at the expense of structure loss on the window shutters, the dots on '`j`' and '`?`', and the hole and tip of the '`a`'. At first glance the median filtered image looks very clean, but much of the fine detail and the letters have been seriously damaged. The weighted median appears to have done a better job overall, but also suffers from corruption of the letters. FUELS with MASS=2 did best of all on the letters.

Figure 6.3 performs the same experiment, but with 10% ($q = 0.1$) corruption. In this environment, single pixel impulse segments are less likely to occur than when $q = 0.05$. The MASS=2 filter still does reasonably well under these conditions — the letters are all legible — but much noise still remains. When MASS=3, less noise remains, but some of the letters are unrecognizable. The two median filters performed similarly, but the centre weighted median did manage to retain more of the letter structure.

Table 6.1 contains RMSE results for the filtered outputs demonstrated in Figures 6.2 and 6.3. When $q = 0.05$, the weighted median and MASS=2 do best. This correlates with the subjective evaluation. When $q = 0.1$, the weighted median again does best in terms of RMSE. MASS=2 does worst, as it can only remove single pixel noise segments, which are in the minority for 10% corruption. MASS=3 and median perform similarly. Once again, this does not differ substantially from the conclusions drawn from the earlier visual examination.

Method	$q = 0.05$	$q = 0.1$
Median	16.4	17.3
W. Median	11.1	13.5
MASS=2	12.7	19.2
MASS=3	15.7	17.9

Table 6.1: RMSE results for filtering impulse noise from `montage`.

Conclusions

One method for adapting the multi-class FUELS algorithm to remove impulse noise was presented. It functioned by treating all local segments containing fewer than a specified number of pixels, the MASS, as noise. Corrupt pixels were replaced by the denoised mean of their largest neighbouring segment. Additive noise was simultaneously removed for those segments not diagnosed as impulse noise.

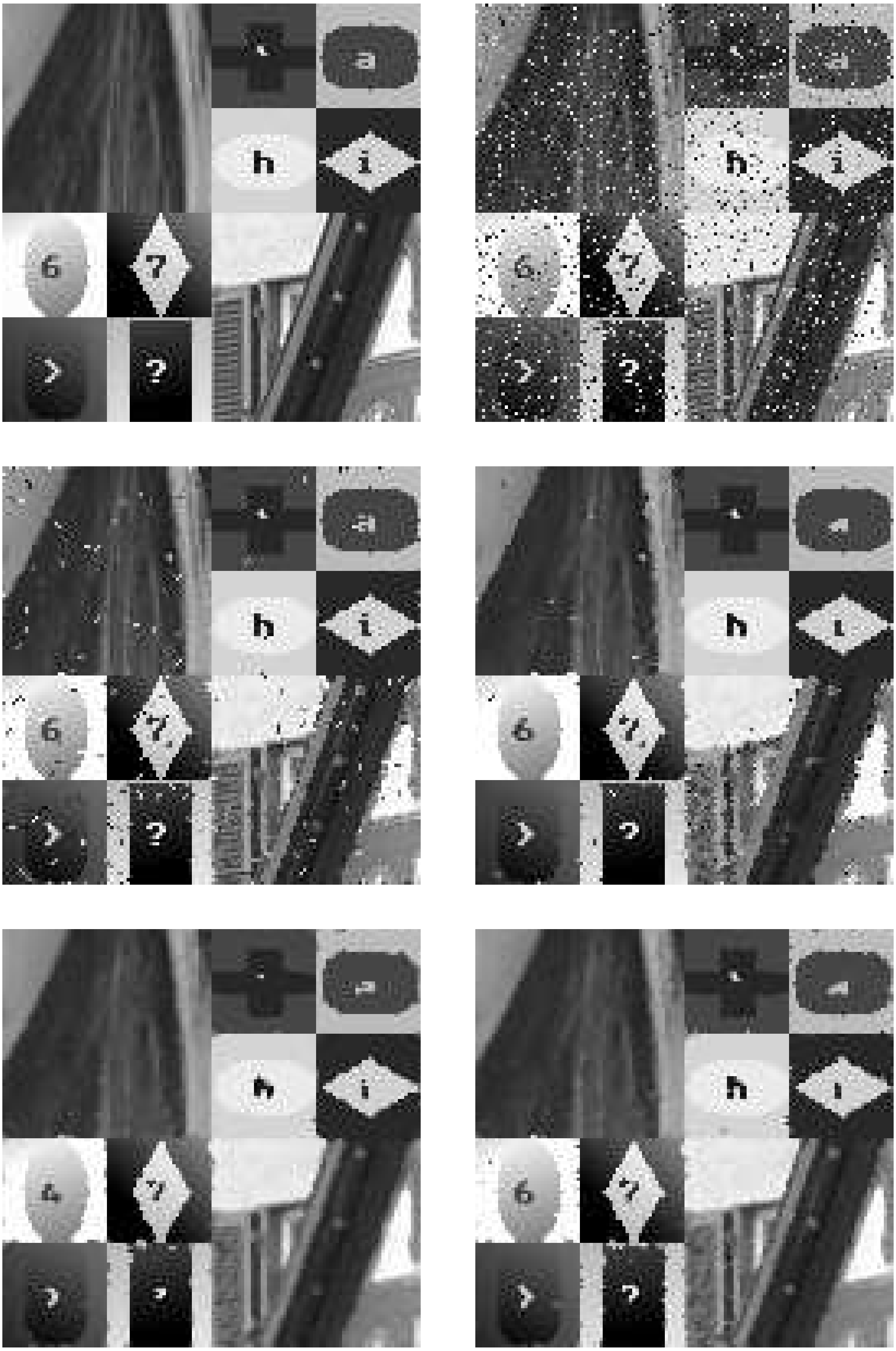

Figure 6.3: (a) clean image; (b) with $q = 0.1$ impulse noise; (c) multi-class FUELS, MASS=2; (d) MASS=3; (e) median; (f) weighted median.

The MASS=2 implementation works well for 5% corruption. It tended to preserve more structure than the median varieties, but could not cope with the occasional pairs of corrupt pixels with similar intensities. As the impulse noise level was increased, higher values for the MASS are required to cope. In those situations it tends to leave more noise behind, but preserve more structure in the areas where it took it away. This should make it more amenable to multiple passes through the data.

The best MASS to use depends on the amount of impulse noise present. It is preferred that the algorithm determine the most suitable MASS for a particular image automatically. One possibility is to apply filters with known MASS and examine the number of pixels which suffer gross changes. From that number it may be possible to estimate the percentage noise corruption, q, and hence a suitable MASS.

6.2.2 Multiplicative noise

Multiplicative noise was first introduced in Section 2.5. It may be considered additive noise, with the noise term being proportional to the intensity of the original pixel. Equation 6.2 illustrates the situation, where $n(x, y)$ is a random value from some unspecified distribution.

$$f'(x, y) = f(x, y) + n(x, y)f(x, y) \tag{6.2}$$

It is common to assume that the multiplicative noise function $n(x, y)$ is normally distributed as $\mathcal{N}(0, \sigma^2)$, with σ constant over the image. Multiplying a variable $\sim \mathcal{N}(0, \sigma^2)$ by a scalar, a, results in a variable $\sim \mathcal{N}(0, a^2\sigma^2)$. In Equation 6.2, a would be the intensity of the original pixel. Thus multiplicative noise could be considered additive noise where the variance is dependent on the original pixel value. There are various ways in which this behaviour could be exploited by the local segmentation algorithms developed in this thesis.

Consider a small block of pixels corrupted by the multiplicative noise model just described. Under the assumption of homogeneity, the mean, μ, of the block would be a reasonable estimate of the original intensities. FUELS would assume the block variance to be σ^2. Under a multiplicative noise model it would be approximately equal to $(\mu\sigma)^2$. Similar arguments would apply to the clusters determined under a heterogeneous model. A modified FUELS model selection criterion would use the per cluster variance to determine suitability, rather than a common noise variance. A simpler alternative is to simply use the average brightness of the block to determine a local noise variance to be used by all derived clusters.

The treatment just described is not ideal. The mean is not necessarily the best estimate to use, as bright pixels are less reliable than dark ones. The binary clustering algorithm used by FUELS assumes the clusters have a common variance. To properly handle multiplicative noise a more complex clustering technique should be used. However, it has been shown that with some modifications, local segmentation could be used for more than just additive noise.

6.2.3 Spatially varying noise

It is possible for the noise to vary from location to location within an image. Consider an additive zero-mean noise component in which the variance depends on the spatial location within the image. This is shown in Equation 6.3, where σ is a function of the pixel position.

$$f'(x, y) = f(x, y) + \mathcal{N}(0, \sigma(x, y)^2) \tag{6.3}$$

Under this model the variance could vary erratically from point to point. It would be difficult to determine an appropriate values of σ for each pixel. A more likely situation is for the variance to vary smoothly across the image, or to be constant within global segments but discontinuous between segments. In these cases, the variance will be mostly consistent on a local scale. The technique used by FUELS and MML to estimate the global noise variance could be applied to small sub-images centered on each pixel. Under the assumption of smoothness, the variance could be measured in a regular fashion for a proportion of all pixels. The variance for the remaining pixels could be interpolated from the estimated ones.

The k-means segmentation algorithm used by FUELS could not be applied unmodified to an image containing known, spatially varying noise. Although it does not take spatial information into account, and the value of a noisy pixel is still the best estimate of its true value, the *equal* averaging of pixels to estimate the segment mean is inappropriate. Instead a weighted mean, based on the noise variance of each pixel value, should be used. The model selection criterion would also need to be modified to handle the different variance of each pixel, and therefore of each segment.

6.3 Different structural models

Image models have two components: the structural model for describing the underlying noiseless signal, and the noise model for describing the way in which the signal was corrupted. It has already been shown how different noise models may be incorporated into local segmentation. In this section, extensions to the structural component will be briefly explored.

6.3.1 Multispectral pixels

The work in this thesis has been derived under the assumption of greyscale pixel data, but local segmentation is not restricted to this. A pixel from an alternative colour space may be interpreted as a vector, rather than scalar. Consider the "RGB" triplet sometimes used for colour pixels. If the noise is considered additive zero-mean Gaussian with the same variance in each band, FUELS and MML may be used with little modification. The k-means clustering algorithm extends naturally to multi-dimensional data, although it is more sensitive to starting conditions. The mean separation criterion would have to be modified to be a distance in RGB space, rather than a simple intensity difference. Under the MML framework, each segment mean and residual would be encoded as a vector. Each colour component in the vector could be considered 3 independent scalars. These ideas also apply to multispectral data, which may have any number of bands.

6.3.2 Planar regions

The structural model used so far has assumed piece-wise constant segments. Piece-wise constant segments are the lowest level in the facet model hierarchy, described in Section 4.2.1. It was argued that, on a local scale, constant facets would be sufficient to model most image behaviour. The success of the FUELS and MML algorithms have shown that a simple model can be taken a long way, but not all image data is well modeled by constant facets. In Section 4.2, it was stated that range images contain segments which vary in a linear, or planar, fashion. Even photographic images like `lenna` and `barb2` have regions which exhibit some planar-like behaviour. It would be useful to incorporate planar segments into the local segmentation framework.

A constant segment has only one parameter, μ, to describe its average intensity. A planar segment has three parameters. The intensity profile for a planar segment may be written like Equation 6.4, where μ is the average brightness, and Δ_x and Δ_y are the horizontal and vertical gradient terms respectively.

$$f(x, y) = \mu + x\Delta_x + y\Delta_y \tag{6.4}$$

If the noise is assumed distributed as $\mathcal{N}(0, \sigma^2)$ for each pixel, the optimal values for the three parameters may be determined using the familiar least-sum-of-squares algorithm [Nie94]. A simpler solution for pixels from a regular square region, like the popular 3×3 window, is given in Figure 6.4. The gradient terms are actually equivalent to the outputs of the horizontal and vertical Prewitt edge detector masks [PM66].

p_1	p_2	p_3
p_4	p_5	p_6
p_7	p_8	p_9

$$\mu = \tfrac{1}{9}\textstyle\sum_{i=1}^{i=9} p_i$$
$$\Delta_x = \tfrac{1}{3}\left[(p_3 - p_1) + (p_6 - p_4) + (p_9 - p_7)\right]$$
$$\Delta_y = \tfrac{1}{3}\left[(p_7 - p_1) + (p_8 - p_2) + (p_9 - p_3)\right]$$

Figure 6.4: Computing the parameters for a homogeneous 3×3 planar model.

It has been shown how to derive a planar model for a homogeneous region. Under the MML framework, this model could be incorporated into the pool of candidates, alongside the homogeneous constant model and two-segment constant models. The model part of its message would be longer, as three parameters need to be encoded to describe the fitted plane. It should be possible to determine the optimal quantization levels for Δ_x and Δ_y. The data part is the residuals between the fitted plane and the noisy pixel values. MML naturally takes into consideration the complexity of each model via its overall message length. The planar model would only be chosen over the constant model as warranted by the data.

Fitting planes to each segment in a heterogeneous window is more complicated. Obviously, when there is more than one segment present, there are fewer pixels in each segment. It is difficult to estimate reliably the gradient terms for the fitted planes, due to the lack of data. For an MML message, the small number of pixels would make it difficult to save enough bits in the data part to make up for the extra bits used for encoding the parameters of the plane.

FUELS and MML allow segments to be unconnected, further complicating the modeling. For a 3×3 window, fitting planes to two or more segments would not probably be justified. A homogeneous planar model can already model ramp edges very well. This phenomenon is what two segment regions were previously forced to approximate.

6.3.3 Higher dimensional data

The digital images processed in this thesis were two dimensional in nature, with each coordinate hosting a pixel. Today, medical imaging techniques are able to generate three dimensional (3D) *volume images*. Figure 6.5 shows a small $8 \times 5 \times 4$ volume image, which has 160 elements in all. Volume images are sometimes treated as a series of 2D image slices. Each slice may also be temporally shifted due to time spent in the acquisition process.

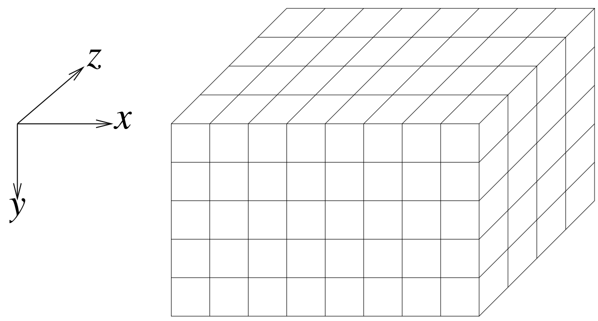

Figure 6.5: An $8 \times 5 \times 4$ volume image.

Each element in a volume image is called a *voxel*, which is shorthand for "volume element". In the simplest case, a voxel stores a scalar measurement, such as a bone density or radioactive dye intensity. Figure 6.6 shows the local neighbourhood of a voxel. Translating the 2D concept of 4-connectivity to 3D results in a neighbourhood of 6 voxels.

Local segmentation is a philosophy, not a particular algorithm or implementation. Its principles apply equally well to 2D greyscale images as they do to 3D volume images. A local neighbourhood for each voxel can be defined. This neighbourhood may then be segmented using any suitable technique. If spatial information is not important, then FUELS or MML-2 could be applied without any major changes. The principles of DNH and overlapping averaging would still apply to volume data.

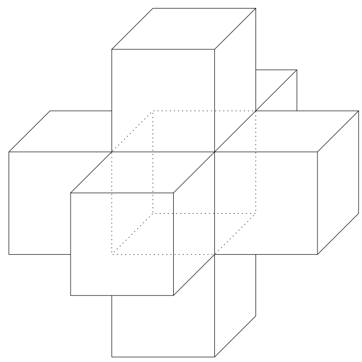

Figure 6.6: Each voxel (dotted) has 6 neighbours in its immediate neighbourhood.

The extension to 3D is only the beginning. Medical scanners are able to take temporal sequences of volume images. For example, to examine blood flow through the body over a period of time. This could be treated as 4D information. The local neighbourhood would consist of voxels around the current voxel in the current "frame", along with voxels around the same position in adjacent frames. A possible complication is that the "time" dimension is inherently different from the spatial ones, whereas FUELS and MML assume certain isotropic behaviour. Planar models might be useful for handling this situation.

These ideas also apply to digital video, which is a time series of 2D images. Local segmentation would be able to detect a change in the number of local segments around each pixel. This change would probably be associated with moving objects in the recorded scenes. Hence, local segmentation could be used as a low level step in object tracking systems.

6.3.4 Larger window sizes

In Section 4.10, FUELS was tested on different windows sizes, the largest of which contained 37 pixels. It was found that smaller windows produced better lower RMSE when denoising. This may not be true if a more sophisticated image model was used. FUELS and MML-2 extend naturally to larger mask sizes, as they only considered a fixed number of candidate models based on thresholding.

MML-256 took the path of considering all possible segment maps. For a 3×3 window, this came to 256 distinct segmentations. This exhaustive approach does not scale well to larger windows. For each extra pixel included in the window, the number of binary segment maps doubles. As the window size increases, the average number of segments present in the underlying image is also expected to rise. This also increases the total number of candidate segmentations to be potentially considered.

Obviously the search space needs to be restricted. The homogeneous and optimally thresholded models should always be included. A segmentation algorithm which exploits spatial information could be used to generate extra candidates. The thresholded segmentation could be used a starting point for the spatial algorithm. Most spatial segmentation algorithms require one or more parameters to control their operation. A few different values for these parameters could be tried, and each resulting segmentation added to the pool.

Reducing the search space is only part of the problem. Each segment map needs to be costed in bits. An *explicit* prior distribution over all possible segment maps is no longer suitable, due to the large number of possibilities. Instead, a method for efficiently coding segment maps is needed, which *implicitly* gives a prior to all segment maps. The field of lossless image compression would be a good place to find a method for doing this. I have already done some preliminary investigations into the use of a low order Markov model for compressing binary segment maps, much like the JBIG algorithm [GGLR81, PMGL88]. The Markov model parameters are dynamically learned using the data-driven prior approach. The details of this work are outside the scope of this thesis.

6.4 Pixel classification

Pixel classification is the process of classifying each pixel in an image as a smooth, shaped, or textured feature point [Cho99]. Smooth points correspond to homogeneous regions, such as those positioned in the interior of global segments with a low natural level of variation. Shaped regions, or edge regions, are those occurring near the boundaries and junctions between global segments. There is no universally accepted definition for texture. Caelli and Reye [CR93] propose that textured regions and edge regions are the same features, but at different scales. They suggest that texture be considered a high resolution area containing many edges in close proximity.

Local segmentation necessarily classifies the region surrounding each pixel. The most important attribute determined by the local segmentation process is k, the number of segments present. This k may be used for pixel classification. Ignoring DNH for the moment, the multi-class 3×3 FUELS algorithm estimates k to be between 1 and 9. When $k = 1$, the window is "smooth". When $k > 1$, the window contains segment boundaries, so a "shaped" classification is more appropriate. In summary, we have:

$$\begin{array}{rcl}\text{“Smooth”} & \iff & k = 1 \\ \text{“Shaped”} & \iff & k \geq 2\end{array}$$

Figure 6.7 shows `lenna` and its corresponding classification map. Black pixels denote smooth points and white pixels denote shaped points. The classification is based on values of k as determined by the multi-class FUELS algorithm with DNH disabled. The noise level was estimated to be $\sigma = 2.658$.

Figure 6.7: (a) original `lenna`; (b) classification into smooth (black) and shaped (white) feature points.

Of the pixels in Figure 6.7b, 59% are smooth and 41% are shaped. The smooth points do indeed correlate with the large homogeneous regions in `lenna`, and the shaped points to the edges and texture. Pixel classification of this type could be used in various ways. For example, a lossy image compression algorithm could allocate more bits to the shaped areas to preserve sharpness. A global segmentation algorithm could attempt to cluster only smooth points first, and then apply region growing to annex shaped points accordingly.

In the example of Figure 6.7, shaped and textured feature points were not distinguished. Windows which are not well modeled by one or more segments could be considered "textured" regions. The multi-class FUELS algorithm uses its DNH mode to identify such windows. In Section 4.12.4 it was also shown that windows where $k = M$, where M is the number of pixels in the window, are equivalent to DNH. Thus local segmentation could be used to classify pixels into three groups as follows:

$$\begin{aligned} \text{“Smooth”} &\iff k = 1 \\ \text{“Shaped”} &\iff k \leq 2 < M \\ \text{“Textured”} &\iff k = M \text{ or DNH} \end{aligned}$$

Figure 6.8 applies these classification rules to `lenna`. Only 5% of pixels are classified as textured. About a third occur near edges, possibly ramps. These are difficult to model with piece-wise constant segments. Another third seem to be isolated patches, perhaps planar regions for which a segment-based model was inappropriate. The last third do actually appear in textured regions of the image, particularly the feathers in `lenna`'s boa. This result is less encouraging than when only smooth and shaped points were considered.

Pixel classification could possibly be improved by examining the image at different *scales*. Chou [Cho99] performs edge strength measurements on the original image and two down-sampled versions thereof. A set of rules relating edge strength at each position over the various scales is then used to make the final decision. For example, a low edge response at all scales gives strong evidence for the point being smooth, and correspondingly, consistently high edge responses suggest a shaped point. If the edge strength varies relatively over scales then it could be considered a textured point. It would be possible to apply these types of rules using the values of k determined at different scales.

6.5 Edge detection

Edge detection describes the general problem of determining the location, magnitude and orientation of (segment) boundaries in an image [HSSB98]. The precision and accuracy to which these edge attributes may be determined is what discriminates between algorithms.

Figure 6.8: (a) original `lenna`; (b) smooth points (white); (c) shaped points (white); (d) textured points (white).

The "shaped" pixel classification in Section 6.4 may be considered a primitive edge detector. It provides a rough location, but no magnitude or orientation. This section will explore some more advanced methods for edge detection from a local segmentation perspective.

6.5.1 Edge strength measurement

Edge strength measurement algorithms are used to estimate the magnitude of an edge transition around each point in the image. They do not attempt to determine the exact position or direction of an edge. Edge strength may be used as a simple edge detector. It should give a low response in homogeneous regions, and a high response in the vicinity of segment boundaries. The response usually increases with both the sharpness and contrast of the edge.

The output of an edge strength algorithm could be used by segmentation algorithms. For example, those pixels furthest from high edge activity may be used as seeds for a region growing algorithm. A map of edge strengths could be used as the "elevation map" for the watershed segmentation algorithm [HH94]. Alternatively, the identified edge pixels may be used to initiate a boundary tracking process.

The commonly used Sobel 3×3 linear convolution masks [Sob70, GW92] treat the image as a functional surface, and attempt to estimate the gradient magnitude in two orthogonal directions. Equation 6.5 shows the horizontal mask, S_x, and the vertical mask, S_y. The edge strength measurement $e_{SOB}(x, y)$ at each point is the Pythagorean combination of the result of convolving the two directional masks with each pixel $f(x, y)$.

$$S_x = \begin{array}{|c|c|c|}\hline 1 & 0 & -1 \\ \hline 2 & 0 & -2 \\ \hline 1 & 0 & -1 \\ \hline \end{array} \quad S_y = \begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -2 & -1 \\ \hline \end{array} \quad e_{SOB}(x, y) = \sqrt{[f(x, y) * S_x]^2 + [f(x, y) * S_y]^2} \tag{6.5}$$

Local segmentation may be applied to edge strength measurement. Consider FUELS' thresholding from Section 4.14, which diagnoses the window as consisting of one or two segments. For the two segment case, the numerical difference between the two cluster means could be considered a measure of edge strength. Equation 6.6 gives an expression for computing an edge strength image, $\mathbf{e}_{LS}$. Because only regions with a contrast difference greater than $C\sigma$ are diagnosed as consisting of two segments, that amount is subtracted from the edge strength. Homogeneous blocks receive an edge strength of zero.

$$e_{LS}(x, y) = \max \{ \, 0 \, , \; | \, \mu_1(x, y) - \mu_2(x, y) \, | - C\sigma \, \} \tag{6.6}$$

This measure of edge strength is only one possibility. An alternative would be to consider the intensity difference between the centre pixel and the neighbouring pixel nearest in intensity, but from the other segment.

Results

Figure 6.9 provides edge strength results for `lenna`. Equation 6.6 was used by the local segmentation based method. The Sobel and local segmentation outputs were linearly stretched to the full intensity range, to improve visibility of the edge strengths. SUSAN's edge detector output is also included[1]. It used the same value of t as used for denoising in Chapter 4.

Figure 6.9: (a) original `lenna`, estimated $\sigma = 2.66$; (b) local segmentation; (c) Sobel; (d) SUSAN using $t = \lfloor 3\sigma \rfloor = 8$.

The local segmentation edge strength measurement algorithm presented here is very simplistic. Its performance could be said to fall somewhere between Sobel and SUSAN. It appears

[1]SUSAN source code: `http://www.fmrib.ox.ac.uk/~steve/susan/`

to be less sensitive to noise than SUSAN, but picks up more potential detail than Sobel. It would be interesting to apply thinning and edge linking to the local segmentation output, but that is outside the scope of this chapter. The results are useful in that they essentially come for free from the local segmentation decomposition, and do not use any spatial information. A more advanced implementation could consider the actual pattern of cluster assignments for each window, and accordingly determine edge direction, or disregard incoherent patterns.

6.5.2 Probabilistic edge detection

The MML-256 local segmentation algorithm from Chapter 5 considers all 256 possible 3×3 segment maps. This results in a posterior probability distribution over possible binary segment maps for the local 3×3 window. The segment model assumes that pixels belong wholly to one segment, and that segment boundaries occur between pixels. Thus a segmentation implicitly defines the existence of segment boundaries.

Consider the 3×3 block of pixels of Figure 6.10. A pixel *interface* is the junction line between two adjacent pixels. There are 12 pixel interfaces in a 3×3 block, 6 of them horizontal and 6 of them vertical. A edge, or *boundary*, is said to exist at a pixel interface if the segment labels for the two pixels at the interface are *different*. At first glance it may appear that there are 2^{12} boundary patterns, but there are really much fewer, because each pattern must form one or more logical segments.

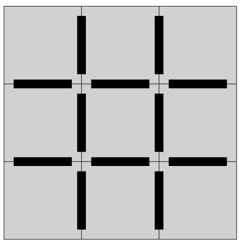

Figure 6.10: The twelve inter-pixel boundaries for a 3×3 window, shown in bold.

Figure 6.11 illustrates three potential segment maps. The homogeneous segment map asserts that no boundaries are present in the window. The window with a vertical edge asserts the existence of a boundary at 3 of the 12 possible interfaces. The diagonal line in the third window is even more complex, insisting that there is a boundary present at 6 interfaces.

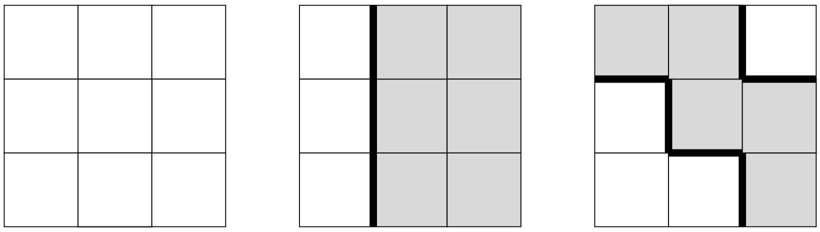

Figure 6.11: Examples of segment maps, with boundaries shown in bold.

The posterior information gleaned from the MML-256 local segmentation process may be used for edge detection. For an $X \times Y$ image there are $(X-1)\times(Y-1)$ horizontal pixel boundaries, and the same number of vertical ones. Each processed pixel produces a probability distribution over possible segment maps. If the segment labels on both sides of an interface are different, then a boundary is present along that interface. For a given segment map, the probability of each interface's boundary existing in the image can be assumed to have the same probability — the posterior probability for the segment map in question.

Thus a probability for the existence of an edge at each pixel interface can be determined. Each interface occurs in 6 overlapping windows, so probabilities can be accumulated. The horizontal and vertical probabilities can be stored in two images, called $\mathbf{h}$ and $\mathbf{v}$, each of resolution $(X-1)\times(Y-1)$. Each image may be visualized alone, or they could be combined into a single image, $\mathbf{e}$. Three different methods for combining the horizontal and vertical edge probability images are considered:

1. Averaging: $\mathbf{e} = \frac{1}{2}(\mathbf{h}+\mathbf{v})$.
2. Euclidean magnitude, or normalized L_2 norm: $\mathbf{e} = \sqrt{\frac{\mathbf{h}^2+\mathbf{v}^2}{2}}$.
3. The maximum, or L_∞ norm: $\mathbf{e} = \max(\mathbf{h}, \mathbf{v})$.

Results

Figure 6.12 gives results for part of `lenna`. Black represents probability zero, and white probability one. The MML-256 algorithm was iterated 6 times to learn priors for the segment maps. The DNH model was treated like a homogeneous window, in that it did not contribute

Figure 6.12: Example of probabilistic pixel boundary detection: (a) part of `lenna`; (b) $\mathbf{h}$; (c) $\mathbf{v}$; (d) $(\mathbf{h}+\mathbf{v})/2$; (e) $\sqrt{(\mathbf{h}^2+\mathbf{v}^2)/2}$; (f) $\max(\mathbf{h},\mathbf{v})$.

any posterior probability to any pixel interfaces. An alternative would have been to treat all pixel interfaces as being active.

This technique seems to work extremely well. The vertical image clearly picks up the hanging feather at the top of the image. This feather does not appear at all in the horizontal image, as desired. The diagonal brim of the hat is clearly detected in both $\mathbf{h}$ and $\mathbf{v}$, because diagonal edges are a blend of horizontal and vertical changes. For visualization purposes, the $\max(\mathbf{v}, \mathbf{h})$ function appears to give an output image with more contrast. The $\max$ operation clearly states whether an edge is present or not, regardless of its orientation. The other two methods for combining $\mathbf{v}$ and $\mathbf{h}$ respond more to diagonal edges than to vertical and horizontal ones.

Figure 6.13 shows the combined probabilistic edge strength image for the complete `lenna` image, using the $\max(\mathbf{h}, \mathbf{v})$ method. It was produced under the same conditions as Figure 6.12f. The results are quite remarkable. All the important boundaries are clearly visible. Most interesting is the amount of detail with respect to individual feathers on the boa hanging from the hat. It has also managed to detect a large proportion of the fine diagonal lines present above the band on the hat.

Conclusions

The results of this section clearly show the power of the MML-256 local segmentation framework. The horizontal and vertical edge strength decomposition derived naturally from the local segmentation models used. It would be possible to transform this orthogonal decomposition into another coordinate space. For example, polar coordinates would describe the edge magnitude and the edge *orientation* at each position. The L_2 norm plotted in Figure 6.12e is effectively the magnitude. The angle of orientation, α, may additionally be derived using the trigonometric expression $\alpha = \arctan \frac{v(x,y)}{h(x,y)}$.

6.6 Image enlargement

Image enlargement increases the resolution of an image. It is also called image expansion, zooming, and interpolation. Consider the simplest problem of doubling the size of an image from $X \times Y$ to $2X \times 2Y$. The doubled image has four times as many pixels as the original. Exactly 75% of the pixels need new values determined for them. This situation is shown in Figure 6.14. The problem is to predict suitable values for the unknown pixels, such that the features of the original image remain intact.

Figure 6.13: Probabilistic edge strength for `lenna` using $\max(\mathbf{v}, \mathbf{h})$.

The simplest technique for image enlargement is *pixel replication*. It makes the assumption that the known pixel values were obtained from a 2×2 down-sampling of an originally larger image. Thus, the "best" replacement pixels have values equal to the known pixel. An example is given in Figure 6.15. The main disadvantage with pixel replication is that, for natural images, the results are very blocky. The next simplest idea is to interpolate the pixel intensities linearly. This often looks better than pixel replication, but has a tendency to produce blurred edges, much like the box filter for denoising.

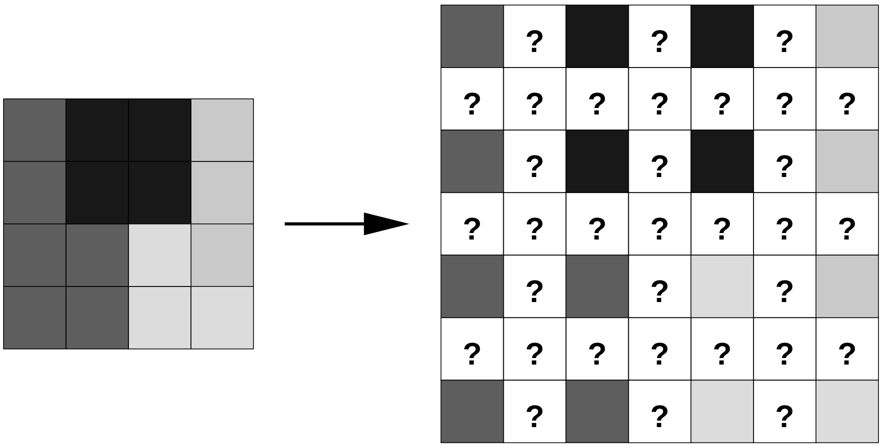

Figure 6.14: Doubling an image's size means predicting all the '?' pixels.

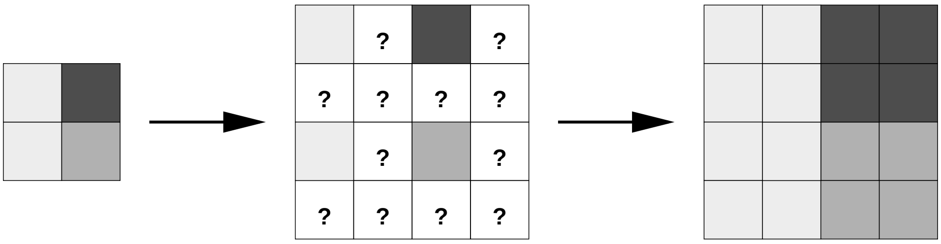

Figure 6.15: Pixel replication is the simplest expansion method.

Structure directed enlargement algorithms attempt to maintain edge coherence when estimating values for the unknown pixels. It is possible to use the MML-256 local segmentation method to aid image enlargement. Consider the $5{\times}5$ window of the enlarged image in Figure 6.16a. It has 9 known pixels, which are numbered. In the original image, assume the best model used the binary segment map denoted by the white and grey pixels. Segment 1 had 4 pixels in it, and segment 2 had 5 pixels in it.

Now consider the $3{\times}3$ sub-window in Figure 6.16b. It has 4 known pixels, and 5 unknown pixels. The segment memberships that the 4 known pixels had in the original segment map can be assumed consistent. All that remains is to determine which segment each of the unknown pixels belong to. When that is achieved, each unknown pixel can be assigned a value equal to its segment's mean. Because only 5 pixels don't belong to a segment, there are only $2^5 = 32$ possible segment maps that could apply.

The MML-256 algorithm can be used to calculate the message lengths for these 32 possible models. The same prior used for the original image could be re-used, ensuring that the

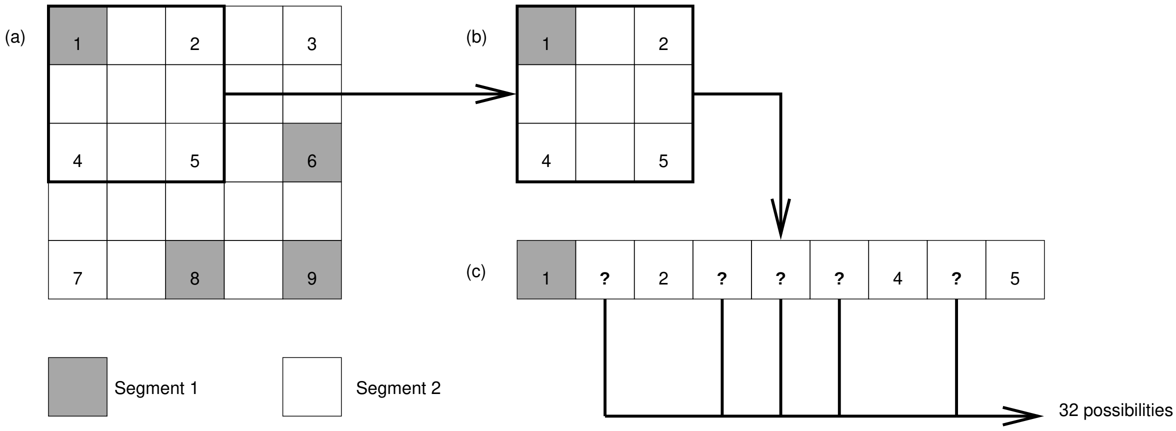

Figure 6.16: Using local segmentation for structure directed interpolation.

original image structure is replicated in the enlarged image. The segment means in each message would be equal to the segment means calculated from the original 9 known pixels in the 5×5 window. The data part of the message would only consist of residuals for the known pixels. Of the 32 possible segment maps, the one with the shortest message length is used to fill in values for the unknown pixels. It is straightforward to incorporate overlapping average and posterior blending into image enlargement too.

Figure 6.17 compares three image enlargement methods for doubling `montage` (only the middle shown). As expected, pixel replication produces blocky output, and linear interpolation produced blurry output. In comparison, the MML-256 result is good. The edges of the artificial shapes and letters look more continuous. MML-256 based zooming may be considered a fractal type approach, because the same prior is used at two different scales, ensuring the image is self-similar at different resolutions.

Digital zooming has many applications. Most digital cameras on the market only support optical zooming up to a point, after which digital zooming takes over. It is important that the "invented" zoomed data appear realistic. Deinterlacing of digitized television signals can be interpreted as taking an interlaced frame of size $2X\times Y$, and producing a full frame of size $2X\times 2Y$. Predicting pixel values for the missing rows could be done by adapting the MML-256 technique just described.

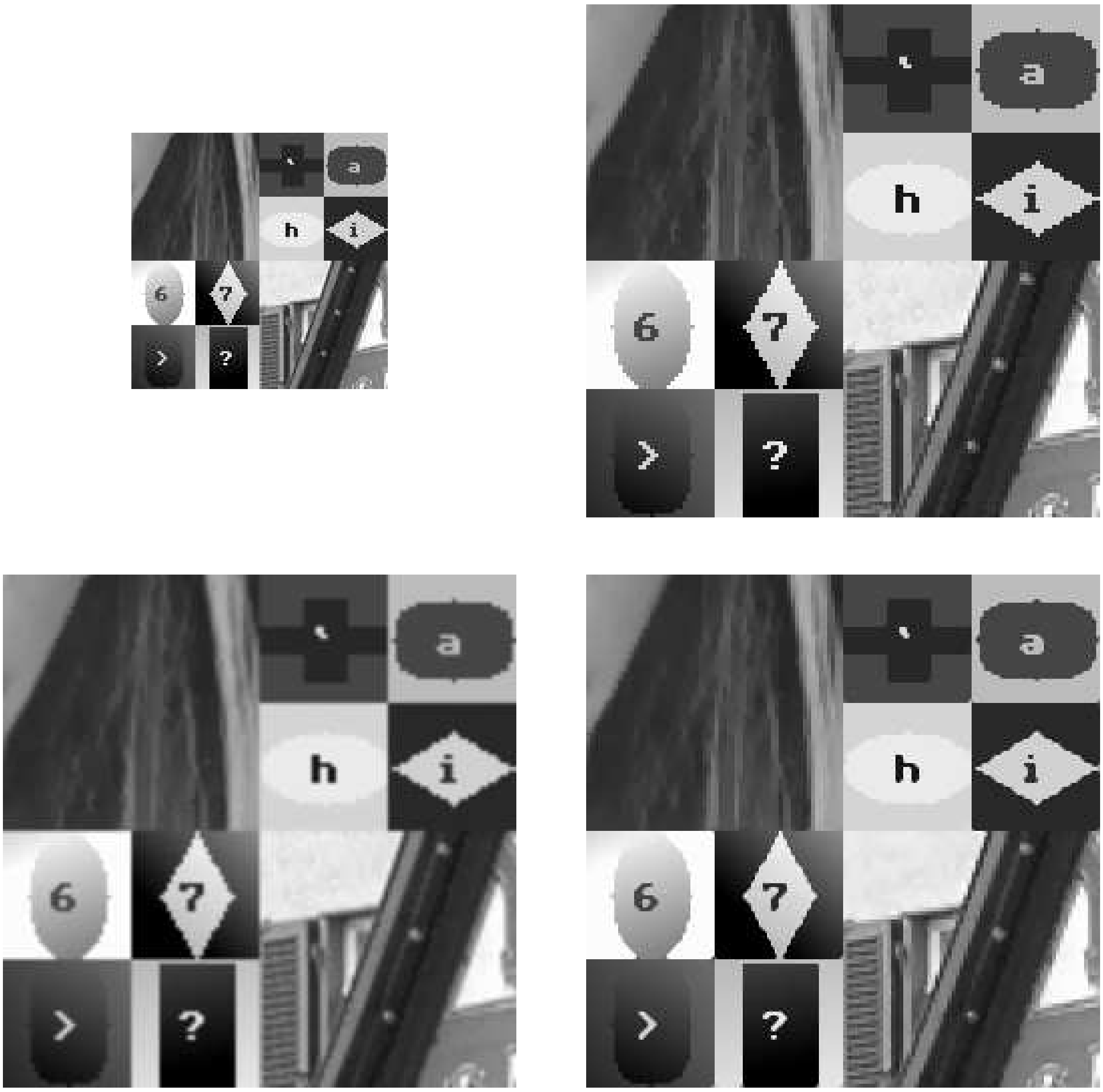

Figure 6.17: Comparison of doubling methods: (a) denoised image; (b) pixel replication; (c) linear interpolation; (d) MML-256 based enlargement.

6.7 Image compression

Image compression is concerned with efficiently encoding images for storage and transmission [RJ91]. *Lossless* image compression encodes the data exactly, the decoded image being identical to the original. *Lossy* image compression encodes an approximation of the original image in order to reduce the encoded length. Lossy algorithms must trade-off the level of compression and the amount of distortion in the reconstructed image.

In the past, image processing and image compression have been treated as separate disciplines. The use of information theoretic techniques for analysing data, such as MML and

MDL, has proffered a merging of the two disciplines. Both image compression and MML are concerned with optimal lossless encodings of the data. The data part of a two part MML message may be considered the noise, while the model part is the structure. This could be useful to a lossy compression algorithm for deciding which image information is important, in addition to any heuristics based on the human visual system. Better compression is intimately linked with better understanding of data.

6.7.1 Aiding lossy compression

Lossy image compression may be considered to consist of two stages, illustrated in Figure 6.18. Firstly, a suitable approximation to the original image is determined. Secondly, the approximate image is losslessly encoded. Although all lossy image compression algorithms effectively do this, it is not always clear how they choose which information to discard.

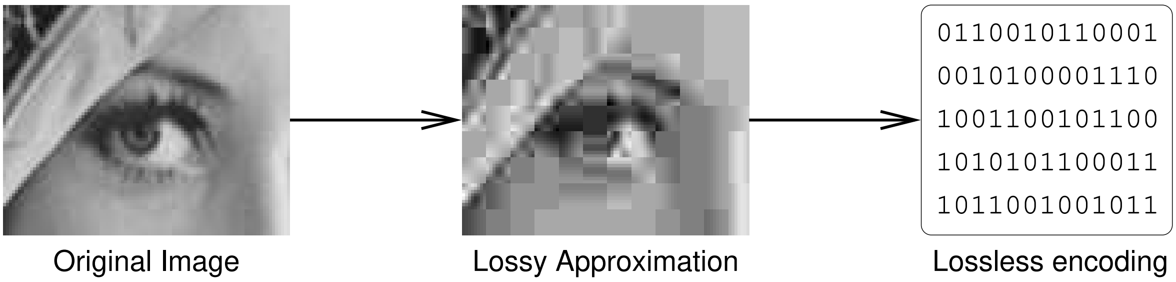

Figure 6.18: Lossy compression can be considered a two stage process.

An alternative to allowing the lossy compression algorithm to distinguish between structure and noise is to preprocess the image to remove the noise first. The parameters to the lossy algorithm can then be chosen to preserve more of the information they are given. That is, the discarding of visually unimportant information (first stage) can be replaced by a denoising algorithm, leaving the encoding (second stage) to the compression algorithm.

In Chapters 4 and 5, local segmentation was shown to produce good denoising algorithms. The denoised output of FUELS could be used as the input to a compression program. Having the noise removed would assist in decreasing the size of the compressed file. An alternative approach would be for a local segmentation technique to determine where bits should be spent. For example, $k = 1$ regions have low information content, and hence fewer bits should be used for encoding those regions. Important structural details, such as edges, occur in $k = 2$ regions, and more bits should be used to ensure that these features are represented accurately in the lossy compressed image.

6.7.2 Adaptive BTC

Block truncation coding (BTC) is a simple and fast technique for image compression, and was discussed in Section 3.3.2. In its original form, BTC processes 4×4 blocks of pixels at a time, and outputs data with a fixed bit rate. For 8 bit per pixel greyscale image data, the resulting compression ratio is 4:1, or 2 bits per pixel. An example of its operation on a sub-image of `lenna` is given in Figure 6.19.

Figure 6.19: Standard BTC: (a) original image at 8 bpp; (b) reconstructed image at 2 bpp; (c) bitmap.

Local segmentation and BTC are closely related. Both techniques involve segmenting a small group of connected pixels. The segmentation includes a bitmap for describing pixel assignments to segments, and representative intensities for each segment. The original BTC method always used two classes, which is similar to FUELS. However, BTC used the block mean as a threshold, and chose the segment "means" to preserve the variance of the overall block. This is in contrast to FUELS' use of the adaptive Lloyd quantizer.

When a block is homogeneous, there is no need to transmit a bitmap and two means — a single mean is sufficient. Some variants of BTC attempt to exploit this property to save bits and improve compression. To make a decodable bitstream, each block's encoding is preceded by one bit to distinguish between homogeneous and standard blocks. An alternative is to send the two means first, and if they are the same, not to send a bitmap. Mitchell *et* *al* [MD80] assume the region is homogeneous if the block variance is less than one, and Nasiopoulos *et al* [NWM91] do the same if the range of the block is less than some threshold.

FUELS already provides a way to estimate a suitable threshold for distinguishing between homogeneous and heterogeneous blocks. Figure 6.20 shows how *Adaptive BTC* can improve the bit rate with little loss in image quality compared to standard BTC. The threshold used is 3σ, where σ is estimated as 2.84. This results in 41% of blocks being declared homogeneous, reducing the overall bit rate to 1.39 bpp. The bitmap image clearly shows how two segments were only used in high activity regions of the image.

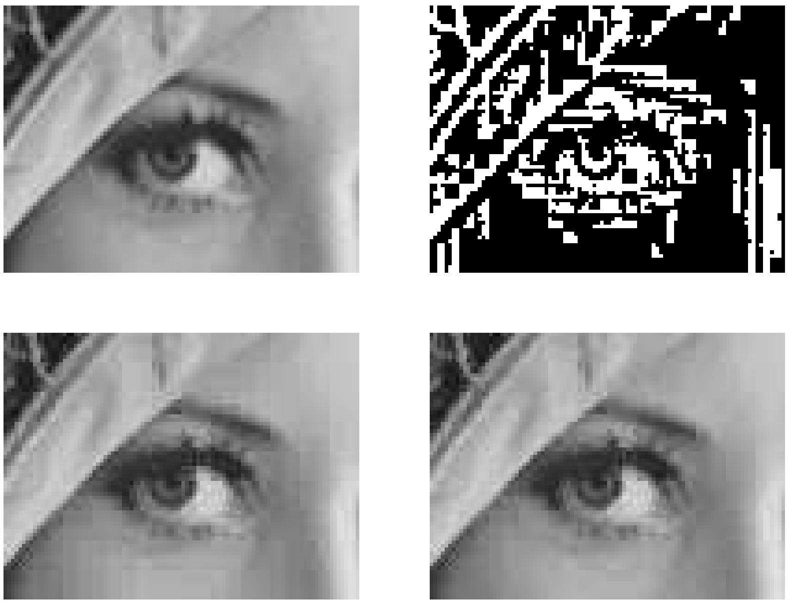

Figure 6.20: Adaptive BTC: (a) original image; (b) bitmap; (c) reconstructed image at 1.39 bpp, 41% homogeneous blocks using threshold of 8.5; (d) reconstructed image at 2 bpp.

Prefixing each block with a bit describing how many segments are in the block implicitly places a prior probability distribution over segmentations. Specifically, it gives probability 0.5 to $k = 1$, and shares the remaining 0.5 between all possible 4×4 clusterings, of which there are $2^{15} = 32768$. This is obviously an inefficient coding, because Section 5.14 showed that not all binary segment maps are equally likely. The MML methodology could be applied to first determine these probabilities from the image. These probabilities could then be sent up front, and the rest of the image encoded using them. If the cost of the upfront information is small enough, a net gain in bit rate would be possible.

6.7.3 Lossless predictive coding

The great majority of lossless compression algorithms are based on predicting the value of the next pixel to be encoded. Examples include CALIC [WM96], LOCO [WSS00], HBB [STM97], TMW [MT97] and Glicbawls [MT01]. The prediction must be formed using only those pixels which have already been encoded — the *causal neighbourhood* shown in Figure 6.21. This is because the decoder does not yet know about future pixels.

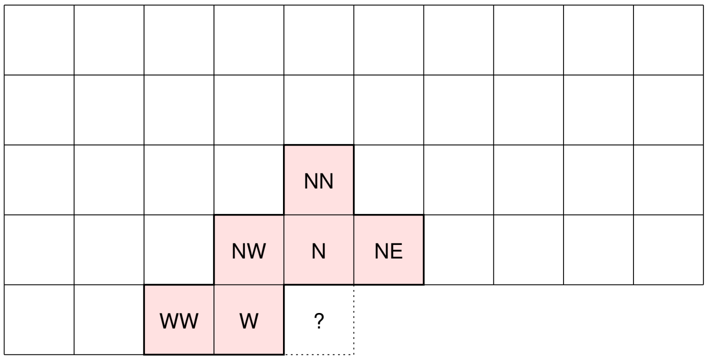

Figure 6.21: The causal local neighbourhood consists of pixels known to both encoder and decoder.

Typically, only pixels in the local neighbourhood are used for prediction. In Figure 6.21 they have been labelled with the standard *compass point notation* [STM97], where N denotes north, W denotes west, and so on. The pixel to be encoded, ?, is usually referred to as *the current pixel*. The predicted value for the current pixel is often taken to be a linear combination of the pixels in the causal neighbourhood. For example, the Pirsch predictor [Pir80] is calculated as 0.5W+0.25N+0.25NE.

Local segmentation could be used to improve the quality of predicted values. The amount of noise in a predicted pixel value is a linear combination of the noise in each of the pixels used for the prediction. It is desirable that this noise be minimized. Instead of using the decoded (original) pixel values for prediction, denoised versions of them could be used. For example, FUELS could be used to filter the image as it is decoded. The overlapping estimate technique would even provide reasonable denoised values for those pixels without a complete two-sided neighbourhood.

When encoding each pixel, it must be with respect to some distribution over possible pixel values. The most commonly used distributions are Gaussian and Laplacian, due to their symmetry and unimodality. The "predicted value" is used to estimate the location parameter of the distribution. The spread parameter captures the level of confidence we have in the predicted value. Besides the prediction formula, the method used for determining an appropriate spread value is what distinguishes most algorithms.

Some techniques implicitly use a single spread parameter for the whole image, such as lossless JPEG [PM93]. Some estimate it directly from the causal neighbourhood, like Glicbawls. CALIC and LOCO use the idea of *context coding*, whereby the local causal neighbourhood is classified into a context. Only a small number of contexts are used, with each one effectively providing a spread estimate to use for encoding the current pixel.

The FUELS approach could be used for determining contexts. Let us assume the decoder has been transmitted an estimate of the global noise variance, σ. The causal neighbourhood could be locally segmented. The optimally determined segment map could be used as a context number. The number of contexts could be controlled through the number of neighbours taking part in the segmentation. Most useful is that local segmentation could distinguish between homogeneous and heterogeneous contexts. The spread parameter for each context could be adaptively learned in a fashion similar to CALIC and LOCO.

A more general approach is to use local segmentation for both the prediction and the context selection. The causal region could be optimally segmented. If $k = 1$, the predicted value will be in the same segment, or another new segment. The predicted value should probably be equal to the mean of the causal neighbourhood. If $k = 2$, the current pixel could belong to either of the causal segments, or be from a new segment. The encoder could examine the current pixel and determine which of the two segments it belongs to. A single binary event could be used to state which of those two segments it is. The mean of that segment could be used as the predicted value. This approach to prediction should be good at predicting large changes in the vicinity of edges.

A more complex version of the previous idea could be based on the MML-256 algorithm. Let us assume the encoder and decoder agree on a prior over segment maps. The causal neighbourhood could be segmented in all possible ways, not including the current pixel. For each segment map considered, there are two new segmentations containing an extra pixel which can be derived by assuming that the current pixel is in each of the two possible segments (assuming $k = 2$). The prior could be used to assign weights to all these segmentations. The predicted values could be blended together based on the posterior probability of each of the segmentations. This is similar to the way TMW blends prediction distributions.

6.8 Conclusions

It has been shown how to apply local segmentation to a variety of image processing tasks besides denoising, including edge detection, image zooming and image compression. Also discussed was how to incorporate different image models and noise models into the local segmentation framework. These followed on from the constant facet and additive noise models used by FUELS and the MML denoising algorithms in Chapters 4 and 5. The extension of the ideas in this thesis from 2D images to 2D video sequences, 3D volume images, and 4D volume image sequences was also considered.

The local segmentation principle is a simple one. It states that whenever a pixel is processed, only those neighbouring pixels in the same segment should take part in the processing. This chapter illustrated how this idea provides a consistent and unified way of examining images in a low level manner. The simplicity of local segmentation allows it to be applied to any algorithms where pixels are processed by looking at the values of neighbouring pixels. Its power and flexibility lies in its simplicity.

Most of the ideas in this chapter could only be explored in brief, as the work required is outside the scope of this thesis. The most promising results are for probabilistic edge detection and impulse noise removal. These ideas in particular would be deserving of further research. The FUELS local segmentation methodology has a simple implementation, requiring little memory and processing power. It could be crafted as a generic image processing tool which could function in real time, potentially embedded in hardware.

Chapter 7

Conclusions

Local segmentation is already used, in some form, by a number of image processing algorithms. Sometimes its use is implicit, like with median filters, or more explicit, such as for SUSAN. The properties of a particular algorithm can be examined by isolating the local segmentation model it uses. This can help to reveal its strengths and weaknesses. The more situations an image processing technique's local segmentation criterion diagnoses correctly, the better its performance. Image denoising is well suited to demonstrating the utility of local segmentation. Good image denoising algorithms attempt to identify and preserve structure, while simultaneously removing noise.

A denoising algorithm called FUELS was developed. Although binary thresholding is a very simple type of segmentation, FUELS showed that it was good enough to outperform existing state-of-the-art denoising algorithms. Unconnected segments are often unavoidable, because the intersection of a window with a global segment may disguise their actual connectedness via pixels outside the window. Thus the main limitation of thresholding may actually be an advantage when applied locally. The windows used in local processing are usually too small to incorporate much spatial information anyway.

The move from assuming homogeneity within the local region to allowing for the presence of two segments lead to a big improvement in performance. This gain was much larger than that which came from also allowing more than two segments. Although this was true for 3×3 windows, it may not be true for larger ones. The popularity of the 3×3 window, along with results for FUELS, suggest that it represents a good trade-off between compact locality and the need for more detailed models.

The work on FUELS introduced the idea of “do no harm” (DNH). FUELS generates two candidate models, and chooses the one it thinks is better. It is possible that both candidates are poor ones for the local region, so using one could do more damage than good. If this situation was identified by the DNH mode, the pixels were left unmodified. This is similar to defaulting to a null model if there is insufficient evidence for any of the alternatives. The DNH idea is not particular to FUELS or even local segmentation — it could be applied to any denoising algorithm to improve its worst case performance, especially at low noise levels.

Most denoising algorithms use the centre pixel of the window as a reference pixel to compare with each other pixel in turn, to produce a denoised estimate of the centre pixel only. Local segmentation differs from this in that it treats all pixels in the window democratically, producing denoised values for them all. Because windows overlap, there are multiple estimates for each pixel. With little extra work, FUELS was able to average these estimates to further improve performance. This illustrates the advantages of combining predictions from different experts, which is harder to do when the centre pixel receives special treatment.

Moving to an MML framework lead to better local segmentation, which itself lead to further improvements in denoising performance. MML made it straightforward to consider a larger set of models. It provides a robust method for comparing the relative merit of different models, but can not rate them in an absolute sense. The DNH principle suggested the addition of a null model, which left the pixels unmodified, to serve as a benchmark. By comparing against this model it was possible to establish whether the other models were useful or not.

The MML denoising algorithms performed best of all when the noise level was high. This showed that thresholding, as a technique for local segmentation, was insufficient for very noisy images. MML provided a way to incorporate spatial information through its use of a data-driven prior for the segment maps. It was able to learn the general characteristics of an image, which guided it in segmenting each local region. It found that for most images, some segment maps are more likely to occur than others. Homogeneous windows were the most common situation diagnosed, which is why simplistic denoising techniques, like box filters, do well for most parts of an image.

Local segmentation could be considered for use in any situation where the current pixel is processed by making reference to the values of its neighbouring pixels. Algorithms that deal with these situations make up a large proportion of algorithms used in image processing. It was shown that local segmentation is applicable to a host of image processing tasks besides denoising. This is because it makes obvious the connection between higher level analysis, which typically involves segmentation, and low level analysis, namely local segmentation.

The principle of local segmentation is a simple one, but has not really been explicitly stated in the literature. The success of FUELS showed that even a simplistic implementation of local segmentation was able to produce high quality results. The application of MML to local segmentation was original, and lead to further small improvements in local image modeling. Because FUELS and SUSAN already represent an excellent trade-off between efficiency and effectiveness, many applications may not be able to exploit MML's better local approximations anyway. The success of local segmentation in denoising bodes well for its wider application in image processing. The idea of data driven priors, using a null model as a reference, and mixing expert predictions can be applied not just to images, but to nearly any data processing domain.

Bibliography

[AG98] C. Ambroise and G. Govaert. Convergence of an EM-type algorithm for spatial clustering. *Pattern Recognition Letters*, 19:919–927, 1998.

[Aka92] H. Akaike. Information theory and an extension of the maximum likelihood principle. In Samuel Kotz and Norman L. Johnson, editors, *Breakthroughs in Statistics*, volume I, pages 599–624. Springer-Verlag., New York, 1992. with an introduction by J. deLeeuw.

[Alp98] Ethem Alpaydin. Soft vector quantization and the EM algorithm. *Neural Networks*, 11:467–477, 1998.

[Ama87] John Amanatides. Realism in computer graphics: A survey. *IEEE Computer Graphics and Applications*, 7(1):44–56, January 1987.

[AW96] Jan Allebach and Ping Wah Wong. Edge-directed interpolation. In *International Conference on Image Processing*, pages 707–710. IEEE, 1996.

[BB95] Léon Bottou and Yoshua Bengio. Convergence properties of the k-means algorithm. In *Advances in Neural Information Processing Systems*, volume 7, Denver, 1995. MIT Press.

[BBB00] Kamel Belkacem-Boussaid and Azeddine Beghdadi. A new image smoothing method based on a simple model of spatial processing in the early stages of human vision. *IEEE Transactions on Image Processing*, 9(2):220–226, February 2000.

[BC90] J. M. H. Du Buf and T. G. Campbell. A quantitative comparison of edge preserving smoothing techniques. *Signal Processing*, 21(4):289–301, 1990.

[Bes86] J. Besag. On the statistical analysis of dirty pictures. *Journal of the Royal Statistical Society, Series B*, 48(3):259–302, 1986.

[Bez81] J. C. Bezdek, editor. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Plenum, NY, 1981.

[BF98] Paul S. Bradley and Usama M. Fayyad. Refining initial points for K-Means clustering. In *Proc. 15th International Conf. on Machine Learning*, pages 91–99. Morgan Kaufmann, San Francisco, CA, 1998.

[BO95] Rohan. A. Baxter and Jonathan J. Oliver. MDL and MML: Similarities and differences (Introduction to minimum encoding inference - Part III). Technical Report 94/207, Department of Computer Science, Monash University, January 1995.

[Bow84] D. R. K. Bownrigg. The weighted median filter. *Communications of the ACM*, 27:807–818, 1984.

[Boz83] H. Bozdogan. Determining the number of component clusters in the standard multivariate normal mixture model using model selection criteria. Technical Report TR UIC/DQM/A83-1, Quantitative Methods Dept., University of Illinois, Chicago, Illinois, 60680, 16 June 1983.

[BS85] R. Bracho and A.C. Sanderson. Segmentation of images based on intensity gradient information. In *Proceedings of CVPR-85 Conference on Computer Vision and Pattern Recognition, San Francisco*, pages 341–347, 1985.

[BS94] J. M. Bernado and A. F. M. Smith. *Bayesian Theory*. Wiley, New York, 1994.

[BSMH98] Michael J. Black, Guillermo Sapiro, David H. Marimont, and David Heeger. Robust anisotropic diffusion. *IEEE Transactions on Image Processing*, 7(3):421–432, March 1998.

[BW72] D. M. Boulton and C. S. Wallace. An information measure for hierarchic classification. *Computer Journal*, 16(3):254–238, 1972.

[Cai96] Z. Q. Cai. Correlation-based block truncation coding. *Electronics Letters*, 32(25):2305–2306, December 1996.

[Can86] John Canny. A computational approach to edge detection. *IEEE Trans. Pattern Analysis and Machine Intelligence*, PAMI-8(6):679–698, November 1986.

[CC94] K. W. Chan and K. L. Chan. Optimized adaptive AMBTC. In *International Conference on Image Processing*, pages 371–374. IEEE, 1994.

[Cho99] Wen-Shou Chou. Classifying image pixels into shaped, smooth, and textured points. *Pattern Recognition*, 32:1697–1706, 1999.

[CHY89] Sungzoon Cho, Robert Haralick, and Seungku Yi. Improvement of Kittler and Illingworth's minimum error thresholding. *Pattern Recognition*, 22(5):609–517, 1989.

[CK72] C. K. Chow and T. Kaneko. Automatic boundary detection of the left ventricle from cineangiograms. *Comput. Biomed. Res.*, 5:388–410, 1972.

[CR93] T. Caelli and D. Reye. On the classification of image regions by color, texture and shape. *Pattern Recognition*, 26:461–470, 1993.

[Cur91] A. Curri. Synthetic aperture radar. *Electronics and Communication Engineering Journal*, 3(4):159–170, 1991.

[CW00] T. Chen and H. R. Wu. Impulse noise removal by multi-state median filtering. In *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, volume IV, pages 2183–2186, June 2000.

[DDF+98] D. L. Dowe, M. Doran, G. E. Farr, A. J. Hurst, D. R. Powell, and T. Seemann. Kullback-Leibler distance, probability and football prediction. In W. Robb, editor, *Proceedings of the Fourteenth Biennial Australian Statistical Conference (ASC-14)*, page 80, July 1998.

[DH73] R. O. Duda and P. E. Hart. *Pattern Classification and scene analysis*. Wiley, New York, 1973.

[DLR77] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm (with discussion). *Journal of the Royal Statistical Society (Series B)*, 39(1):1–38, 1977.

[DM79] E. J. Delp and O. R. Mitchell. Image coding using block truncation coding. *IEEE Trans. Commun.*, 27:1335–1342, 1979.

[Ede99] Shimon Edelman. *Representation and recognition in vision*. MIT Press, Cambridge, Mass., 1999.

[EH81] B. S. Everitt and D. J. Hand. *Finite Mixture Distributions*. Chapman and Hall Ltd, 1981.

[ELM91] N. Efrati, H. Liciztin, and H. B. Mitchell. Classified block truncation coding-vector quantization: an edge sensitive image compression algorithm. *Signal Process. Image Commun.*, 3:275–283, 1991.

[EM00] How-Lung Eng and Kai-Kuang Ma. Noise adaptive soft-switching median filter for image denoising. In *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, volume IV, pages 2175–2178, June 2000.

[Far99] G. Farr. *Information Theory and MML Inference (Lecture Notes)*. School of Computer Science and Software Engineering, Monash University, 1999.

[FCG85] J.P. Fitch, E.J. Coyle, and N.C. Gallagher, Jr. Root properties and convergence rates of median filters. *IEEE Trans. Acoustics, Speech, and Signal Processing*, 33:230–240, 1985.

[FH96] Winfried A. Fesslenz and Georg Hartmann. Preattentive grouping and attentive selection for early visual computation. In *Proc. 13th International Conference on Pattern Recognition*, volume IV: Parallel and Connectionist Systems, pages 340–345, Los Alamitos, CA., 1996. IEEE Computer Science.

[Fis12] R. A. Fisher. *Messenger Math.*, 41:155–160, 1912.

[Fis36] R. A. Fisher. *Annals of Eugenics*, 7:376–386, 1936.

[FKN80] Henry Fuchs, Zvi M. Kedem, and Bruce F. Naylor. On visible surface generation by a priori tree structures. *Computer Graphics (SIGGRAPH '80 Proceedings)*, 14(3):124–133, July 1980.

[FM81] K. S. Fu and J. K. Mui. A survey on image segmentation. *Pattern Recognition*, 13:3–16, 1981.

[FNK94] Pasi Fränti, Olli Nevalainen, and Timo Kaukoranta. Compression of digital images by block truncation coding: A survey. *The Computer Journal*, 37(4):308–332, 1994.

[GG84] S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images. *IEEE Trans. Patt. Anal. Machine Intell.*, 6(6):712–741, November 1984.

[GG91] A. Gersho and R. Gray. *Vector Quantization and Signal Compression*. Kluwer Academic Publishers, 1991.

[GGLR81] Jr. Glen G. Langdon and Jorma Rissanen. Compression of black-white images with arithmetic coding. *IEEE Transactions on Communications*, 29(6):858–867, June 1981.

[Gla93] C. A. Glasbey. An analysis of histogram-based thresholding algorithms. *CVGIP: Graphical Models and Image Processing*, 55(6):532–537, November 1993.

[GN98] Robert M. Gray and David L. Neuhoff. Quantization. *IEEE Transactions on Information Theory*, 44(6):1–63, October 1998.

[GO96] Mehmet I. Gürelli and Levent Onural. A class of adaptive directional image smoothing filters. *Pattern Recognition*, 29(12):1995–2004, 1996.

[Gro88] Richard A. Groeneveld. *Introductory Statistical Methods : An Integrated Approach Using Minitab*. PWS-KENT Publishing Company, 1988.

[GW92] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing*. Addison-Wesley, 3rd edition, 1992.

[HB88] R. J. Hathaway and J. C. Bezdek. Recent convergence results for the fuzzy c-means clustering algorithms. *J. Classification*, 5(2):237–247, 1988.

[HH94] Michael W. Hansen and William E. Higgins. Watershed-driven relaxation labeling for image segmentation. In *International Conference on Image Processing*, pages 460–463. IEEE, 1994.

[HH95] Michael W. Hansen and William E. Higgins. Image enhancement using watershed-based maximum homogeneity filtering. In *IEEE 2nd International Conference on Image Processing*, volume 2, pages 482–485. IEEE, 1995.

[HH99] Michael W. Hansen and William E. Higgins. Watershed-based maximum-homogeneity filtering. *IEEE Trans. Image Processing*, 8(7):982–988, July 1999.

[Hig91] W. E. Higgins. A flexible implementation of maximum-homogeneity filtering for 2-D and 3-D images. *IEEE Trans. Signal Processing*, 39(10):2325–2331, October 1991.

[HipBC] Hippocrates. The oath. 400 B.C. Translated by Francis Adams. `http://classics.mit.edu/Hippocrates/hippooath.html`.

[HM01] Pierre Hansen and Nenad Mladenović. J-MEANS: a new local search heuristic for minimum sums of squares clustering. *Pattern Recognition*, 34:405–413, 2001.

[HP74] S. Horowitz and T. Pavlidis. Picture segmentation by a directed split-and-merge procedure. In *Proc. 2nd Int. Joint Conf. Pattern Recognition (ICPR'74)*, pages 424–433, 1974.

[HR94] Iftekhar Hussain and Todd R. Reed. Segmentation-based nonlinear image smoothing. In *International Conference on Image Processing*, pages 507–511. IEEE, 1994.

[HRA96] D. Hagyard, M. Razaz, and P. Atkin. Analysis of watershed algorithms for greyscale images. In *International Conference on Image Processing*, pages 41–44. IEEE, 1996.

[HS85] Robert M. Haralick and Linda G. Shapiro. Image segmentation techniques. *Computer Vision, Graphics, and Image Processing*, 29:100–132, 1985.

[HSD73] R. M. Haralick, K. Shanmugam, and I. Dinstein. Texture features for image classification. *IEEE Trans. Systems, Mans and Cybernetics*, SMC-3:610–621, 1973.

[HSSB98] Mike Heath, Sudeep Sarkar, Thomas Sanocki, and Kevin Bowyer. Comparision of edge detectors: A methodology and initial study. *Computer Vision and Image Understanding*, 69(1):38–54, January 1998.

[Hua84] N. Huang. Markov model for image segmentation. In *22nd Allerton Conference on Communication, Control, and Computing*, pages 775–781, October 3–5 1984.

[Hub81] Peter J. Huber, editor. *Robust Statistics*. Wiley series in probability and mathematical statistics. John Wiley & Sons, Inc., 1981.

[HW79] J. A. Hartigan and M. A. Wong. A K-means clustering algorithm. *Applied Statistics*, 28:100–108, 1979.

[HW81] R. M. Haralick and L. Watson. A facet model for image data. *Computer Graphics and Image Processing*, 15:113–129, 1981.

[Imm96] John Immerkær. Fast noise variance estimation. *Computer Vision and Image Understanding*, 64(2):300–302, September 1996.

[Jäh93] Bernd Jähne. *Digital Image Processing - Concepts, Algorithms and Scientific Applications*. Springer-Verlag, second edition, 1993.

[Jai89] Anil K. Jain. *Fundamentals of Digital Image Processing*. Prentice Hall, 1989.

[JRvdH93] R. C. Jansen, K. Reinink, and G. W. A M. van der Heijden. Analysis of grey level histograms by using statistical methods for mixtures of distributions. *Pattern Recognition Letters*, 14:585–590, July 1993.

[KD95] Helmut Kopka and Patrick W. Daly. *A Guide to LaTeX2e - Document Preparation for Beginners and Advanced Users*. Addison-Wesley, second edition, 1995.

[KI86] J. Kittler and J. Illingworth. Minimum error thresholding. *Pattern Recognition*, 19(1):41–47, 1986.

[KL51] S. Kullback and R. A. Leibler. On information and sufficiency. *Annals of Mathematical Statistics*, 22(1):79–86, 1951.

[KL91] S. Ko and Y. Lee. Center weighted median filters and their applications to image enhancement. *IEEE Trans. on Circuits and Systems*, 38(9):984–993, September 1991.

[Knu73] D. E. Knuth. *Sorting and Searching*, volume 3 of *The Art of Computer Programming*. Addison-Wesley, 2nd edition, 1973.

[KOA92] T. Kurita, N. Otsu, and N. Abdelmalek. Maximum likelihood thresholding based on population mixture models. *Pattern Recognition*, 25(10):1231–1240, 1992.

[KS00] Klaus Köster and Michael Spann. MIR: An approach to robust clustering — application to range segmentation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 22(5):430–444, May 200.

[KSSC87] Darwin T. Kuan, Alexander A. Sawchuk, Timothy C. Strand, and Pierre Chavel. Adaptive restoration of images with speckle. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-35(3):373–383, March 1987.

[KSW85] J. N. Kapur, P. K. Sahoo, and A. K. C. Wong. A new method for gray-level picture thresholding using the entropy of the histogram. *Computer Vision, Graphics, and Image Processing*, 29:273–285, 1985.

[Kuh00] J. Kuha. Model assessment and model choice: an annotated bibliography. Technical report, Department of Statistics, Pennsylvania State University, April 2000. `http://www.stat.psu.edu/~jkuha/msbib/biblio.html`.

[Kuw76] M. Kuwahara. Processing of RI-angio-cardiographic images. In K. Preston and M. Onoe, editors, *Digital Processing of Biomedical Images*, pages 187–203. Plenum Press, New York, NY, 1976.

[KWT87] M. Kass, A.P. Witkin, and D. Terzopoulos. Snakes: Active contour models. In *International Conference on Computer Vision (ICCV)*, pages 259–268, London, 1987. IEEE.

[LCP90] Sang Uk Lee, Seok Yoon Chung, and Rae Hong Park. A comparitive performance study of several global thresholding techniques for segmentation. *Computer Vision, Graphics, and Image Processing*, 52:171–190, 1990.

[Lee81a] J. S. Lee. Speckle analysis and smoothing of synthetic aperture radar images. *Computer Graphics and Image Processing*, 17:24–32, 1981.

[Lee81b] Jong-Sen Lee. Refined filtering of image noise using local statistics. *Computer Graphics and Image Processing*, 15:380–389, 1981.

[Lee83] J-S. Lee. Digital image smoothing and the sigma filter. *Computer Vision, Graphics, and Image Processing*, 24:255–269, 1983.

[len72] Playmate of the Month: Lenna Sjööblom. *Playboy: Entertainment for Men*, pages 134–141, November 1972.

[Li01] S. Z. Li. *Markov random field modeling in image analysis*. Springer, 2001.

[Lin96] Tetra Lindarto. MML segmentation-based image coding. Master's thesis, Department of Computer Science, Monash University, May 1996.

[LL96] C. K. Leung and F. K. Lam. Maximum segmented-scene spatial entropy thresholding. In *International Conference on Image Processing*, pages 963–966. IEEE, 1996.

[Llo82] S. P. Lloyd. Least squares quantization in PCM. *IEEE Trans. Inf. Theory*, 28:129–137, 1982.

[LM84] M. D. Lema and O. R. Mitchell. Absolute moment block truncation coding and its application to color images. *IEEE Trans. Commun.*, 32:1148–1157, 1984.

[Mas85] G. A. Mastin. Adaptive filters for digital image noise smoothing: An evaluation. *Computer Vision, Graphics, and Image Processing*, 31:103–121, 1985.

[MB90] F. Meyer and S. Beucher. Morphological segmentation. *J. Vis. Commun. Image Represent.*, 1(1):21–46, 1990.

[McD81] M. J. McDonnell. Box-filtering techniques. *Computer Graphics and Image Processing*, 17:65–70, 1981.

[McQ67] J. B. McQueen. Some methods of classification and analysis of multivariate observations. In L.M. Le Cam and J. Neyman, editors, *Proceedings of Fifth Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297, 1967.

[MD80] O. R. Mitchell and E. J. Delp. Multilevel graphics representation using block truncation coding. *Proc. IEEE*, 68:868–873, 1980.

[MJ66] C. E. K. Mees and T. H. James. *The Theory of the Photographics Process*. Macmillan, 1966.

[MJR90] P. Meer, J. Jolion, and A. Rosenfeld. A fast parallel algorithm for blind estimation of noise variance. *IEEE Trans. Pattern Anal. Mach. Intelligence*, 12(2):216–223, 1990.

[MM99] Jean-Bernard Martens and Lydia Meesters. The role of image dissimilarity in image quality models. In B. E. Rogowitz and T. N. Pappas, editors, *Proceedings of SPIE: Human Vision and Electronic Imaging IV*, volume 3644, pages 258–269, January 1999.

[MP00] Geoffrey McLachlan and David Peel. *Finite Mixture Models*. Wiley series in probability and statistics. John Wiley & Sons, Inc., 2000.

[MPA00] Volker Metzler, Marc Puls, and Til Aach. Restoration of ultrasound images by nonlinear scale-space filtering. In Edward R. Dougherty and Jaakko T. Astola, editors, *SPIE Proceedings: Nonlinear Image Processing XI*, volume 3961, pages 69–80. SPIE, 2000.

[MR74] Frederick Mosteller and Robert E. K. Rourke, editors. *Sturdy Statistics: Nonparametric and Order Statistics*. Addison-Wesley, 1974.

[MR95] Kai-Kuang Ma and Sarah A. Rajala. New properties of AMBTC. *IEEE Signal Processing Letters*, 2(2):34–36, February 1995.

[MT97] Bernd Meyer and Peter E. Tischer. TMW – a new method for lossless image compression. In *Picture Coding Symposium (PCS'97)*, pages 533–538, Berlin, Germany, 1997. VDE-Verlag GMBH.

[MT01] Bernd Meyer and Peter Tischer. Glicbawls — grey level image compression by adaptive weighted least squares. In J. A. Storer and M. Cohn, editors, *Proceedings of the Data Compression Conference (DCC'2001)*, Piscataway, NJ, USA, March 2001. IEEE Service Center.

[NH88] A. Netravali and B. Haskell. *Digital Pictures, Representation and Compression.* Plenum, New York, 1988.

[Nie94] Yves Nievergelt. Total least squares: State-of-the-art regression in numerical analysis. *SIAM Review*, 36(2):258–264, June 1994.

[NR79] Yasuo Nakagawa and Azriel Rosenfeld. Some experiments on variable thresholding. *Pattern Recognition*, 11:191–204, 1979.

[NWM91] P. Nasiopoulos, R. Ward, and D. Morse. Adaptive compression coding. *IEEE Trans. Commun.*, 39:1245–1254, 1991.

[OB94] Jonathan J. Oliver and Rohan A. Baxter. MML and Bayesianism: Similarities and differences. Technical Report 206, Department of Computer Science, Monash University, December 1994.

[OH94] Jonathan J. Oliver and David Hand. Introduction to minimum encoding inference. Technical Report 205, Department of Computer Science, Monash University, November 1994.

[Ols93] S. I. Olsen. Estimation of noise in images: An evaluation. *CVGIP: Graphical Models and Image Processing*, 55(4):319–323, July 1993.

[Ots79] N. Otsu. A threshold selection method from gray level histograms. *IEEE Trans. Systems, Man and Cybernetics*, 9:62–66, March 1979.

[PG94] N. Papamarkos and B. Gatos. A new approach for multilevel threshold selection. *CVGIP: Graphical Models and Image Processing*, 56(5):357–370, September 1994.

[PH66] E. S. Pearson and H. O. Heartly, editors. *Biometrika Tables for Statisticians*, volume 1. Cambridge University Press, 3rd edition, 1966.

[Pir80] P. Pirsch. A new predictor design for DPCM coding of TV signals. In *ICC Conference Report (International Conf. on Communications)*, pages 31.2.1–31.2.5, Seattle, WA, 1980.

[Pit95] Ioannis Pitas. *Digital Image Processing Algorithms*. Prentice Hall International Series in Acoustics, Speech and Signal Processing. Prentice Hall, 1995.

[PM66] J. M. Prewitt and M. L. Mendelsohn. The analysis of cell images. *Ann. New York Acad. Sci.,*, 128:1025–1053, 1966.

[PM90] P. Perona and J. Malik. Scale-space and edge detection using anisotropic diffusion. *IEEE Trans. Pattern Anal. Machine Intell.*, 12(7):629–639, July 1990.

[PM93] W. Pennebaker and J. Mitchell. *JPEG: Still Image Compression Standard*. Van Nostrand Reinhold, New York, 1993.

[PMGL88] W.B. Pennebaker, J.L. Mitchell, and R.B. Arps G.G. Langdon. An overview of the basic principles of the Q-coder adaptive binary arithmetic coder. *IBM Journal of research and development*, 32(6):771–726, November 1988.

[PP93] Nikhil R. Pal and Sankar K. Pal. A review on image segmentation techniques. *Pattern Recognition*, 26(9):1277–1294, 1993.

[Pre89] S. J. Press. *Bayesian Statistics*. Wiley, 1989.

[PV90] I. Pitas and A. N. Venetsanopoulos. *Nonlinear Digital Filters: Principles and Applications*. Kluwer Academic, 1990.

[QS95] M. K. Quweider and E. Salari. Gradient-based block truncation coding. *Electronics Letters*, 31(5):353–354, March 1995.

[RC78] T.W. Ridler and S. Calvard. Picture thresholding using an iterative selection method. *IEEE Transactions on Systems, Man and Cybernetics*, SMC-8(8):630–632, August 1978.

[RHZ76] A. Rosenfeld, R. A. Hummel, and S. W. Zucker. Scene labeling by relaxation operations. *IEEE Trans. on Systems, Man and Cybernetics*, 6:420–433, 1976.

[Ris78] J. Rissanen. Modeling by shortest data description. *Automatica*, 14:465–471, 1978.

[Ris87] Jorma Rissanen. Stochastic complexity. *J. R. Statist. Soc. B*, 49(3):223–239, 1987.

[Ris00] Jorma Rissanen. MDL denoising. *IEEE Trans. on Information Theory*, 46(7):2537–2543, 2000.

[RJ91] Majid Rabbani and Paul W. Jones. *Digital Image Compression Techniques*. SPIE Optical Engineering Press, 1991.

[RL87] Peter J. Rousseeuw and Annick. M. Leroy, editors. *Robust regression and outlier detection*. Wiley series in probability and mathematical statistics: Applied probability and statistics. John Wiley & Sons, Inc., 1987.

[RLU99] K. Rank, M. Lendl, and R. Unbehauen. Estimation of image noise variance. *IEE Proc.-Vis. Image Signal Process.*, 146(2):80–84, April 1999.

[Ros81] A. Rosenfeld. Image pattern recognition. *Proc. IEEE*, 69:596–605, 1981.

[RR98] Stephen J. Roberts and Iead Rezek. Bayesian approaches to Gaussian mixture modeling. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 20(11):1133–1142, November 1998.

[RRK84] S. S. Reddi, S. F. Rudin, and H. R. Keshavan. An optimal multiple threshold scheme for image segmentation. *IEEE Trans. Systems, Man, and Cybernetics*, 14(4):661–665, July/August 1984.

[SB97] S. M. Smith and J. M. Brady. SUSAN - a new approach to low level image processing. *International Journal of Computer Vision*, 23(1):45–78, May 1997.

[Sha48] C. E. Shannon. A mathematical theory of communication. *Bell Systems Technical Journal*, 27:379–423, 1948.

[She90] Q. Shen. Fuzzy image smoothing. In *proc. Int. Conf. on Pattern Recognition*, pages 74–78, 1990.

[SKYS] Brian V. Smith, Paul King, Ken Yap, and Supoj Sutanthavibul. XFig: vector graphics drawing software. `http://www.xfig.org/`.

[SMCM91] Philippe Saint-Marc, Jer-Sen Chen, and Gérard Medioni. Adaptive smoothing: A general tool for early vision. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 13(6):514–529, June 1991.

[Smi95] S. M. Smith. SUSAN - a new approach to low level image processing. Technical Report TR95SMS1, Defence Research Agency, Chobham Lane, Chertsey, Surrey, UK, 1995. `http:/www.fmrib.ox.ac.uk/~steve/`.

[Smi96] S. M. Smith. Flexible filter neighbourhood designation. In *Proc. 13th Int. Conf. on Pattern Recognition*, volume 1, pages 206–212, 1996.

[Sob70] I. E. Sobel. *Camera Models and Machine Perception*. PhD thesis, Electrical Engineering Department, Stanford University, Stanford, CA., 1970.

[SSW88] P. K. Sahoo, S. Soltani, and A. K. C. Wong. A survey of thresholding techniques. *Computer Vision, Graphics and Image Processing*, 41:233–260, 1988.

[ST97] Torsten Seemann and Peter E. Tischer. An MML approach to noise cleaning. In *Picture Coding Symposium (PCS'97)*, pages 39–43, Berlin, Germany, September 1997. VDE-Verlag GMBH.

[ST98] Torsten Seemann and Peter E. Tischer. Structure preserving noise filtering using explicit local segmentation. In *Proc. IAPR Conf. on Pattern Recognition (ICPR'98)*, volume II, pages 1610–1612, August 1998.

[Sta99] R. C. Staunton. Hexagonal sampling in image processing. *Advances in Imaging and Electron Physics*, 107:231–307, April 1999.

[STM97] Torsten Seemann, Peter E. Tischer, and Bernd Meyer. History based blending of image sub-predictors. In *Picture Coding Symposium (PCS'97)*, pages 147–151, Berlin, Germany, September 1997. VDE-Verlag GMBH.

[TA97] Mark Tabb and Narendra Ahuja. Multiscale image segmentation by integrated edge and region detection. *IEEE Trans. Image Processing*, 6(5):642–655, May 1997.

[Tag96] Akira Taguchi. A design method of fuzzy weighted median filters. In *International Conference on Image Processing*, pages 423–426. IEEE, 1996.

[TH94] Patrick Teo and David Heeger. Perceptual image distortion. In *First IEEE International Conference on Image Processing*, volume 2, pages 982–986, November 1994.

[Tho18] W. M. Thorburn. The myth of Occam's razor. *Mind*, 27:345–353, 1918.

[Tis94] Peter E. Tischer. Optimal predictors for image compression. Technical Report 189, Department of Computer Science, Monash University, 1994.

[TSM85] D. M. Titterington, A. F. M. Smith, and U. E. Makov. *Statistical Analysis of Finite Mixture Distributions*. John Wiley & Sons Ltd., 1985.

[Tuk71] J. W. Tukey. *Exploratory Data Analysis*. Addison-Wesley, 1971.

[VS91] Luc Vincent and Pierre Soille. Watersheds in digital spaces: An efficient algorithm based on immersion simulations. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 13(6):583–598, 1991.

[WB68] C. S. Wallace and D. M. Boulton. An information measure for classification. *Computer Journal*, 11(2):185–194, 1968.

[WD00] C. S. Wallace and D. L. Dowe. MML clustering of multi-state, Poisson, von Mises circular and Gaussian distributions. *Journal of Statistics and Computing*, 10(1):73–83, January 2000.

[Wei97] Joachim Weickert. A review of nonlinear diffusion filtering. In B. ter Haar Romney, L. Florack, J. Koenderink, and M. Viergever, editors, *Scale-space theory in computer vision*, volume 1252, pages 3–28. Springer-Verlag, 1997.

[Wes78] Joan S. Weszka. A survey of threshold selection techniques. *Computer Graphics and Image Processing*, 7:259–265, 1978.

[WF87] C. S. Wallace and P. R. Freeman. Estimation and inference by compact coding. *J. R. Statist. Soc. B*, 49(3):240–265, 1987.

[WKL+] Thomas Williams, Colin Kelley, Russell Lang, Dave Kotz, John Campbell, Gershon Elber, and Alexander Woo. gnuplot: graph plotting software. `http://www.gnuplot.info/`.

[WM96] Xiaolin Wu and Nasir Memon. CALIC - a context based adaptive lossless image codec. *IEEE ASSP*, 4:1890–1893, 1996.

[WR78] Joan S. Weszka and Azriel Rosenfeld. Threshold evaluation techniques. *IEEE Trans. Systems, Man, and Cybernetics*, 8(8):622–629, August 1978.

[WSS00] Marcelo J. Weinberger, Gadiel Seroussi, and Guillermo Sapiro. The LOCO-I lossless image compression algorithm: Principles and standardization into JPEG-LS. volume 9, pages 1309–1324, August 2000.

[WVL81] D. C. C. Wang, A. H. Vagnucci, and C. C. Li. Gradient inverse weighted smoothing scheme and the evaluation of its performance. *CGIP*, 15(2):167–181, February 1981.

[YG01] G. Z. Yang and D. F. Gillies. *Computer Vision Lecture Notes*. Department of Computing, Imperial College, UK, 2001. `http://www.doc.ic.ac.uk/~gzy`.

[ZG94] Y. J. Zhang and J. J. Gerbrands. Objective and quantitative segmentation evaluation and comparison. *Signal Processing*, 39:43–54, 1994.

[Zha97] Y. J. Zhang. Evaluation and comparison of different segmentation algorithms. *Pattern Recognition Letters*, 18:963–974, 1997.