Master's thesis

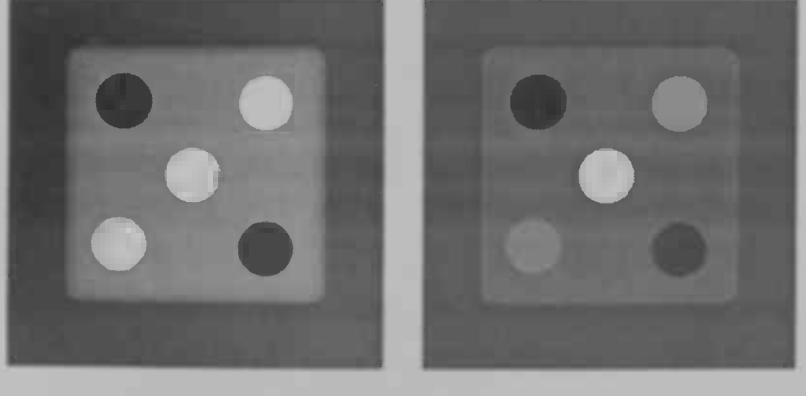
Local quantisation based on Robust Automatic Threshold Selection

Ralf van den Broek

Institute for Mathematics and Computing Science
University of Groningen
The Netherlands

Supervisor: Michael H. F. Wilkinson

31st August 2004



Contents

1 Introduction 1
1.1 Related work . . . 2

2 Binarisation 3
2.1 Introduction . . . 3
2.2 Otsu's algorithm . . . 3
2.3 Robust Automatic Threshold Selection . . . 5
2.4 Local RATS . . . 6

3 Quantisation 9
3.1 Introduction . . . 9
3.2 Contrast enhancement . . . 9
3.3 Generalised Otsu's algorithm . . . 10

4 Multilevel RATS 13
4.1 Introduction . . . 13
4.2 Naive method . . . 13
4.3 Clustering . . . 14
4.4 Step edges, the continuous case . . . 15
4.5 Step edges, the discrete case . . . 17
4.6 Blurred step edges . . . 18
4.7 Modified algorithm . . . 21
4.8 Results . . . 22

5 Local Multilevel RATS 25
5.1 Introduction . . . 25
5.2 Matching thresholds . . . 25
5.3 Window size and shape . . . 26
5.4 Algorithm . . . 27
5.5 Results . . . 28

6 Noise reduction techniques 31
6.1 Introduction . . . 31
6.2 $\lambda\sigma$ method . . . 31
6.3 Mean shift method . . . 32
6.4 Area filtering . . . 33

7 Experiments 35
7.1 Introduction . . . 35
7.2 Phantom images . . . 35
7.3 Window size and shape . . . 36
7.4 Gaussian blurring . . . 36
7.5 Mean shift filtering . . . 36
7.6 Parameters . . . 40
7.7 Real images . . . 41
7.8 Image references . . . 41

8 Conclusions 45
8.1 Future work . . . 45

Bibliography 47

GNU Free Documentation License 49

Abstract

Binarisation is a common image operation, which separates an image into two classes, and there are many algorithms for this operation. However, there are fewer algorithms for the generalisation of binarisation to multiple classes, quantisation.

Robust Automatic Threshold Selection (RATS) is a well-known technique for binarisation based on a simple image statistic, due to Kittler *et al.* [4]. However, as opposed to some other methods, such as Otsu's algorithm [7], RATS does not readily extend to quantisation.

In this paper, we will propose a method to perform quantisation using an adaptation of RATS. We will prove that this method is correct with respect to step edges, and use it as a basis for a local quantisation algorithm. We will experiment with the local version of the algorithm, which segments an image into a number of objects, to find good values for the algorithm's parameters. Finally, we will show some results of applying the local algorithm to a number of real-life examples.

Chapter 1

Introduction

In many image analysis problems, we are presented with an image of a number of objects, taken by a camera or other recording device. Many such problems require that we for instance be able to track or count those objects, or be able to make statements about the size or shape of the objects. The original grey-scale images are often too complex to work with without preprocessing, so we must somehow be able to determine roughly where the objects end.

Traditionally, this is often done using binarisation algorithms, which divide the image in two classes, usually using thresholding. This is often a useful method, because the foreground objects share the same intensity level, for instance if we are interested in counting the number pips on a die we rolled, where the pips are all black, and the die is white. In this case, binarisation will separate the pips from the die.

However, imagine we have a more colourful die, with a midtone background, and white and black pips. Now, binarisation will either lump in the white pips with the background, or the black pips with the background, because we can only use two classes. In this case, we will need to be able to segment the image in more than two classes, and then decide for each class whether it is a pip or the background, using for instance the area of the object.

In this paper, we will examine the Robust Automatic Threshold Selection (RATS), a binarisation algorithm due to Kittler *et al.* [4], which is based on a weighted average of image grey values, and we propose a multi-level adaptation of this algorithm, using weighted grey level histograms instead of averages. We will prove the correctness of this algorithm when dealing with step edges, both in the continuous and the discrete case, and we will show some improvements to the basic method.

Then, we will make a local adaptation of this method, which is more suitable for segmentation, such as the dice problem mentioned above, especially in the presence of gradients. We will experiment with this method to find suitable values for the algorithm's parameters, and show some results on real-life images.

Finally, we will briefly mention some noise reduction techniques and their application in our algorithm.

1.1 Related work

The RATS algorithm was first described by Kittler *et al.* [4]. A global version is described, using the maximum of the orthogonal gradients as the edge detection method. Local thresholding is referred to, but not explored. However, Gaussian noise is considered and a method to reduce the bias due to noise is described.

The simple edge detection method used in traditional RATS can be replaced with more sophisticated methods, and it has been shown by Wilkinson [10] that all reasonable weight functions, i.e. those that give an even response to a step edge at the origin, yield the optimum threshold for noise-free orthogonal step edges.

Other extensions include a multi-scale moving-window method due to Wilkinson [11], which employs knowledge about the distribution of noise to compute the reliability of thresholds at a particular scale, and uses Gaussian windows to capture information about the entire image in each point.

A well-known traditional histogram-based thresholding algorithm is due to Otsu [7], which employs a number of statistical criteria to determine a probabilistically optimal threshold. Though it is usually used for binarisation, it can be generalised trivially to multi-thresholding.

A field related to multi-thresholding is contrast enhancement. In [1], Beghdadi and Le Negrate describe a method for contrast enhancement using a similar method to RATS to calculate the edge gray value in each pixel and then calculate contrast based on that value.

Chapter 2

Binarisation

2.1 Introduction

A binarisation algorithm is an algorithm that, for each pixel in the input image, decides whether it is part of the foreground, or the background of the image. Generally, a threshold is calculated which divides foreground and background pixels. This is often desirable, because this usually makes the image easier to process with image analysis algorithms, such as OCR applications.

The assumption here is that the image has a bimodal distribution. This is reasonable because we are trying to separate the image in two distinct classes, which generally does not make sense for a unimodal or multi-modal distribution.

Some algorithms, such as Otsu's algorithm [7], described in Section 2.2, use a histogram of the image to find the threshold, essentially looking for a valley in the assumed bimodal distribution.

Other algorithms, such as RATS [4], described in Section 2.3, do not make use of a histogram, but calculate the threshold as a function of certain image characteristics. In the case of RATS, these are the grey values of the pixels and their response to an edge detection filter.

Both these methods calculate a single threshold for the entire image. While we can easily adapt these to find localised thresholds, thresholds that depend on the local grey level distribution, by simply dividing the image into segments, there are also algorithms specifically adapted to local thresholding. For instance, Wilkinson proposed [11] a localised variant of RATS, which uses Gaussian windows to give a greater weight to pixel's neighbourhood. This method is described in Section 2.4.

2.2 Otsu's algorithm

A well-known histogram-based thresholding algorithm is due to Otsu [7], which attempts to evaluate the "goodness" of a threshold by assuming that the grey-scale image's pixels can be split into two classes, and then maximises a "goodness" measure to find the optimal threshold.

Let the image consist of pixels represented in L grey levels $[1...L]$. The number of pixels at level i is n_i and the total number of pixels is $N = \sum_{i=1}^{L} n_i$.

(a) Original (b) Otsu's algorithm

Figure 2.1: Binarisation using Otsu's algorithm

We consider the grey level histogram as a probability distribution where $p_i = n_i/N$. We can then calculate the class probabilities ω_0 and ω_1, the class means μ_0 and μ_1, the class variances σ_0^2 and σ_1^2 and the total mean level μ_T.

These values are used to calculate the within-class variance σ_W^2 and the between-class variance σ_B^2, as well as the total variance σ_T^2:

$$\sigma_W^2 = \omega_0\sigma_0^2 + \omega_1\sigma_1^2, \tag{2.1}$$

$$\sigma_B^2 = \omega_0(\mu_0 - \mu_T)^2 + \omega_1(\mu_1 - \mu_T)^2, \tag{2.2}$$

$$\sigma_T^2 = \sum_{i=1}^{L}(i - \mu_T)^2 p_i. \tag{2.3}$$

Otsu then defines three measures of goodness,

$$\lambda = \sigma_B^2/\sigma_W^2, \tag{2.4}$$

$$\kappa = \sigma_T^2/\sigma_W^2 \text{ and} \tag{2.5}$$

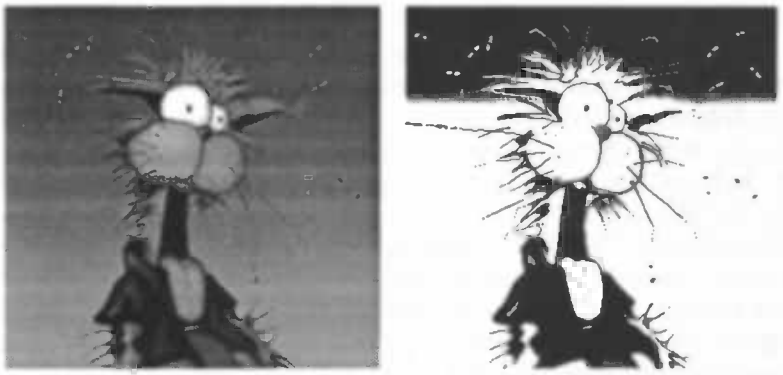
$$\eta = \sigma_B^2/\sigma_T^2. \tag{2.6}$$

Maximising any of these will lead to an optimal threshold, however, η is the simplest of these measures with respect to the threshold k, as σ_B^2 is based on first-order statistics whereas σ_W^2 is based on second-order statistics, which are more difficult to compute. As σ_T^2 is constant with respect to the threshold k, we only need to maximise σ_B^2. Rewriting (2.2) yields

$$\sigma_B^2(k) = \frac{(\mu_T\omega(k) - \mu(k))^2}{\omega(k)(1 - \omega(k))}, \tag{2.7}$$

where $\omega(k) = \Sigma_{i=1}^{k} p_i$ and $\mu(k) = \Sigma\, i = 1^k i p_i$. The optimal threshold k^* can then be found by

$$\sigma_B^2(k*) = \max_{i \le k < L} \sigma_B^2(k). \tag{2.8}$$



(a) Original (b) RATS

Figure 2.2: Binarisation using RATS

The value of η at the optimal threshold k^*, η^*, is also a measure of separability, where the extremes $\eta^* = 0$ denotes an image with a single constant grey value, and $\eta^* = 1$ denotes an image with exactly two grey values.

As we can see in Fig. 2.1, Otsu's algorithm yields a good result in this case. The background gradient is split in two classes, but this is a common issue with global thresholding. Also, the shirt is classified in a single class, while in the original image it is clear that there are two discernible grey levels in the shirt.

2.3 Robust Automatic Threshold Selection

The Robust Automatic Threshold Selection (RATS) algorithm is due to Kittler *et al.* [4]. Rather than using histogram analysis, RATS calculates the threshold based on the average grey value of the image, weighted according to the pixel's response to an edge detection filter.

The edge detection filter used by Kittler *et al.* is the maximum of the magnitudes of the x and y derivatives of the image p,

$$e(x, y) = \max(|g_x(x, y)|, |g_y(x, y)|), \tag{2.9}$$

where $g_x = p(x-1, y) - p(x+1, y)$ and similarly for g_y, and $p(x, y)$ is the grey value of the pixel at position (x, y).

The threshold is then calculated by taking the sum of the products of the pixel values and the edge responses and dividing it by the sum of the edge responses:

$$T_0 = \frac{\Sigma_x \Sigma_y p(x, y) e(x, y)}{\Sigma_x \Sigma_y e(x, y)}. \tag{2.10}$$

While Kittler *et al.* have shown only that RATS yields the optimal threshold when using $e(x, y)$ (2.9), Wilkinson shows [10] that all reasonable weight functions, that is, all weight functions that are even and integrable, and are

limited to a domain $[-a, a]$, yield the optimal threshold for noise-free, straight step edges, in the presence of symmetric blur.

Wilkinson also approaches the problems of curvature bias and noise bias. Curvature bias is caused by the fact that the weight function will also respond to pixels near an edge. In fact, we can use this effect to calculate the effective radius r_{eff}, which measures how sharply the weight function peaks around an edge. To reduce curvature bias, then, we should select a weight function which peaks sharply around edges.

Noise bias is a bias due to noise, usually uniform Gaussian noise, and Poisson noise. We can generally reduce sensitivity to noise by thresholding the values returned by the edge filter and ignoring any pixels that fall below this threshold, however, this comes at a price of reduced sensitivity to actual edges.

Another consideration is that it is usually preferable to have a weight function that is invariant to edge orientation. The $e(x, y)$ filter, for instance, has a response to blurred diagonal edges which is $1/\sqrt{2}$ times the response to blurred edges parallel to the coordinate axes.

Wilkinson compares eight weight functions, and the results show that the quadratic Sobel filter, g^2_{Sobel} (see Eq. (4.1)), performed the best compared to the other filters, in this experimental setting.

An example of an application of RATS is shown in Fig. 2.2. The quality of the thresholding is comparable to that of Otsu's algorithm, and the same issues are present concerning the background gradient and the cat's shirt.

2.4 Local RATS

A multi-scale, moving-window variant of RATS is due to Wilkinson [11]. This algorithm uses the g^2_{Sobel} edge filter, in combination with the $\lambda\sigma$ noise threshold to suppress noise, which becomes $\lambda^2\sigma^2$ because the algorithm uses the g^2_{Sobel} filter.

The weight function and the weight-image product are convolved with a Gaussian kernel to achieve local thresholding. At each point, the value of the convolved functions is contributed to by each pixel in the image, and the threshold at a point is calculated by dividing the convolved weight-image product at that pixel by the convolved weight at that pixel:

$$T_\sigma(x, y) = \frac{(G_\sigma * (w \cdot p))(x, y)}{(G_\sigma * w)(x, y)}. \tag{2.11}$$

A problem with local thresholds is that if the window is too small, there will not be an edge inside the window, and although the Gaussian has an infinite impulse response, the threshold will be dominated by noise gradients if there is no edge nearby. To prevent this, Wilkinson uses multiple scales in succession, and for each pixel the algorithm uses the threshold provided by the smallest scale where $(G_\sigma * w)(x, y)$ exceeds a threshold so that this value differs significantly from the value expected from noise. If no such scale can be found among the scales used, a global threshold is used.

As we can see in Fig. 2.3, local thresholding solves the issue with the cat's shirt, which is separated in two classes in this image. There is also an more detail in the hair. However, the background has become fragmented as it is

(a) Original (b) MS-RATS

Figure 2.3: Binarisation using Multiscale RATS

classified as foreground or background depending on the contents of the local window. Where there are bright image details present, such as the flies, the background is black in a circle around the detail.

Chapter 3

Quantisation

3.1 Introduction

In binarisation, an image is reduced to two colours, or classes, generally indicated with white and black. This corresponds to the assumption that the grey value distribution of the image is bimodal.

For some applications, this is not the case, and more classes are necessary. This can be achieved by generalising the concept of binarisation from two colours to n colours. This generalisation is called *quantisation*. Binarisation itself, then, is a specific instance of quantisation where $n = 2$.

There are two approaches to quantisation. The simpler approach is to supply the algorithm with the desired value of n. This approach is used in the multilevel variant of Otsu's algorithm [7]. If the number of classes in the image's distribution is known, this is an adequate method. We will examine this method in more detail in Section 3.3.

The other approach is to let the algorithm choose a suitable value of n, and we will use this approach in the multilevel extension to RATS, in Chapter 4.

We will also briefly mention contrast enhancement in Section 3.2.

3.2 Contrast enhancement

A field closely related to multi-thresholding is contrast enhancement. Rather than enhancing the image by reducing the number of colours in the image, we enhance the contrast, usually by improving the contrast around the edges in the image.

Beghdadi and Le Negrate [1] describe a method based on Gordon's method [3], which is remarkably similar to RATS.

At each point, the mean edge grey level is calculated in a square window W_{kl} around the point, by using the absolute value of the Laplacian Δ_{ij} as the edge detection filter:

$$\bar{E}_{kl} = \frac{\Sigma_{W_{kl}} \Delta_{ij} \cdot X_{ij}}{\Sigma_{W_{kl}} \Delta_{ij}}. \tag{3.1}$$

Given the edge values, the contrast in each point can be calculated by

$$C_{kl} = \frac{|X_{kl} - \bar{E}_{kl}|}{|X_{kl} + \bar{E}_{kl}|} \tag{3.2}$$

and the contrast can then be enhanced. This can be done by any function which is greater than or equal to its input on the range $[0, 1]$ and is limited to $[0, 1]$ for input in the range $[0, 1]$. The square root function is a simple function that satisfies these conditions, but there are other, more esoteric functions mentioned by Beghdadi and Le Negrate. After calculating the new contrast value, we can solve the equation given above for X_{kl} to find the new pixel grey value.

Beghdadi and Le Negrate compared their method to Gordon's method [3], and a number of classical contrast enhancement algorithms. They conclude that their method is less sensitive to noise and digitisation effects, and sharpens the edges in the image.

3.3 Generalised Otsu's algorithm

Otsu's algorithm is easily extended to perform quantisation instead of binarisation. For example, see Fig. 3.1.

The "goodness" measures used in the basic variant can be generalised to any number of classes. As noted by Otsu [7], the easiest goodness measure is σ_B^2, the between-class variance. For the binary case, this is defined as (2.2)

$$\sigma_B^2 = \omega_0(\mu_0 - \mu_T)^2 + \omega_1(\mu_1 - \mu_T)^2,$$

where ω_0 and ω_1 are the class occurrence probabilities, μ_0 and μ_1 are the class means and μ_T is the total mean level.

This formula can be extended to more than two classes by adding terms for the additional classes to the formula:

$$\sigma_B^2 = \sum_{i=1}^{n} \omega_i(\mu_i - \mu_T)^2. \tag{3.3}$$

In the binary case, the value of σ_B^2 is dependent on only a single threshold, thus, finding the optimal threshold is a single- variable optimisation problem. In the general case, there are $n - 1$ thresholds in strict ascending order, so the problem becomes an $n - 1$-variable optimisation problem. The number of possible combinations of thresholds can become very large as the number of thresholds increases.

Also, there are several issues with the algorithm that limit its usefulness as a generic quantisation algorithm. Most importantly, the number of classes has to be passed to the algorithm as a parameter. Also, although there are good arguments for the validity of the goodness measures in the binary case, it is not certain that the measures give good results in multilevel cases.

The images in Fig. 3.1 are a good result if we are interested in reducing the number of colours in the image. However, as a segmentation algorithm, this algorithm is less suitable, as there is no clear correspondence between objects in the image and colours in the quantisation. The algorithm cannot distinguish between objects with the same grey value. Also, the fact that we have to supply the number of classes can be a large problem in some cases, for instance if

(a) Original (b) Otsu (n=2) (c) Otsu (n=3) (d) Otsu (n=4)

Figure 3.1: Quantisation using generalised Otsu's algorithm

an image contains more classes due to an anomaly, for instance, a tumour in the human brain. In this case, the number of classes may be specified with a healthy brain in mind, and the quantisation may be incorrect when an additional class, the tumour, is introduced.

Chapter 4

Multilevel RATS

4.1 Introduction

As noted before, the RATS algorithm does not readily extend to quantisation. The metric used by RATS, a weighted average over the image, does not lend itself to be extended, so we will need another approach.

The central idea is that, instead of using the weighted average, the quantisation algorithm uses a weighted histogram, and extracts the thresholds from that histogram. Using a clustering algorithm, the histogram is divided in clusters, and the cluster centres are used as the thresholds. Because the edges of the image have a higher weight, they are likely to be used as thresholds, which should give a good segmentation.

In this chapter, we will first show a basic implementation of this idea, and prove its correctness for step edges in the continuous and the discrete case. We will then show that this method cannot cope with noise, and show a method to fix this problem. We will also prove that this method is still correct.

4.2 Naive method

Our initial approach, shown in Algorithm 4.1, was to create a grey level histogram of the image, weighted according to the response to an edge detection filter. The resulting histogram was then clustered, and the cluster centres were used as thresholds. The clustering algorithm is described in the next section.

Algorithm 4.1 MULTILEVELRATS

$H \leftarrow$ HISTOGRAM(f, w) {f is the image function, w is the weight function}
CLUSTER$(H, -1)$
$T \leftarrow$ CLUSTERCENTRES(H)
MULTITHRESHOLD(f, T)

The rationale behind this is that, because we are weighing the histogram according to edge strength, the grey values at the edges of objects would be strongly represented in the histogram. After clustering, we would expect these grey values to become cluster centres, and thus thresholds. In this way, we

should get thresholds which will separate the classes at the edges of objects in the image. The HISTOGRAM function we use in the naive method is shown in Algorithm 4.2.

Algorithm 4.2 HISTOGRAM(f, w)

```
H[0...255] ← {0,0,...,0}
for all image pixels f(x,y) do
  if f(x,y) is not a noise pixel then {See Chapter 6}
    H[f(x,y)] ← H[f(x,y)] + w(x,y)
  end if
end for
return H
```

It should be noted that we chose the squared Sobel gradient as our weight function:

$$w = g^2_{\text{Sobel}} = \Delta^2_{x,\text{Sobel}} + \Delta^2_{y,\text{Sobel}}, \tag{4.1}$$

where

$$\Delta_{x,\text{Sobel}}(x,y) = \frac{1}{4}(\Delta_x(x,y-1) + 2\Delta_x(x,y) + \Delta_x(x,y+1)) \tag{4.2}$$

$$\Delta_x = f(x-1,y) - f(x+1,y), \tag{4.3}$$

and $\Delta_{y,\text{Sobel}}$ is defined similarly.

As shown by Wilkinson [10], this weight function has better properties than a number of other common weight functions, as well as being easy to calculate.

4.3 Clustering

In [2], Fukunaga and Hostetler describe a method for non-parametric density gradient estimation with a generalised kernel approach. Of particular interest regarding multilevel thresholding is the gradient clustering algorithm described in the article.

In this algorithm, each observation is shifted in the direction of the gradient at that point, proportional to the magnitude of the gradient. This has the effect that the observations are transformed in a number of tight clusters around the maxima in the distribution.

If we define $X^0_j \equiv X_j$ for all j, the observations are repeatedly transformed according to

$$X^{i+1}_j = X^i_j + a\nabla_x \ln p(X^i_j). \tag{4.4}$$

Fukunaga and Hostetler show that for sufficiently small a, X will converge.

The normalised gradient $\nabla_x \ln p(X) = \nabla_x p(X)/p(X)$ can be estimated using the mean shift gradient estimate

$$M_h(X) = \sum_{S_{h_C}(x)} (X_i - X) \tag{4.5}$$

where $S_{h_C}(X)$ is a small region around X, using

$$\hat{\nabla}_x \ln p_N(X) \equiv \frac{n+2}{h_C^2} M_{h_C}(X) \tag{4.6}$$

Silverman [8] notes that for

$$a = \frac{h_C^2}{d+2}, \tag{4.7}$$

where d is the number of dimensions, the clustering algorithm is very simple: at each step, an observation is shifted to the mean value of all observations lying within Euclidean distance h_C of the observation, and this step is repeated until the histogram no longer changes, as shown in Algorithm 4.3. We will use the value of a given in (4.7).

Fukunaga and Hostetler note that it is possible to cluster a Gaussian distribution in one step by correctly choosing a value for a. They show this for $\sigma = 1$, but in general, a can also be calculated.

We assume the distribution is a circular Gaussian distribution:

$$p(X) = (2\pi\sigma^2)^{-n/2} \exp(-(X-M)^T(X-M)(2\sigma^2)^{-1}). \tag{4.8}$$

We can then solve $X_j^1 = M$ for a, using (4.4)

$$\begin{aligned} X_j^1 &= X_j + a\nabla_x \ln p(X_j) \\ &= X_j + a\nabla_x(\ln(2\pi\sigma^2)^{-n/2} + (-(X_j-M)^T(X_j-M)(2\sigma^2)^{-1})) \\ &= X_j + a\nabla_x(-(X_j-M)^T(X_j-M)(2\sigma^2)^{-1}) \\ &= X_j + a(-(X_j-M)\sigma^{-2}) \end{aligned}$$

and filling in $a = \sigma^2$ yields

$$X_j^1 = X_j + \sigma^2(-(X_j-M)\sigma^{-2}) = X_j - (X_j - M) = M. \tag{4.9}$$

To find the value of h_C corresponding with this value of a, we solve (4.7). In our case, $d = 1$, so after rewriting (4.7) to $h_C = \sqrt{a(n+2)}$, this solves to

$$h_C = \sqrt{a(n+2)} = \sqrt{3\sigma^2} = \sqrt{3}\sigma. \tag{4.10}$$

After we cluster the histogram, we will use the cluster centres to find the thresholds which will be used in the multilevel RATS algorithm.

4.4 Step edges, the continuous case

Without loss of generality, we will use a step edge with the edge at $x = 0$, and the value zero for $x < 0$, and one for $x > 0$. The optimum threshold is then the average of these two values, or $t = \frac{1}{2}$.

In the continuous case, this step edge is defined using the unit step, or Heaviside step function $H(x)$ (see Fig. 4.1(a)). This function is defined as:

$$H(x) = \begin{cases} 0 \text{ if } x < 0 \\ \frac{1}{2} \text{ if } x = 0 \\ 1 \text{ if } x > 0. \end{cases} \tag{4.11}$$

Algorithm 4.3 CLUSTER(H, n)

```
H_0 ← H
k ← 0
repeat
  k ← k + 1
  H_k[0...255] ← {0, 0, ..., 0}
  for i ← 0 to 255 do
    i' ← round(Σ_{j=i−h_C}^{i+h_C} j H_{k−1}[j] / Σ_{j=i−h_C}^{i+h_C} H_{k−1}[j])
    H_k[i'] ← H_{k−1}[i]
    if i = n then
      n ← i' {Keeping track of where n is shifted to, see Section 6.3}
    end if
  end for
until H_k = H_{k−1}
H ← H_k
return n
```

The first derivative of the Heaviside step function is the Dirac delta function:

$$\frac{\mathrm{d}}{\mathrm{d}x}H(x) = \delta(x). \tag{4.12}$$

The continuous weighted histogram is defined as

$$T(u) = \int_{-\infty}^{\infty} w(x)\delta(u - f(x))\mathrm{d}x. \tag{4.13}$$

where $w(x)$ is the weight function, and $f(x)$ is the image function. As we use the square of the gradient $g^2(x)$ as our weight function (4.1), this reduces $w(x)$ to the square of the first derivative of $f(x)$ in this case, $w(x) = (f(x)')^2$. We use (4.12) to find $w(x) = \delta^2(x) = \delta(x)$. Note that the square of the delta function is the delta function itself. Using $f(x) = H(x)$ as defined by (4.11), and filling this in in (4.13), yields:

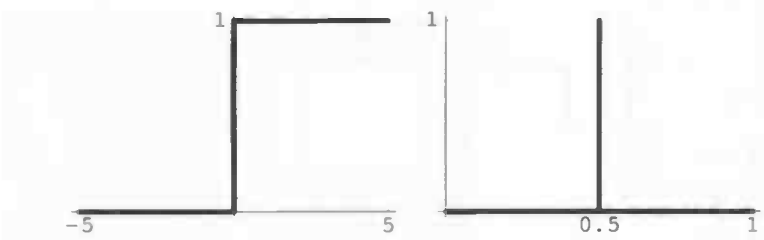
$$T(u) = \int_{-\infty}^{\infty} \delta(x)\delta(u - H(x))\mathrm{d}x. \tag{4.14}$$

The a product over two Dirac deltas in the integral is only non-zero if both parameters are zero. In this case, x should be zero, and then $H(x) = \frac{1}{2}$, thus, u should be $\frac{1}{2}$ to make the parameter of the second delta function zero as well. This means that $T(u)$ is zero everywhere except at $u = \frac{1}{2}$. In fact, because the integral of the Dirac delta is again the Heaviside step function, $T(\frac{1}{2}) = 1$ and thus $T(u) = \delta_{u\frac{1}{2}}$ (see Fig. 4.1(b), where $\delta_{u\frac{1}{2}}$ is the Kronecker delta:

$$\delta_{ij} = \begin{cases} 0 \text{ if } i \neq j \\ 1 \text{ if } i = j. \end{cases} \tag{4.15}$$

After the trivial clustering step we will find a single threshold $t = \frac{1}{2}$, which is indeed the theoretical optimum for the threshold.

While we have shown that a single step edge will be thresholded in an optimal way by using ML-RATS, this could already be achieved by using the



(a) Continuous step edge (b) Weighted histogram

Figure 4.1: The histogram of a continuous step edge

traditional binary RATS algorithm. The interesting feature of ML-RATS, after all, is that it does quantisation instead of binarisation. So, we will also show that ML-RATS correctly thresholds a succession of step edges.

We will again use the histogram function $T(u)$ (4.13), but we will use a generalised function $f(x)$ (see Fig. 4.2(a), where $N = 3$),

$$f(x) = \sum_{i=0}^{N-1} H(x - i). \tag{4.16}$$

The value of N is the number of steps in the function. We will again define the weight function $w(x)$ as the squared gradient of $f(x)$:

$$w(x) = (f(x)')^2 = \sum_{i=0}^{N-1} \delta(x - i). \tag{4.17}$$

We can derive the value of $T(u)$ for these functions in a similar way to the one-step case. Only when both $x - i$ and $u - f(x)$ are zero will we have a response in the histogram. As there are N possible values for i we will have N peaks in the histogram. As we can see in Fig. 4.2(b) for $N = 3$,

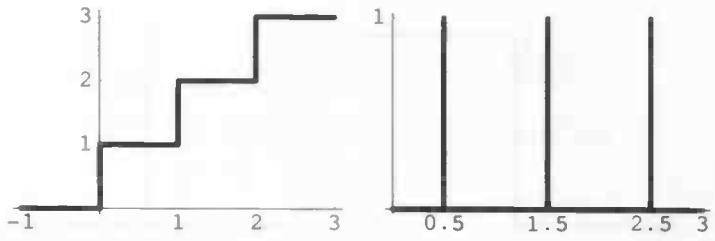
$$T(u) = \sum_{i=0}^{N-1} \delta_{u(i+\frac{1}{2})}. \tag{4.18}$$

The peaks of $T(u)$ correspond to the optimal values for the thresholds, as these optimal values are the average of the grey value before and after each step. In this case, these averages are $\frac{1}{2}, \frac{3}{2}, \ldots, N - \frac{1}{2}$, which are also the peaks of $T(u)$.

4.5 Step edges, the discrete case

If we apply the naive ML-RATS algorithm to a discrete step edge in an image, we will find two sharp peaks in the histogram, one for each side of the edge, because the halfway value $H(0) = \frac{1}{2}$ is not present in the image.

Discretely, the step function is only defined on the integer domain. To create a sharp edge, we will place the edge between 0 and 1, thus, we have the discrete step function



(a) Multiple step edges (b) Weighted histogram

Figure 4.2: The histogram of multiple step edges

$$H_d(x) = \begin{cases} 0 \text{ if } x \leq 0 \\ 1 \text{ if } x > 0. \end{cases} \tag{4.19}$$

We take the discrete derivative to be

$$f'_d(x) = f(x+1) - f(x-1), \tag{4.20}$$

Using (4.19) and (4.20), we find that

$$H'_d(x) = \begin{cases} 1 \text{ if } x = 0 \text{ or } x = 1 \\ 0 \text{ otherwise.} \end{cases} \tag{4.21}$$

Taking a discrete weighted histogram of this, this yields

$$T_d(u) = \begin{cases} 1 \text{ if } u = 0 \text{ or } u = 1 \\ 0 \text{ otherwise.} \end{cases} \tag{4.22}$$

This means that the algorithm will find two thresholds. If there is no noise present, this will be an acceptable result, however, if we consider a noisy step edge, the histogram will resemble two Gaussian distributions (see Fig. 4.3(f)).

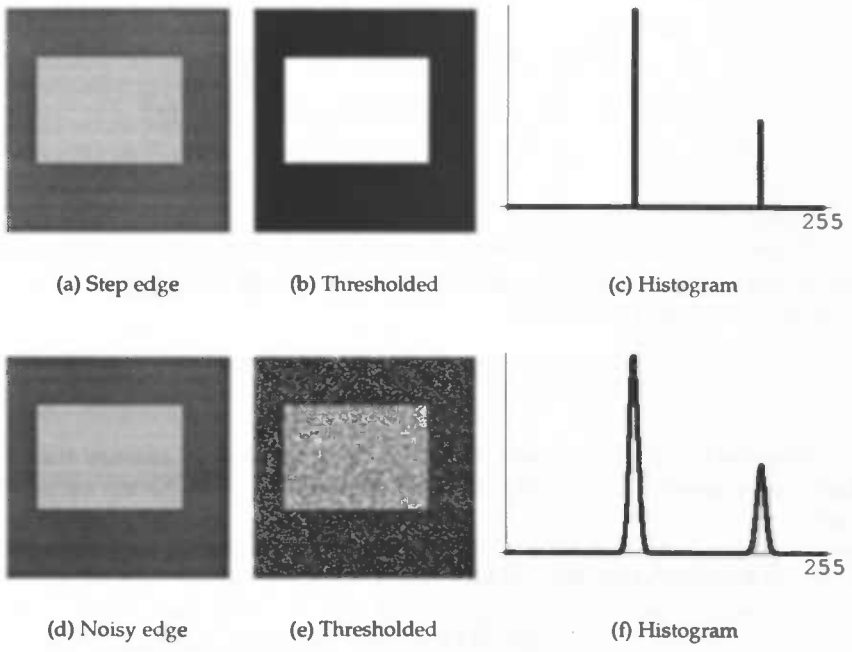
After clustering, two thresholds will be found at the centres of the two distributions. Thus, the thresholds found will be the same as if no noise was present. As seen in Fig. 4.3(e), this will yield unacceptable results, as part of the foreground and background are classified together as a third class.

As the thresholds are only based on grey values existing in the image, it is not possible to get better thresholds with this image. However, if we were to blur the image, using a Gaussian kernel, there would be additional grey values present in the image, which would lead to more thresholds. We would then expect a better separation of classes. Indeed, for the simple and noisy step edges, we can see in Fig. 4.4 that this is the case.

Note that the blurred images are only used to find the thresholds; the actual thresholding is still performed on the unblurred image.

4.6 Blurred step edges

While we have a visual indication that convolving the image with a Gaussian kernel will yield correct thresholds, we will need to prove this as well. We



(a) Step edge (b) Thresholded (c) Histogram

(d) Noisy edge (e) Thresholded (f) Histogram

Figure 4.3: Thresholding step edges using naive ML-RATS

(a) Step edge (b) Noisy edge

Figure 4.4: ML-RATS using Gaussian blur applied to Fig. 4.3(a) and Fig. 4.3(d)

will show that this operation can be performed, on the image and the weight function as well as only on the image.

Recall the definition of $T(u)$ (4.13)

$$T(u) = \int_{-\infty}^{\infty} w(x)\delta(u - f(x))\mathrm{d}x.$$

The function $f(x)$ was the Heaviside step function $H(x)$ (4.11). However, because we are blurring the step edge with a Gaussian kernel, we convolve this function with a Gaussian kernel (see Fig. 4.5(a)). We assume, without loss of generality, that $\sigma = 1$. We also need to calculate $w(x)$ (4.24), which is the squared gradient of $f(x)$.

$$f(x) = (H * G_\sigma)(x) = \frac{1 + \mathrm{erf}(x/\sqrt{2})}{2} \tag{4.23}$$

$$\begin{aligned} w(x) &= ((H * G_\sigma)(x)')^2 = (H' * G_\sigma)^2(x) \\ &= (\delta * G_\sigma)^2(x) = G_\sigma^2(x), \end{aligned} \tag{4.24}$$

where $\mathrm{erf}(z) = 2/\sqrt{\pi} \int_0^z e^{-t^2} \mathrm{d}t$.

Because $f(x)$ is a bijection with domain $\mathbb{R}$ and range $[0, 1]$, each point in the continuous histogram is due to exactly one point in $f(x)$. Specifically, in this case we have (see Fig. 4.5(b))

$$T(u) = w(f^{-1}(u)) = \frac{e^{-2(\mathrm{erf}^{-1}(2u-1))^2}}{2\pi}, \tag{4.25}$$

where erf^{-1} is the inverse error function. We will need to show that $T(u)$ is symmetric around $u = \frac{1}{2}$, because in that case $u = \frac{1}{2}$ will be the single cluster centre.

Proof. It is known that $\mathrm{erf}^{-1}(x)$ is an odd function,

$$\mathrm{erf}^{-1}(x) = -\mathrm{erf}^{-1}(-x).$$

Also, we know that the product of two odd functions is an even function,

$$(\mathrm{erf}^{-1}(x))^2 = (\mathrm{erf}^{-1}(-x))^2.$$

The composition of a monotonically increasing or decreasing function and an even function is an even function,

$$e^{-(\mathrm{erf}^{-1}(x))^2} = e^{-(\mathrm{erf}^{-1}(-x))^2}.$$

Finally, we multiply with some constants:

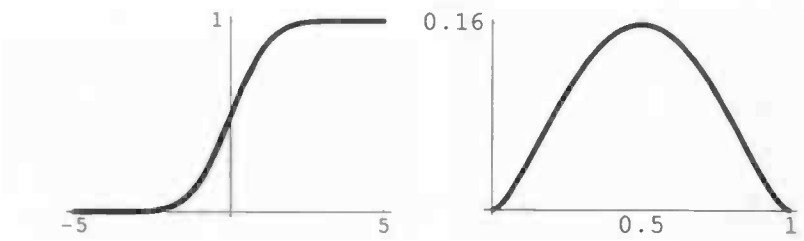
$$e^{-2(\mathrm{erf}^{-1}(x))^2}/2\pi = e^{-2(\mathrm{erf}^{-1}(-x))^2}/2\pi.$$

An even function is symmetric around $x = 0$. If we substitute $x = 2u - 1$, we move the symmetry axis to $u = \frac{1}{2}$:

$$e^{-2(\mathrm{erf}^{-1}(2u-1))^2}/2\pi = e^{-2(\mathrm{erf}^{-1}(-2u+1))^2}/2\pi. \tag{4.26}$$

This is the same as $T(2u - 1) = T(-2u + 1)$, which means $T(u)$ is symmetric around $u = \frac{1}{2}$. Thus, the cluster centre will be at $u = \frac{1}{2}$. □

Although we have shown here that this method guarantees the optimum threshold for a step edge, in practice this method may not be desirable. Blurring the weight function will lead to an increase in the effective radius of the weight function, and this may lead to the loss of small details. A better approach would be to use the blurred image for the grey values, yet use the original image to find the weight values. Specifically, this means $w(x) = \delta(x)$. Filling this, and $f(x)$ (4.23) in in $T(u)$ yields



(a) Blurred step edge (b) Weighted histogram

Figure 4.5: The histogram of a continuous blurred step edge

$$T(u) = \int_{-\infty}^{\infty} \delta(x)\delta(u - f(x))\mathrm{d}x. \tag{4.27}$$

Because $f(x)$ is a bijection, there will only be one value of $T(u)$ that is non-zero, specifically where $\delta(x)\delta(u - f(x))$ is non-zero. Thus, $x = 0$ to make the first factor non-zero, and $u - f(0) = 0$, or $u = f(0)$ to make the second factor non-zero. In this case, $f(0) = \frac{1}{2}$, thus $T(u) = \delta_{u\frac{1}{2}}$.

As we can see, this approach also gives the correct threshold, but the histogram is much sharper. In this case, small details will not be lost, so this approach is preferable to the previous approach.

We can extend this method to multiple step edges as well. Convolving the generalised step function $f(x)$ (4.16), which we will call $f_H(x)$ here, with a Gaussian kernel (we assume, again without loss of generality, that $\sigma = 0.1$) yields

$$f(x) = (f_H * G_\sigma)(x) = \sum_{i=0}^{N-1} \frac{1 + \mathrm{erf}((x - i)/(0.1\sqrt{2}))}{2}. \tag{4.28}$$

We will use the unblurred weight function, as this will yield better results than using the blurred weight function. The weight function is thus identical to (4.17).

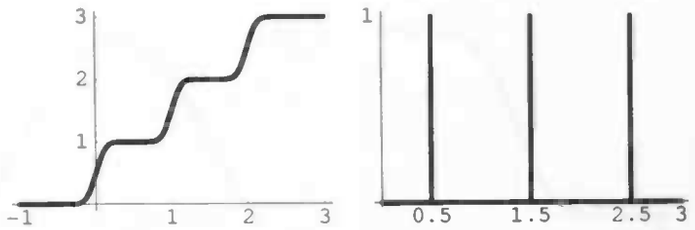
We can calculate $T(u)$ similarly to the single-step case. There are three peaks in the weight function, and each of these corresponds to a peak in the histogram. The result is the same as (4.18), namely

$$T(u) = \sum_{i=0}^{N-1} \delta_{u(i+\frac{1}{2})}.$$

This function yields N thresholds, which correspond to the optimum thresholds $t = i + \frac{1}{2}$ for i from 0 to $N - 1$.

4.7 Modified algorithm

As we have seen, using the grey values of a blurred version of the image when creating the histogram can improve the quality of the quantisation.



(a) Multiple blurred edges (b) Weighted histogram

Figure 4.6: The histogram of multiple blurred step edges

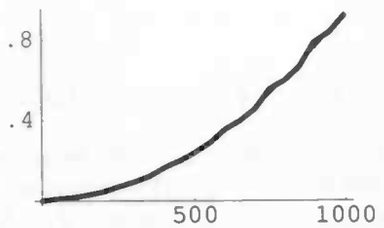


Figure 4.7: Timing graph for the ML-RATS algorithm, plotting the image dimension against the average time in seconds per calculation

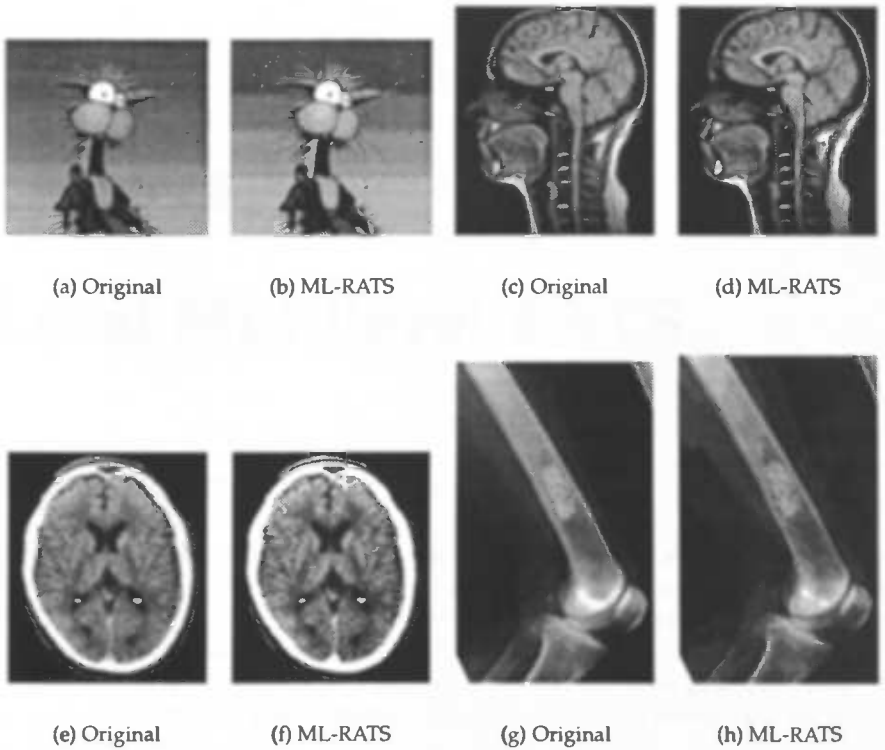
In the modified algorithm, Algorithm 4.4, we blur the image with a Gaussian kernel before creating the histogram. However, the weight function is not blurred, and the test to determine whether a pixel is noise or not is also performed using the non-blurred image.

Algorithm 4.4 HISTOGRAM(f, w)

$f_G = \text{GAUSSIANBLUR}(f, \sigma)$
$H[0 \ldots 255] \leftarrow \{0, 0, \ldots, 0\}$
for all image pixels $f_G(x, y)$ **do**
  if $f(x, y)$ is not a noise pixel **then** {Note f, not f_G; see Chapter 6}
    $H[f_G(x, y)] \leftarrow H[f(x, y)] + w(x, y)$ {w is not blurred}
  end if
end for
return H

4.8 Results

We have performed some timing experiments on our implementation of the ML-RATS algorithm, using ten real images and scaling them to squared images sized from 50×50 to 1000×1000. We ran the ML-RATS algorithm using each of these images, and took the average running time. The results are plotted in Fig. 4.7. The tests were run on a system with an Intel Pentium 4 2.8 GHz



(a) Original (b) ML-RATS (c) Original (d) ML-RATS

(e) Original (f) ML-RATS (g) Original (h) ML-RATS

Figure 4.8: Quantisation using Multilevel RATS

processor with 512 Mb RAM.

We performed a quadratic fit on the results and plotted this as a thin line in the same figure. This line is only visible at the two points where the actual graph is slightly discontinuous, so the fit is fairly accurate. This is as we expected, as the algorithm has a complexity which is linear in the number of pixels in the image, thus quadratic in the image dimension given our square images.

In Fig. 4.8, we show a number of results, created by our algorithm. The number of colours in the images are reduced, but not as drastically as with generalised Otsu's algorithm. Again, as with Otsu's algorithm, segmentation is not optimal, but, because we retain more classes, the objects are more clearly segmented is than with Otsu's algorithm, though there is a significant amount of oversegmentation. Part of the problem is that most objects are covered in a gradient, and as a global algorithm, ML-RATS is not able to cope with gradients. As we will see, a local approach will yield significantly better results.

Chapter 5

Local Multilevel RATS

5.1 Introduction

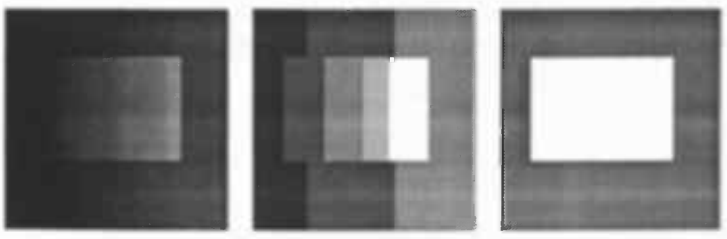
As with binarisation, it is often the case that global quantisation does not yield desirable results. Figure 5.1(a), for instance, has a strong gradient running through the image, and these are in practical applications usually due to shadows, slanted camera positions and such.

Because of this, it is usually desirable to remove such gradients from the quantised image, and global quantisation is incapable of doing this, while local quantisation can do this.

In this chapter, we will first consider an important issue in extending multilevel RATS to local quantisation, how to match the thresholds for different pixels to each other, as there may be a different number of thresholds. We will then briefly consider window sizes and shapes, and we will finally give the algorithm for the local approach.

5.2 Matching thresholds

Local quantisation suffers from the problem that, if the number of classes is unknown, as it is in ML-RATS, finding matches between classifications in ad-



(a) Gradient image (b) ML-RATS (c) Local ML-RATS

Figure 5.1: Local ML-RATS

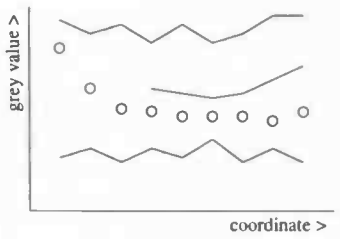


Figure 5.2: Threshold matching in local quantisation

jacent pixels is non-trivial. For instance, in Fig. 5.2, it is not directly obvious whether the pixels in the area with two thresholds should be classified the same as those in the area with three thresholds, or differently.

The method we will use to solve this problem is to try to match neighbouring pixels by applying the thresholds found for one pixel to the pixel value of the pixel and its neighbour. If these are both in the same class, and vice versa with the thresholds of the neighbour pixel, the two pixels are considered to be in the same class. Specifically, if

$$\begin{aligned}(\exists i:\ T_a[i] < f_a \leq T_a[i+1] \ \wedge\ T_a[i] < f_b \leq T_b[i+1]) \quad \wedge \\ (\exists j:\ T_b[j] < f_a \leq T_b[j+1] \ \wedge\ T_b[j] < f_b \leq T_b[j+1])\end{aligned} \tag{5.1}$$

holds, there will be a link between a and b. In this formula, f_a and f_b are the grey values of a and b, and T_a and T_b are the sets of thresholds corresponding to a and b respectively. $T[-1]$ is considered to be $-\infty$, and $T[\#T]$ is likewise considered to be ∞.

For instance, in Fig. 5.2, the third pixel would be classified in the same class as the fourth pixel, if we use the thresholds corresponding to the fourth pixel, and vice versa. Thus, these pixels belong to the same class. This obviously holds for all other adjacent pairs of pixels, so all pixels belong to a single class in this case.

We repeat this procedure for all adjacent pairs of pixels, using four-connectivity. This creates a map consisting of links between pixels. To find the classes, we perform a flood-fill operation on each area. Alternatively, a union-find algorithm [9, 6] could be used, which performs the same function and is faster. However, because the operation is performed only once during the execution of the algorithm, the speed-up is negligible. This will yield an image where the different classes each have a different colour. This algorithm is summarised in Algorithm 5.1. $T_{(x,y)}$ denotes all local sets of thresholds; the set of thresholds for a point (a, b) is denoted as $T_{(a,b)}$.

5.3 Window size and shape

An important question in local quantisation, as in local binarisation, is determining the size and shape of the window around each pixel. In general, smaller windows compute faster, but are more sensitive to noise. The idea is then to use the smallest window size which still gives acceptable results. In Section 7.3, we will examine this problem in more detail.

Algorithm 5.1 LINK$(f, T_{(x,y)})$

```
L ← ∅
for all adjacent pairs of pixels f(x1, y1) and f(x2, y2) do
  if f(x1, y1) and f(x2, y2) match then {See (5.1)}
    L ← L ∪ (x1, y1) ↔ (x2, y2)
  end if
end for
f ← {−1, −1, ... − 1} {All pixels are unclassified}
color ← 0
for all pixel f(x, y) do
  if f(x, y) = −1 then
    FLOODFILL(f, x, y, L, color) {Use links for flood-filling}
    color ← color + 1
  end if
end for
```

A related problem is the choice of window shape. Given a point (x_p, y_p) and a window size h_w, there are several methods to determine whether a point (x, y) is included in the window around (x_p, y_p). Alternatives include square windows, where

$$\max(|x_p - x|, |y_p - y|) \leq h_w, \tag{5.2}$$

circular windows, where the Euclidean distance

$$\sqrt{(x_p - x)^2 + (y_p - y)^2} \leq h_w, \tag{5.3}$$

diamond-shaped windows, where the Manhattan distance

$$x_p - x + y_p - y \leq h_w, \tag{5.4}$$

and Gaussian-weighted windows, where a pixel isn't so much part of the window or not, but rather contributes to the window weighted to the value of the Gaussian function centred around (x_p, y_p) with standard deviation h_w. In Section 7.3, we will compare two of these alternatives, the square window and the Gaussian-weighted window.

To implement local windows in our algorithm, we need to adapt the histogram method, Algorithm 4.4, as only pixels inside the window should be taken into account. Because the local histogram is centred around a single pixel, we will need to add the pixel's coordinates, (x_p, y_p), to the parameter list of the function. We will also need to add W, the window function, which returns 1 if the pixel (relative to the centre of the window) is inside the window, and 0 otherwise. Weighted windows, such as the Gaussian window, can also return values between 0 and 1 to represent the pixel's weight.

5.4 Algorithm

As shown in Algorithm 5.3, for each pixel we create a histogram of the surrounding area. These histograms are clustered, and the linking algorithm, Al-

Algorithm 5.2 LOCALHISTOGRAM(f, w, x_p, y_p, W)

$f_G = \text{GAUSSIANBLUR}(f, \sigma)$
$H[0 \ldots 255] \leftarrow \{0, 0, \ldots, 0\}$
for all image pixels $f_G(x, y)$ **do**
  if $f(x, y)$ is not a noise pixel **then** {Note f, not f_G; see Chapter 6}
    $H[f_G(x, y)] \leftarrow H[f(x, y)] + w(x, y)W(x - x_p, y - y_p)$ {w is not blurred}
  end if
end for
return H

gorithm 5.1, is used to create the segmented image.

Algorithm 5.3 LOCALMULTILEVELRATS

for all image pixels $f(x, y)$ **do**
  $H_{(x,y)} \leftarrow \text{LOCALHISTOGRAM}(f, w, x, y, W)$ {W is the window function}
  $\text{CLUSTER}(H_{(x,y)})$
  $T_{(x,y)} \leftarrow \text{CLUSTERCENTRES}(H_{(x,y)})$
end for
$\text{LINK}(f, T_{(x,y)})$ {$T_{(x,y)}$ denotes all sets of thresholds}

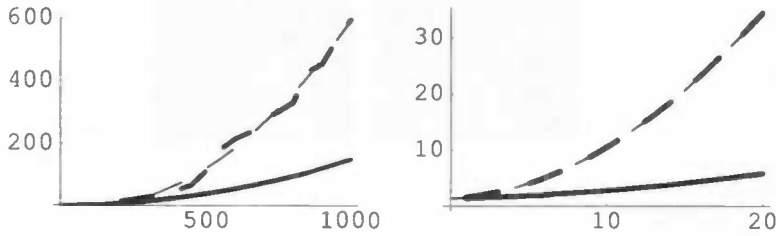
5.5 Results

We also performed timing experiments on the local version of our algorithm, using the same set of ten images we used to time the ML-RATS algorithm in Section 4.8. In Fig. 5.3(a), we plotted the image dimension against the time, using squared images scaled from 50×50 to 1000×1000 and square and Gaussian windows of size 5. In Fig. 5.3(b), we plotted the window size against the time, using windows sized from 1 to 20 and square images of 100×100. These tests were run on a system with an Intel Pentium 4 2.8 GHz processor with 512 Mb RAM.

We fitted quadratic functions to the results, and plotted these as the thin lines in the figures. The lines are a very good fit for the data, and we can see that the algorithm is linear both in the number of pixels in the image, as well as the number of pixels in the window, as we expect it to be. Because the images were square, the algorithm is quadratic in the image dimension, and similarly, because the number of pixels in the window is quadratically related to the window size, the algorithm is also quadratic in the window size.

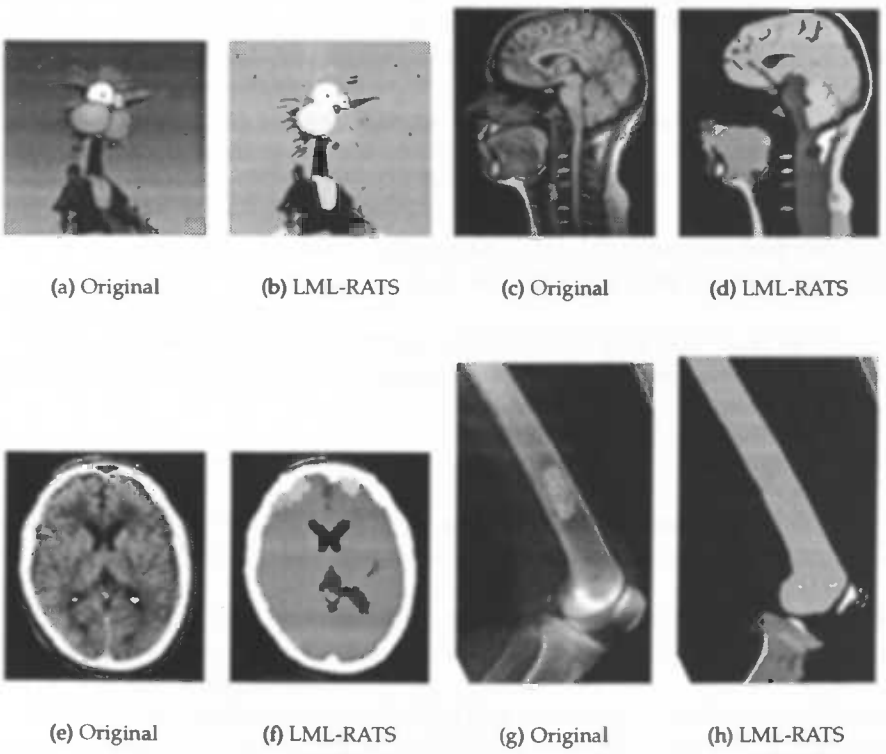
In Fig. 5.4, we show the same images as in Fig. 4.8, but this time we segmented them using the local algorithm, using Gaussian windows with $\sigma = 5$. As opposed to the global quantisation algorithms in Chapters 3 and 4, the results of this algorithm do provide a fairly accurate segmentation of the image. Also, the segmentations provided by this algorithm could be used as a starting guess for algorithms which compute more precise segmentations but which need a good starting guess.

An interesting effect can be seen in the MRI scan, Fig. 5.4(c), where the nose is not shown in the segmented image. This is due to the fact that the nose



(a) Image dimension (b) Window size

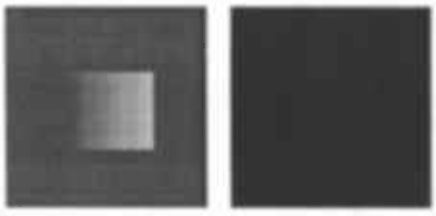
Figure 5.3: Timing graphs for the Local ML-RATS algorithm, plotting the image dimension and window size against the average time in seconds per calculation. The solid line denotes square windows, and the dashed line denotes Gaussian windows



(a) Original (b) LML-RATS (c) Original (d) LML-RATS

(e) Original (f) LML-RATS (g) Original (h) LML-RATS

Figure 5.4: Segmentation using Local Multilevel RATS



(a) Original (b) LML-RATS

Figure 5.5: Segmenting a gradient blending into the background

blends in the background via a gradient, and the algorithm thus considers it to be a single class. This can be clearly seen in Fig. 5.5, where we can see that the gradient blends into the background, and is considered to belong to the same class.

Chapter 6

Noise reduction techniques

6.1 Introduction

While simple step edges may be noise free, real images suffer from at least some amount of noise and blur. This is fundamental to light, due to its quantum nature.

Noise causes problems in binarisation and quantisation algorithms, especially in the case of RATS as it will not only offset the grey values of the images, but it will also introduce "edges" when two adjacent pixels have a different amount of noise added. It is therefore desirable to reduce the amount of noise in the image by some method.

In this chapter, we will show three methods for reducing noise. The first method, $\lambda\sigma$ thresholding [4], reduces the noise while creating the histogram, or calculating the weighted average in the original RATS algorithm, by only using pixels whose weight is above a certain threshold.

Another approach, the mean shift method, reduces noise as a preprocessing step by setting the value of a pixel to its average amongst pixels in the same class in its neighbourhood.

The third method we will discuss reduces noise in a post-processing step, by removing classes in the image whose area is smaller than a certain threshold.

6.2 $\lambda\sigma$ method

In real-life images, there is always an amount of noise available. If we know the distribution of the noise, it is generally relatively easy to reduce the noise in the image.

However, in realistic applications, the distribution of the noise is usually unknown. We will have to estimate the distribution of the noise and the parameters of this distribution.

We will assume that the noise is due to a Gaussian distribution, with zero mean. This leaves us with only the standard deviation to estimate. Kittler *et al.* devised a formula to estimate the amount of noise present [4]:

$$\hat{\sigma} = \frac{\sqrt{2\pi}}{4n} \sum_x \sum_y e(x, y) \tag{6.1}$$

where $e(x, y)$ is the edge magnitude $e(x, y) = \max(|e_x|, |e_y|)$, with $e_x = f(x+1, y) - f(x-1, y)$ and e_y chosen similarly. Also, n is the number of pixels in the image.

Kittler *et al.* also describe a method to reduce noise, by ignoring pixels whose edge response falls below a certain threshold, for instance, to eliminate 99.99% of non-edge pixels this threshold should be 5.75σ, where σ is the standard deviation of the image noise. The constant 5.75 can be varied, leading to the general threshold $\lambda\sigma$.

Once we have this estimate of σ, we will ignore all pixels with a weight value of less than a certain threshold in our histogram calculations, effectively setting those weight values to zero.

To eliminate 99.99% of non-edge pixels this threshold should be 5.75σ, where σ is the standard deviation of the image noise. The constant 5.75 can be varied, leading to the general threshold $\lambda\sigma$.

Wilkinson [10] shows that $\lambda = 5$ gives near optimal results for first-derivative filters, and $\lambda = 3$ for first-derivative Sobel filters.

Note that the above method is valid only for edge filters which are linear in the gradient of the image. Higher-power filters, such as the squared gradient filter g^2, will need to use a corresponding threshold. In the case of the squared gradient filter, for instance, we will also need to square the noise threshold, which means we will need to use $\lambda^2\sigma^2$ as the noise threshold.

6.3 Mean shift method

A well-known method to smooth an image is to convolve the image with a Gaussian kernel. This method also reduces the variance of noise present, however, it has the undesirable side effect of also blurring the edges in the image. This makes the method unsuitable for noise reduction.

However, we can use the observation that convolving with an averaging kernel reduces the noise variance to find a method to reduce noise without blurring edges. First note that we could very well use a square window kernel to blur the image, instead of a Gaussian kernel.

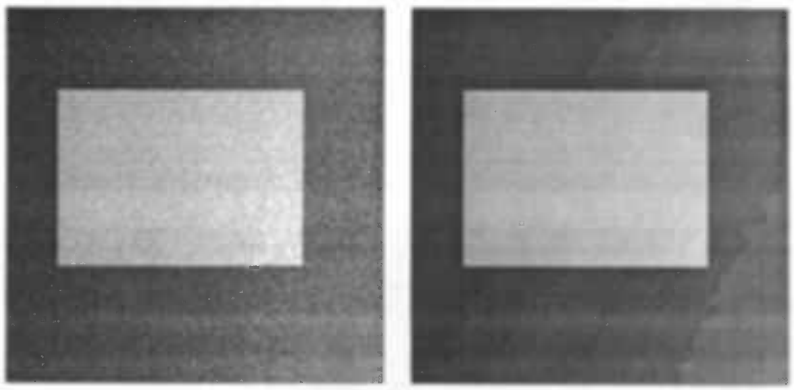
What we would like is to know which pixels in the window are similar to the pixel we are examining, as we can calculate the new grey value of this pixel by averaging over all similar pixels in the window. Essentially, we would like to cluster the pixels in the window, and use only those pixels who are in the same cluster.

The approach we will use is to create a histogram of the local window, and apply the clustering technique described in Section 4.3. We will keep track of where the grey value of the current pixel is shifted to, as it will be shifted towards its cluster centre. We will then set the pixel to the final value it was shifted towards. This is summarised in Algorithm 6.1

Some results are shown in Fig. 6.1. The noise is almost completely removed in the second image. However, the gradient is still present, so we will still need to use a segmentation algorithm to segment the image.

Algorithm 6.1 MEANSHIFT(f, h_w)

for all image pixels $f(x, y)$ **do**
  $H[0 \ldots 255] \leftarrow \{0, 0, \ldots, 0\}$
  for all image pixels $f(x', y')$ with $\max(|x' - x|, |y' - y|) \leq h_w$ **do**
    $H[f(x', y')] \leftarrow H[f(x', y')] + 1$
  end for
  $f(x, y) \leftarrow$ CLUSTER$(H, f(x, y))$
end for



(a) Original image (b) Mean shift

Figure 6.1: The mean shift method with $h_w = 5$

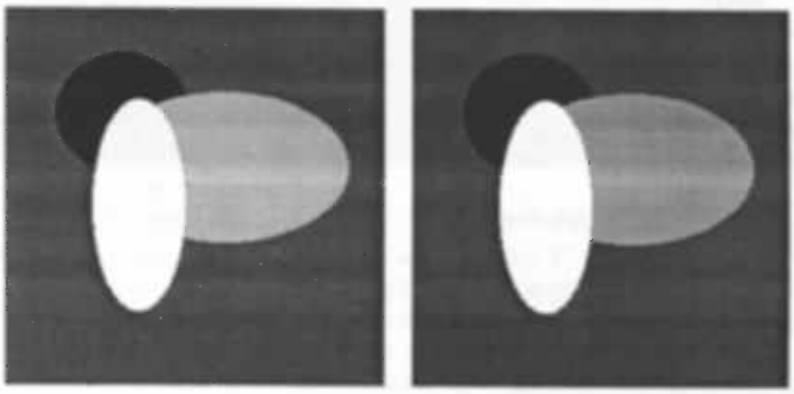
6.4 Area filtering

The local multilevel RATS algorithm often generates a number of small classes which are due to noise, in Fig. 6.2(a) for instance. It may be desirable to remove these artifacts, and this can be achieved by performing a post-processing step which removes all classes smaller than a certain threshold. This is a fairly rigorous method, and it should be used with caution. However, if it is known that the image does not contain small classes, it can be used safely.

Initially, the size of each class is determined. Then, classes whose size is below the threshold a are assigned to neighbouring classes on a pixel for pixel basis, working from the outside of the class inwards. The assignments can for example be done based on how closely the average colour matches that of the neighbouring class, or each pixel can be assigned to the largest neighbouring class. In our implementation, we have chosen the first approach. The algorithm is summarised in Algorithm 6.2.

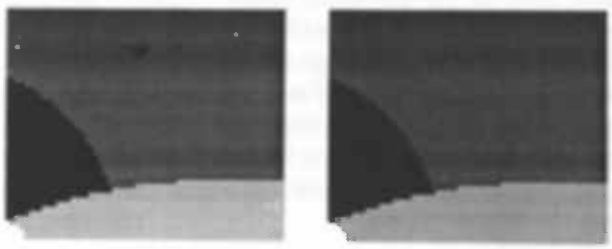
Algorithm 6.2 AREAFILTER(f, a)

for all image pixels $f(x, y)$ **do**
  $c(x, y) \leftarrow$ area of class $f(x, y)$ belongs to
end for
while $\min(c(x, y)) < a$ **do**
  for all image pixels $f(x, y)$ with $c(x, y) < a$ **do**
    if $f(x, y)$ has at least one neighbour $f(x', y')$ with $c(x', y') \geq a$ **then**
      $f'(x, y) \leftarrow$ best matching neighbour $f(x_N, y_N)$ with $c(x_N, y_N) \geq a$
      $c'(x, y) \leftarrow c(x_N, y_N)$
    end if
  end for
  $f \leftarrow f'$
  $c \leftarrow c'$
end while



(a) Unfiltered (b) $a = 15$

Figure 6.2: The area filtering method



(a) Unfiltered (b) $a = 15$

Figure 6.3: Small sections of the images in Fig. 6.2, showing some of the small classes which are filtered out

Chapter 7

Experiments

7.1 Introduction

In this chapter, we will experiment with the local multilevel RATS algorithm described in Chapter 5. We will first perform experiments on phantom images, images with known amounts of noise and gradient, and a known reference segmentation, to find good values for the algorithm parameters.

After this, we will experiment using real images, using the parameters we found while experimenting with phantom images.

7.2 Phantom images

First, we will perform experiments using phantom images. We will modify a baseline image in some way, by adding noise and gradients for instance, and then use our algorithm to reconstruct the image. By comparing this reconstruction to the original image, we can see how well the algorithm copes with degraded images.

To compare the two images, we will use the segmentation measure due to Levine and Nazif [5]. Given a test segmentation with M regions $\{\tau_1, \tau_2, \ldots, \tau_M\}$ having areas $\{\alpha_1, \alpha_2, \ldots, \alpha_M\}$, and a reference segmentation with N regions $\{R_1, R_2, \ldots, R_N\}$ having areas $\{A_1, A_2, \ldots, A_N\}$, Levine and Nazif define two measures. The first is the under-merging error measure

$$UM = \sum_{j=1}^{M} \frac{(A_K - \cap[\tau_j, R_K]) \cap [\tau_j, R_K]}{A_K}, \tag{7.1}$$

where $\cap[\tau_j, R_K]$ is defined as the intersection of regions τ_j and R_K, that is, the area both regions share, and R_K is a region such that

$$\cap[\tau_j, R_K] = \max_{1 \le i \le N} \cap[\tau_j, R_i]. \tag{7.2}$$

The second measure is the over-merging error measure

$$OM = \sum_{j=1}^{M} (\alpha_j - \cap[\tau_j, R_K]). \tag{7.3}$$

These two measures can then be combined, for instance by considering them distances along orthogonal axes, normalising them, and taking the Euclidean length as the combined measure:

$$M = \sqrt{(UM/A_I)^2 + (OM/A_I)^2}, \tag{7.4}$$

where A_I is the total area of the image.

Using this measure, we will try to find correlations between image parameters, such as noise and gradients, and algorithm parameters, such as window size.

7.3 Window size and shape

As mentioned in Section 5.3, there are many possibilities for the window shape used in the local multilevel RATS algorithm. In this section, we will look at two different window shapes, and try to find a relationship between the noise and gradient in the image, and the optimal window size.

Data was accumulated by running the algorithm using specific phantom images for noise values of $\sigma = 1$ to $\sigma = 20$, and using window sizes ranging from 1 to 20, with both square and Gaussian windows. A square window with window size h is a $2h + 1$ by $2h + 1$ square window, and in a Gaussian window, the size is equal to the standard deviation. All other parameters in the algorithm were constant.

We also used noiseless images with gradients running diagonally over the image, with a difference in intensity ranging from 10 (gradient strength 1) to 200 (gradient strength 20), and tested this with the same range of window sizes.

The resulting images were compared to the original using the segmentation measure given in (7.4), and the results are plotted in Figures 7.1 and 7.2. Contrary to our expectations, window size has no consistent impact on the quality of the segmentation, and neither has window shape. As square windows are faster to compute, these will be used, and we will use window size 8, which seems to give good enough results.

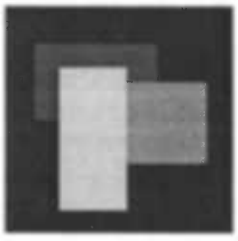
7.4 Gaussian blurring

We also performed some experiments to find a good value to use for the Gaussian blurring step in the algorithm. Again, we used images with a range of noise values and gradients, as in the previous section, and we plotted this against Gaussian blur standard deviations running from $\sigma = 1$ to $\sigma = 20$.

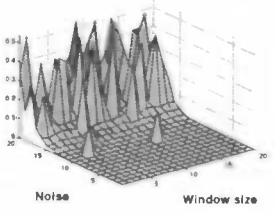
As we can see in Fig. 7.3, $\sigma = 2$ seems to be the optimum value, as the results don't improve for larger values of σ, and $\sigma = 1$ shows a significant decrease in performance.

7.5 Mean shift filtering

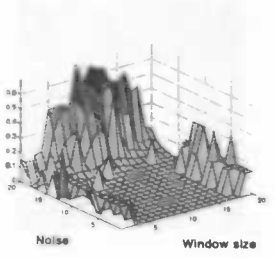
To find the best values for mean shift noise filtering, we added different amounts of noise to our test images, and tried to remove this noise with different win-



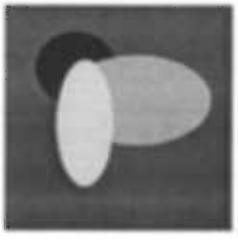
(a) Test image



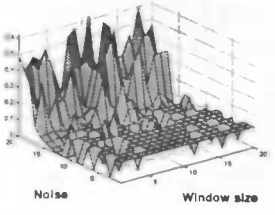
(b) Square windows



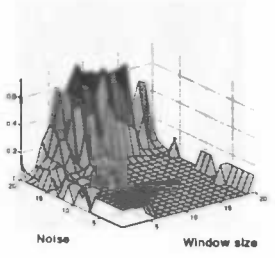
(c) Gaussian windows



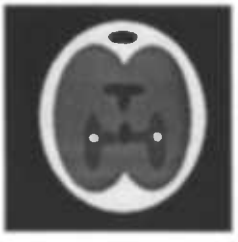
(d) Test image



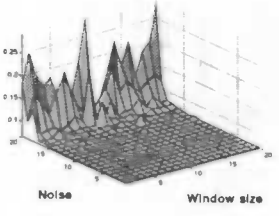
(e) Square windows



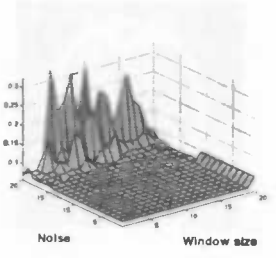
(f) Gaussian windows



(g) Test image

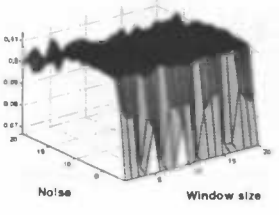


(h) Square windows

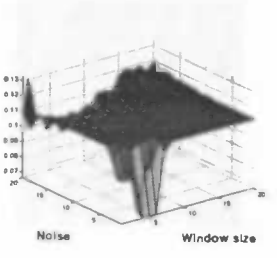


(i) Gaussian windows

(j) Test image

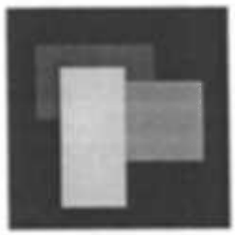


(k) Square windows

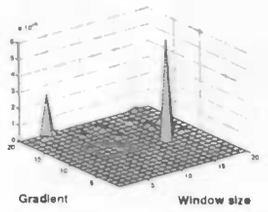


(l) Gaussian windows

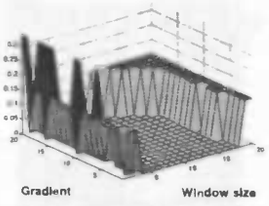
Figure 7.1: Segmentation measure plots of noise versus window size



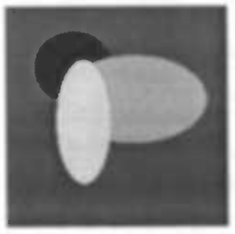
(a) Test image



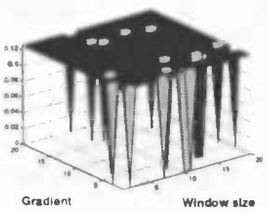
(b) Square windows



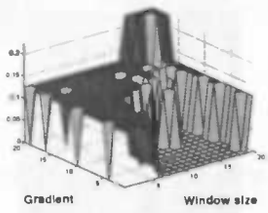
(c) Gaussian windows



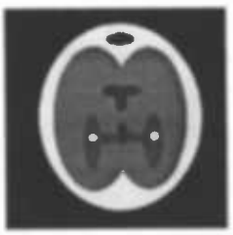
(d) Test image



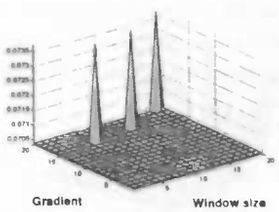
(e) Square windows



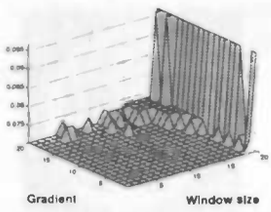
(f) Gaussian windows



(g) Test image

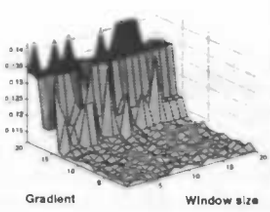


(h) Square windows

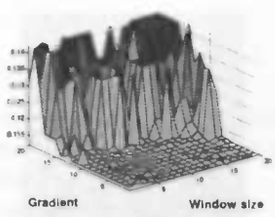


(i) Gaussian windows

(j) Test image

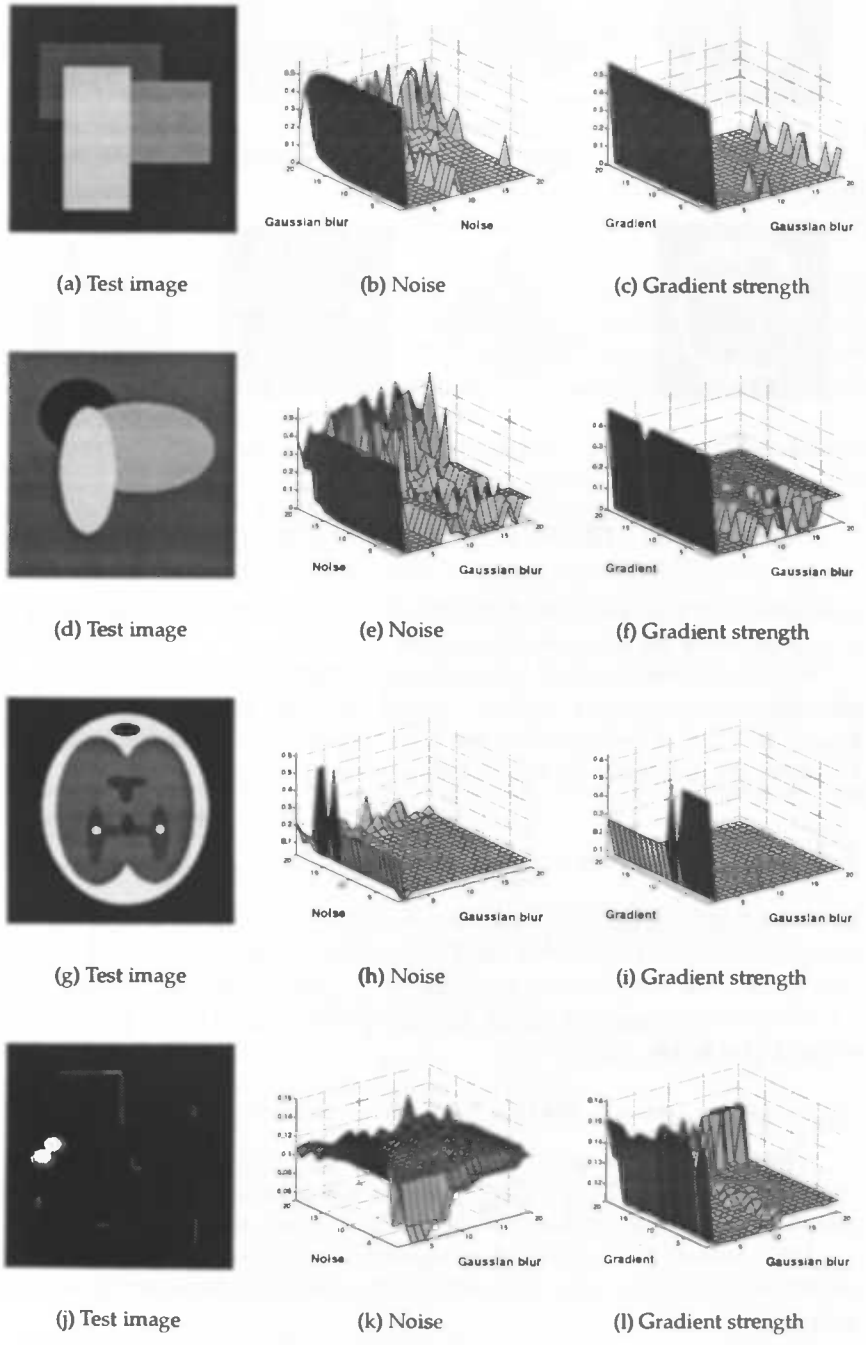


(k) Square windows



(l) Gaussian windows

Figure 7.2: Segmentation measure plots of gradient strength versus window size



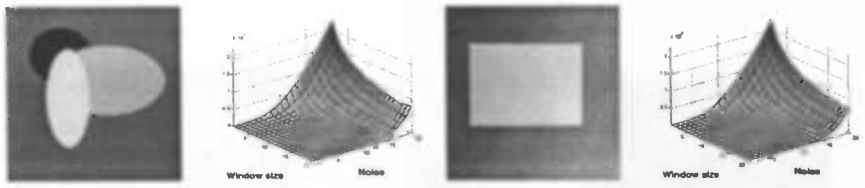
(a) Test image (b) Noise (c) Gradient strength

(d) Test image (e) Noise (f) Gradient strength

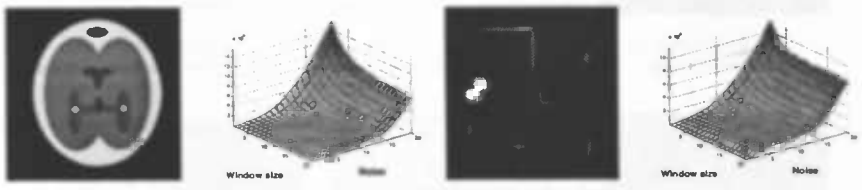
(g) Test image (h) Noise (i) Gradient strength

(j) Test image (k) Noise (l) Gradient strength

Figure 7.3: Segmentation measure plots of noise and gradient versus Gaussian blur



(a) Test image (b) Plot (c) Test image (d) Plot



(e) Test image (f) Plot (g) Test image (h) Plot

Figure 7.4: Squared error plots of noise versus mean shift window size

dow sizes. We then calculated the squared sum of errors over the entire image as a measure of the amount of noise left.

We found that the best value for the window size h_{MS} depends on the amount of noise present, and as a rough estimate, we found that the relation $h_{MS} = 0.075\eta^2$, where η is the standard deviation of the noise, is the smallest value for the window size which still gives good results.

7.6 Parameters

In the previous sections, we have shown some experiments to find correlations between noise and gradients, and algorithm parameters. In this section, we will summarise the parameter values we will use to test real images.

We will use square windows, where a point (x, y) is in the window centred around (x_P, y_P) if

$$\max(|x - x_P|, |y - y_P|) \leq h_W,$$

where $h_W = 8$. We have found no consistent correlation between noise and window size, so we have chosen a somewhat arbitrary window size. We will use $\sigma_{\text{blur}} = 2$ as the standard deviation for the Gaussian blurring step,

The clustering algorithm also requires a window size parameter, h_C. As shown in (4.10), this value depends on the standard deviation σ of the noise, where $h_C = \sqrt{3}\sigma$.

We will experiment both with $\lambda\sigma$ filtering and with mean shift filtering. We will use $\lambda = 3$, which was shown by Wilkinson [10] to be a good value for λ. For mean shift filtering, we will use $h_{MS} = 0.075\eta^2$.

Due to time constraints, we have not performed experiments to find the best value for the minimum area in the area filtering algorithm, but as an educated

guess, we will use 10 for this step, based on our preliminary results in Section 5.5.

7.7 Real images

Using the parameters we found in our experiments with phantom images, we applied the LML-RATS algorithm to a number of real-life images.

We performed two sets of experiments, one with $\lambda\sigma$ noise reduction, and one with mean shift noise reduction. The results are shown in Figures 7.5, 7.6 and 7.7.

The segmentation of some images is somewhat dissapointing. Especially those with a high calculated noise level suffer from a relatively poor segmentation. This is likely due to estimating the amount of noise too high, as Kittler's formula assumes textureless images, and textures, which are often present in real images, are also considered to be noise.

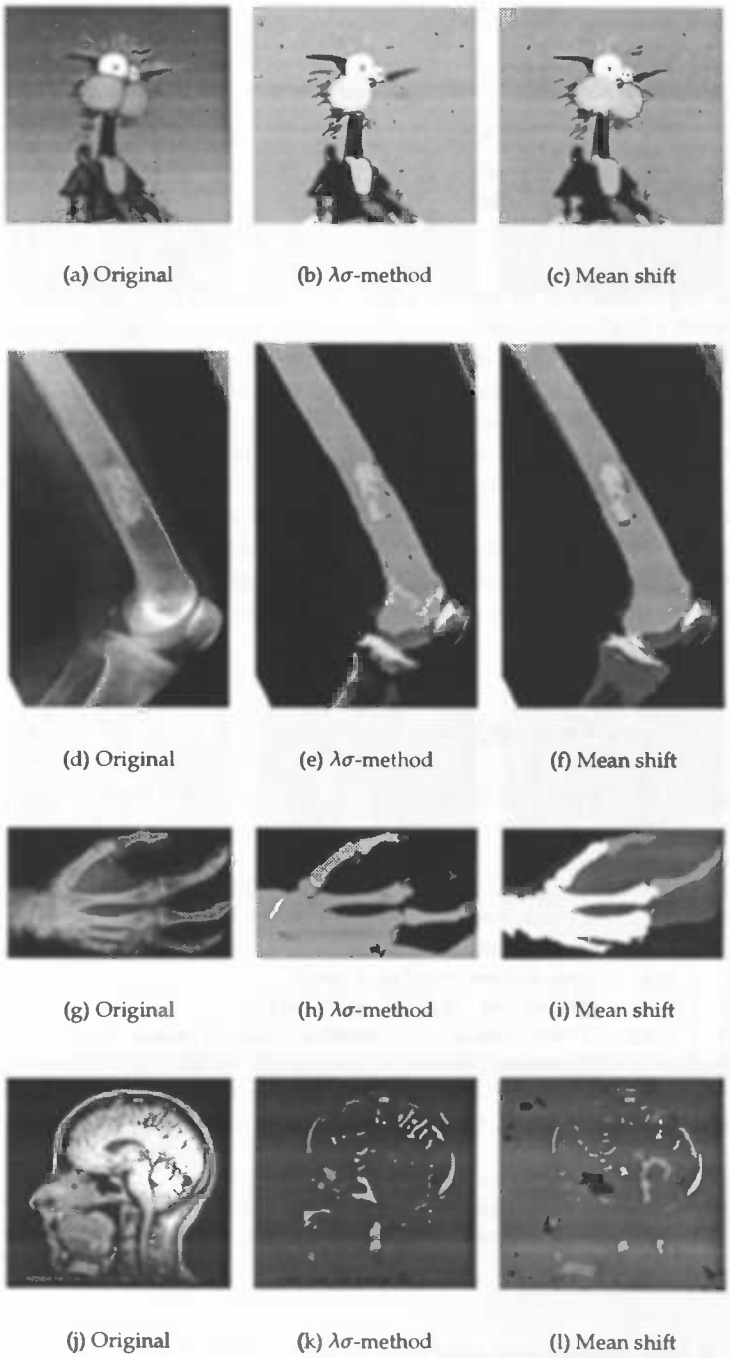
Also, we can see that some of the images are segmented better using the parameters we used in Fig. 5.4, which were only educated guesses. Clearly, the parameter selection we made in the previous section does not work for all pictures, so individual changes to the parameters will be needed.

Finally, we can see that, while there is often some difference between the $\lambda\sigma$-method and the mean shift method, neither of them is consistently better than the other.

7.8 Image references

The images in Figures 7.5, 7.6 and 7.7 were taken from the following sources:

Figure	Reference
7.5(a)	http://johnnyvw.home.mindspring.com/
7.5(d)	http://web.kanazawa-u.ac.jp/~med23/NMC/PrepCase0211.html
7.5(g)	http://www.epilepsy.org.uk/news/archive/2002/20020527.html
7.5(j)	http://www.midwest-medical.net/
7.6(a)	http://focus.aps.org/story/v4/st13
7.6(d)	http://freud.tau.ac.il/~shakhar/neuro/images.html
7.6(g)	http://yourmedicalsource.com/library/ctscan/CTS_whatis.html
7.6(j)	http://www.cnsv.net/cnsv/diagnosis.html
7.7(a)	http://www.ddc.musc.edu/ddc_pro/pro_development/case_studies/case011.htm
7.7(d)	http://www.cairoscan.com.eg/issue8_1.htm
7.7(g)	http://www.lowback-pain.com/interesting%20cases.htm
7.7(j)	http://www.bintel.com.au/Solar%20Spots%202.html



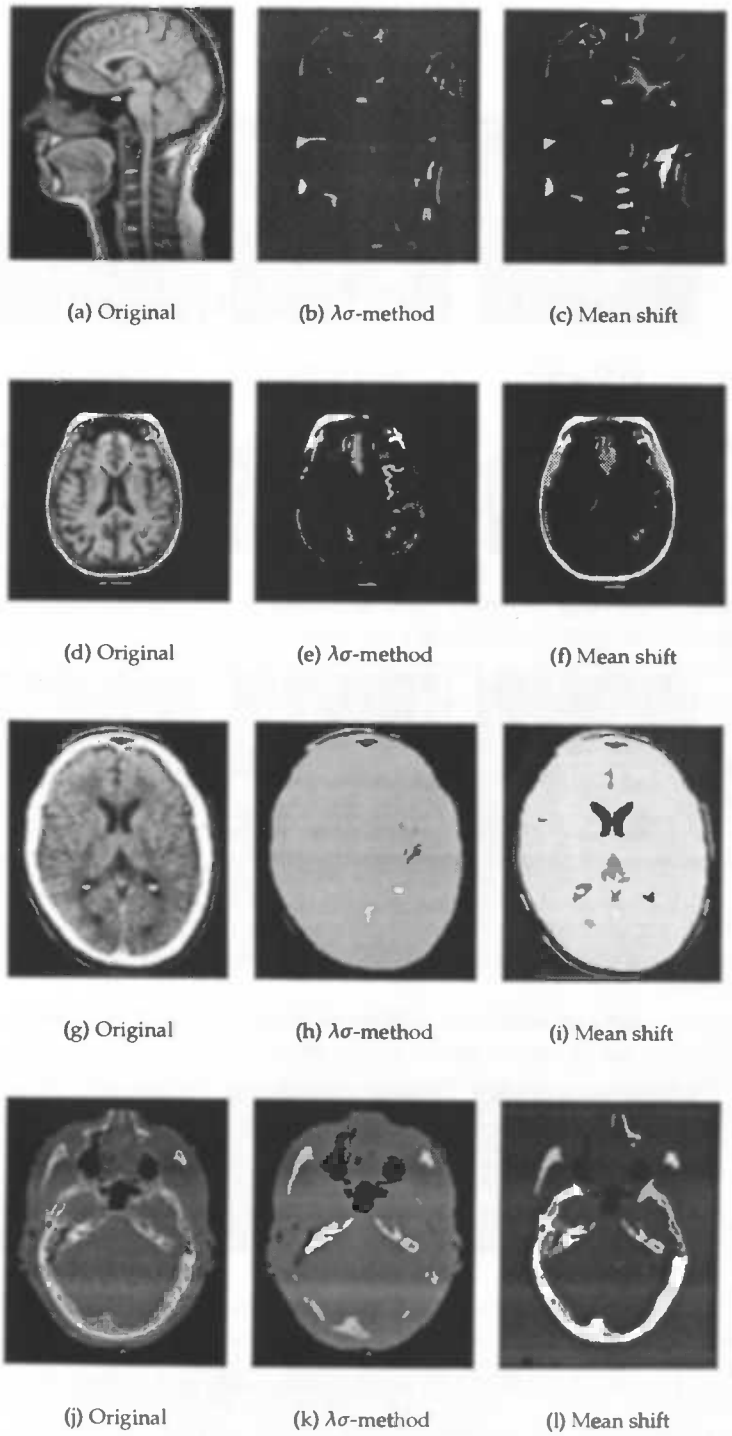
(a) Original (b) $\lambda\sigma$-method (c) Mean shift

(d) Original (e) $\lambda\sigma$-method (f) Mean shift

(g) Original (h) $\lambda\sigma$-method (i) Mean shift

(j) Original (k) $\lambda\sigma$-method (l) Mean shift

Figure 7.5: LML-RATS experiments with real images, part 1



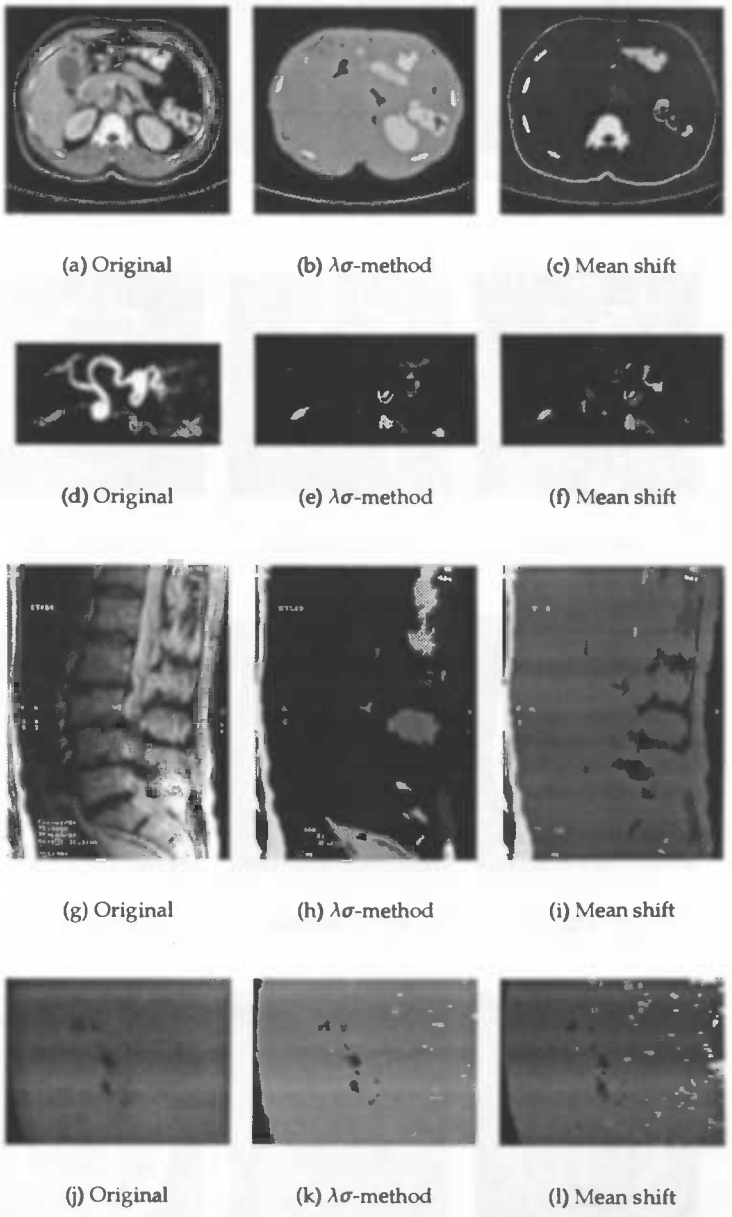
(a) Original (b) $\lambda\sigma$-method (c) Mean shift

(d) Original (e) $\lambda\sigma$-method (f) Mean shift

(g) Original (h) $\lambda\sigma$-method (i) Mean shift

(j) Original (k) $\lambda\sigma$-method (l) Mean shift

Figure 7.6: LML-RATS experiments with real images, part 2



(a) Original (b) $\lambda\sigma$-method (c) Mean shift

(d) Original (e) $\lambda\sigma$-method (f) Mean shift

(g) Original (h) $\lambda\sigma$-method (i) Mean shift

(j) Original (k) $\lambda\sigma$-method (l) Mean shift

Figure 7.7: LML-RATS experiments with real images, part 3

Chapter 8

Conclusions

In this paper, we have shown methods for performing both global quantisation and local quantisation, based on an adaption of the RATS algorithm.

The global method is not particularly suitable for segmentation purposes, as we are often left with a large number of classes. The local method, on the other hand, often yields a smaller number of classes, and this method is therefore better suited to segmentation. The segmentation results are good when using phantom images, but not nearly as good when using real images. This is also somewhat depending on the algorithm parameters, and the calculation of the amount of noise due to Kittler, which overestimates the amount of noise present in images with high amounts of texture.

The bleeding effect in particular, where two objects are merged together in the segmenation if they are connected by a gradient in the original image, is a real problem.

It is interesting to note that the window size and shape, as far as we tested this, did not seem to have any significant influence on the quality of the segmentation. The performance, on the other hand, is negatively influenced by a larger window, or by choosing Gaussian windows rather than square windows.

The performance of the global algorithm is reasonable, however, the local algorithm is often rather slow, particularly when we are using larger Gaussian windows. We can decrease the algorithm's running time by using square windows, but we will still need to optimise the algorithm to achieve acceptable performance.

8.1 Future work

We can perhaps improve upon the performance of the algorithm by calculating the square windows in an incremental approach. Currently, the square windows are recalculated at every pixel, but only a single row of pixels changes in the square window of two neighbouring pixels. By using this information, we can calculate the window's histogram incrementally, and this may improve the algorithm's performance. However, we need to keep in mind that most of the running time is used by the clustering algorithm, so unless this algorithm is also made incremental, the improvements may not be very impressive.

Currently, we have experimented with each of the algorithm's parameters separately. However, it may be interesting to experiment with combinations of parameters, as this may give more accurate values for the parameters. Also, it would be useful to perform experiments with the area filtering method, to find optimal values of the minimum area relating to, for instance, the size of the image.

It may also be interesting to consider additional noise reduction methods. One method which may be promising is to perform an area opening of the gradient image, so that small areas of similar gradient are discarded. as these are likely to be noise, while larger areas of similar gradients are preserved, as these are likely to be the structures we are interested in.

Bibliography

[1] A. Beghdadi and A. Le Negrate. Contrast enhancement technique based on local detection of edges. *Computer Vision, Graphics, and Image Processing*, 46:162–174, 1989.

[2] K. Fukunaga and L. D. Hostetler. Estimation of the gradient of a density function with applications in pattern recognition. *IEEE Transactions on Information Theory*, IT-21:32–40, 1975.

[3] R. Gordon and R. M. Rangayan. Feature enhancement of film mammograms using fixed and adaptive neighborhoods. *Applied Optics*, 23(4):560–564, 1984.

[4] J. Kittler, J. Illingworth, and J. Föglein. Threshold selection based on a simple image statistic. *Computer Vision, Graphics, and Image Processing*, 30:125–147, 1985.

[5] M. D. Levine and A. M. Nazif. An experimental rule based system for testing low level segmentation strategy. In K. Preston and L. Uhr, editors, *Multicomputers and Image Processing: Algorithms and Programs*, pages 149–160. Academic Press, New York, 1982.

[6] A. Meijster and M. H. F. Wilkinson. A comparison of algorithms for connected set openings and closings. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(4):484–494, 2002.

[7] N. Otsu. A threshold selection method from grey-level histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-9:62–66, 1979.

[8] B. W. Silverman. *Density Estimation for Statistics and Data Analysis*. Chapman & Hall, 1986. Page 133.

[9] R. E. Tarjan. Efficiency of a good but not linear set union algorithm. *Journal of the ACM*, 22(2):215–225, 1975.

[10] M. H. F. Wilkinson. Optimizing edge detectors for robust automatic threshold selection: Coping with edge curvature and noise. *CVGIP: Graphical Models and Image Processing*, 60:385–401, 1998.

[11] M. H. F. Wilkinson. Gaussian-weighted moving-window robust automatic threshold selection. In *Computer Analysis of Image and Patterns, 10th International Conference*, pages 369–376, Groningen, 2003.

GNU Free Documentation License

Preamble

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. Applicability and Definitions

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary

word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. Verbatim Copying

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. Copying in Quantity

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. Modifications

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties–for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. Combining Documents

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. Collections of Documents

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. Aggregation with Independent Works

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket

the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. Translation

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. Termination

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. Future Revisions of this License

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See `http://www.gnu.org/copyleft/`.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.