Image Processing: The Fundamentals. Maria Petrou and Panagiota Bosdogianni

Print ISBN 0-471-99883-4 Electronic ISBN 0-470-84190-7

Chapter 7

Image Segmentation and Edge Detection

What is this chapter about?

This chapter is about those Image Processing techniques that are used in order to prepare an image as an input to an automatic vision system. These techniques perform *image segmentation* and *edge detection*, and their purpose is to **extract** information from an image in such a way that the output image contains much **less** information than the original one, but the little information it contains is much more relevant to the other modules of an automatic vision system than the discarded information.

What exactly is the purpose of image segmentation and edge detection?

The purpose of image segmentation and edge detection is to extract the outlines of different regions in the image; i.e. to divide the image in to regions which are made up of pixels which have something in common. For example, they may have similar brightness, or colour, which may indicate that they belong to the same object or facet of an object.

How can we divide an image into uniform regions?

One of the simplest methods is that of histogramming and *thresholding*. If we plot the number of pixels which have a specific grey level value, versus that value, we create the histogram of the image. Properly normalized, the histogram is essentially the probability density function for a certain grey level value to occur. Suppose that we have images consisting of bright objects on a dark background and suppose that we want to extract the objects. For such an image, the histogram will have two peaks and a valley between them.

We can choose as the threshold then the grey level value which corresponds to the valley of the histogram, indicated by t_0 in *Figure* 7.1, and *label* all pixels with grey

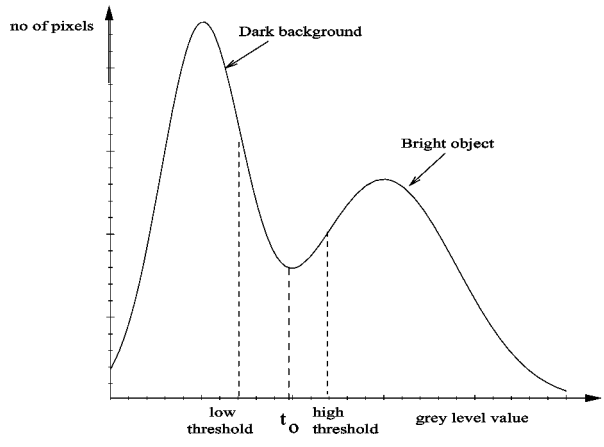


Figure 7.1: The histogram of an image with a bright object on a dark background.

level values greater than t_0 as object pixels and pixels with grey level values smaller than t_0 as background pixels.

What do we mean by "labelling" an image?

When we say we "extract" an object in an image, we mean that we identify the pixels that make it up. To express this information, we create an array of the same size as the original image and we give to each pixel a *label*. All pixels that make up the object are given the same label and all pixels that make up the background are given a different label. The label is usually a number, but it could be anything: a letter or a colour. It is essentially a name and it has symbolic meaning only. Labels, therefore, cannot be treated as numbers. Label images cannot be processed in the same way as grey level images. Often label images are also referred to as *classified images* as they indicate the *class* to which each pixel belongs.

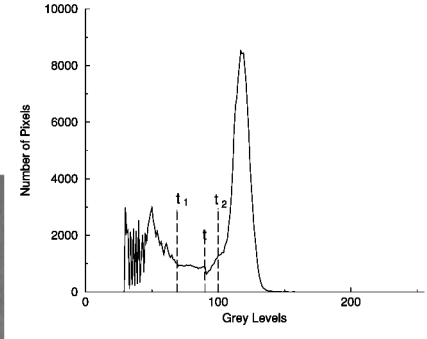
What can we do if the valley in the histogram is not very sharply defined?

If there is no clear valley in the histogram of an image, it means that there are several pixels in the background which have the same grey level value as pixels in the object and vice versa. Such pixels are particularly encountered near the boundaries of the objects which may be fuzzy and not sharply defined. One can use then what is called *hysteresis thresholding*: instead of one, two threshold values (see *Figure* 7.1) are chosen on either side of the valley.

The highest of the two thresholds is used to define the "hard core" of the object. The lowest is used in conjunction with spatial proximity of the pixels: a pixel with intensity value greater than the smaller threshold but less than the larger threshold is labelled as object pixel only if it is adjacent to a pixel which is a core object pixel.

Figure 7.2 shows an image depicting a dark object on a bright background and its histogram. In 7.2c the image is segmented with a single threshold, marked with a t in the histogram, while in 7.2d it has been segmented using two thresholds marked t_1 and t_2 in the histogram.

(a) Original image



(b) Histogram of (a)

(c) Thresholded with t=91

(d) Thresholded with $t_1 = 68$ and $t_2 = 100$

Figure 7.2: Simple thresholding versus hysteresis thresholding.

Alternatively, we may try to choose the global threshold value in an optimal way. Since we know we are bound to misclassify some pixels, we may try to minimize the number of misclassified pixels.

How can we minimize the number of misclassified pixels?

We can minimize the number of misclassified pixels if we have some prior knowledge about the distributions of the grey values that make up the object and the background.

For example, if we know that the objects occupy a certain fraction θ of the area of the picture then this θ is the prior probability for a pixel to be an object pixel. Clearly the background pixels occupy $1-\theta$ of the area and a pixel has $1-\theta$ prior probability to be a background pixel. We may choose the threshold then so that the pixels we classify as object pixels are a θ fraction of the total number of pixels. This method is called *p-tile* method. Further, if we also happen to know the probability density functions of the grey values of the object pixels and the background pixels, then we may choose the threshold that exactly minimizes the error.

B7.1 Differentiation of an integral with respect to a parameter.

Suppose that the definite integral $I(\lambda)$ depends on a parameter λ as follows:

$$I(\lambda) = \int_{a(\lambda)}^{b(\lambda)} f(x;\lambda)dx$$

Its derivative with respect to λ is given by the following formula, known as the *Leibnitz rule*:

$$\frac{dI(\lambda)}{d\lambda} = \frac{db(\lambda)}{d\lambda}f(b(\lambda);\lambda) - \frac{da(\lambda)}{d\lambda}f(a(\lambda);\lambda) + \int_{a(\lambda)}^{b(\lambda)} \frac{\partial f(x;\lambda)}{\partial\lambda}dx \qquad (7.1)$$

How can we choose the minimum error threshold?

Let us assume that the pixels which make up the object are distributed according to the probability density function $p_o(x)$ and the pixels which make up the background are distributed according to function $p_b(x)$.

Suppose that we choose a threshold value t (see *Figure* 7.3). Then the error committed by misclassifying object pixels as background pixels will be given by:

$$\int_{-\infty}^{t} p_o(x)dx$$

and the error committed by misclassifying background pixels as object pixels is:

$$\int_{t}^{+\infty} p_b(x)dx$$

In other words, the error that we commit arises from misclassifying the two tails of the two probability density functions on either side of threshold t. Let us also assume that the fraction of the pixels that make up the object is θ, and by inference, the

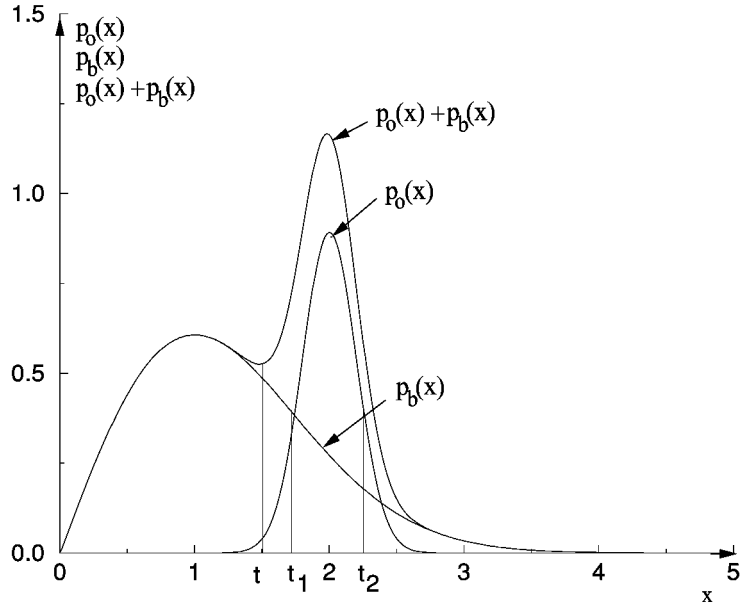


Figure 7.3: The probability density functions of the grey values of the pixels that make up the object ($p_o(x)$) and the background ($p_b(x)$). Their sum, normalized to integrate to 1, is what we obtain if we take the histogram of an image and normalize it.

fraction of the pixels that make up the background is $1-\theta$. Then the total error is:

$$E(t) = \theta \int_{-\infty}^{t} p_o(x)dx + (1-\theta)\int_{t}^{+\infty} p_b(x)dx \tag{7.2}$$

We would like to choose t so that $E(t)$ is minimum. We take the first derivative of $E(t)$ with respect to t (see Box **B7.1**) and set it to zero:

$$\frac{\partial E}{\partial t} = \theta p_o(t) - (1-\theta)p_b(t) = 0 \Rightarrow$$
$$\theta p_o(t) = (1-\theta)p_b(t) \tag{7.3}$$

The solution of this equation gives the minimum error threshold, for any type of distributions the two pixel populations have.

Example 7.1 (B)

Derive equation (7.3) from (7.2).

We apply the Leibnitz rule given by equation (7.1) to perform the differentiation of $E(t)$ given by equation (7.2). We have the following correspondences:
Parameter λ corresponds to t.
For the first integral:

$$\begin{aligned} a(\lambda) &\to -\infty && \textit{(a constant, with zero derivative)} \\ b(\lambda) &\to t \\ f(x;\lambda) &\to p_0(x) && \textit{(independent from the parameter with respect to which we differentiate)} \end{aligned}$$

For the second integral:

$$\begin{aligned} a(\lambda) &\to t \\ b(\lambda) &\to -\infty && \textit{(a constant, with zero derivative)} \\ f(x;\lambda) &\to p_b(x) && \textit{(independent from } t\textit{)} \end{aligned}$$

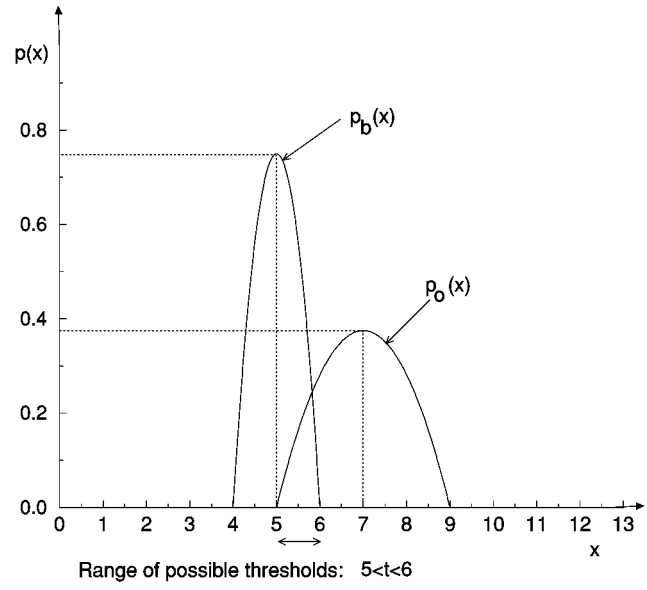
Equation (7.3) then follows.

Example 7.2

The grey level values of the object and the background pixels are distributed according to the probability density function:

$$p(x) = \begin{cases} \frac{3}{4a^3}\left[a^2 - (x-b)^2\right] & \textbf{for } b-a \leq x \leq b+a \\ 0 & \textbf{otherwise} \end{cases}$$

with $a = 1$, $b = 5$ for the background and $a = 2$, $b = 7$ for the object. Sketch the two distributions and determine the range of possible thresholds.



Example 7.3

If the object pixels are eight-ninths $(\frac{8}{9})$ of the total number of pixels, determine the threshold that minimizes the fraction of misclassified pixels for the problem of Example 7.2.

We substitute into equation (7.3) the following:

$$\theta = \frac{8}{9} \Rightarrow 1 - \theta = \frac{1}{9}$$

$$p_b(t) = \frac{3}{4}(-t^2 - 24 + 10t) \qquad p_0(t) = \frac{3}{32}(-t^2 - 45 + 14t)$$

Then:

$$\frac{1}{9}\ \frac{3}{4}(-t^2 - 24 + 10t) = \frac{8}{9}\ \frac{3}{32}(-t^2 - 45 + 14t) \Rightarrow$$

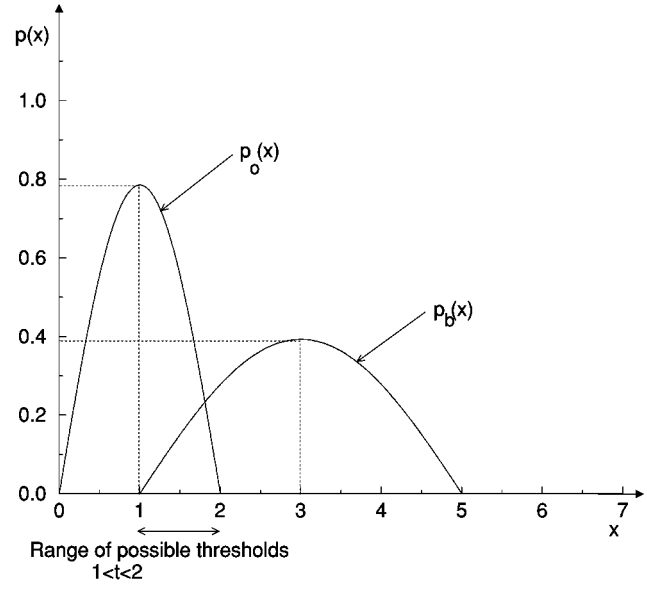
$$-24 + 10t = -45 + 14t \Rightarrow 4t = 21 \Rightarrow t = \frac{21}{4} = 5.25 \qquad (7.4)$$

Example 7.4

The grey level values of the object and the background pixels are distributed according to the probability density function:

$$p(x) = \begin{cases} \frac{\pi}{4a} \cos \frac{(x-x_0)\pi}{2a} & \text{for } x_0 - a \leq x \leq x_0 + a \\ 0 & \text{otherwise} \end{cases}$$

with $x_0 = 1$, $a = 1$ for the objects, and $x_0 = 3$, $a = 2$ for the background. Sketch the two probability density functions. If one-third of the total number of pixels are object pixels, determine the fraction of misclassified object pixels by optimal thresholding.



Apply formula (7.3) with:

$$\theta = \frac{1}{3} \Rightarrow 1 - \theta = \frac{2}{3}$$

$$p_o(x) = \frac{\pi}{4} \cos \frac{(x-1)\pi}{2} \qquad p_b(x) = \frac{\pi}{8} \cos \frac{(x-3)\pi}{4}$$

Equation (7.3) becomes:

$$\frac{1}{3} \frac{\pi}{4} \cos \frac{(t-1)\pi}{2} = \frac{2}{3} \frac{\pi}{8} \cos \frac{(t-3)\pi}{4} \Rightarrow$$

$$\Rightarrow \cos\frac{(t-1)\pi}{2} = \cos\frac{(t-3)\pi}{4} \Rightarrow$$
$$\Rightarrow \frac{(t-1)\pi}{2} = \stackrel{+}{-} \frac{(t-3)\pi}{4}$$

Consider first $\frac{t-1}{2}\pi = \frac{t-3}{4}\pi \Rightarrow 2t-2 = t-3 \Rightarrow t = -1$.
This value is outside the acceptable range, so it is a meaningless solution. Then:

$$\frac{(t-1)\pi}{2} = -\frac{(t-3)\pi}{4} \Rightarrow 2t-2 = -t+3 \Rightarrow 3t = 5 \Rightarrow t = \frac{5}{3}$$

This is the threshold for minimum error. The fraction of misclassified object pixels will be given by all those object pixels that have grey value greater than $\frac{5}{3}$. *We define a new variable of integration* $y \equiv x-1$, *to obtain:*

$$\int_{\frac{5}{3}}^{2} \frac{\pi}{4}\cos\frac{(x-1)\pi}{2}dx = \frac{\pi}{4}\int_{\frac{2}{3}}^{1}\cos\frac{y\pi}{2}dy = \frac{\pi}{4}\frac{\sin\frac{y\pi}{2}}{\frac{\pi}{2}}\Bigg|_{\frac{2}{3}}^{1}$$
$$= \frac{1}{2}\left(\sin\frac{\pi}{2} - \sin\frac{\pi}{3}\right) = \frac{1}{2}(1-\sin 60^o) = \frac{1}{2}\left(1-\frac{\sqrt{3}}{2}\right)$$
$$= \frac{2-1.7}{4} = \frac{0.3}{4} = 0.075 = 7.5\%$$

What is the minimum error threshold when object and background pixels are normally distributed?

Let us assume that the pixels that make up the object are normally distributed with mean μ_o and standard deviation σ_o and the pixels that make up the background are normally distributed with mean μ_b and the standard deviation σ_b:

$$\begin{aligned} p_o(x) &= \frac{1}{\sqrt{2\pi}\sigma_o}\exp\left[-\frac{(x-\mu_o)^2}{2\sigma_o^2}\right] \\ p_b(x) &= \frac{1}{\sqrt{2\pi}\sigma_b}\exp\left[-\frac{(x-\mu_b)^2}{2\sigma_b^2}\right] \end{aligned} \tag{7.5}$$

Upon substitution into equation (7.3) we obtain:

$$\theta\frac{1}{\sqrt{2\pi}\sigma_o}\exp\left[-\frac{(t-\mu_o)^2}{2\sigma_o^2}\right] = (1-\theta)\frac{1}{\sqrt{2\pi}\sigma_b}\exp\left[-\frac{(t-\mu_b)^2}{2\sigma_b^2}\right] \Rightarrow$$
$$\exp\left[-\frac{(t-\mu_o)^2}{2\sigma_o^2} + \frac{(t-\mu_b)^2}{2\sigma_b^2}\right] = \frac{1-\theta}{\theta}\frac{\sigma_o}{\sigma_b} \Rightarrow$$

$$-\frac{(t-\mu_o)^2}{2\sigma_o^2}+\frac{(t-\mu_b)^2}{2\sigma_b^2}=\ln\left[\frac{\sigma_o}{\sigma_b}\frac{1-\theta}{\theta}\right]\Rightarrow$$

$$(t^2+\mu_b^2-2t\mu_b)\sigma_o^2-(t^2+\mu_o^2-2\mu_o t)\sigma_b^2=2\sigma_o^2\sigma_b^2\ln\left[\frac{\sigma_o}{\sigma_b}\frac{1-\theta}{\theta}\right]\Rightarrow$$

$$(\sigma_o^2-\sigma_b^2)t^2+2(-\mu_b\sigma_o^2+\mu_o\sigma_b^2)t+\mu_b^2\sigma_o^2-\mu_o^2\sigma_b^2-2\sigma_o^2\sigma_b^2\ln\left[\frac{\sigma_o}{\sigma_b}\frac{1-\theta}{\theta}\right]=0 \quad (7.6)$$

This is a quadratic equation in t. It has two solutions in general, except when the two populations have the same standard deviation. If $\sigma_o=\sigma_b$, the above expression takes the form:

$$2(\mu_o-\mu_b)\sigma_o^2t+(\mu_b^2-\mu_o^2)\sigma_o^2-2\sigma_o^4\ln\left(\frac{1-\theta}{\theta}\right)=0\Rightarrow$$

$$t=\frac{\sigma_o^2}{\mu_o-\mu_b}\ln\left(\frac{1-\theta}{\theta}\right)-\frac{\mu_o+\mu_b}{2} \quad (7.7)$$

This is the minimum error threshold.

What is the meaning of the two solutions of (7.6)?

When $\sigma_o\neq\sigma_b$, the quadratic term in (7.6) does not vanish and we have two thresholds, t_1 and t_2. These turn out to be one on either side of the sharpest distribution. Let us assume that the sharpest distribution is that of the object pixels (see *Figure* 7.3). Then the correct thresholding will be to label as object pixels only those pixels with grey value x such that $t_1<x<t_2$.

The meaning of the second threshold is that the flatter distribution has such a long tail that the pixels with grey values $x\geq t_2$ are more likely to belong to the long tail of the flat distribution, than to the sharper distribution.

Example 7.5

Derive the optimal threshold for image 7.4a and use it to threshold it.

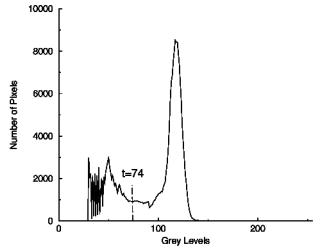
Figure 7.4d shows the image of Figure 7.2a thresholded with the optimal threshold method. First the two main peaks in its histogram were identified. Then a Gaussian was fitted to the peak on the left and its standard deviation was chosen by trial and error so that the best fitting was obtained. The reason we fit first the peak on the left is because it is flatter, so it is expected to have the longest tails which contribute to the value of the peak of the other distribution. Once the first peak has been fitted, the values of the fitting Gaussian are subtracted from the histogram. If the result of this subtraction is negative, it is simply set to zero. Figure 7.4a shows the full histogram. Figure 7.4b shows the histogram with the Gaussian with which the first peak has been fitted superimposed. The mean and the standard deviation of this Gaussian are $\mu_o = 50$ *and* $\sigma_o = 7.5$. *Figure 7.4c shows the histogram that is left after we subtract this Gaussian, with the negative numbers set to zero, and a second fitting Gaussian superimposed. The second Gaussian has* $\mu_b = 117$ *and* $\sigma_b = 7$. *The amplitude of the first Gaussian was* $A_o = 20477$, *and of the second* $A_b = 56597$. *We can estimate* θ, *i.e. the fraction of object pixels, by integrating the two fitting functions:*

$$\frac{\theta}{1-\theta} = \frac{A_o \int_{-\infty}^{\infty} e^{-\frac{(x-\mu_o)^2}{2\sigma_o^2}} dx}{A_b \int_{-\infty}^{\infty} e^{-\frac{(x-\mu_b)^2}{2\sigma_b^2}} dx} = \frac{A_o \sqrt{2\pi}\sigma_o}{A_b \sqrt{2\pi}\sigma_b} = \frac{A_o \sigma_o}{A_b \sigma_b}$$

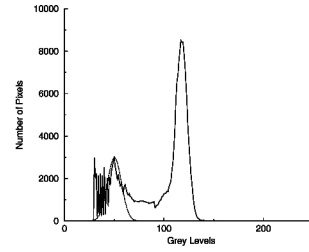
Therefore

$$\theta = \frac{A_o \sigma_o}{A_o \sigma_o + A_b \sigma_b}$$

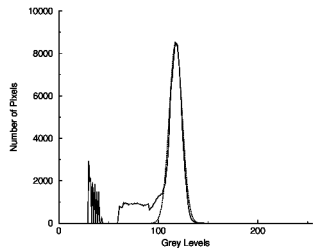
In our case we estimate $\theta = 0.272$. *Substituting these values into equation (7.6) we obtain two solutions* $t_1 = -1213$ *and* $t_2 = 74$. *The original image 7.2a thresholded with* t_2 *is shown in Figure 7.4d. After thresholding, we may wish to check how the distributions of the pixels of each class agree with the assumed distributions. Figures 7.4e and 7.4f show the histograms of the pixels of the object and the background respectively, with the assumed Gaussians superimposed. One can envisage an iterative scheme according to which these two histograms are used to estimate new improved parameters for each class which are then used to define a new threshold, and so on. However, it is not certain that such a scheme will converge. What is more, this result is worse than that obtained by hysteresis thresholding with two heuristically chosen thresholds. This demonstrates how powerful the combination of using criteria of spatial proximity and attribute similarity is.*



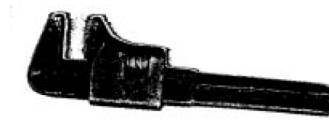
(a) Histogram with the optimal threshold marked



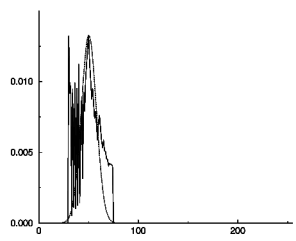
(b) Histogram with the Gaussian model for the object pixels superimposed



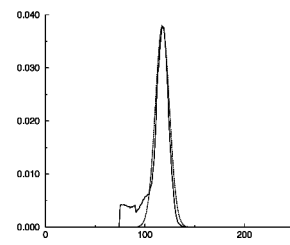
(c) Histogram after subtraction of the Gaussian used to model the object pixels with the Gaussian model for the background pixels superimposed.



(d) Image thresholded with the optimal threshold



(e) Gaussian model used for the object pixels, and their real histogram



(f) Gaussian model used for the background pixels, and their real histogram

Figure 7.4: Optimal thresholding ($t = 74$).

Example 7.6

The grey level values of the object pixels are distributed according to the probability density function:

$$p_o = \frac{1}{2\sigma_o} \exp\left(-\frac{|x-\mu_o|}{\sigma_o}\right)$$

while the grey level values of the background pixels are distributed according to the probability density function:

$$p_b = \frac{1}{2\sigma_b} \exp\left(-\frac{|x-\mu_b|}{\sigma_b}\right)$$

If $\mu_o = 60$, $\mu_b = 40$ and $\sigma_o = 10$ and $\sigma_b = 5$, find the thresholds that minimize the fraction of misclassified pixels when we know that the object occupies two-thirds of the area of the image.

We substitute in equation (7.3):

$$\frac{\theta}{2\sigma_o} e^{-\frac{|t-\mu_o|}{\sigma_o}} = \frac{1-\theta}{2\sigma_b} e^{-\frac{|t-\mu_b|}{\sigma_b}}$$

$$\Rightarrow \exp\left(-\frac{|t-\mu_o|}{\sigma_o} + \frac{|t-\mu_b|}{\sigma_b}\right) = \frac{\sigma_o(1-\theta)}{\sigma_b\theta}$$

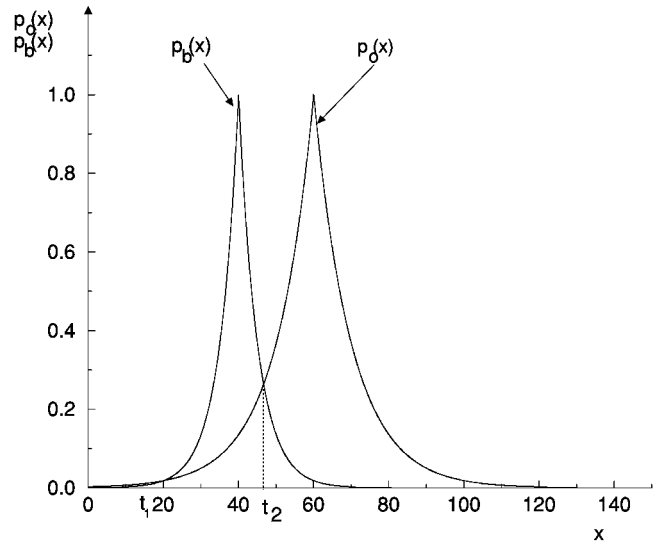
We have $\theta = \frac{2}{3}$ (therefore $1-\theta = \frac{1}{3}$) and $\sigma_o = 10$, $\sigma_b = 5$. Then:

$$-\frac{|t-\mu_o|}{\sigma_o} + \frac{|t-\mu_b|}{\sigma_b} = \ln\frac{10 \times \frac{1}{3}}{5 \times \frac{2}{3}} = \ln 1 = 0$$

We have the following cases:

1. $t < \mu_b < \mu_o \Rightarrow |t-\mu_o| = -t+\mu_o$ *and* $|t-\mu_b| = -t+\mu_b \Rightarrow \frac{t-\mu_o}{\sigma_o} + \frac{-t+\mu_b}{\sigma_b} = 0 \Rightarrow (\sigma_b - \sigma_o)t = \mu_o\sigma_b - \sigma_o\mu_b \Rightarrow t = \frac{\mu_o\sigma_b - \sigma_o\mu_b}{\sigma_b - \sigma_o} \Rightarrow t = \frac{60\times5-10\times40}{-5} \Rightarrow t_1 = 20$
2. $\mu_b < t < \mu_o \Rightarrow |t-\mu_o| = -t+\mu_o$ *and* $|t-\mu_b| = t-\mu_b \Rightarrow \frac{t-\mu_o}{\sigma_o} + \frac{t-\mu_b}{\sigma_b} = 0 \Rightarrow (\sigma_o + \sigma_b)t = \mu_o\sigma_b + \sigma_o\mu_b \Rightarrow t = \frac{\mu_o\sigma_b + \sigma_o\mu_b}{\sigma_b + \sigma_o} = \frac{60\times5+10\times40}{15} = \frac{700}{15} = 47 \Rightarrow t_2 = 47$
3. $\mu_b < \mu_o < t \Rightarrow |t-\mu_o| = t-\mu_o$ *and* $|t-\mu_b| = t-\mu_b \Rightarrow -\frac{t-\mu_o}{\sigma_o} + \frac{t-\mu_b}{\sigma_b} = 0 \Rightarrow (\sigma_o - \sigma_b)t = \sigma_o\mu_b - \mu_o\sigma_b \Rightarrow t = \frac{-\mu_o\sigma_b + \sigma_o\mu_b}{\sigma_o - \sigma_b} = \frac{-60\times5+10\times40}{5} = 20 < \mu_o$, *rejected because t was assumed $> \mu_o$.*

So, there are two thresholds, $t_1 = 20$ and $t_2 = 47$.



Only pixels with the grey level values between t_1 and t_2 should be classified as background pixels in order to minimize the error.

What are the drawbacks of the minimum error threshold method?

The method has various drawbacks. For a start, we must know the prior probabilities for the pixels to belong to the object or the background; i.e. we must know θ. Next, we must know the distributions of the two populations. Often it is possible to approximate these distributions by normal distributions, but even in that case one would have to estimate the parameters σ and μ of each distribution.

Is there any method that does not depend on the availability of models for the distributions of the object and the background pixels?

A method which does not depend on modelling the probability density functions has been developed by Otsu. Unlike the previous analysis, this method has been developed directly in the discrete domain.

Consider that we have an image with L grey levels in total and its normalized histogram, so that for each grey level value x, p_x represents the frequency with which the particular value arises. Then suppose that we set the threshold at t. Let us assume that we are dealing with the case of a bright object on a dark background.

The fraction of pixels that will be classified as background ones will be:

$$\theta(t) = \sum_{x=1}^{t} p_x \tag{7.8}$$

The fraction of pixels that will be classified as object pixels will be:

$$1 - \theta(t) = \sum_{x=t+1}^{L} p_x \tag{7.9}$$

The mean grey level value of the background pixels and the object pixels respectively will be:

$$\mu_b = \frac{\sum_{x=t+1}^{t} xp_x}{\sum_{x=1}^{t} p_x} \equiv \frac{\mu(t)}{\theta(t)} \tag{7.10}$$

$$\mu_o = \frac{\sum_{x=t+1}^{L} xp_x}{\sum_{x=t+1}^{L} p_x} = \frac{\sum_{x=1}^{L} xp_x - \sum_{x=1}^{t} xp_x}{1-\theta(t)} = \frac{\mu - \mu(t)}{1-\theta(t)} \tag{7.11}$$

where we defined $\mu(t) \equiv \sum_{x=1}^{t} xp_x$, and μ is the mean grey level value over the whole image, defined by:

$$\mu \equiv \frac{\sum_{x=1}^{L} xp_x}{\sum_{x=1}^{L} p_x} \tag{7.12}$$

Similarly, we may define the variance of each of the two populations created by the choice of a threshold t as:

$$\begin{aligned} \sigma_b^2 &\equiv \frac{\sum_{x=1}^{t} (x-\mu_b)^2 p_x}{\sum_{x=1}^{t} p_x} = \frac{1}{\theta(t)} \sum_{x=1}^{t} (x-\mu_b)^2 p_x \\ \sigma_o^2 &\equiv \frac{\sum_{x=t+1}^{L} (x-\mu_o)^2 p_x}{\sum_{x=t+1}^{L} p_x} = \frac{1}{1-\theta(t)} \sum_{x=t+1}^{L} (x-\mu_o)^2 p_x \end{aligned} \tag{7.13}$$

Let us consider next the total variance of the distribution of the pixels in the image:

$$\sigma_T^2 = \sum_{x=1}^{L} (x-\mu)^2 p_x$$

We may split this sum into two:

$$\sigma_T^2 = \sum_{x=1}^{t} (x-\mu)^2 p_x + \sum_{x=t+1}^{L} (x-\mu)^2 p_x$$

As we would like eventually to involve the statistics defined for the two populations, we add and subtract inside each sum the corresponding mean:

$$\begin{aligned}\sigma_T^2 &= \sum_{x=1}^{t}(x-\mu_b+\mu_b-\mu)^2 p_x + \sum_{x=t+1}^{L}(x-\mu_o+\mu_o-\mu)^2 p_x \\ &= \sum_{x=1}^{t}(x-\mu_b)^2 p_x + \sum_{x=1}^{t}(\mu_b-\mu)^2 p_x + 2\sum_{x=1}^{t}(x-\mu_b)(\mu_b-\mu)p_x \\ &\quad + \sum_{x=t+1}^{L}(x-\mu_o)^2 p_x + \sum_{x=t+1}^{L}(\mu_o-\mu)^2 p_x + 2\sum_{x=t+1}^{L}(x-\mu_o)(\mu_o-\mu)p_x\end{aligned}$$

Next we substitute the two sums on the left of each line in terms of σ_b^2 and σ_o^2 using equations (7.13). We also notice that the two sums in the middle of each line can be expressed in terms of equations (7.8) and (7.9), since μ, μ_b and μ_o are constants and do not depend on the summing variable x:

$$\begin{aligned}\sigma_T^2 &= \theta(t)\sigma_b^2 + (\mu_b-\mu)^2\theta(t) + 2(\mu_b-\mu)\sum_{x=1}^{t}(x-\mu_b)p_x \\ &\quad + (1-\theta(t))\sigma_o^2 + (\mu_o-\mu)^2\theta(t) + 2(\mu_o-\mu)\sum_{x=t+1}^{L}(x-\mu_b)p_x\end{aligned}$$

The two terms with the sums are zero, since, for example:

$$\sum_{x=1}^{t}(x-\mu_b)p_x = \sum_{x=1}^{t}xp_x - \sum_{x=1}^{t}\mu_b p_x = \mu_b\theta(t) - \mu_b\theta(t) = 0$$

Then by rearranging the remaining terms:

$$\begin{aligned}\sigma_T^2 &= \underbrace{\theta(t)\sigma_b^2 + (1-\theta(t))\sigma_o^2}_{\text{terms depending on the variance } \textbf{within} \text{ each class}} + \underbrace{(\mu_b-\mu)^2\theta(t) + (\mu_o-\mu)^2(1-\theta(t))}_{\text{terms depending on the variance } \textbf{between} \text{ the two classes}} \\ &\equiv \sigma_W^2(t) + \sigma_B^2(t) \qquad (7.14)\end{aligned}$$

where $\sigma_W^2(t)$ is defined to be the within-class variance and $\sigma_B^2(t)$ is defined to be the between-class variance. Clearly σ_T^2 is a constant. We want to specify t so that $\sigma_W^2(t)$ is as small as possible, i.e. the classes that are created are as compact as possible, and $\sigma_B^2(t)$ is as large as possible. Suppose that we choose to work with $\sigma_B^2(t)$, i.e. choose t so that it maximizes $\sigma_B^2(t)$. We substitute in the definition of $\sigma_B^2(t)$ the expression for μ_b and μ_o as given by equations (7.10) and (7.11) respectively:

$$\begin{aligned}\sigma_B^2(t) &= (\mu_b-\mu)^2\theta(t) + (\mu_o-\mu)^2(1-\theta(t)) \\ &= \left[\frac{\mu(t)}{\theta(t)}-\mu\right]^2\theta(t) + \left[\frac{\mu-\mu(t)}{1-\theta(t)}-\mu\right]^2(1-\theta(t))\end{aligned}$$

$$
\begin{aligned}
&= \frac{[\mu(t) - \mu\theta(t)]^2}{\theta(t)} + \frac{[\mu - \mu(t) - \mu + \mu\theta(t)]^2}{1 - \theta(t)} \\
&= \frac{[\mu(t) - \mu\theta(t)]^2[1 - \theta(t)] + \theta(t)[-\mu(t) + \mu\theta(t)]^2}{\theta(t)[1 - \theta(t)]} \\
&= \frac{[\mu(t) - \mu\theta(t)]^2}{\theta(t)[1 - \theta(t)]} \qquad (7.15)
\end{aligned}
$$

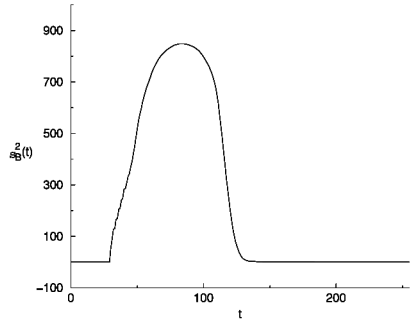
This function expresses the **interclass** variance $\sigma_B^2(t)$ in terms of the mean grey value of the image μ, and quantities that can be computed once we know the values of the image histogram up to the chosen threshold t.

The idea is then to start from the beginning of the histogram and test each grey level value for the possibility of being the threshold that maximizes $\sigma_B^2(t)$, by calculating the values of $\mu(t) = \sum_{x=1}^{t} xp_x$ and $\theta(t) = \sum_{x=1}^{t} p_x$ and substituting into equation (7.15). We stop testing once the value of σ_B^2 starts decreasing. This way we identify t for which $\sigma_B^2(t)$ becomes maximal. This method tacitly assumes that function $\sigma_B^2(t)$ is well-behaved; i.e. that it has only one maximum.

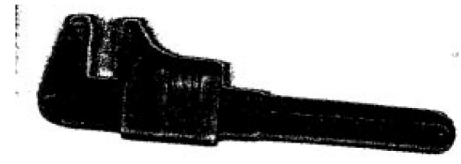
Example 7.7

Calculate Otsu's threshold for the image of Figure 7.2a and use it to threshold the image.

Figure 7.5a shows how $\sigma_B^2(t)$ varies as t scans all possible grey values. The first maximum of this function is at $t = 84$ and we choose this threshold to produce the result shown in Figure 7.5b. We can see that the result is not noticeably different from the result obtained with the empirical threshold (Figure 7.5c) and a little worse than the optimal threshold result (Figure 7.4d). It is much worse than the result obtained by hysteresis thresholding, reinforcing again the conclusion that spatial and grey level characteristics used in thresholding is a powerful combination.



(a) $\sigma_B(t)$ *versus* t.



(b) *Image thresholded with Otsu's threshold.*

Figure 7.5: Otsu's thresholding ($t = 84$).

Are there any drawbacks to Otsu's method?

Yes, a few:

1. Although the method does not make any assumption about the probability density functions $p_o(x)$ and $p_b(x)$, it describes them by using only their means and variances. Thus it tacitly assumes that these two statistics are sufficient in representing them. This may not be true.
2. The method breaks down when the two populations are very unequal. When the two populations become very different in size from each other, $\sigma_B^2(t)$ may have two maxima and actually the correct maximum is not necessarily the global maximum. That is why in practice the correct maximum is selected from among all maxima of $\sigma_B^2(t)$ by checking that the value of the histogram at the selected threshold, p_t, is actually a valley (i.e. $p_t < p_{\mu_o}$, $p_t < p_{\mu_b}$) and only if this is true should t be accepted as the best threshold.
3. The method, as presented above, assumes that the histogram of the image is bimodal; i.e. the image contains two classes. For more than two classes present in the image, the method has to be modified so that multiple thresholds are defined which maximize the *interclass* variance and minimize the *intraclass* variance.
4. The method will divide the image into two classes even if this division does not make sense. A case when the method should not be directly applied is that of variable illumination.

How can we threshold images obtained under variable illumination?

In the chapter on image enhancement, we saw that an image is essentially the **product** of a reflectance function $r(x,y)$ which is intrinsic to the viewed surfaces, and an illumination function $i(x,y)$:

$$f(x,y) = r(x,y)i(x,y)$$

Thus any spatial variation of the illumination results in a multiplicative interference to the reflectance function that is recorded during the imaging process. We can convert the **multiplicative** interference into **additive**, if we take the logarithm of the image:

$$\ln f(x,y) = \ln r(x,y) + \ln i(x,y) \tag{7.16}$$

Then instead of forming the histogram of $f(x,y)$, we can form the histogram of $\ln f(x,y)$.

If we threshold the image according to the histogram of $\ln f(x,y)$, are we thresholding it according to the reflectance properties of the imaged surfaces?

The question really is, what the histogram of $\ln f(x,y)$ is in terms of the histograms of $\ln r(x,y)$ and $\ln i(x,y)$. For example, if $\ln f(x,y)$ is the sum of $\ln r(x,y)$ and $\ln i(x,y)$ which may be reasonably separated functions apart from some overlap, then by thresholding $\ln f(x,y)$ we may be able to identify the $\ln r(x,y)$ component; i.e. the component of interest.

Let us define some new variables:

$$\begin{aligned} z(x,y) &\equiv \ln f(x,y) \\ \tilde{r}(x,y) &\equiv \ln r(x,y) \\ \tilde{i}(x,y) &\equiv \ln i(x,y) \end{aligned}$$

Therefore, equation (7.16) can be written as:

$$z(x,y) = \tilde{r}(x,y) + \tilde{i}(x,y) \tag{7.17}$$

If $f(x,y)$, $r(x,y)$ and $i(x,y)$ are thought of as random variables, then $z(x,y)$, $\tilde{r}(x,y)$ and $\tilde{i}(x,y)$ are also random variables. So, the question can be rephrased into: **What is the histogram of the sum of two random variables in terms of the histograms of the two variables?** A histogram can be thought of as a probability density function. Rephrasing the question again, we have: **What is the probability density function of the sum of two random variables in terms of the probability density functions of the two variables?** We have seen that the probability density function of a random variable is the derivative of the distribution function of the variable. So, we can rephrase the question again: **What is the distribution**

function of the sum of two random variables in terms of the probability density functions or the distribution functions of the two variables? In the $(\tilde{i}, \tilde{r})$ space, equation (7.17) represents a line for a given value of z. By definition, we know that:

$$\text{Distribution function of } z = P_z(u) = \text{Probability of } z \leq u \equiv \mathcal{P}(z \leq u)$$

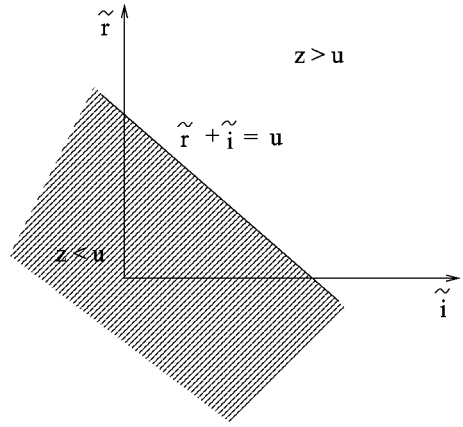


Figure 7.6: z is less than u in the shadowed half plane.

The line $\tilde{r} + \tilde{i} = u$ divides the $(\tilde{i}, \tilde{r})$ plane into two half planes, one in which $z > u$ and one where $z < u$. The probability of $z < u$ is equal to the integral of the probability density function of pairs $(\tilde{i}, \tilde{r})$ over the area of the half plane in which $z < u$ (see *Figure* 7.6):

$$P_z(u) = \int_{\tilde{i}=-\infty}^{+\infty} \int_{\tilde{r}=-\infty}^{u-\tilde{i}} p_{\tilde{r}\tilde{i}}(\tilde{r}, \tilde{i}) d\tilde{r} d\tilde{i}$$

where $p_{\tilde{r}\tilde{i}}(\tilde{r}, \tilde{i})$ is the joint probability density function of the two random variables $\tilde{r}$ and $\tilde{i}$.

To find the probability density function of z, we differentiate $P_z(u)$ with respect to u, using Leibnitz's rule (see Box **B7.1**), applied twice, once with

$$f(x;\lambda) \rightarrow \int_{\tilde{r}=-\infty}^{u-\tilde{i}} p_{\tilde{u}\tilde{i}}(\tilde{r}, \tilde{i}) d\tilde{r}$$
$$b(\lambda) \rightarrow +\infty$$
$$a(\lambda) \rightarrow -\infty$$

and once more when we need to differentiate $f(x;\lambda)$ which itself is an integral that

depends on parameter u with respect to which we differentiate:

$$\begin{aligned} p_z(u) &= \frac{dP_z u}{du} \int_{\tilde{i}=-\infty}^{\infty} \frac{d}{du} \left[\int_{\tilde{r}=-\infty}^{u-\tilde{i}} p_{\tilde{r}\tilde{i}}(\tilde{r},\tilde{i}) d\tilde{r} \right] d\tilde{i} \\ &= \int_{\tilde{i}=-\infty}^{\infty} p_{\tilde{r}\tilde{i}}(u-\tilde{i},\tilde{i}) d\tilde{i} \end{aligned} \tag{7.18}$$

The two random variables $\tilde{r}$ and $\tilde{i}$, one associated with the imaged surface and one with the source of illumination, are independent, and therefore their joint probability density function can be written as the product of their two probability density functions:

$$p_{\tilde{r}\tilde{i}}(\tilde{r},\tilde{i}) = p_{\tilde{r}}(\tilde{r}) p_{\tilde{i}}(\tilde{i})$$

Upon substitution in (7.18) we obtain:

$$p_z(u) = \int_{-\infty}^{\infty} p_{\tilde{r}}(u-\tilde{i}) p_{\tilde{i}}(\tilde{i}) d\tilde{i} \tag{7.19}$$

which shows that the histogram (= probability density function) of z is equal to the **convolution** of the two histograms of the two random variables $\tilde{r}$ and $\tilde{i}$.

If the illumination is uniform, then:

$$i(x,y) = \text{constant} \quad \Rightarrow \quad \tilde{i} = \ln i(x,y) = \tilde{i}_o = \text{constant}$$

Then $p_{\tilde{i}}(\tilde{i}) = \delta(\tilde{i} - \tilde{i}_o)$, and after substitution in (7.19) and integration we obtain $p_z(u) = p_{\tilde{r}}(u)$.

That is, under uniform illumination, the histogram of the reflectance function (intrinsic to the object) is essentially unaffected. If, however, the illumination is not uniform then, even if we had a perfectly distinguishable object, the histogram is badly distorted and the various thresholding methods break down.

Since straightforward thresholding methods break down under variable illumination, how can we cope with it?

There are two ways in which we can circumvent the problem of variable illumination:

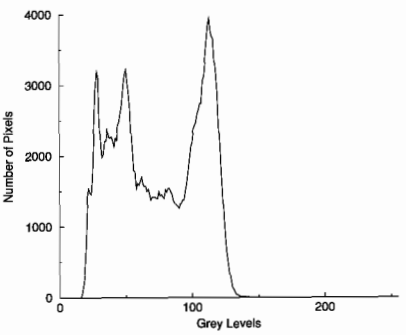
1. Divide the image into more or less uniformly illuminated patches and histogram and threshold each patch as if it were a separate image. Some adjustment may be needed when the patches are put together as the threshold essentially will jump from one value in one patch to another value in a neighbouring patch.
2. Obtain an image of just the illumination field, using the image of a surface with uniform reflectance and divide the image $f(x,y)$ by $i(x,y)$; i.e. essentially subtract the illumination component $\tilde{i}(x,y)$ from $z(x,y)$. Then multiply $\frac{f(x,y)}{i(x,y)}$ with a reference value, say $i(0,0)$, to bring the whole image under the same illumination and proceed using the corrected image.

Example 7.8

Threshold the image of Figure 7.7a.

This image exhibits an illumination variation from left to right. Figure 7.7b shows the histogram of the image. Using Otsu's method, we identify threshold $t = 75$. *The result of thresholding the image with this threshold is shown in Figure 7.7c. The result of dividing the image into four subimages from left to right and applying Otsu's method to each subimage separately is shown in Figure 7.7d.*

(a) Original image



(b) Histogram

(c) Global thresholding

(d) Local thresholding

Figure 7.7: Global versus local thresholding for an image with variable illumination.

Are there any shortcomings of the thresholding methods?

With the exception of hysteresis thresholding which is of limited use, the spatial proximity of the pixels in the image is not considered at all in the segmentation process. Instead, only the grey level values of the pixels are used.

For example, consider the two images in *Figure* 7.8.

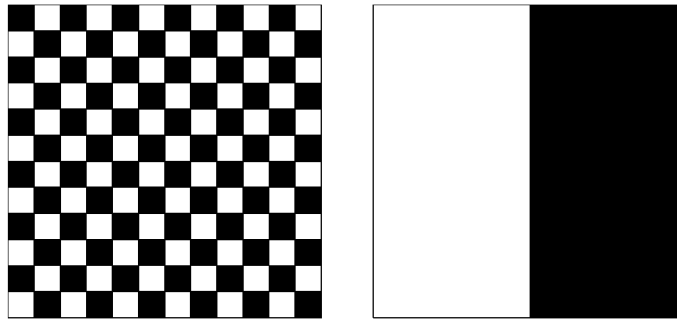
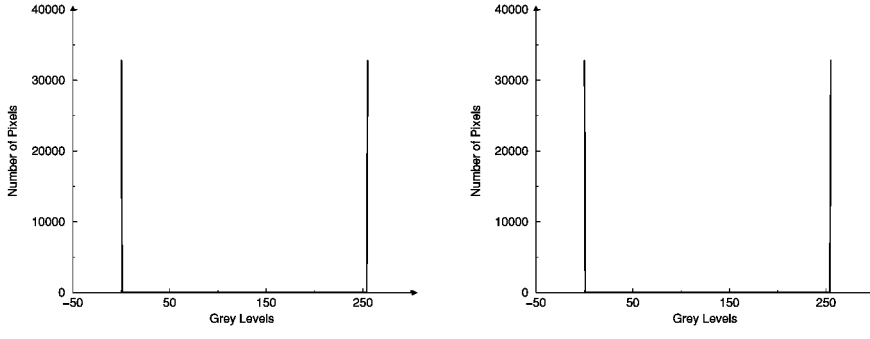


Figure 7.8: Two very different images with identical histograms.

Clearly, the first image is the image of a uniform region, while the second image contains two quite distinct regions. Even so, both images have identical histograms, shown in *Figure* 7.9. Their histograms are bimodal and we can easily choose a threshold. However, if we use it to segment the first image, we shall get nonsense.



(a) Histogram of image in Figure 7.8a

(b) Histogram of image in Figure 7.8b

Figure 7.9: The two identical histograms of the very different images shown in *Figure* 7.8.

How can we cope with images that contain regions that are not uniform but they are *perceived* as uniform?

Regions that are not uniform in terms of the grey values of their pixels but are perceived as uniform, are called *textured regions*. For segmentation purposes then, each pixel cannot only be characterized by its grey level value but also by another number or numbers which quantify the variation of the grey values in a small patch around that pixel. The point is that the problem posed by the segmentation of textured images can be solved by using more than one attribute to segment the image. We can envisage that each pixel is characterized not by one number but by a vector of numbers, each component of the vector measuring something at the pixel position. Then each pixel is represented by a point in a multidimensional space, where we measure one such number, *a feature*, along each axis. Pixels belonging to the same region will have similar or identical values in their attributes and thus will cluster together. The problem then becomes one of identifying clusters of pixels in a multidimensional space. Essentially it is similar to histogramming only now we deal with multidimensional histograms. There are several *clustering* methods that may be used but they are in the realm of *Pattern Recognition* and thus beyond the scope of this book.

Are there any segmentation methods that take into consideration the spatial proximity of pixels?

Yes, they are called *region growing* methods. In general, one starts from some seed pixels and attaches neighbouring pixels to them provided the attributes of the pixels in the region created in this way vary within a predefined range. So, each seed grows gradually by accumulating more and more neighbouring pixels until all pixels in the image have been assigned to a region.

How can one choose the seed pixels?

There is no clear answer to this question, and this is the most important drawback of this type of method. In some applications the choice of seeds is easy. For example, in target tracking in infrared images, the target will appear bright, and one can use as seeds the few brightest pixels. A method which does not need a predetermined number of regions or seeds is that of split and merge.

How does the split and merge method work?

Initially the whole image is considered as one region. If the range of attributes within this region is greater than a predetermined value, then the region is split into four quadrants and each quadrant is tested in the same way until every square region created in this way contains pixels with range of attributes within the given value. At the end all adjacent regions with attributes within the same range may be merged.

An example is shown in *Figure* 7.10 where for simplicity a binary 8×8 image is considered. The tree structure shows the successive splitting of the image into quadrants. Such a tree is called *quadtree.*

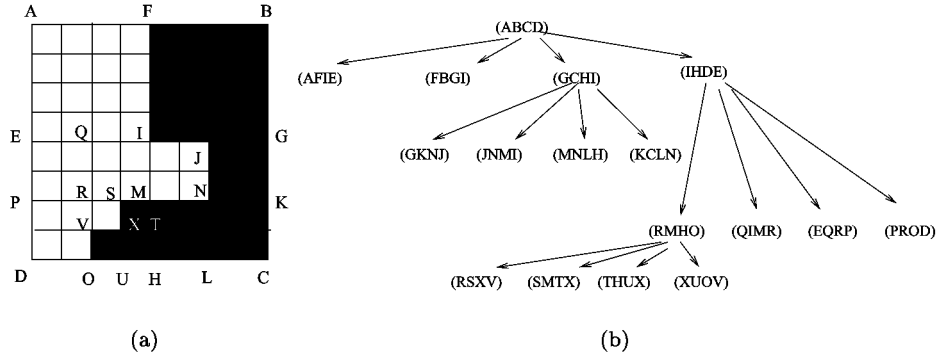


Figure 7.10: Image segmentation by splitting.

We end up having the following regions:

$$(AFIE)(FBGI)(GKNJ)(JNMI)(MNLH)(KCLN)$$
$$(RSXV)(SMTX)(THUX)(XUOV)(QIMR)(EQRP)(PROD)$$

i.e all the children of the quadtree. Any two adjacent regions then are checked for merging and eventually only the two main regions of irregular shape emerge. The above quadtree structure is clearly favoured when the image is square with $N = 2^n$ pixels in each side.

Split and merge algorithms often start at some intermediate level of the quadtree (i.e some blocks of size $2^l \times 2^l$ where $l < n$) and check each block for further splitting into four square sub-blocks and any two adjacent blocks for merging. At the end again we check for merging any two adjacent regions.

Is it possible to segment an image by considering the *dissimilarities* between regions, as opposed to considering the similarities between pixels?

Yes, in such an approach we examine the differences between neighbouring pixels and say that pixels with different attribute values belong to different regions and therefore we postulate a boundary separating them. Such a boundary is called an *edge* and the process is called *edge detection.*

How do we measure the dissimilarity between neighbouring pixels?

We may slide a window across the image and at each position calculate the statistical properties of the pixels within each half of the window and compare the two results.

The places where these statistical properties differ most are where the boundaries of the regions are.

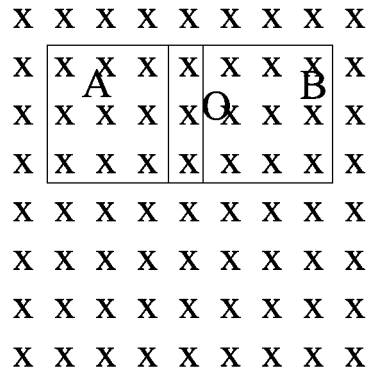


Figure 7.11: Measuring the dissimilarity between two image regions using a sliding widow.

For example, consider the 8×8 image in *Figure* 7.11. Each X represents a pixel. The rectangle drawn is a 3×7 window which could be placed so that its centre O coincides with every pixel in the image, apart from those too close to the edge of the image. We can calculate the statistical properties of the nine pixels on the left of the window (part A) and those of the nine pixels on the right of the window (part B) and assign their difference to pixel O. For example, we may calculate the standard deviation of the grey values of the pixels within each half of the window, say σ_A and σ_B, calculate the standard deviation of the pixels inside the whole window, say σ, and assign the value $E \equiv 2\sigma - \sigma_A - \sigma_B$ to the central pixel. We can slide this window horizontally to scan the whole image. Local maxima of the assigned values are candidate positions for vertical boundaries. Local maxima where the value of E is greater than a certain threshold are accepted as vertical boundaries between adjacent regions. We can repeat the process by rotating the window by 90° and sliding it vertically to scan the whole image again. Clearly the size of the window here plays a crucial role as we need a large enough window to calculate the statistics properly and a small enough window to include within each half only part of a single region and avoid contamination from neighbouring regions.

What is the smallest possible window we can choose?

The smallest possible window we can choose consists of two adjacent pixels. The only "statistic" we can calculate from such a window is the difference of the grey values of the two pixels. When this difference is high we say we have an *edge* passing between the two pixels. Of course, the difference of the grey values of the two pixels is not a statistic but is rather an estimate of the first derivative of the intensity function with respect to the spatial variable along the direction of which we take the difference.

This is because first derivatives are approximated by first differences in the discrete case:

$$\Delta f_x = f(i+1, j) - f(i, j)$$
$$\Delta f_y = f(i, j+1) - f(i, j)$$

Calculating Δf_x at each pixel position is equivalent to convolving the image with a mask (filter) of the form $\boxed{-1}\boxed{+1}$ in the x direction, and calculating Δf_y is equivalent to convolving the image with the filter $\begin{array}{|c|}\hline -1 \\ \hline +1 \\ \hline \end{array}$ in the y direction.

The first and the simplest edge detection scheme then is to convolve the image with these two masks and produce two outputs. Note that these small masks have even lengths so their centres are not associated with any particular pixel in the image as they slide across the image. So, the output of each calculation should be assigned to the position in between the two adjacent pixels. These positions are said to constitute the *dual* grid of the image grid. In practice, we seldom invoke the dual grid. We usually adopt a convention and try to be consistent. For example, we may always assign the output value to the first pixel of the mask. If necessary, we later may remember that this value actually measures the difference between the two adjacent pixels at the position half an interpixel distance to the left or the bottom of the pixel to which it is assigned. So, with this understanding, and for simplicity from now on, we shall be talking about *edge pixels.*

In the first output, produced by convolution with mask $\boxed{-1}\boxed{+1}$, any pixel that has an absolute value larger than values of its left and right neighbours is a candidate pixel to be a vertical edge pixel. In the second output, produced by convolution with mask $\begin{array}{|c|}\hline -1 \\ \hline +1 \\ \hline \end{array}$, any pixel that has an absolute value larger than the values of its top and bottom neighbours is a candidate pixel to be a horizontal edge pixel. The process of identifying the local maxima as candidate edge pixels (=*edgels*) is called *non-maxima suppression.*

In the case of zero noise this scheme will clearly pick up the discontinuities in intensity.

What happens when the image has noise?

In the presence of noise every small and irrelevant fluctuation in the intensity value will be amplified by differentiating the image. It is common sense then that one should smooth the image first with a lowpass filter and then find the local differences. *Figure* 7.12 shows an original image and the output obtained if the $\boxed{-1}\boxed{+1}$ and $\begin{array}{|c|}\hline -1 \\ \hline +1 \\ \hline \end{array}$ convolution filters are applied along the horizontal and vertical directions respectively. The outputs of the two convolutions are squared, added and square rooted to produce the gradient magnitude associated with each pixel. *Figure* 7.13 shows the results obtained using these minimal convolution filters and some more sophisticated filters that take consideration the noise present in the image.

(a) (b)

Figure 7.12: (a) Original image. (b) The image of the gradient magnitudes computed by simple differencing without applying any smoothing. For displaying purposes the gradient image has been subjected to histogram equalization.

Let us consider for example a 1-dimensional signal. Suppose that one uses as lowpass filter a simple averaging procedure. We smooth the signal by replacing each intensity value with the average of three successive intensity values:

$$A_i \equiv \frac{I_{i-1} + I_i + I_{i+1}}{3} \tag{7.20}$$

Then we estimate the derivative at position i by averaging the two differences between the value at position i and its left and right neighbours:

$$F_i \equiv \frac{(A_{i+1} - A_i) + (A_i - A_{i-1})}{2} = \frac{A_{i+1} - A_{i-1}}{2} \tag{7.21}$$

If we substitute from (7.20) into (7.21), we obtain:

$$F_i = \frac{1}{6}[I_{i+2} + I_{i+1} - I_{i-1} - I_{i-2}] \tag{7.22}$$

It is obvious from this example that one can combine the two linear operations of smoothing and finding differences into one operation if one uses large enough masks. In this case, the first difference at each position could be estimated by using a mask like $\boxed{-\frac{1}{6} \mid -\frac{1}{6} \mid 0 \mid \frac{1}{6} \mid \frac{1}{6}}$. It is clear that the larger the mask used, the better

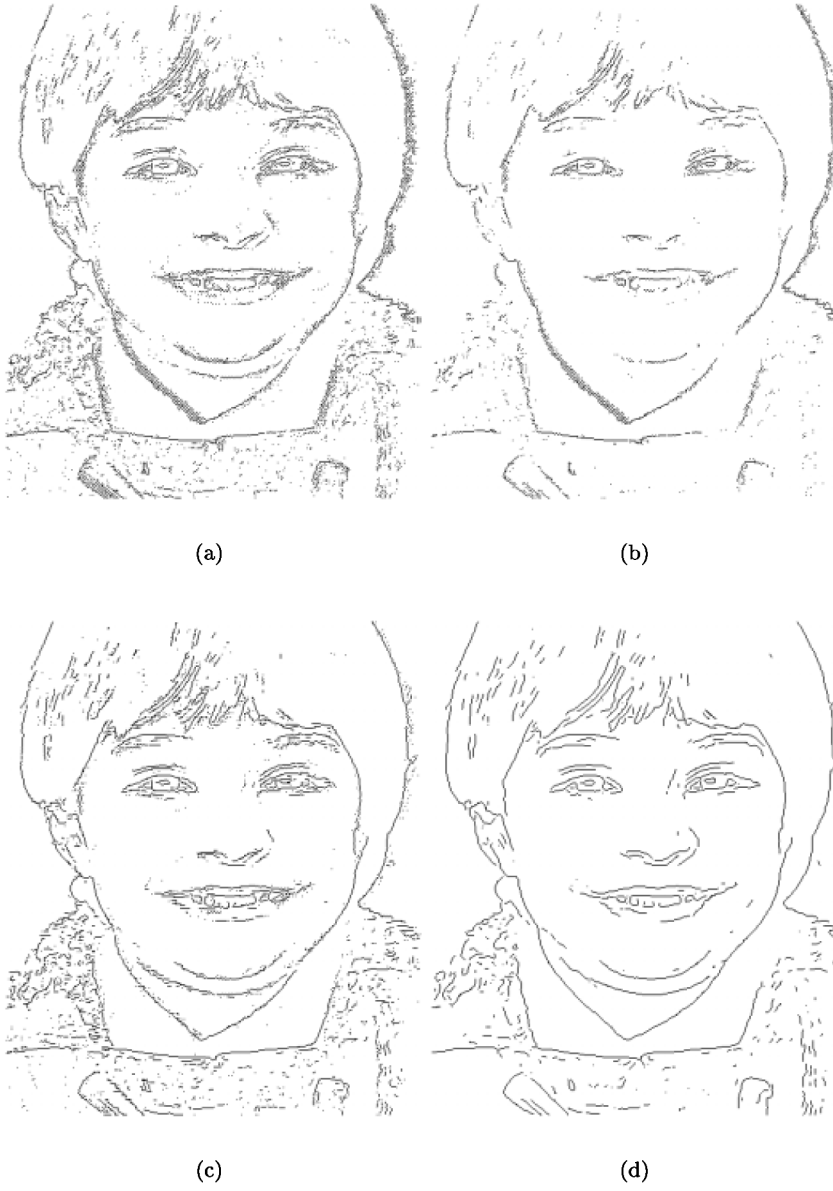


Figure 7.13: (a) Result obtained by simply thresholding the gradient values obtained without any smoothing. (b) The same as in (a) but using a higher threshold, so some noise is removed. However, useful image information has also been removed. (c) Result obtained by smoothing first along one direction and differentiating afterwards along the orthogonal direction, using a Sobel mask. (d) Result obtained using the optimal filter of size 7×7. In all cases the same value of threshold was used.

is the smoothing. But it is also clear that the more blurred the edge becomes so the more inaccurately its position will be specified (see *Figure* 7.14).

For an image which is a 2D signal one should use 2-dimensional masks. The smallest mask that one should use to combine minimum smoothing with differencing is a 3×3 mask. In this case we also have the option to smooth in one direction and take the difference along the other. This implies that the 2D mask may be the result of applying in a cascaded way first a 3×1 smoothing mask and then a 1×3 differencing masks, or vice versa. In general, however, a 2D 3×3 mask will have the form:

a_{11}	a_{12}	a_{13}
a_{21}	a_{22}	a_{23}
a_{31}	a_{32}	a_{33}

How can we choose the weights of a 3×3 mask for edge detection?

We are going to use one such mask to calculate Δf_x and another to calculate Δf_y. Such masks must obey the following conditions:

1. The mask which calculates Δf_x must be produced from the mask that calculates Δf_y by rotation by 90°. Let us consider from now on the mask which will produce Δf_y only. The calculated value will be assigned to the central pixel.
2. We do not want to give any extra weight to the left or the right neighbours of the central pixel, so we must have identical weights in the left and right columns. The 3×3 mask therefore must have the form:

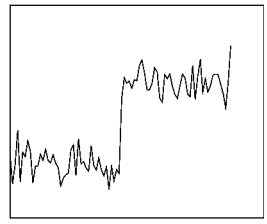
a_{11}	a_{12}	a_{11}
a_{21}	a_{22}	a_{21}
a_{31}	a_{32}	a_{31}

3. Let us say that we want to subtract the signal "in front" of the central pixel from the signal "behind" it, in order to find local differences, and we want these two subtracted signals to have equal weights. The 3×3 mask therefore must have the form:

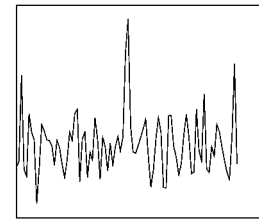
a_{11}	a_{12}	a_{11}
a_{21}	a_{22}	a_{21}
$-a_{11}$	$-a_{12}$	$-a_{11}$

4. If the image is absolutely smooth we want to have zero response. So the sum of all the weights must be zero. Therefore, $a_{22} = -2a_{21}$:

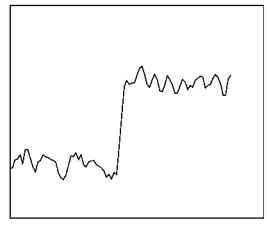
a_{11}	a_{12}	a_{11}
a_{21}	$-2a_{21}$	a_{21}
$-a_{11}$	$-a_{12}$	$-a_{11}$



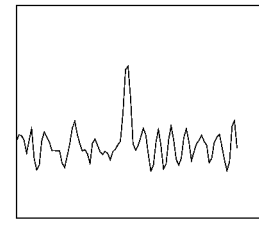
(a) A noisy signal



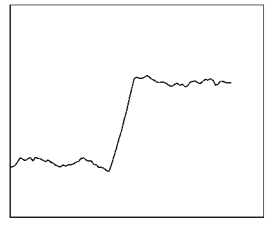
(b) First derivative of (a) by differencing



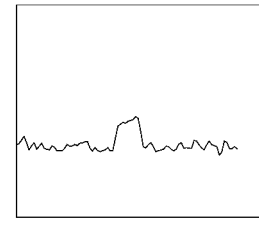
(c) Signal smoothed by averaging over every 3 samples



(d) First derivative of (c) by differencing



(e) Signal smoothed by averaging over every 10 samples



(f) First derivative of (e) by differencing

Figure 7.14: The left column shows a noisy signal and two smoothed versions of it. On the right is the result of estimating the first derivative of the signal by simple differencing. The edge manifests itself with a sharp peak. However, the more the noise is left in the signal, the more secondary peaks will be present, and the more the smoothing, the more blunt the main peak becomes.

5. In the case of a smooth signal, and as we differentiate in the direction of columns, we expect each column to produce 0 output. Therefore, $a_{21} = 0$:

a_{11}	a_{12}	a_{11}
0	0	0
$-a_{11}$	$-a_{12}$	$-a_{11}$

We can divide these weights throughout by a_{11} so that finally, this mask depends only on one parameter:

1	K	1
0	0	0
-1	$-K$	-1

(7.23)

What is the best value of parameter K?

It can be shown that the orientations of edges which are almost aligned with the image axes are not affected by the differentiation, if we choose $K = 2$. We have then the *Sobel* masks for differentiating an image along two directions:

1	2	1
0	0	0
-1	-2	-1

-1	0	1
-2	0	2
-1	0	1

Note that we have changed convention for the second mask and subtract the values of the pixels "behind" the central pixel from the values of the pixels "in front". This is intentional so that the orientation of the calculated edge, computed by using the components of the gradient vector derived using these masks, is measured from the horizontal axis anticlockwise (see Box **B7.2**).

B7.2: Derivation of the Sobel masks

Edges are characterized by their strength and orientation defined by:

$$\text{Strength} = E(i,j) = \sqrt{[\Delta f_x(i,j)]^2 + [\Delta f_y(i,j)]^2}$$

$$\text{Orientation: } a(i,j) = \tan^{-1}\frac{\Delta f_y(i,j)}{\Delta f_x(i,j)}$$

The idea is to try to specify parameter K of mask (7.23) so that the output of the operator is as faithful as possible to the true values of E and a which correspond to the non-discretized image:

$$E = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

$$a = \tan^{-1}\frac{\frac{\partial f}{\partial y}}{\frac{\partial f}{\partial x}}$$

Consider a straight step edge in the scene, passing through the middle of a pixel. Each pixel is assumed to be a tile of size 1×1. Suppose that the edge has orientation θ and suppose that θ is small enough so that the edge cuts lines AB and CD as opposed to cutting lines AC and BD $(0 \leq \theta \leq \tan^{-1}(\frac{1}{3}))$ (see *Figure* 7.15).

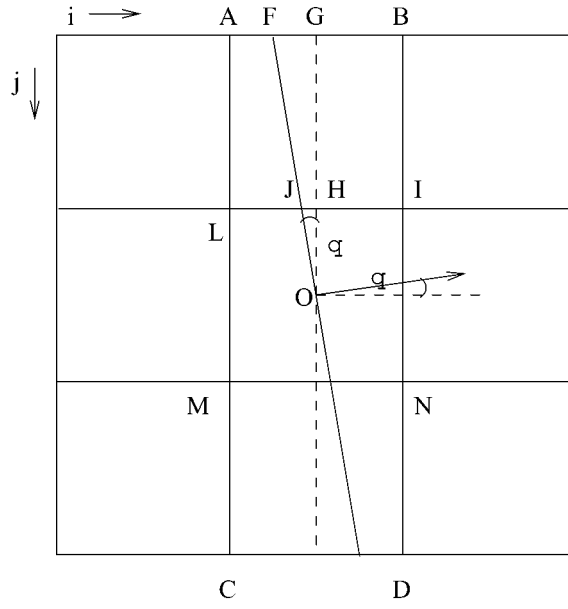


Figure 7.15: Zooming into a 3×3 patch of an image around pixel (i, j).

Also, assume that on the left of the edge we have a dark region with grey value G_1 and on the right a bright region with grey value G_2. Then clearly the pixel values inside the mask are:

$$f(i-1, j-1) = f(i-1, j) = f(i-1, j+1) = G_1$$
$$f(i+1, j-1) = f(i+1, j) = f(i+1, j+1) = G_2$$

The pixels in the central column have mixed values. If we assume that each pixel is like a tile with dimensions 1×1 and denote the area of a polygon by the name of the polygon inside brackets, then pixel $ABIL$ will have value:

$$f(i,j-1) = G_1(AFJL)+G_2(FBIJ) = G_1[\frac{1}{2}-(FGHJ)]+G_2[\frac{1}{2}+(FGHJ)] \quad (7.24)$$

We must find the area of trapezium $FGHJ$. From the triangles OJH and OFG we have:

$$JH = \frac{1}{2}\tan\theta, \qquad FG = \frac{3}{2}\tan\theta$$

Therefore, $(FGJH) = \frac{1}{2}(JH + FG) = \tan\theta$, and by substitution into equation (7.24) we obtain:

$$f(i,j-1) \quad = \quad G_1(\frac{1}{2} - \tan\theta) + G_2(\frac{1}{2} + \tan\theta)$$

By symmetry:

$$f(i,j+1) = G_2(\frac{1}{2} - \tan\theta) + G_1(\frac{1}{2} + \tan\theta)$$

Clearly

$$f(i,j) = \frac{G_1 + G_2}{2}$$

Let us see now what mask (7.23) will calculate in this case:

$$\begin{aligned}
\Delta f_x &= f(i+1,j+1) + f(i+1,j-1) + Kf(i+1,j) \\
&\quad - [f(i-1,j+1) + f(i-1,j-1) + Kf(i-1,j)] \\
&= (G_2 + G_2 + KG_2) - (G_1 + G_1 + KG_1) = (G_2 - G_1)(2+K) \\
\Delta f_y &= -[f(i-1,j+1) + f(i+1,j+1) + Kf(i,j+1)] \\
&\quad + f(i-1,j-1) + f(i+1,j-1) + Kf(i,j-1) \\
&= -G_1 - G_2 - KG_2\left(\frac{1}{2} - \tan\theta\right) - KG_1\left(\frac{1}{2} + \tan\theta\right) + G_1 + G_2 \\
&\quad + KG_1\left(\frac{1}{2} - \tan\theta\right) + KG_2\left(\frac{1}{2} + \tan\theta\right) \\
&= -K(G_2 - G_1)\left(\frac{1}{2} - \tan\theta\right) + K(G_2 - G_1)\left(\frac{1}{2} + \tan\theta\right) \\
&= K(G_2 - G_1)\left(-\frac{1}{2} + \tan\theta + \frac{1}{2} + \tan\theta\right) \\
&= 2K(G_2 - G_1)\tan\theta
\end{aligned}$$

The magnitude of the edge will then be

$$E \quad = \quad \sqrt{(G_2 - G_1)^2(2+K)^2 + (2K)^2(G_2 - G_1)^2\tan^2\theta}$$

$$= (G_2 - G_1)(2+K)\sqrt{1+\left(\frac{2K}{2+K}\right)^2 \tan^2\theta}$$

and the orientation of the edge:

$$\tan\alpha = \frac{\Delta f_y}{\Delta f_x} = \frac{2K\tan\theta}{2+K}$$

Note that if we choose $K = 2$:

$$\tan\alpha = \tan\theta$$

i.e. the calculated orientation of the edge will be equal to the true orientation.

One can perform a similar calculation for the case $\tan^{-1}\frac{1}{3} \leq \theta \leq 45^o$. In that case we have distortion in the orientation of the edge.

Example 7.9

Write down the formula which expresses the output $O(i,j)$ at position (i,j) of the convolution of the image with the Sobel mask that differentiates along the i axis, as a function of the input values $I(i,j)$. (Note: Ignore boundary effects, i.e. assume that (i,j) is sufficiently far from the image border.)

$$\begin{aligned} O(i,j) &= -I(i-1,j-1) - 2I(i-1,j) - I(i-1,j+1) \\ &\quad + I(i+1,j-1) + 2I(i+1,j) + I(i+1,j+1) \end{aligned}$$

Example 7.10

You have a 3×3 image which can be represented by a 9×1 vector. Construct a 9×9 matrix which, when it operates on the image vector, will produce another vector each element of which is the estimate of the gradient component of the image along the i axis. (To deal with the boundary pixels assume that the image is repeated periodically in all directions.)

Consider a 3×3 image:

$$\begin{matrix} & \overset{i}{\rightarrow} & & \\ j\downarrow & \begin{pmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{pmatrix} & & \end{matrix}$$

Periodic repetition of this image implies that we have:

$$\begin{matrix} f_{33} & f_{31} & f_{32} & f_{33} & f_{31} \\ f_{13} & \begin{pmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{pmatrix} & & & f_{11} \\ f_{23} & & & & f_{21} \\ f_{33} & & & & f_{31} \\ f_{13} & f_{11} & f_{12} & f_{13} & f_{11} \end{matrix}$$

Using the result of Example 7.9, we notice that the first derivative at position $(1,1)$ *is given by:*

$$f_{32} + 2f_{12} + f_{22} - f_{33} - 2f_{13} - f_{23}$$

The vector representation of the image is:

$$\begin{pmatrix} f_{11} \\ f_{21} \\ f_{31} \\ f_{12} \\ f_{22} \\ f_{32} \\ f_{13} \\ f_{23} \\ f_{33} \end{pmatrix}$$

If we operate with a 9×9 *matrix on this image, we notice that in order to get the desired output, the first row of the matrix should be:*

$$0 \quad 0 \quad 0 \quad 2 \quad 1 \quad 1 \quad -2 \quad -1 \quad -1$$

The derivative at position $(1,2)$*, i.e. at pixel with value* f_{21}*, is given by:*

$$f_{12} + 2f_{22} + f_{32} - f_{13} - 2f_{23} - f_{33}$$

Therefore, the second row of the 9×9 *matrix should be:*

$$0 \quad 0 \quad 0 \quad 1 \quad 2 \quad 1 \quad -1 \quad -2 \quad -1$$

The derivative at position $(1,3)$ *is given by:*

$$f_{22} + 2f_{32} + f_{12} - f_{23} - 2f_{33} - f_{13}$$

Therefore, the third row of the 9×9 *matrix should be:*

$$0 \quad 0 \quad 0 \quad 1 \quad 1 \quad 2 \quad -1 \quad -1 \quad -2$$

Reasoning this way, we conclude that the matrix we require must be:

$$\begin{pmatrix} 0 & 0 & 0 & 2 & 1 & 1 & -2 & -1 & -1 \\ 0 & 0 & 0 & 1 & 2 & 1 & -1 & -2 & -1 \\ 0 & 0 & 0 & 1 & 1 & 2 & -1 & -1 & -2 \\ -2 & -1 & -1 & 0 & 0 & 0 & 2 & 1 & 1 \\ -1 & -2 & -1 & 0 & 0 & 0 & 1 & 2 & 1 \\ -1 & -1 & -2 & 0 & 0 & 0 & 1 & 1 & 2 \\ 2 & 1 & 1 & -2 & -1 & -1 & 0 & 0 & 0 \\ 1 & 2 & 1 & -1 & -2 & -1 & 0 & 0 & 0 \\ 1 & 1 & 2 & -1 & -1 & -2 & 0 & 0 & 0 \end{pmatrix}$$

Example 7.11

Using the matrix derived in Example 7.10, calculate the first derivative along the i axis of the following image:

$$\begin{pmatrix} 3 & 1 & 0 \\ 3 & 1 & 0 \\ 3 & 1 & 0 \end{pmatrix}$$

What is the component of the gradient along the i axis at the centre of the image?

$$\begin{pmatrix} 0 & 0 & 0 & 2 & 1 & 1 & -2 & -1 & -1 \\ 0 & 0 & 0 & 1 & 2 & 1 & -1 & -2 & -1 \\ 0 & 0 & 0 & 1 & 1 & 2 & -1 & -1 & -2 \\ -2 & -1 & -1 & 0 & 0 & 0 & 2 & 1 & 1 \\ -1 & -2 & -1 & 0 & 0 & 0 & 1 & 2 & 1 \\ -1 & -1 & -2 & 0 & 0 & 0 & 1 & 1 & 2 \\ 2 & 1 & 1 & -2 & -1 & -1 & 0 & 0 & 0 \\ 1 & 2 & 1 & -1 & -2 & -1 & 0 & 0 & 0 \\ 1 & 1 & 2 & -1 & -1 & -2 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 3 \\ 3 \\ 3 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 4 \\ 4 \\ 4 \\ -12 \\ -12 \\ -12 \\ 8 \\ 8 \\ 8 \end{pmatrix}$$

At the centre of the image the component of the gradient along the i *axis is* -12.

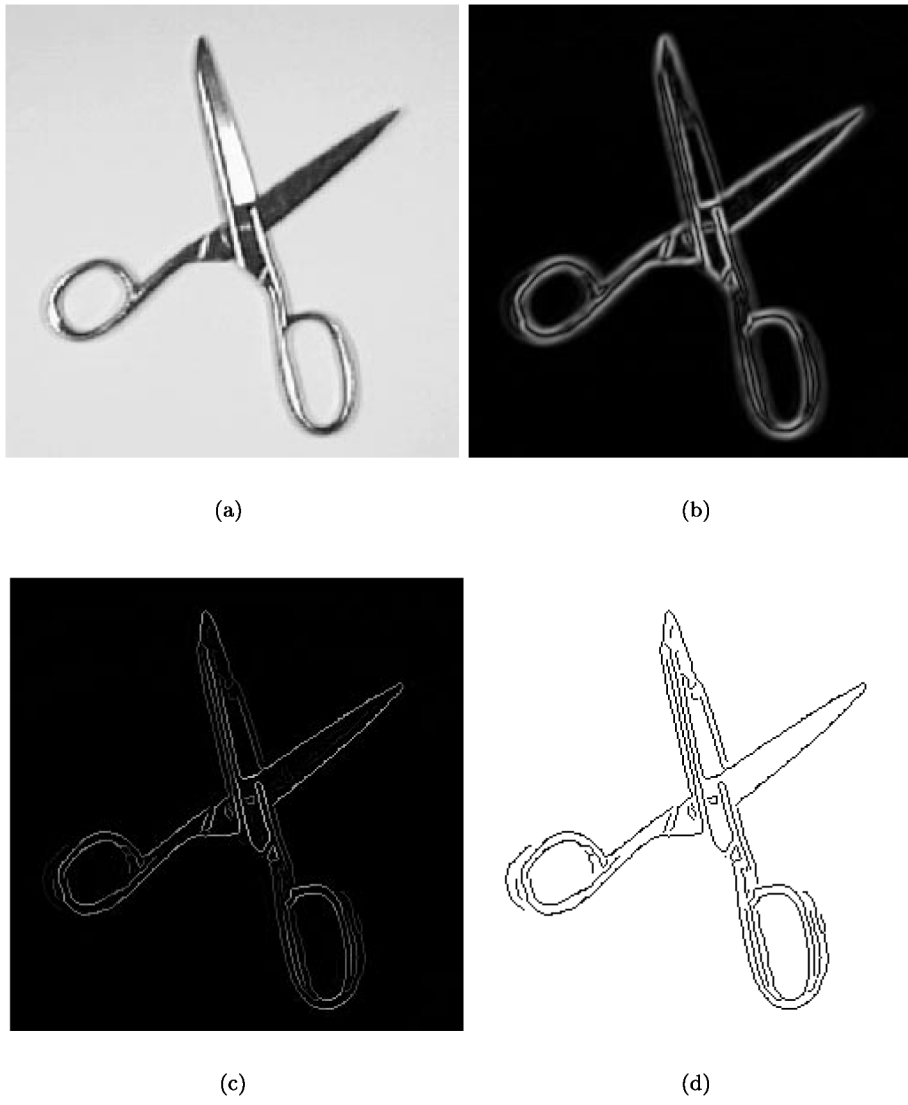


Figure 7.16: (a) Original image. (b) The gradient magnitude "image": the brighter the pixel, the higher the value of the gradient of the image at that point. (c) After non-maxima suppression: only pixels with locally maximal values of the gradient magnitude retain their values. The gradient values of all other pixels are set to zero. (d) The edge map: the gradient values in (c) are thresholded: all pixels with values above the threshold are labelled 0 (black) and all pixels with values below the threshold are labelled 255 (white).

In the general case, how do we decide whether a pixel is an edge pixel or not?

Edges are positions in the image where the image function changes. As the image is a 2D function, to find these positions we have to calculate the gradient of the function $\nabla f(x, y)$. The gradient of a 2D function is a 2D vector with components the partial derivatives of the function along two orthogonal directions. In the discrete case these partial derivatives are the partial differences computed along two orthogonal directions by using masks like, for example, the Sobel masks. If we convolve an image with these masks, we have a gradient vector associated with each pixel. Edges are the places where the magnitude of the gradient vector is a local maximum along the direction of the gradient vector (i.e. the orientation of the gradient at that pixel position). For this purpose, the local value of the gradient magnitude will have to be compared with the values of the gradient estimated along this orientation and at unit distance on either side away from the pixel. In general, these gradient values will not be known because they will happen to be at positions in between the pixels. Then, either a local surface is fitted to the gradient values and used for the estimation of the gradient magnitude at any interpixel position required, or the value of the gradient magnitude is calculated by interpolating the values of the gradient magnitudes at the known integer positions.

After this process of *non-maxima suppression* takes place, the values of the gradient vectors that remain are thresholded and only pixels with gradient values above the threshold are considered as edge pixels identified in the *edge map* (see *Figure* 7.16).

Example 7.12

The following image is given

0	1	2	0	4	5
0	0	1	1	4	5
0	2	0	4	5	4
0	0	5	4	6	6
0	0	6	6	5	6
5	4	6	5	4	5

Use the following masks to estimate the magnitude and orientation of the local gradient at all pixel positions of the image except the boundary pixels:

−1	0	1
−3	0	3
−1	0	1

−1	−3	−1
0	0	0
1	3	1

Then indicate which pixels represent horizontal edge pixels and which represent vertical.

If we convolve the image with the first mask, we shall have an estimate of the gradient component along the x (horizontal) axis:

	5	4	16	17	
	6	11	19	6	
	21	20	7	6	
	24	23	−4	2	

ΔI_x

If we convolve the image with the second mask, we shall have an estimate of the gradient component along the y (vertical) axis:

	1	−1	11	6	
	4	15	15	10	
	0	18	12	4	
	18	8	2	−6	

ΔI_y

The gradient magnitude is given by: $|G| = \sqrt{(\Delta I_x)^2 + (\Delta I_y)^2}$

	$\sqrt{26}$	$\sqrt{17}$	$\sqrt{377}$	$\sqrt{325}$	
	$\sqrt{52}$	$\sqrt{346}$	$\sqrt{586}$	$\sqrt{136}$	
	$\sqrt{441}$	$\sqrt{724}$	$\sqrt{193}$	$\sqrt{52}$	
	$\sqrt{900}$	$\sqrt{593}$	$\sqrt{20}$	$\sqrt{40}$	

(7.25)

The gradient orientation is given by:

$$\theta = \tan^{-1}\frac{\Delta I_y}{\Delta I_x}$$

	$\tan^{-1}\frac{1}{5}$	$-\tan^{-1}\frac{1}{4}$	$\tan^{-1}\frac{11}{16}$	$\tan^{-1}\frac{6}{17}$	
	$\tan^{-1}\frac{2}{3}$	$\tan^{-1}\frac{15}{11}$	$\tan^{-1}\frac{15}{19}$	$\tan^{-1}\frac{5}{3}$	
	$\tan^{-1}0$	$\tan^{-1}\frac{9}{10}$	$\tan^{-1}\frac{12}{7}$	$\tan^{-1}\frac{2}{3}$	
	$\tan^{-1}\frac{3}{4}$	$\tan^{-1}\frac{8}{23}$	$-\tan^{-1}\frac{1}{2}$	$-\tan^{-1}3$	

We know that for an angle θ to be in the range $0°$ to $45°$ ($0° \leq |\theta| \leq 45°$) its tangent must satisfy $0 \leq |\tan\theta| \leq 1$. Also, an angle θ is in the range $45°$ to $90°$ ($45° < |\theta| \leq 90°$) if $1 < |\tan\theta| \leq \infty$.

As we want to quantize the gradient orientations to vertical and horizontal ones only, we shall make all orientations $0° \leq |\theta| \leq 45°$ horizontal, i.e. set them to 0, and all orientations with $45° < |\theta| \leq 90°$ vertical, i.e. we shall set them to $90°$.

By inspecting the orientation array above we infer the following gradient orientations:

	0	0	0	0	
	0	90°	0	90°	
	0	0	90°	0	
	0	0	0	90°	

A pixel is a horizontal edge pixel if the magnitude of its gradient is a local maximum when compared with its vertical neighbours and its orientation is 0. A pixel is a vertical edge if its orientation is $90°$ and its magnitude is a local maximum in the horizontal direction.

By inspecting the gradient magnitudes in (7.25) and the quantized orientations we derived, we identify the following horizontal $(-)$ edges only:

			−		
		−			

Example 7.13

Are the masks used in Example 7.12 separable? How can we take advantage of the separability of a 2D mask to reduce the computational cost of convolution?

The masks used in Example 7.12 are separable because they can be implemented as a sequence of two 1D filters applied in a cascaded way one after the other:

$$\begin{bmatrix} -1 & 0 & 1 \\ -3 & 0 & 3 \\ -1 & 0 & 1 \end{bmatrix} \Rightarrow \begin{bmatrix} 1 \\ 3 \\ 1 \end{bmatrix} \text{ followed by } \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} -1 & -3 & -1 \\ 0 & 0 & 0 \\ 1 & 3 & 1 \end{bmatrix} \Rightarrow \begin{bmatrix} 1 & 3 & 1 \end{bmatrix} \text{ followed by } \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}$$

Any 2D $N \times N$ separable mask can be implemented as a cascade of two 1D masks of size N. This implementation replaces N^2 multiplications and additions per pixel by $2N$ such operations per pixel.

Are Sobel masks appropriate for all images?

Sobel masks are appropriate for images with low levels of noise. They are inadequate for noisy images. See for example *Figure* 7.17 which shows the results of edge detection using Sobel masks for a very noisy image.

How can we choose the weights of the mask if we need a larger mask owing to the presence of significant noise in the image?

We shall consider the problem of detecting abrupt changes in the value of a 1-dimensional signal, like the one shown in *Figure* 7.18.

Let us assume that the feature we want to detect is $u(x)$ and it is immersed in additive white Gaussian noise $n(x)$. The mask we want to use for edge detection should have certain desirable characteristics, called *Canny's criteria*:

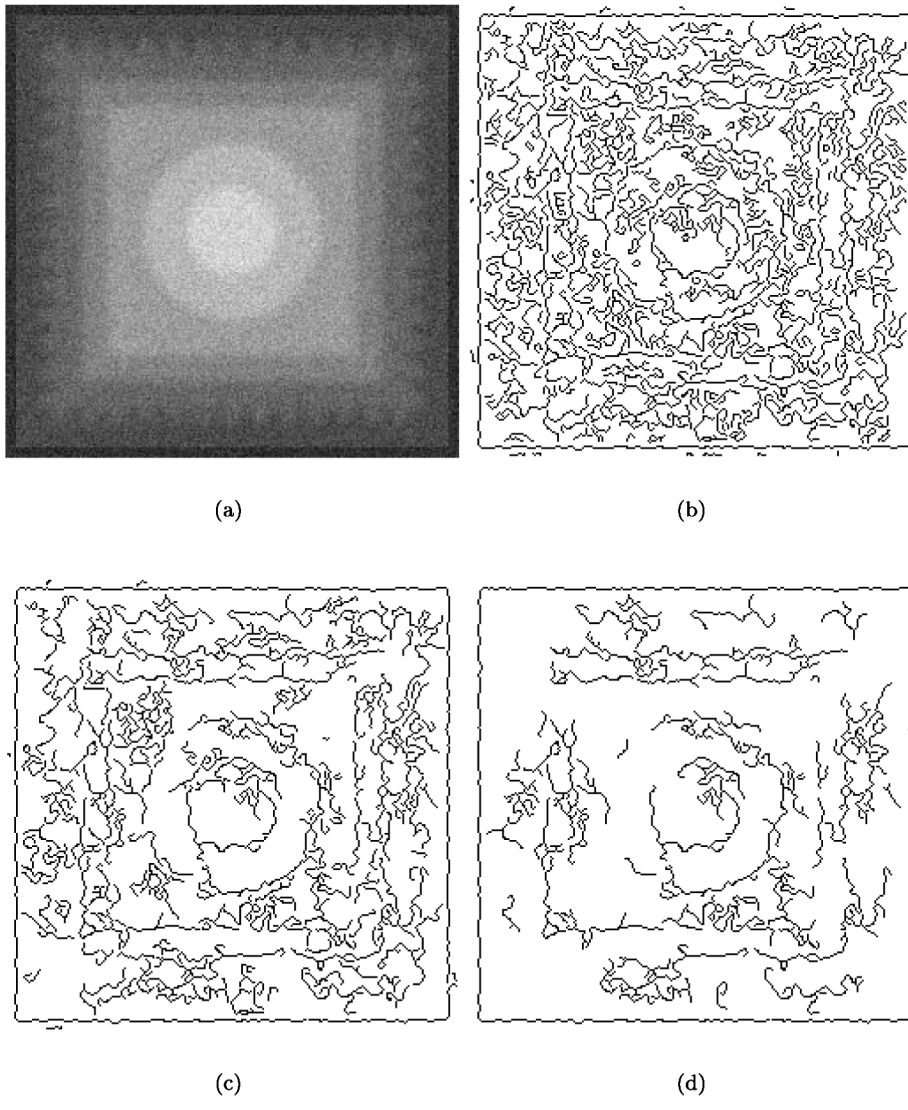


Figure 7.17: Trying to detect edges in a blurred and noisy image using the Sobel edge detector may be very tricky. One can play with different threshold values but the result is not satisfatory. The three results shown have been obtained by using different thresholds within the same edge detection framework, and they were the best among many others obtained for different threshold values.

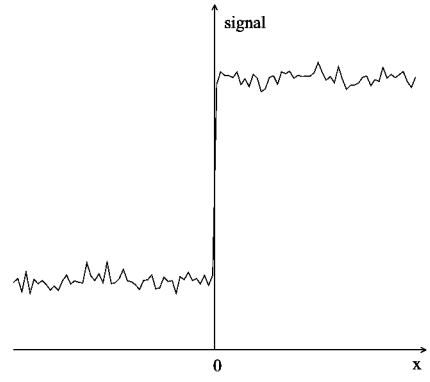


Figure 7.18: A noisy 1D edge.

1. Good signal to noise ratio.
2. Good locality; i.e. the edge should be detected where it actually is.
3. Small number of false alarms; i.e. the maxima of the filter response should be mainly due to the presence of the true edges in the image rather than to noise.

Canny showed that a filter function $f(x)$ has maximal signal to noise ratio if it is chosen in such a way that it maximizes the quantity:

$$SNR = \frac{\int_{-\infty}^{\infty} f(x)u(-x)dx}{n_0\sqrt{\int_{-\infty}^{\infty} f^2(x)dx}} \tag{7.26}$$

where n_0 is the standard deviation of the additive Gaussian noise. As this is a constant for a particular image, when we try to decide what function $f(x)$ should be, we omit it from the computation. So, we try to maximize the quantity:

$$S \equiv \frac{\int_{-\infty}^{\infty} f(x)u(-x)dx}{\sqrt{\int_{-\infty}^{\infty} f^2(x)dx}}$$

Canny also showed that the filter function $f(x)$ detects an edge with the minimum deviation from its true location if it is chosen in such a way that its first and second derivatives maximize the quantity:

$$L \equiv \frac{\int_{-\infty}^{\infty} f''(x)u(-x)dx}{\sqrt{\int_{-\infty}^{\infty} [f'(x)]^2 dx}} \tag{7.27}$$

Finally, he showed that the output of the convolution of the signal with the filter $f(x)$ will contain minimal number of false responses if function $f(x)$ is chosen in such

a way that its first and second derivatives maximize the quantity:

$$C \equiv \sqrt{\frac{\int_{-\infty}^{\infty} (f'(x))^2 dx}{\int_{-\infty}^{\infty} (f''(x))^2 dx}} \tag{7.28}$$

One, therefore, can design an optimal edge enhancing filter by trying to maximize the above three quantities:

$$S, \quad L, \quad C$$

Canny combined the first two into one performance measure P, which he maximized under the constraint that $C = const.$ He derived a convolution filter that way, which, however, he did not use because he noticed that it could be approximated by the derivative of a Gaussian. So he adopted the derivative of a Gaussian as a good enough edge enhancing filter.

Alternatively, one may combine all three quantities S, L, and C into a single performance measure P:

$$P \equiv (S \times L \times C)^2 = \frac{\left[\int_{-\infty}^{\infty} f(x)u(-x)dx\right]^2 \left[\int_{-\infty}^{\infty} f''(x)u(-x)dx\right]^2}{\int_{-\infty}^{\infty} f^2(x)dx \int_{-\infty}^{\infty} (f''(x))^2 dx} \tag{7.29}$$

and try to choose $f(x)$ so that P is maximum. The free parameters which will appear in the functional expression of $f(x)$ can be calculated from the boundary conditions imposed on $f(x)$ and by substitution into the expression for P and selection of their numerical values so that P is maximum.

It must be stressed that the filters defined in this way are appropriate for the detection of **antisymmetric features**; i.e **step** or **ramp edges** when the noise in the signal is **additive**, **white** and **Gaussian**.

Can we use the optimal filters for edges to detect lines in an image in an optimal way?

No. Edge detection filters respond badly to lines. The theory for optimal edge detector filters has been developed under the assumption that there is a single isolated step edge. We can see that from the limits used in the integrals in equation (7.26), for example: the limits are from $-\infty$ to $+\infty$, assuming that in the whole infinite length of the signal there is only a single step edge. If we have a line, its profile looks like the one shown in *Figure* 7.19.

It looks like two step edges back to back. The responses of the filter to the two step edges interfere and the result is not satisfactory: the two steps may or may not be detected. If they are detected, they tend to be detected as two edges noticeably misplaced from their true position and shifted away from one another.

Apart from this, there is another more fundamental difference between step edges and lines.

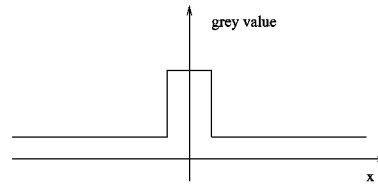


Figure 7.19: The profile of a line.

What is the fundamental difference between step edges and lines?

A step edge is scale invariant: it looks the same whether we stretch it or shrink it. A line has an intrinsic length-scale: it is its width. From the moment the feature we wish to detect has a characteristic "size", the size of the filter we must use becomes important. Similar considerations apply also if one wishes to develop appropriate filters for ramp edges as opposed to step edges: the length over which the ramp rises (or drops) is characteristic of the feature and it cannot be ignored when designing the optimal filter. Petrou and Kittler used the criterion expressed by equation (7.29), appropriately modified, to develop optimal filters for the detection of ramp edges of various slopes. *Figure* 7.20 shows attempts to detect edges in the image of *Figure* 7.17a, using the wrong filter; the edges are blurred, so they actually have ramp-like profiles and the filter used is the optimal filter for step edges. *Figure* 7.21 shows the results of using the optimal filter for ramps on the same image, and the effect the size of the filter has on the results. *Figure* 7.22 shows an example of a relatively "clean" and "unblurred" image for which one does not need to worry too much about which filter one uses for edge detection.

Petrou also modified Canny's criteria for good edge filters for the case of lines and used them to derive optimal filters for line detection that depend on the width and sharpness of the line.

Example 7.14(B)

Assume that $u(x)$ is defined for positive and negative values of x and that we want to enhance a feature of $u(x)$ at position $x = 0$. Use equation (7.26) to show that if the feature we want to enhance makes $u(x)$ an even function, we must choose an even filter, and if it makes it an odd function, we must choose an odd filter.

Any function $f(x)$ can be written as the sum of an odd and an even part:

$$f(x) \equiv \underbrace{\frac{1}{2}[f(x) - f(-x)]}_{f_o(x)(odd)} + \underbrace{\frac{1}{2}[f(x) + f(-x)]}_{f_e(x)(even)}$$

In general, therefore, $f(x)$ can be written as:

$$f(x) = f_e(x) + f_o(x)$$

Suppose that $u(x)$ is even. Then the integral in the numerator of S is:

$$\int_{-\infty}^{\infty} f(x)u(-x)dx = \int_{-\infty}^{\infty} f_e(x)u(-x)dx + \underbrace{\int_{-\infty}^{\infty} f_o(x)u(-x)dx}_{\text{odd intergand integrated over a symmetric interval: it vanishes}}$$

$$= \int_{-\infty}^{\infty} f_e(x)u(-x)dx$$

The integral in the denominator in the expression for S is:

$$\int_{-\infty}^{\infty} f^2(x)dx = \int_{-\infty}^{\infty} f_e^2(x)dx + \int_{-\infty}^{\infty} f_o^2(x)dx + \underbrace{2\int_{-\infty}^{\infty} f_e f_o(x)dx}_{\text{odd intergand integrated over a symmetric interval: it vanishes}}$$

$$= \int_{-\infty}^{\infty} f_e^2(x)dx + \int_{-\infty}^{\infty} f_o^2(x)dx$$

So, we see that the odd part of the filter does not contribute at all in the signal response, while it contributes to the noise response; i.e. it reduces the signal to noise ratio. Thus, to enhance an even feature we must use even filters. Similarly, to enhance an odd feature, we must use odd filters.

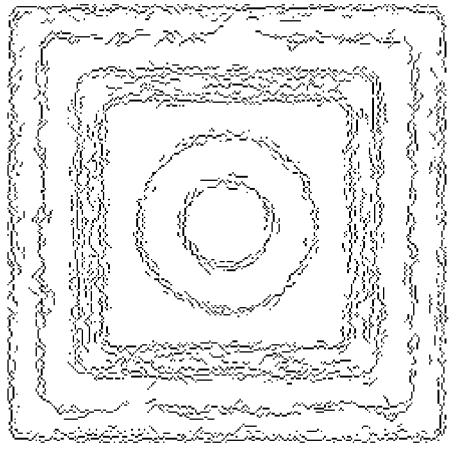
Example 7.15

What type of feature are the digital filter masks below appropriate for enhancing?

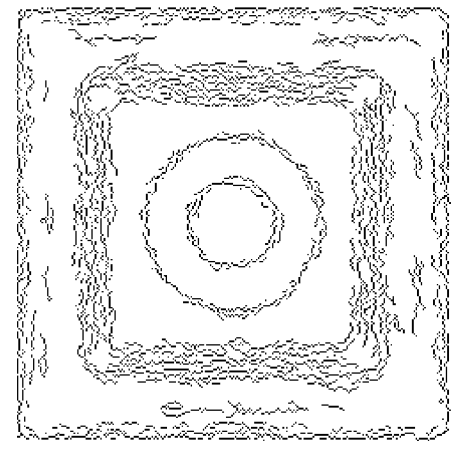
−1	2	−1
−1	2	−1
−1	2	−1

−1	−1	−1
2	2	2
−1	−1	−1

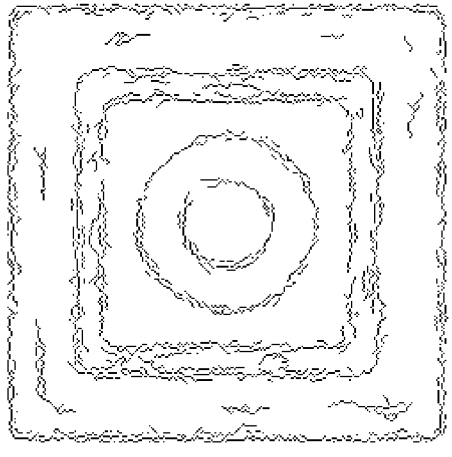
The first enhances vertical lines and the second horizontal lines, in both cases brighter than the background.



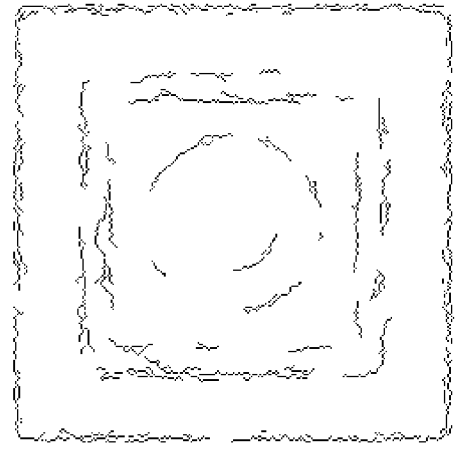
(a) Assuming a step edge and using a filter of size 17×17



(b) The same parameters as in (a) with the size of the filter increased to 21×21. Some detail is lost and no improvement has been achieved

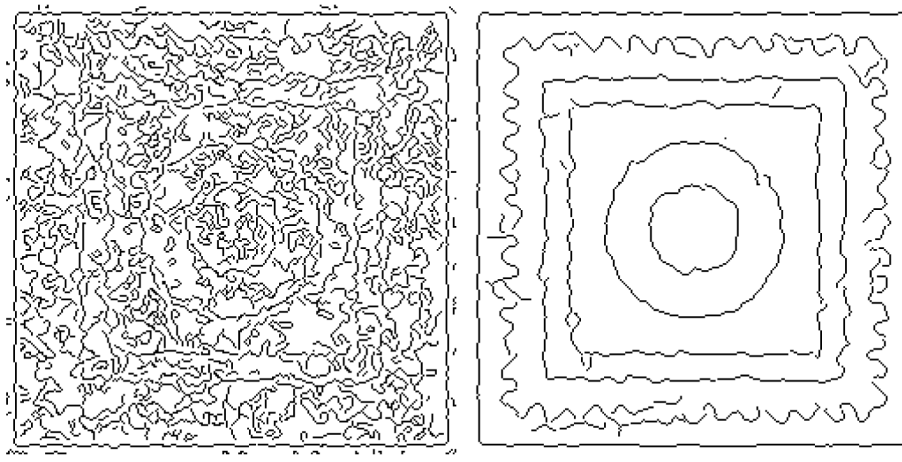


(c) Assuming a step edge and a filter of size 17×17 and increasing the threshold



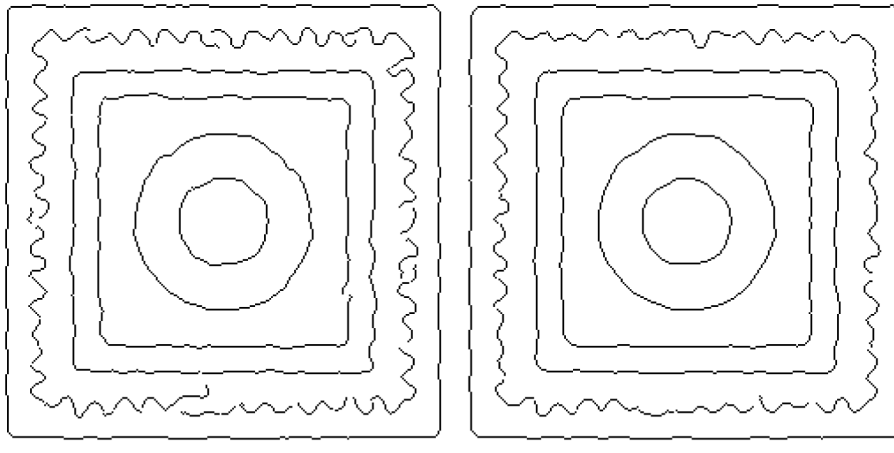
(d) As in (c) with the threshold increased even further

Figure 7.20: Trying to deal with the high level of noise by using an optimal filter of large size and playing with the thresholds may not help, if the edges are blurred and resemble ramps, and the "optimal" filter has been developed to be optimal for step edges.



(a) Assuming a shallow ramp and using a filter of size 7×7

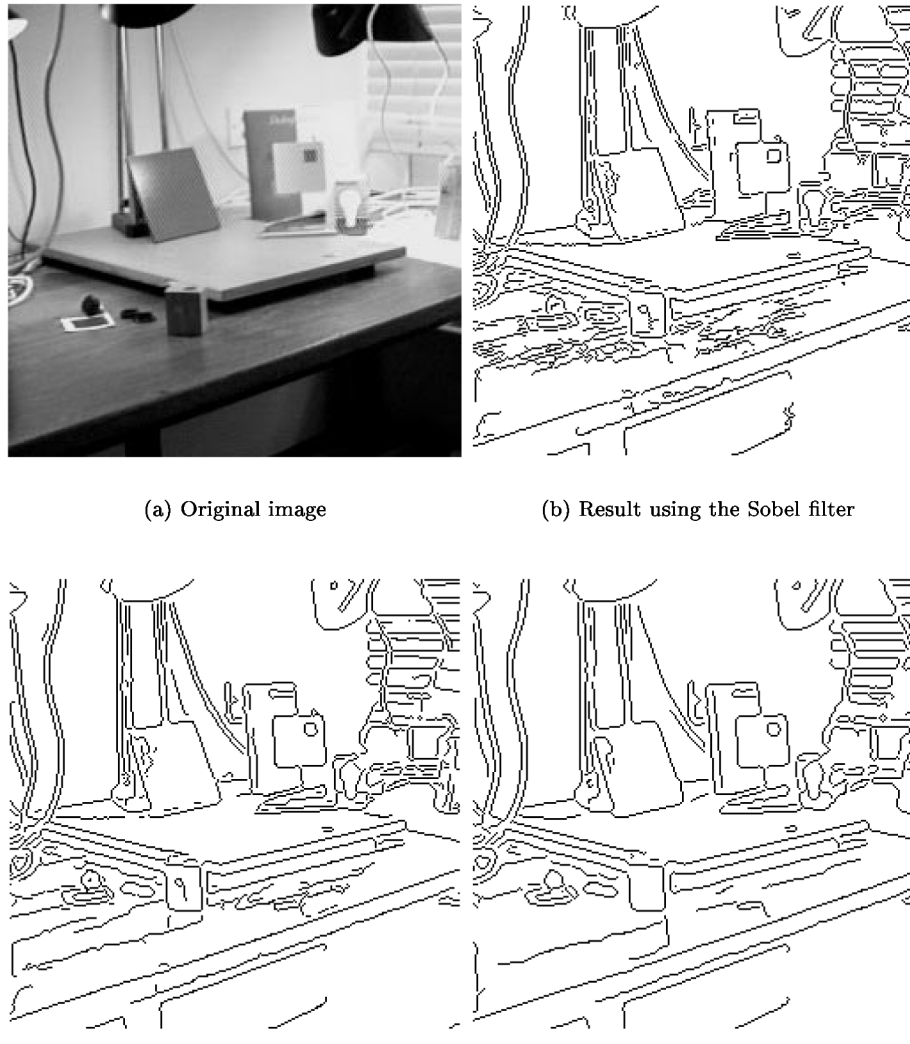
(b) Exactly the same as in (a) but setting the filter size to 13×13



(c) Assuming a ramp edge and filter of size 17×17

(d) Assuming a ramp edge and using a filter of size 21×21

Figure 7.21: Having chosen the right model for the features we wish to detect, using the right filter size for the level of noise may prove crucial.



(a) Original image

(b) Result using the Sobel filter

(c) Result with the optimal filter of size 5×5

(d) Result with the optimal filter of size 7×7

Figure 7.22: Everyday life images are relatively clean and good results can be obtained by using relatively small filters. Note that the smaller the filter, the more details are preserved.

Example 7.16

Use the masks of Example 7.15 to process the image below (do not process the border pixels).

1	1	5	3	0
0	1	4	1	0
1	1	3	2	1
0	2	5	3	0
1	0	4	2	0

Then by choosing an appropriate threshold for the output images calculated, produce the feature maps of the input image.

The result of the convolution of the image with the first mask is:

	−8	15	−1	
	−5	14	−1	
	−8	14	1	

The result of the convolution of the image with the second mask is:

	−2	−3	−4	
	−2	−4	−1	
	4	8	4	

(We do not have output for the border pixels.)
Note that the outputs contain values that can easily be divided into two classes: positive and negative. We choose therefore as our threshold the zero value $t = 0$. This threshold seems to be in the largest gap between the two populations of the output values (i.e. one population that presumably represents the background and one that represents the features we want to detect). When thresholding we shall give one label to one class (say all numbers above the threshold will be labelled 1) and another label to the other class (all numbers below the threhold will be labelled 0). The two outputs then become:

0	1	0
0	1	0
0	1	0

0	0	0
0	0	0
1	1	1

These are the feature maps of the given image.

B7.3: Convolving a random noise signal with a filter.

Suppose that our noise signal is $n(x)$ and we convolve it with a filter $f(x)$. The result of the convolution will be:

$$g(x_0) = \int_{-\infty}^{\infty} f(x)n(x_0 - x)dx \tag{7.30}$$

As the input noise is a random process, this quantity is a random process too. If we take its expectation value, it will be 0 as long as the expectation value of the input noise is 0 too. We may also try to characterize it by calculating its variance, its mean square value. This is given by:

$$E\{[g(x_0)]^2\} = E\{g(x_0)g(x_0)\} \tag{7.31}$$

This is the definition of the autocorrelation function of the output calculated at argument 0. So we must calculate first the autocorrelation function of g, $R_{gg}(0)$. We multiply both sides of equation (7.30) with $g(x_0 + \tau)$ and take expectation values:

$$\begin{aligned} E\{g(x_0)g(x_0+\tau)\} &= E\left\{\int_{-\infty}^{\infty} f(x)g(x_0+\tau)n(x_0-x)dx\right\} \\ \Rightarrow R_{gg}(\tau) &= \int_{-\infty}^{\infty} f(x)E\{g(x_0+\tau)n(x_0-x)\}dx \\ \Rightarrow R_{gg}(\tau) &= \int_{-\infty}^{\infty} f(x)R_{gn}(x_0+\tau-x_0+x)dx \\ \Rightarrow R_{gg}(\tau) &= \int_{-\infty}^{\infty} f(x)R_{gn}(\tau+x)dx \end{aligned} \tag{7.32}$$

where $R_{gn}(a)$ is the cross-correlation function between the input noise signal and the output noise signal at relative shift a. We must calculate R_{gn}. We multiply both sides of (7.30) with $n(x_0-\tau)$ and then take expectation values. (We multiply with $n(x_0 - \tau)$ instead of $n(x_0 + \tau)$ because in (7.32) we defined the argument of R_{gn} as the difference of the argument of g minus the argument of n.)

$$\begin{aligned} E\{g(x_0)n(x_0-\tau)\} &= \int_{-\infty}^{\infty} f(x)E\{n(x_0-x)n(x_0-\tau)\}dx \\ &\Rightarrow \quad R_{gn}(\tau) = \int_{-\infty}^{\infty} f(x)R_{nn}(x_0-x-x_0+\tau)dx \\ &\Rightarrow \quad R_{gn}(\tau) = \int_{-\infty}^{\infty} f(x)R_{nn}(\tau-x)dx \end{aligned} \tag{7.33}$$

where $R_{nn}(a)$ is the autocorrelation function of the input noise signal at relative shift a. However, $n(x)$ is white Gaussian noise, so its autocorrelation function is a delta function given by $R_{nn}(\tau) = n_0^2\delta(\tau)$ where n_0^2 is the variance of the noise. Therefore:

$$R_{gn}(\tau) = \int_{-\infty}^{\infty} f(x)n_0^2\delta(\tau-x)dx = n_0^2 f(\tau) \tag{7.34}$$

Therefore, R_{gn} with argument $\tau + x$ as it appears in (7.32) is:

$$R_{gn}(\tau+x) = n_0^2 f(\tau+x) \tag{7.35}$$

If we substitute in (7.32) we have:

$$R_{gg}(\tau) = n_0^2\int_{-\infty}^{\infty} f(x)f(\tau+x)dx \Rightarrow R_{gg}(0) = n_0^2\int_{-\infty}^{\infty} f^2(x)dx \tag{7.36}$$

If we substitute in equation (7.31) we obtain:

$$E\left\{[g(x_0]^2\right\} = n_0^2\int_{-\infty}^{\infty} f^2(x)dx \tag{7.37}$$

The mean filter response to noise is given by the square root of the above expression.

B7.4: Calculation of the signal to noise ratio after convolution of a noisy edge signal with a filter.

Suppose that $f(x)$ is the filter we want to develop, so that it enhances an edge in a noisy signal. The signal consists of two components: the deterministic signal of interest $u(x)$ and the random noise component $n(x)$:

$$I(x) = u(x) + n(x)$$

The response of the filter due to the deterministic component of the noisy signal, when the latter is convolved with the filter, is:

$$s(x_0) = \int_{-\infty}^{\infty} f(x)u(x_0 - x)dx \tag{7.38}$$

Let us assume that the edge we wish to detect is actually at position $x_0 = 0$. Then:

$$s_0 = \int_{-\infty}^{\infty} f(x)u(-x)dx \tag{7.39}$$

The mean response of the filter due to the noise component (see Box **B7.3**) is $n_0\sqrt{\int_{-\infty}^{\infty} f^2(x)dx}$. The signal to noise ratio, therefore, is:

$$SNR = \frac{\int_{-\infty}^{\infty} f(x)u(-x)dx}{n_0\sqrt{\int_{-\infty}^{\infty} f^2(x)dx}} \tag{7.40}$$

B7.5: Derivation of the good locality measure.

The position of the edge is marked at the point where the output is maximum. The total output of the convolution is the output due to the deterministic signal $u(x)$ and the output due to the noise component $n(x)$. Let us say that the noisy signal we have is:

$$I(x) \quad = \quad u(x) + n(x)$$

The result of the convolution with the filter $f(x)$ will be:

$$\begin{aligned} O(x_0) \quad &\equiv \quad \int_{-\infty}^{\infty} f(x)I(x_0 - x)dx = \int_{-\infty}^{\infty} f(x_0 - x)I(x)dx \\ &= \quad \int_{-\infty}^{\infty} f(x_0 - x)u(x)dx + \int_{-\infty}^{\infty} f(x_0 - x)n(x)dx \equiv s(x_0) + g(x_0) \end{aligned}$$

where $s(x_0)$ is the output due to the deterministic signal of interest and $g(x_0)$ is the output due to noise.

The edge is detected at the local maximum of this output; i.e. at the place where its first derivative with respect to x_0 becomes 0:

$$\frac{dO(x_0)}{dx_0} = \frac{ds(x_0)}{dx_0} + \frac{dg(x_0)}{dx_0} = \int_{-\infty}^{\infty} f'(x_0 - x)u(x)dx + \int_{-\infty}^{\infty} f'(x_0 - x)n(x)dx \quad (7.41)$$

We expand the derivative of the filter, about point $x_0 = 0$, the true position of the edge, and keep only the first two terms of the expansion:

$$f'(x_0 - x) \simeq f'(-x) + x_0 f''(-x) + \ldots$$

Upon substitution into $\frac{ds(x_0)}{dx_0}$ we obtain:

$$\frac{ds(x_0)}{dx_0} \simeq \int_{-\infty}^{\infty} f'(-x)u(x)dx + x_0 \int_{-\infty}^{\infty} f''(-x)u(x)dx \quad (7.42)$$

The feature we want to detect is an antisymmetric feature; i.e. an edge has a shape like $\sqcup$.

According to the result proven in Example 7.14, $f(x)$ should also be an antisymmetric function. This means that its first derivative will be symmetric and when it is multiplied with the antisymmetric function $u(x)$ it will produce an antisymmetric integrand which upon integration over a symmetric interval will make the first term in equation (7.42) vanish. Then:

$$\frac{ds(x_0)}{dx_0} \simeq \underbrace{x_0 \int_{-\infty}^{\infty} f''(-x)u(x)dx}_{\text{set } \tilde{x} \equiv -x} = x_0 \int_{-\infty}^{\infty} f''(\tilde{x})u(-\tilde{x})d\tilde{x} \quad (7.43)$$

The derivative of the filter response to noise may be written in a more convenient way:

$$\frac{dg(x_0)}{dx_0} = \int_{-\infty}^{\infty} f'(x_0 - x)n(x)dx = \int_{-\infty}^{\infty} f'(x)n(x_0 - x)dx \quad (7.44)$$

The position of the edge will be marked at the value of x that makes the sum of the right hand sides of equations (7.43) and (7.44) zero:

$$\frac{ds(x_0)}{dx_0} + \frac{dg(x_0)}{dx_0} = 0 \Rightarrow$$

$$x_0 \int_{-\infty}^{\infty} f''(x)u(-x)dx + \int_{-\infty}^{\infty} f'(x)n(x_0 - x)dx = 0 \Rightarrow$$

$$x_0 \int_{-\infty}^{\infty} f''(x)u(-x)dx = -\int_{-\infty}^{\infty} f'(x)n(x_0 - x)dx \quad (7.45)$$

The right hand side of this expression is a random variable, indicating that the location of the edge will be marked at various randomly distributed positions around the true position, which is at $x_0 = 0$. We can calculate the mean shifting away from the true position as the expectation value of x_0. This, however, is expected to be 0. So, we calculate instead the variance of the x_0 values. We square both sides of (7.45) and take the expectation value of:

$$E\{x_0^2\}\left[\int_{-\infty}^{\infty} f''(x)u(-x)dx\right]^2 = E\left\{\left[\int_{-\infty}^{\infty} f'(x)n(x_0 - x)dx\right]^2\right\}$$

Note that the expectation value operator applies only to the random components.

In Box **B7.3** we saw that if a noise signal with variance n_0^2 is convolved by a filter $f(x)$, the mean square value of the output signal is given by:

$$n_0^2 \int_{-\infty}^{\infty} f^2(x)dx$$

(see equation (7.37)). The right hand side of equation (7.45) here indicates the convolution of the noise component by filter $f'(x)$. Its mean square value therefore is $n_0^2 \int_{-\infty}^{\infty} [f'(x)]^2 dx$. Then:

$$E\{x_0^2\} = \frac{n_0^2 \int_{-\infty}^{\infty} [f'(x)]^2 dx}{\left[\int_{-\infty}^{\infty} f''(x)u(-x)dx\right]^2} \tag{7.46}$$

The smaller this expectation value is, the better the localization of the edge. We can define, therefore, the good locality measure as the inverse of the square root of the above quantity. We may also ignore factor n_0 as it is the standard deviation of the noise during the imaging process, over which we do not have control. So a filter is optimal with respect to good locality, if it maximizes the quantity:

$$L \equiv \frac{\int_{-\infty}^{\infty} f''(x)u(-x)dx}{\sqrt{\int_{-\infty}^{\infty} [f'(x)]^2 dx}} \tag{7.47}$$

Example 7.17 (B)

Show that the differentiation of the output of a convolution of a signal $u(x)$ with a filter $f(x)$ can be achieved by convolving the signal with the derivative of the filter.

The output of the convolution is:

$$s(x_0) = \int_{-\infty}^{\infty} f(x)u(x_0 - x)dx$$

or

$$s(x_0) = \int_{-\infty}^{\infty} f(x_0 - x)u(x)dx \tag{7.48}$$

Applying Leibnitz's rule for differentiating an integral with respect to a parameter (Box **B7.1***) we obtain from equation (7.48):*

$$\frac{ds(x_0)}{dx_0} = \int_{-\infty}^{\infty} f'(x_0 - x)u(x)dx$$

which upon changing variables can be written as:

$$\frac{ds(x_0)}{dx_0} = \int_{-\infty}^{\infty} f'(x)u(x_0 - x)dx$$

B7.6: Derivation of the count of false maxima.

It has been shown by Rice that the average density of zero crossings of the convolution of a function h with Gaussian noise is given by:

$$x_{av} = \pi \left[\frac{-R_{hh}(0)}{R''_{hh}(0)} \right]^{\frac{1}{2}} \tag{7.49}$$

where $R_{hh}(\tau)$ is the spatial autocorrelation function of function $h(x)$, i.e:

$$R_{hh}(\tau) = \int_{-\infty}^{\infty} h(x)h(x + \tau)dx \tag{7.50}$$

Therefore:

$$R_{hh}(0) = \int_{-\infty}^{\infty} (h(x))^2 dx$$

Using Leibnitz's rule (see Box **B7.1**) we can differentiate (7.50) with respect to τ to obtain:

$$R'_{hh}(\tau) = \int_{-\infty}^{\infty} h(x)h'(x+\tau)dx$$

We define a new variable of integration $\tilde{x} \equiv x + \tau \Rightarrow x = \tilde{x} - \tau$ and $dx = d\tilde{x}$ to obtain:

$$R'_{hh}(\tau) = \int_{-\infty}^{\infty} h(\tilde{x} - \tau)h'(\tilde{x})d\tilde{x} \tag{7.51}$$

We differentiate (7.51) once more:

$$R''_{hh}(\tau) = -\int_{-\infty}^{\infty} h'(\tilde{x} - \tau)h'(\tilde{x})d\tilde{x} \Rightarrow R''_{hh}(0) = -\int_{-\infty}^{\infty} (h'(x))^2 dx$$

Therefore, the average distance of zero crossings of the output signal when a noise signal is filtered by function $h(x)$ is given by:

$$x_{av} = \pi\sqrt{\frac{\int_{-\infty}^{\infty}(h(x))^2dx}{\int_{-\infty}^{\infty}(h'(x))^2dx}} \tag{7.52}$$

From the analysis in Box **B7.5**, we can see that the false maxima in our case will come from equation $\int_{-\infty}^{\infty} f'(x)n(x_0 - x)dx = 0$ which will give the false alarms in the absence of any signal ($u(x) = 0$). This is equivalent to saying that the false maxima coincide with the zeroes in the output signal when the noise signal is filtered with function $f'(x)$. So, if we want to reduce the number of false local maxima, we should make the average distance between the zero crossings as large as possible, for the filter function $f'(x)$. Therefore, we define the good measure of scarcity of false alarms as:

$$C \equiv \sqrt{\frac{\int_{-\infty}^{\infty}(f'(x))^2dx}{\int_{-\infty}^{\infty}(f''(x))^2dx}}$$

What is the "take home" message of this chapter?

This chapter dealt with the reduction of the information content of an image so that it can be processed more easily by a computer vision system. It surveyed the two basic approaches for this purpose: region segmentation and edge detection. Region segmentation tries to identify spatially coherent sets of pixels that appear to constitute uniform patches in the image. These patches may represent surfaces or parts of surfaces of objects depicted in the image. Edge detection seeks to identify boundaries between such uniform patches. The most common approach for this is based on the estimate of the first derivative of the image. For images with low levels of noise the

Sobel masks are used to enhance the edges. For noisy images the Canny filters should be used. Canny filters can be approximated by the derivative of a Gaussian; i.e. they have the form $xe^{-\frac{x^2}{2\sigma^2}}$. Although parameter σ in this expression has to have a **specific** value for the filters to be optimal according to Canny's criteria, often people treat σ as a free parameter and experiment with various values of it. Care must be taken in this case when discretizing the filter and truncating the Gaussian, not to create a filter with sharp ends. Whether the Sobel masks are used or the derivatives of the Gaussian, the result is the enhancement of edges in the image. The output produced has to be further processed by non-maxima suppression (i.e. the identification of the local maxima in the output array) and thresholding (i.e. the retention of only the significant local maxima).

Edge detectors consist of all three stages described above and produce fragmented edges. Often a further step is involved of linking the fragments together to create closed contours that identify uniform regions. Alternatively, people may bypass this process by performing region segmentation directly and, if needed, extract the boundaries of the regions afterwards. Region-based methods are much more powerful when both attribute similarity and spatial proximity are taken into consideration when deciding which pixels form which region.