8. MACHINE VISION

8.1 Introduction

As stated in chapter 1, robotics is the study, design and development of robotic systems. Machine vision (also known as "computer vision" and "computational vision") is an important component of many robotic systems. Machine vision is being applied in a growing number of applications, including inspection of manufactured products, optical character recognition, medicine, security and biometrics. In this chapter we will study some of the basic techniques involved.

8.2 Definitions

An "image" is a digitized representation of a scene. Although 3D images exist (*e.g.* medical CAT scans) we will limit ourselves to the more common 2D images. The images are normally obtained using a digital video camera. This is known as "image acquisition". Note that the pixel values are always positive integers or zero. The small sections that make up the image are called "pixels" (derived from "picture cells"). There are three commonly used image formats:

- Binary:
  Intensity of light represented by 1 bit/pixel.
  A pixel value of 0 = black.
  A pixel value of 1 = white.
- Grayscale (or greyscale):
  Intensity of light represented by 8 bits/pixel.
  A pixel value of 0 = black.
  A pixel value of 255 = white.
  Pixel values from 1-254 represent different shades of gray.
- Colour:
  Colour of each pixel represented by 24 bits. (8 bits for the intensity of the Red component, another 8 for the Green component, and the remaining 8 for the Blue component (RGB)).

"Image processing" is the act of improving or altering the original image. Many image processing methods are available. For example the original image may be filtered to remove noise and clarify the image. Sometimes image processing is used alone, but it is normally followed by "image analysis". With "image analysis" information is extracted from the processed image. This information can include the number of objects in the scene, the identity of the objects, or various measurements. "Machine vision" refers to the combination of image acquisition, image processing, and image analysis.

8.3 Grayscale Image Processing Methods

8.3.1 Point Operations

Point operations alter the gray levels of the pixels in an image without regard to their row and column positions within the image. These operations are usually used to improve the contrast of the image. Linear point operations can be written in the general form:

$$g_{new} = Gg_{old} + b \tag{8.1}$$

where g_{old} is the original gray level for a given pixel, g_{new} is the new gray level for the given pixel, G is the "gain" and b is the "bias". G and b are scalar constants. This equation can be applied to the pixels of the image in any order. The linear point operation will affect the image as follows:

- G=1, b=0: No change in image.
- G>1, b=0: Increase in contrast.
- G<1, b=0: Decrease in contrast.
- G=1, b>0: Image made lighter.
- G=1, b<0: Image made darker.
- G= -1, b=255: Produces photographic negative of the image.

An example will be shown in class.

A histogram is a useful tool for studying and modifying the gray levels of an image. A histogram represents the total number of pixels at each gray level. An example of an image and its histogram is shown in Figure 8.1. The image is dark and lacks contrast. Correspondingly, the histogram shows that the gray levels of most of the pixels in the image are in the range 0-100.

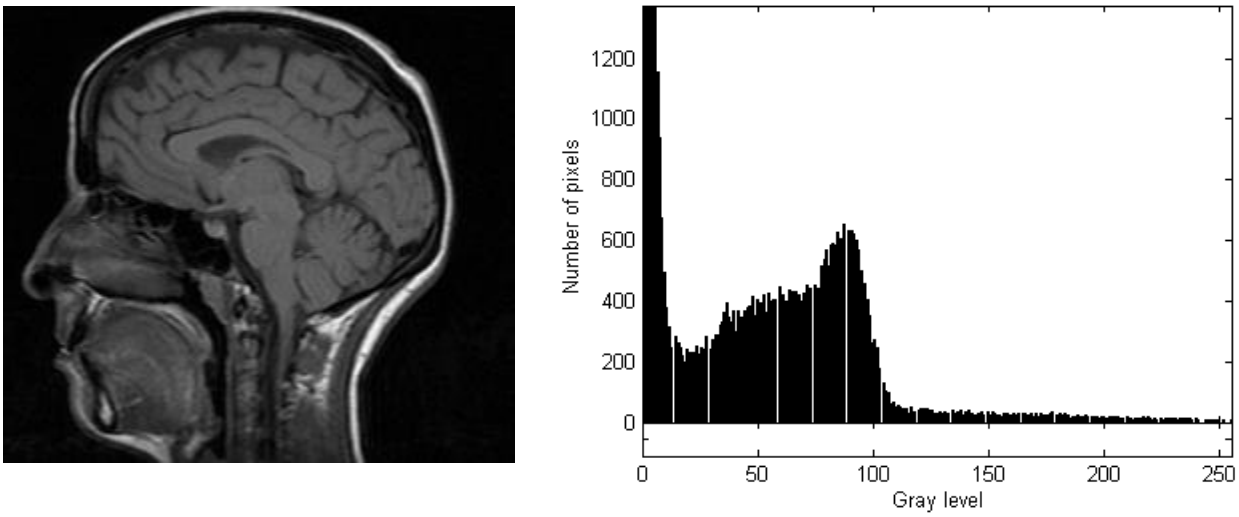


Figure 8.1 A gray scale image (left) and its histogram (right).

This situation could be improved by applying equation (8.1) with G>1 and b=0. This would spread out the histogram and increase the contrast. Another option is to use the standard technique known as "histogram equalization". Histogram equalization is a non-linear point operation that modifies the gray levels of the image such that the new histogram approximates a uniform distribution. This is usually more effective than a linear point operation at improving the contrast of the image. The results of applying histogram equalization to the image from Figure 8.1 are given in Figure 8.2.

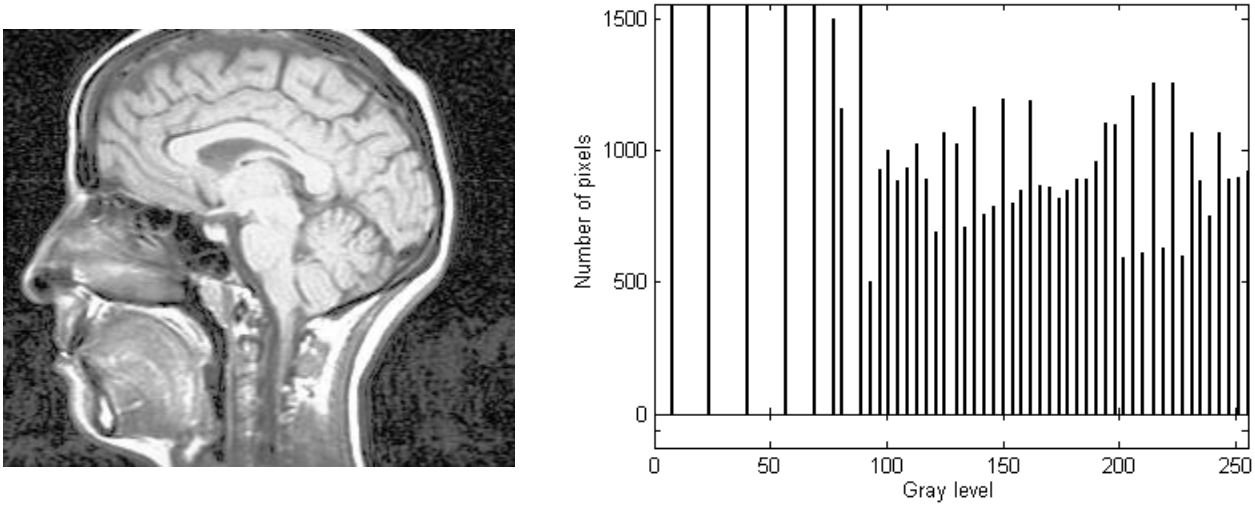


Figure 8.2 The results of applying histogram equalization.

Another point operation is termed "thresholding". This will be described in a later section.

8.3.2 Algebraic Operations

"Algebraic operations" operate on entire images. There are two common operations. With "image averaging" several images of the same scene are averaged together to obtain the output image. In equation form:

$$B = \frac{1}{N}\sum_{i=1}^{N} A_i \tag{8.2}$$

where B is the output image, A_i is an input image, and N is the number of input images. Assuming the scene and camera are fixed, this operation will reduce the effects of random noise (The random noise is typically caused by lighting fluctuations and electrical noise). An example is shown in Figure 8.3. The scene is from an optical character recognition application. The images have been enlarged so individual pixels may be clearly seen. Note that any fractional values from equation (8.2) must be rounded to integers.

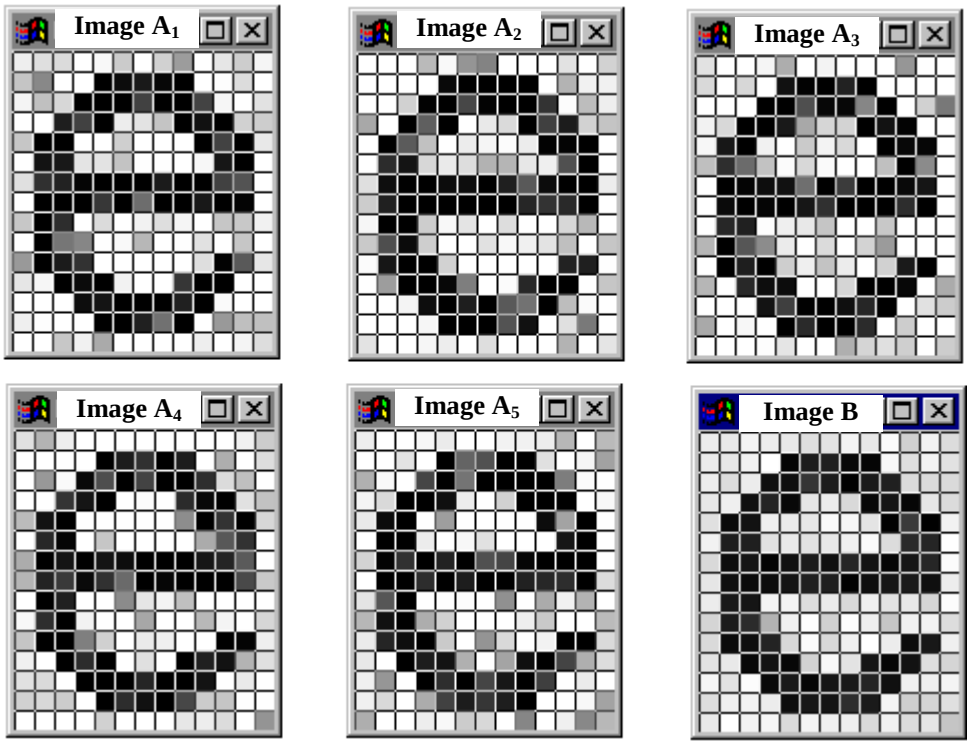


Figure 8.3 Example of five noisy input images and the averaged output image.

The second common algebraic operation is "image differencing" or "image subtraction". This may be written:

$$B = A_1 - A_0 \tag{8.3}$$

where B is the output image, A_1 is the input image, and A_0 is a reference image. Note that if negative values are produced the data must be forced into the range 0-255 by either using equation (8.1) or by taking their absolute values. If we use (8.1) for this purpose then G=0.5 and b=128 are suitable values. Equation (8.3) has several applications. If we wish to separate an object from the background then we can use (8.3) with an input image of the object and background together, and a reference image of the background alone. An example is shown in Figure 8.4. Another application is inspection for defects. In this case the reference image is of a defect-free part. The input image must have the same lighting and part location for this to be effective. An example is shown in Figure 8.5. If the images A_0 and A_1 are from a timed sequence then their

difference may be used to study object motion. An example is shown in Figure 8.6. These examples will be discussed further in class.

Figure 8.4 Example of using image subtraction to remove the background.

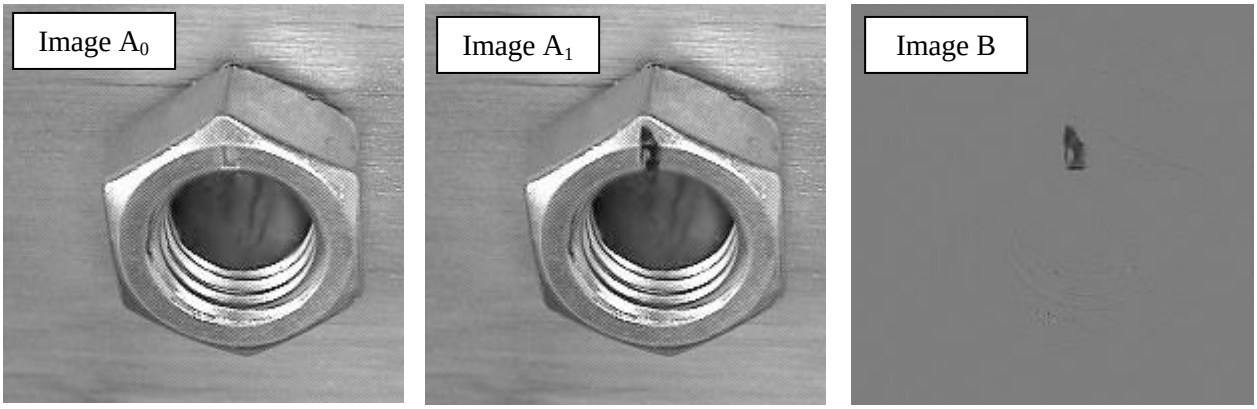


Figure 8.5 Example of using image subtraction to help locate defects.

Figure 8.6 Example of using image subtraction for detecting object motion.

8.3.2 Filtering Operations

Filtering may be used to improve the image quality or to enhance certain features. Before we discuss filtering, it is useful to further study the pixel levels of grayscale images. It is easier to study the pixel levels along a row or column since they can be plotted in 2D. An example is given in Figure 8.7. Examining the plot for column 25 and the image along this column, we can observe that sudden changes in the pixel levels occur at or near the edges of the object (e.g. row 35). If the pixel changes are interpreted as being plotted vs. time these rapid changes represent high frequency signals. Therefore the edges of objects in scenes occur at or near high frequency signals. So if we apply a high-pass filter to the image the edges it contains will be

enhanced. This idea is the basis for most edge detection techniques. This works well when the object surfaces have a smooth appearance as they do along column 25, and on the left side of the image. Examining the plot for column 120, the top and bottom edges of the object are near to rows 45 and 73, respectively. However, there are many sudden changes in the gray levels along the object's surface due to its rough appearance. This makes proper edge detection much more difficult (because there are many high frequency signals that do not correspond to edges). Similar difficulties are caused by noise in the image, since noise tends to be high frequency. Other types of filters may be used to help this situation.

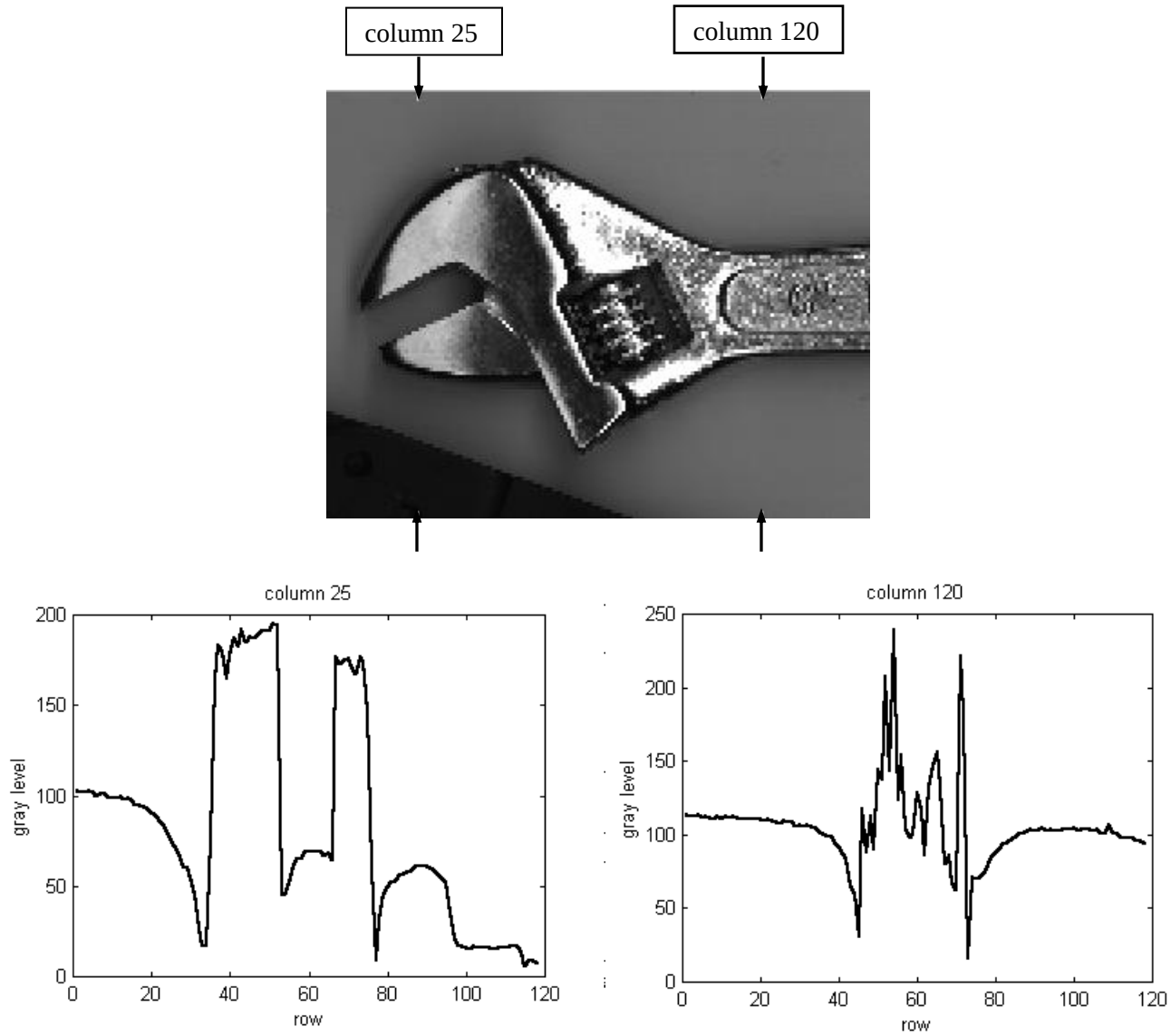


Figure 8.7 Plots of pixel gray levels vs. row for two columns of the image shown. (Column 25 on the left and column 120 on the right).

The most common method for filtering uses a "convolution mask". This mask is applied to each pixel in the input image to produce the output image. Since the mask includes neighbouring pixels the result depends on the row and column location of the pixel, unlike the point processing methods. Larger masks are more effective but require more computing power. The masks are usually square with dimensions of 3×3, 5×5 or 7×7. For example, a 3×3 mask can be written:

$$M = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix} \tag{8.4}$$

For an $N_{row} \times N_{col}$ image and an $n \times n$ mask, filtering is applied as follows:

for $i=(n+1)/2$ to $N_{row}-(n+1)/2+1$
  for $j=(n+1)/2$ to $N_{col}-(n+1)/2+1$

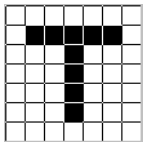
$$f_{ij} = \frac{1}{S}\left|\sum_{k=1}^{n}\sum_{p=1}^{n} m_{kp}\, a_{i+k-(n+1)/2,\, j+p-(n+1)/2}\right| \tag{8.5}$$

  next j
next i

where f_{ij} is the pixel value for the ith row and jth column of the filtered output image F; a are the pixels from the input image A; and $S = \left|\sum_{k=1}^{n}\sum_{p=1}^{n} m_{kp}\right|$ if the summation ≠ 0 and S = 1 if the summation = 0. Note that the first and last (n+1)/2−1 rows and columns of the image are left unchanged. Also note that taking the absolute value in procedure (8.5) is only necessary if some of the elements of M are negative. Some scaling may also be required to keep the filtering results in the grayscale range 0-255.

Example 8.1

The sample 7 × 7 image shown at the right can be written in the matrix form:



$$A = \begin{bmatrix} 255 & 255 & 255 & 255 & 255 & 255 & 255 \\ 255 & 0 & 0 & 0 & 0 & 0 & 255 \\ 255 & 255 & 255 & 0 & 255 & 255 & 255 \\ 255 & 255 & 255 & 0 & 255 & 255 & 255 \\ 255 & 255 & 255 & 0 & 255 & 255 & 255 \\ 255 & 255 & 255 & 0 & 255 & 255 & 255 \\ 255 & 255 & 255 & 255 & 255 & 255 & 255 \end{bmatrix} \tag{8.6}$$

Let's try applying the following filter mask:

$$M = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \tag{8.7}$$

For this mask:

$$S = \left|\sum_{k=1}^{n}\sum_{p=1}^{n} m_{kp}\right| = 16 \tag{8.8}$$

Since n=3, (n+1)/2=2, and we start procedure (8.5) at the (2,2) pixel. For this pixel:

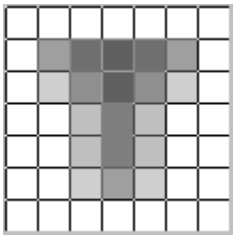
$$\begin{aligned} f_{22} &= \frac{1}{16}\left|\sum_{k=1}^{3}\sum_{p=1}^{3} m_{kp}\, a_{i+k-(n+1)/2,\, j+p-(n+1)/2}\right| \\ &= \frac{1}{16}\left|m_{11}a_{11} + m_{12}a_{12} + m_{13}a_{13} + m_{21}a_{21} + m_{22}a_{22} + m_{23}a_{23} + m_{31}a_{31} + m_{32}a_{32} + m_{33}a_{33}\right| = 159 \end{aligned} \tag{8.9}$$

Similarly, for the next pixel:

$$f_{23} = \frac{1}{16}\left|m_{11}a_{12} + m_{12}a_{13} + m_{13}a_{14} + m_{21}a_{22} + m_{22}a_{23} + m_{23}a_{24} + m_{31}a_{32} + m_{32}a_{33} + m_{33}a_{34}\right| = 111 \quad (8.10)$$

So why is M called a mask? Looking at equations (8.9) and (8.10) it becomes clear. The mask is always located with its centre element (in this example m_{22}) at the current pixel, and then the overlaid mask elements and pixels are multiplied and summed. Applying this to the rest of the image gives:

$$F = \begin{bmatrix} 255 & 255 & 255 & 255 & 255 & 255 & 255 \\ 255 & 159 & 111 & 95 & 111 & 159 & 255 \\ 255 & 207 & 143 & 95 & 143 & 207 & 255 \\ 255 & 255 & 191 & 127 & 191 & 255 & 255 \\ 255 & 255 & 191 & 127 & 191 & 255 & 255 \\ 255 & 255 & 207 & 159 & 207 & 255 & 255 \\ 255 & 255 & 255 & 255 & 255 & 255 & 255 \end{bmatrix} \quad \text{or} \quad \text{(8.11)}$$



The filter has reduced the sudden changes in the pixel levels and in doing so it has attenuated the high frequencies. Therefore it is a low-pass filter. Optically this makes the image softer or blurrier. This has the advantage that high frequency noise is reduced and the disadvantage that edges become less distinct. A mask of this type is also sometimes called "neighbourhood averaging".

End of example 8.1.

There are several standard filter masks. These include:

3×3 Gaussian filter (low-pass) $M = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$

3×3 Mean filter (low-pass) $M = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$

3×3 Laplacian1 filter (high-pass) $M = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$

3×3 Laplacian2 filter (high-pass) $M = \begin{bmatrix} 1 & -2 & 1 \\ -2 & 5 & -2 \\ 1 & -2 & 1 \end{bmatrix}$

3×3 Sobel edge enhancer (high-pass) $M_h = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$ and $M_v = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$

3×3 Prewitt edge enhancer (high-pass) $M_h = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$ and $M_v = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$

We used the Gaussian filter in example 8.1. The mean filter is another common low-pass filter. The Gaussian filter is better at reducing image noise if it has a Gaussian distribution. The Laplacian1 and Laplacian2 filters can be used to sharpen an image. The Laplacian2 sharpens more at the cost of higher noise amplification.

A different method for sharpening is known as the "unsharp filter". With this approach a sharpened image is obtained by subtracting a low-pass filtered image from the input image. A scaled version of this image difference is then added back to the original as follows:

$$F = A + c(A - F_{smooth}) \tag{8.12}$$

where A is the input image, F_{smooth} is the output image resulting from mean filtering, F is the output image after sharpening and c is a scalar constant between 0.2 and 0.7. A larger c value produces a greater sharpening effect. This sharpening method is less aggressive than the Laplacian filters.

The Sobel and Prewitt edge enhancers (also called "edge detectors") require further explanation. If desired, the M_h mask may be used alone to enhance horizontal edges, and the M_v mask may be used alone to enhance vertical edges. If all edges are to be enhanced then each mask should be used separately and the results summed, that is:

$$F = F_h + F_v \tag{8.13}$$

where F_h is the output image after horizontal edge enhancement, F_v is the output image after vertical edge enhancement, and F is the output image with all edges enhanced. The difference between the two operators is that the Sobel operator emphasizes the pixels closer to the centre of the mask. This gives more enhancement to both curved edges and individual noisy pixels.

Examples will be shown later on.

8.3.3 The Median Filter

The median filter is a different type of filter that deserves special mention. We know that real images will contain high-frequency noise that will corrupt the results of image processing and image analysis if we are not careful. The low-pass filters reduce this noise but also make the image blurry. If multiple images of a stationary scene are available, and we have the computing power, then image averaging is a better approach. Another superior method is the median filter. The output of a median filter is the median of the pixel values in the $n \times n$ neighbourhood of the each pixel in the input image. This reduces noise in the image without a significant blurring effect. However it also requires sorting the pixel values which makes it more time consuming than the other low-pass filters. An example is shown in Figure 8.8. In this example the input pixel value of 49 becomes 38 in the output, median filtered, image. Another example will be given later on.

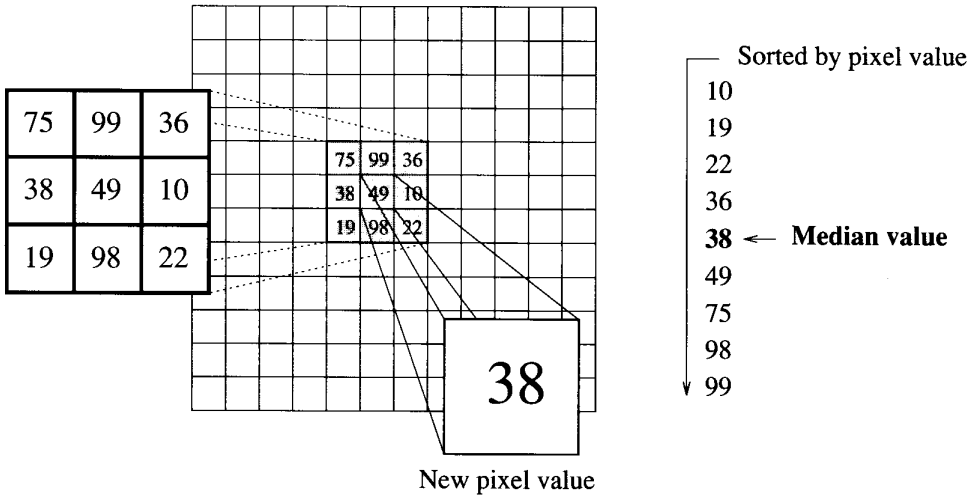


Figure 8.8 Example of applying a 3 × 3 median filter to one pixel in an image [1].

8.3.4 Edge Detection

Detecting the object edges contained in an image and producing a output binary image consisting of the edges and a blank background is called "edge detection". There are several reasons why edge detection is an important part of machine vision, including:

- The line drawing resulting from edge detection is easier than a regular image to store and further process (and analyse).
- The edges may be used to help separate the objects in the scene from each other and from the background. This problem is known as segmentation.
- The edges may be used to analyse the 3D shape of objects (*e.g.* an image of a cube) and object features such as holes.

Edge detection normally involves three sequential steps [1]:

1) Noise reduction.
2) Edge enhancement.
3) Detection.

The output image from each step becomes the input image for the next step. We've already looked at methods for noise reduction and edge enhancement, so only a discussion of the detection step is needed.

The goal of the detection step is to convert the edge enhanced grayscale image into a binary image of containing only the edges. The standard method for this conversion is to use a "thresholding" operation. Thresholding is a point operation that uses the grayscale levels of an image to divide it into different portions. In this course we will limit this to the two portion case (*i.e.* binary). The thresholding operation is then:

$$p_{new} = \begin{cases} 1 & \text{if } T_{lower} \le p_{old} \le T_{upper} \\ 0 & \text{otherwise} \end{cases} \tag{8.14}$$

where p_{new} is the binary output value, p_{old} is the grayscale input value, T_{lower} is the lower threshold

and T_{upper} is the upper threshold. This must be applied to every image pixel. The resulting 1 pixels are the detected edge pixels and the 0 pixels are the background pixels. The difficulty is selecting the thresholds such that only true edges and not noise is detected. These may be tuned manually (automated methods using the histogram also exist). An edge detection example will be presented in the next section.

8.3.5 Grayscale Image Processing Examples

An image of a gear, and its pixel values along row 30, are shown in Figure 8.9. The image is 120 × 100 pixels, so row 30 is 1/4 from the top of the image. This image was used as an input image for the filtering and edge detection methods discussed in the prior sections. The results of Gaussian filtering, mean filtering, median filtering, Laplacian1 filtering, Laplacian2 filtering, and unsharp filtering are shown in Figures 8.10-8.15, respectively. These results will be discussed in class.

The Prewitt and Sobel edge enhancers produced similar results for this input image as illustrated by Figures 8.16-8.17. The edge detection result shown in Figure 8.18 used median filtering for step 1, followed by the Sobel edge enhancer. Thresholding, with T_{lower} equal to half the maximum pixel level of the edge enhanced image, and T_{upper}=255, was used for the step 3. Note that a lower T_{lower} value would have detected more of the edge but also could have falsely detected noisy pixels.

8.4 Binary Image Processing Methods

Binary images require less memory for storage and can be processed faster that grayscale images. Normally the acquired image is grayscale and must first be converted to binary using the thresholding operation discussed previously (equation 8.14). This should be done such that the 1 pixels correspond to the object in the scene while the 0 pixels correspond to the background. This will be described further is section 8.5.1.

We will only cover three binary image processing methods, namely "expanding", "shrinking" and "size filtering" [1]. With "expanding" a pixel is changed from 0 to 1 if any of its four nearest neighbours are 1. The nearest neighbours are the adjacent pixels to the north, south, east and west. With "shrinking" a pixel is changed from 1 to 0 if any of its four nearest neighbours are 0. Performing expanding followed by shrinking helps to fill in holes but does not affect pixels due to noise. An example is shown in Figure 8.19b. Performing shrinking followed by expanding helps to eliminate isolated pixels caused by noise. An example is shown in Figure 8.19c.

Groups of connected 1 pixels are termed "blobs". With size filtering, blobs with less than or equal to T_S pixels are removed by setting all of their pixels equal to zero. This may be used to eliminate blobs due to noise. An example is shown in Figure 8.20. Note that if T_S is made too small then the noise will not be eliminated, but if it is made too large then some objects will be eliminated. For example if T_S=20 was used with the image from Figure 8.20 the dot on the "i" would have been filtered out.

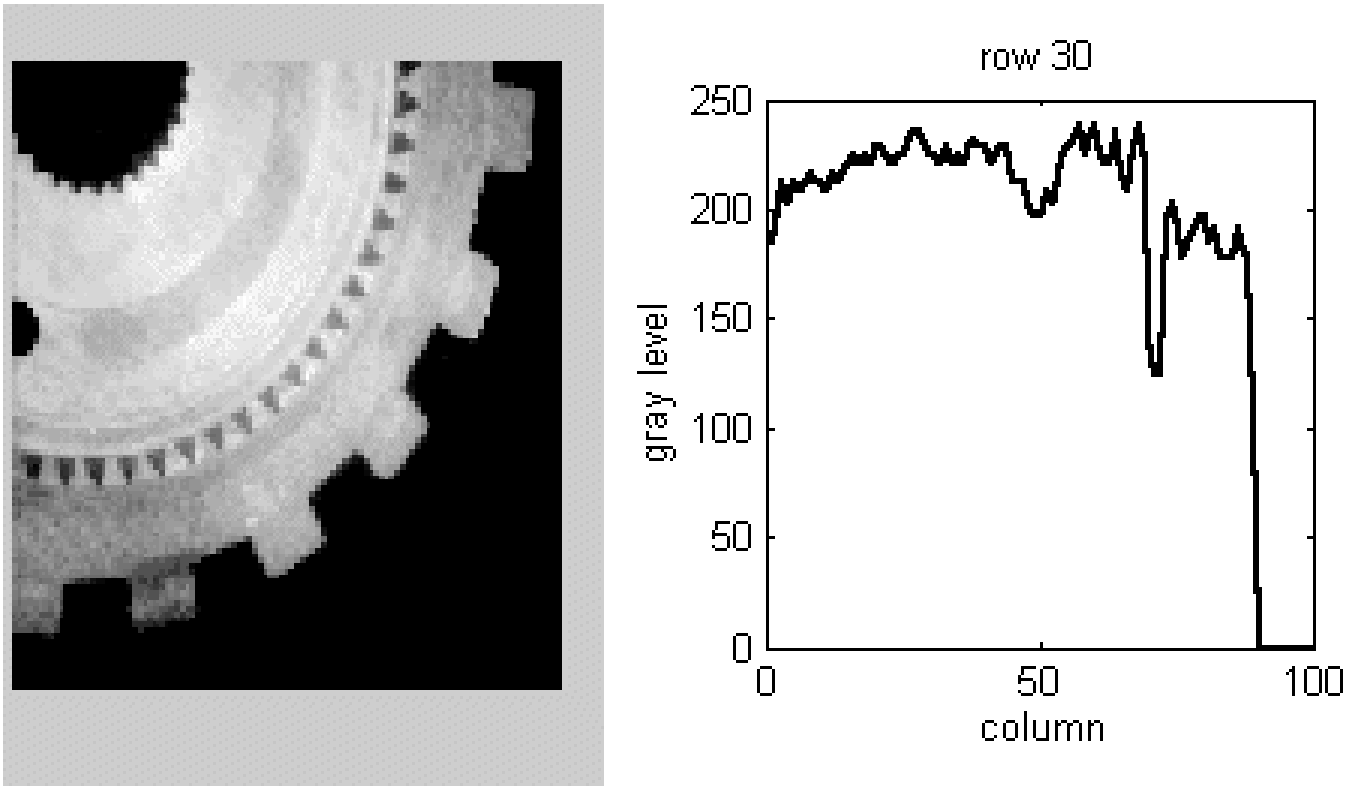


Figure 8.9 Left: Image of a gear against and a dark background. Right: The pixel levels along row 30 of the image.

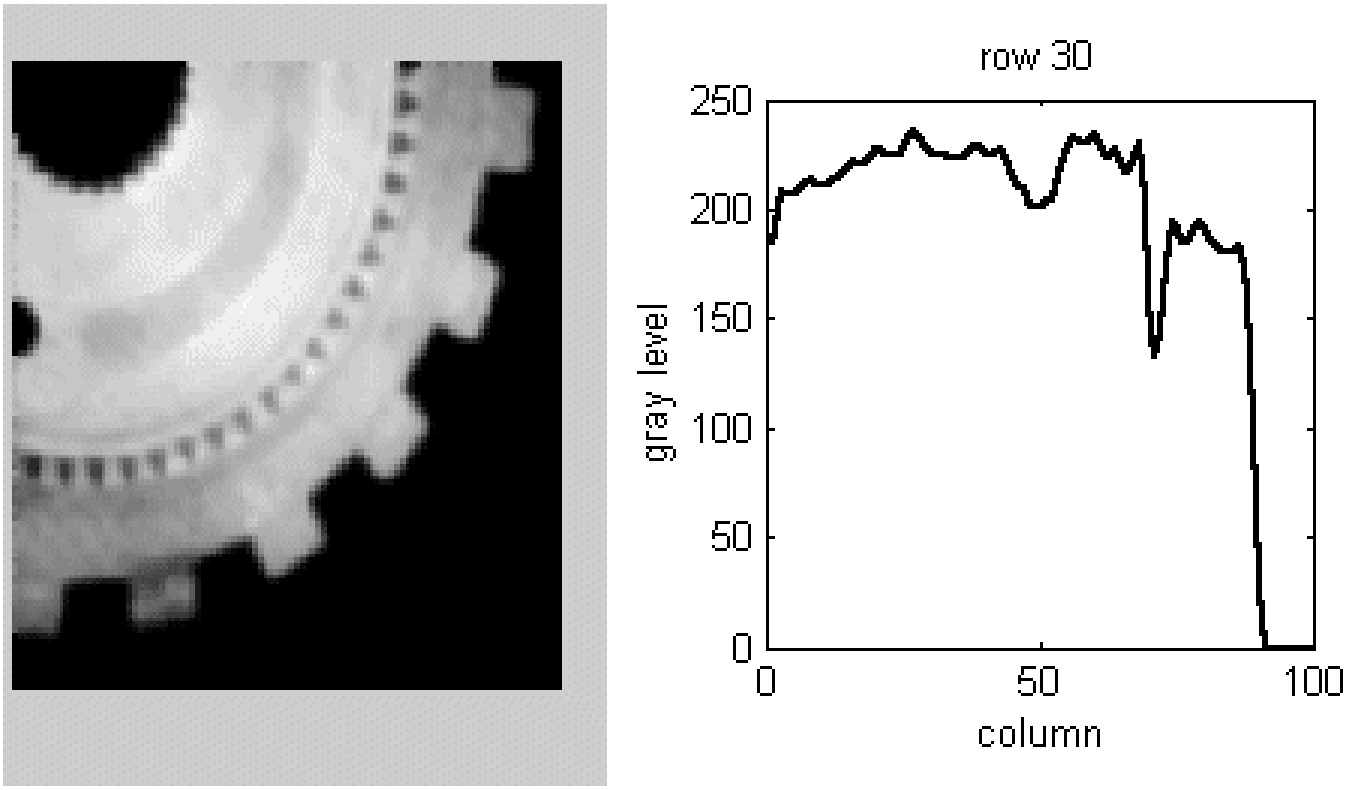


Figure 8.10 Gaussian filtering result.

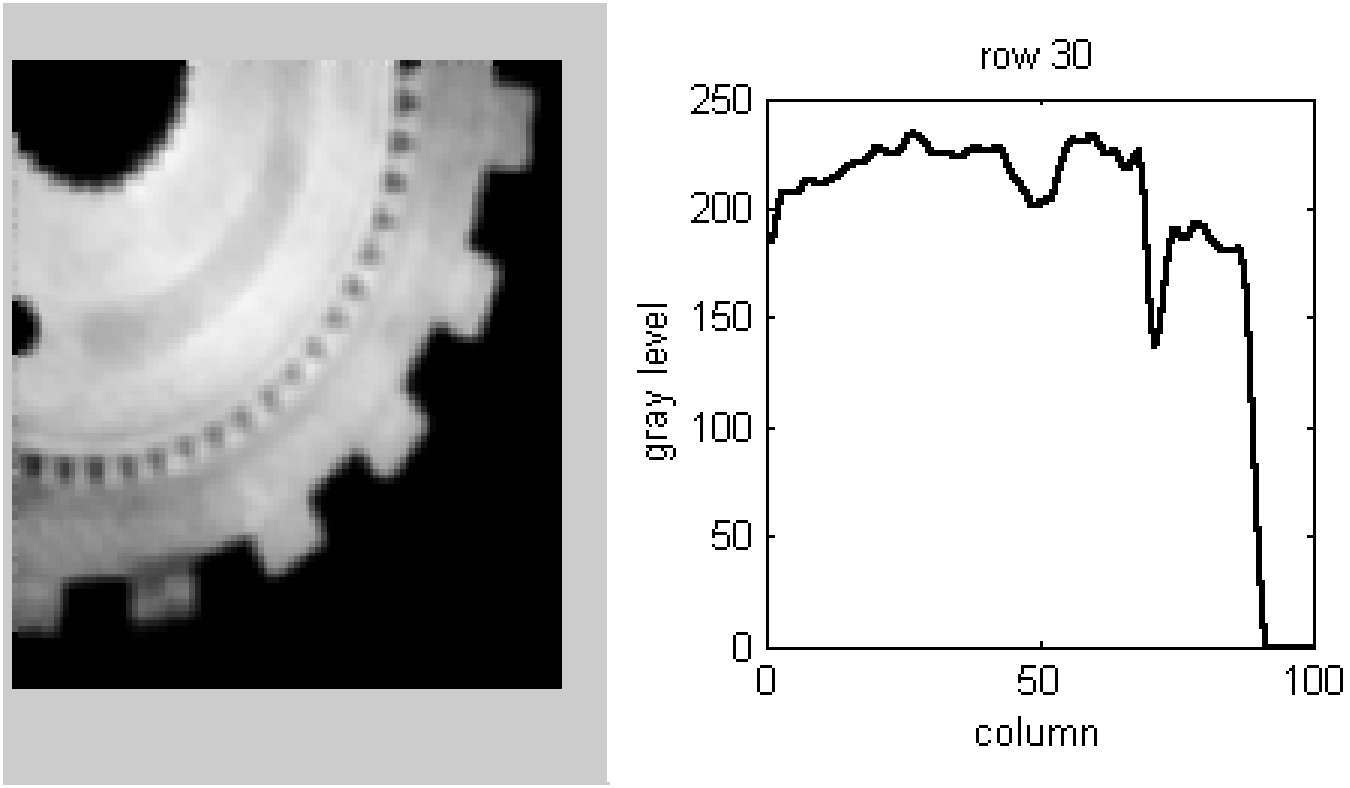


Figure 8.11 Mean filtering result.

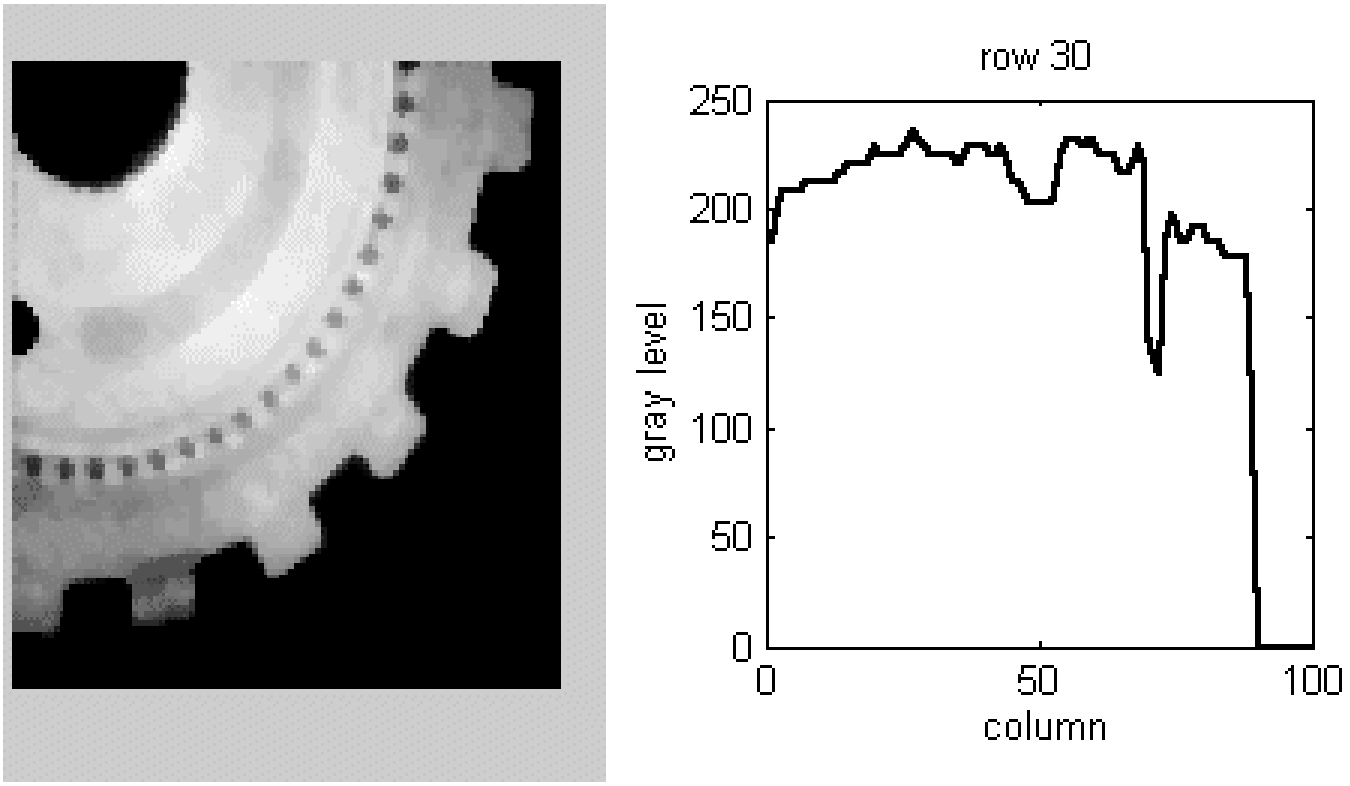


Figure 8.12 Median filtering result.

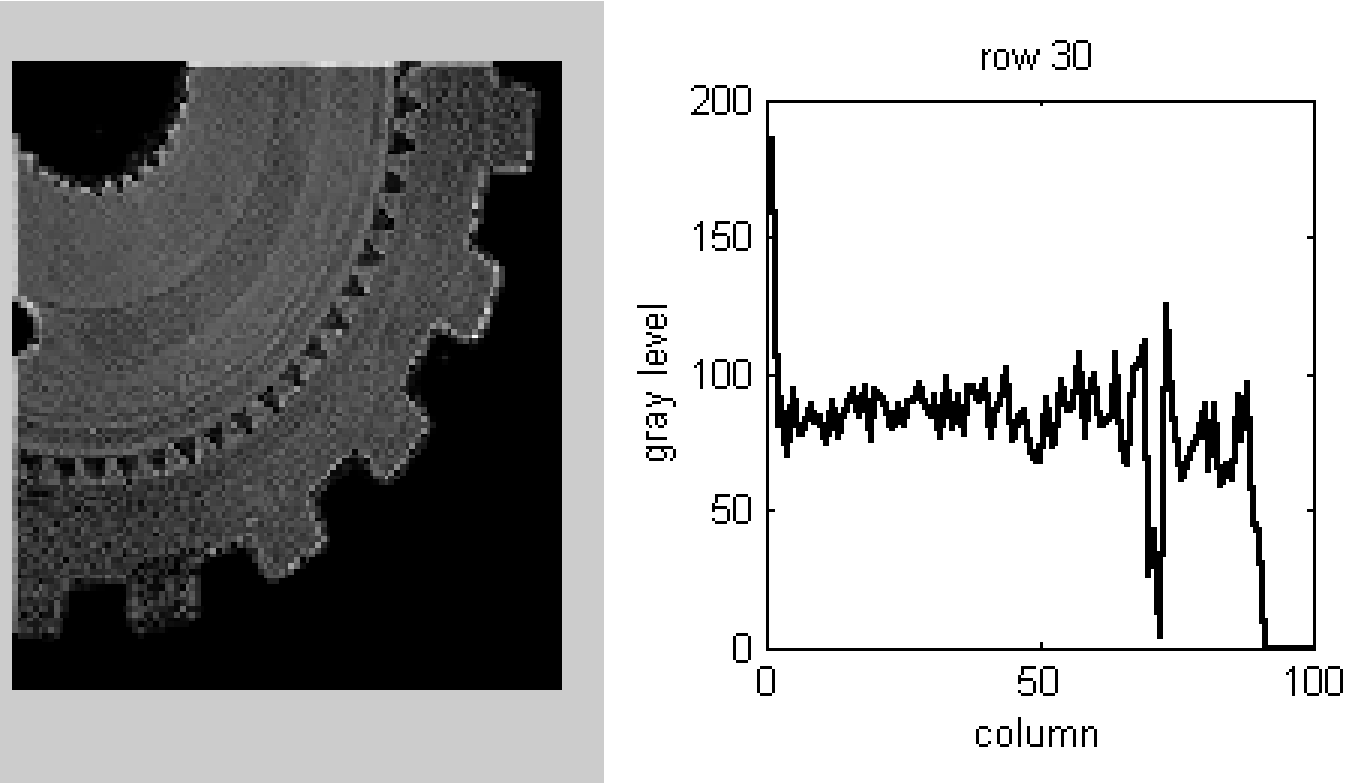


Figure 8.13 Laplacian1 filtering result.

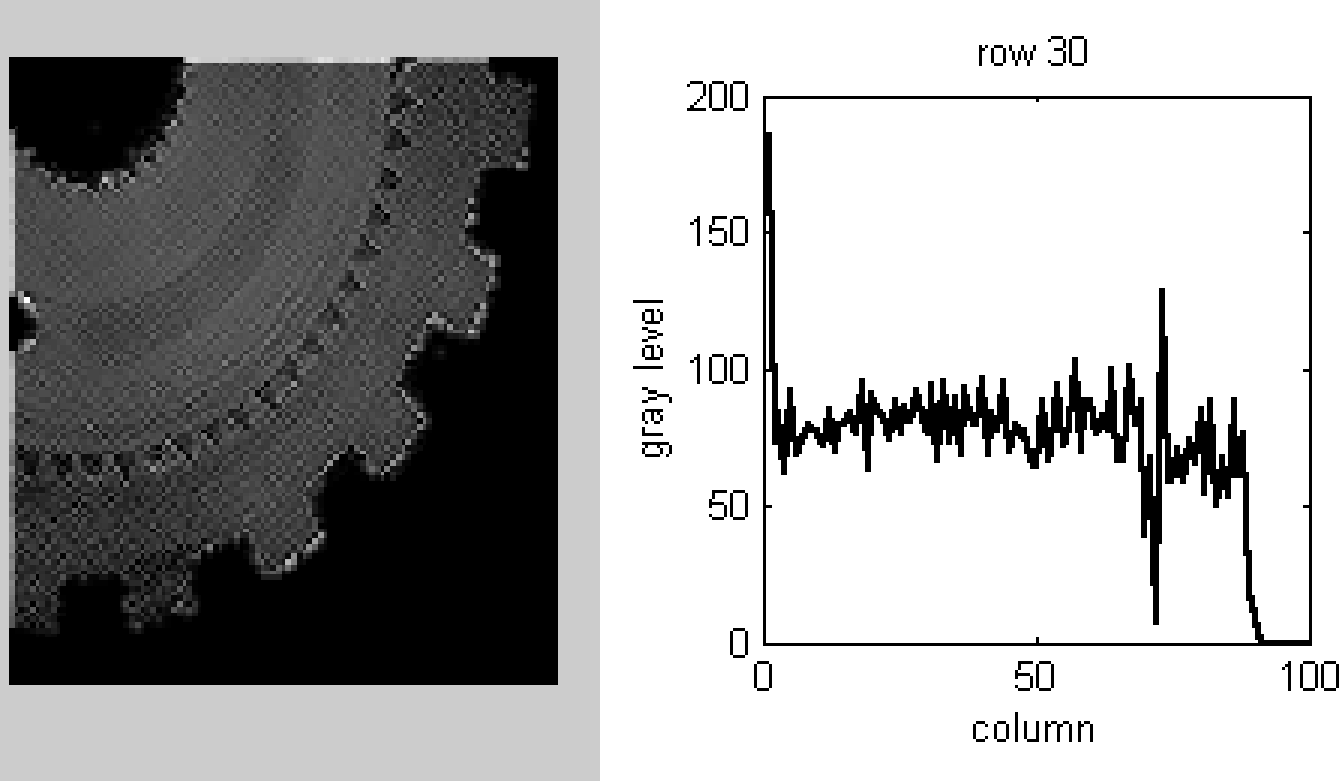


Figure 8.14 Laplacian2 filtering result.

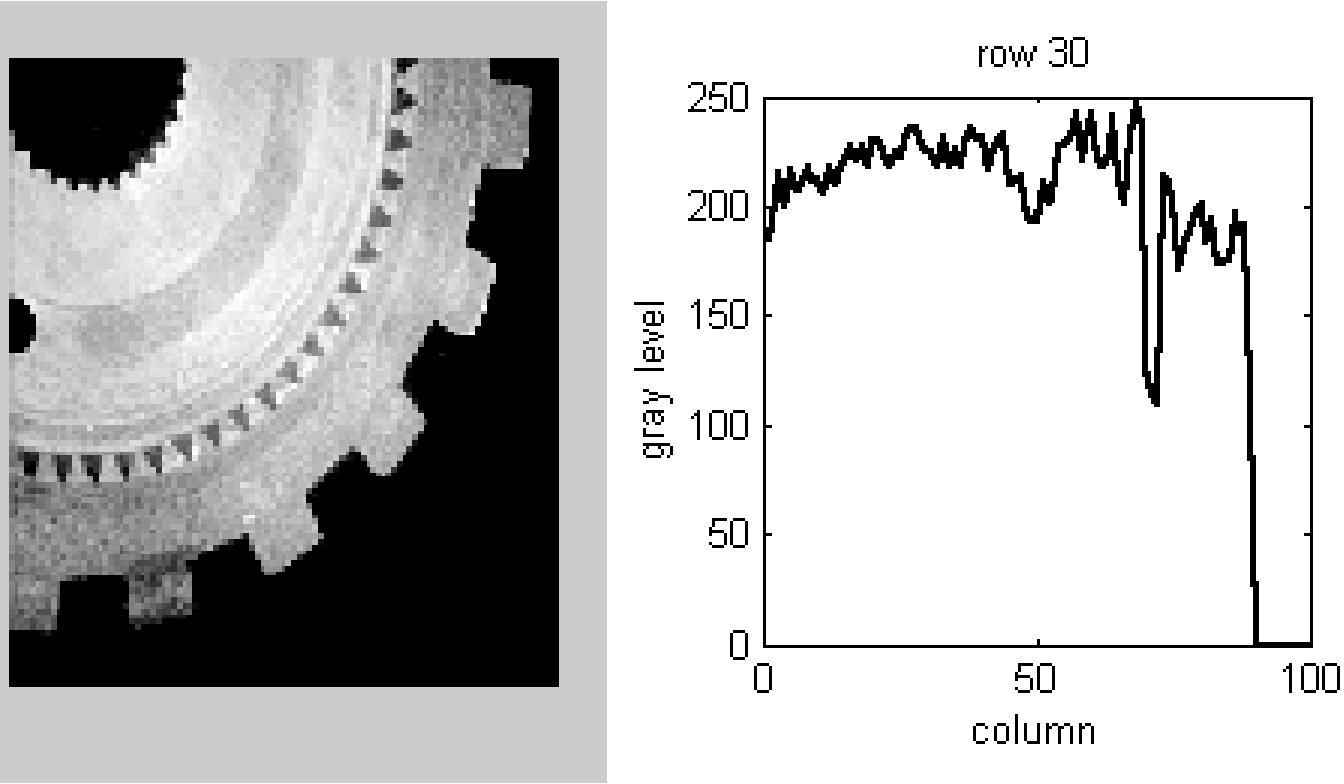


Figure 8.15 Unsharp filtering result using c=0.7.

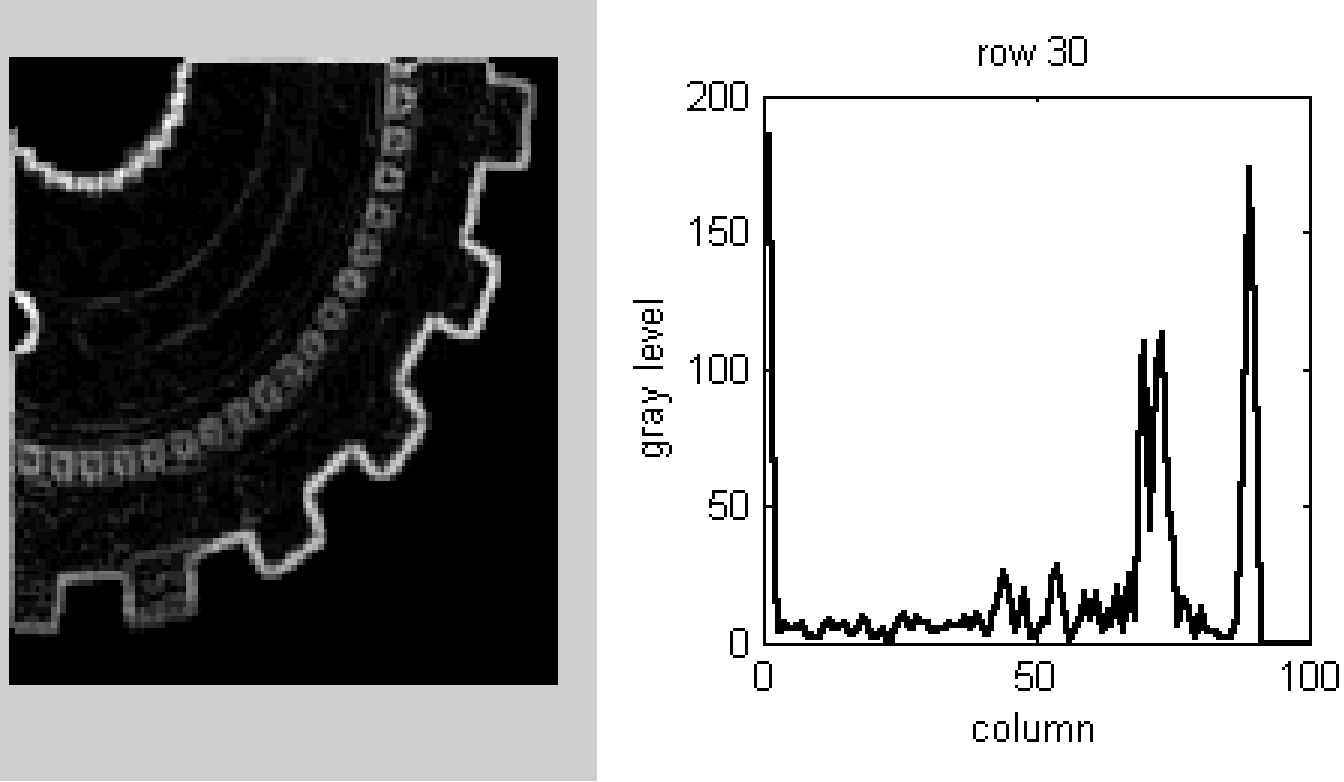


Figure 8.16 Prewitt edge enhancing result.

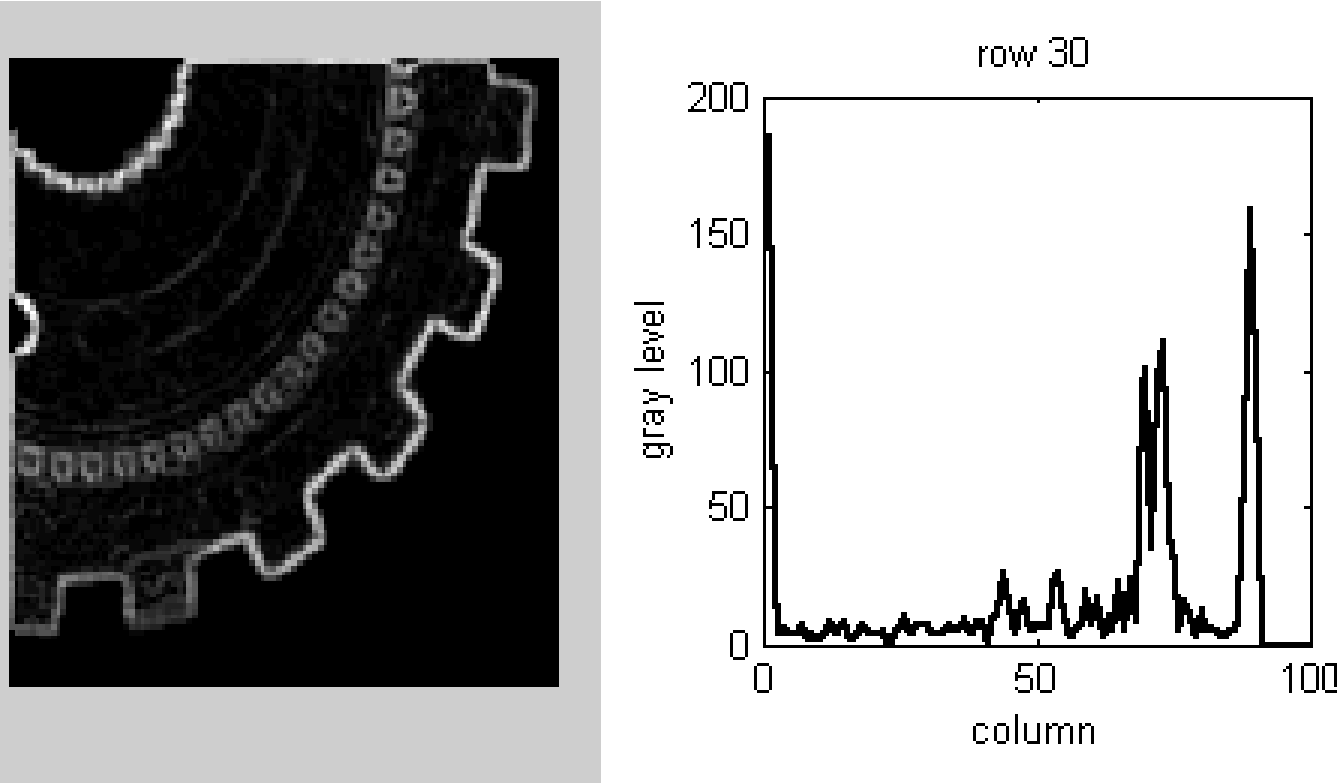


Figure 8.17 Sobel edge enhancing result.

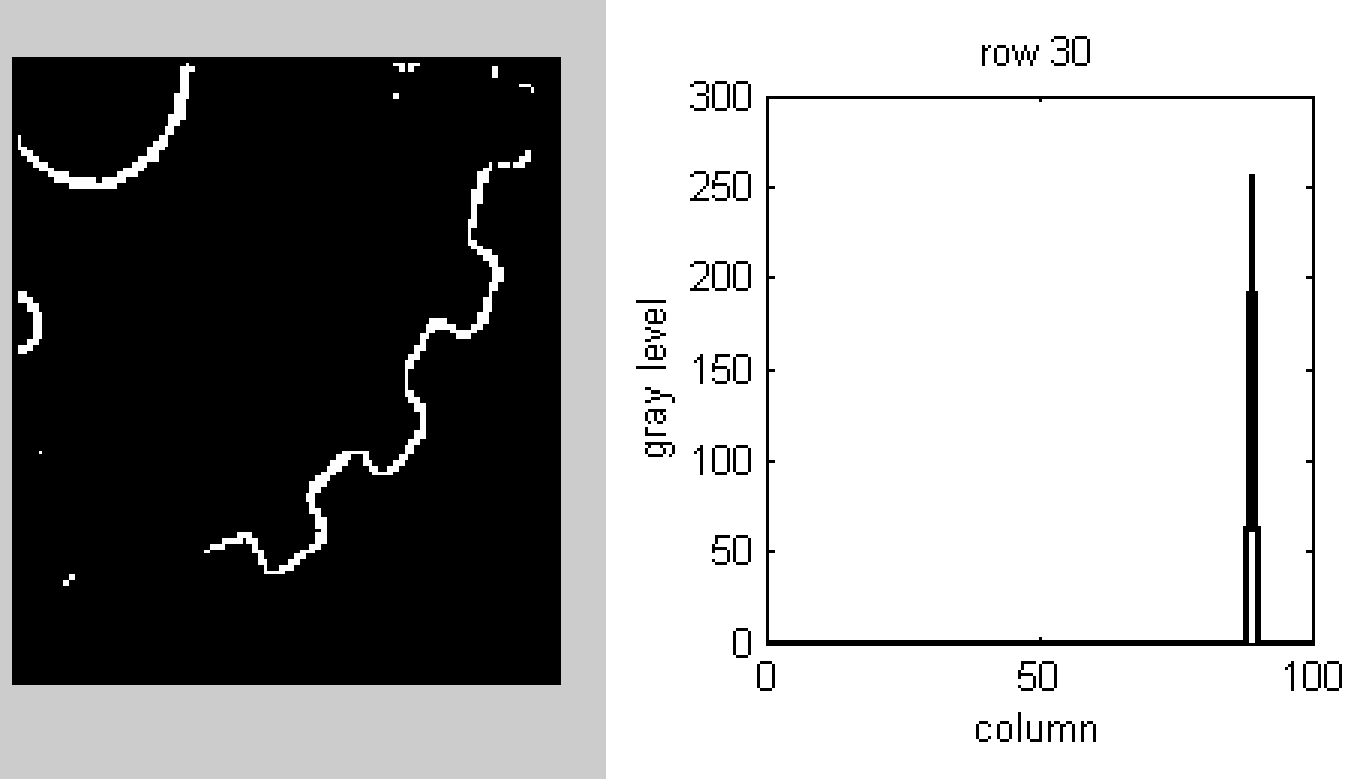


Figure 8.18 Edge detection result. Note that the pixel values were scaled by 255 for display purposes.

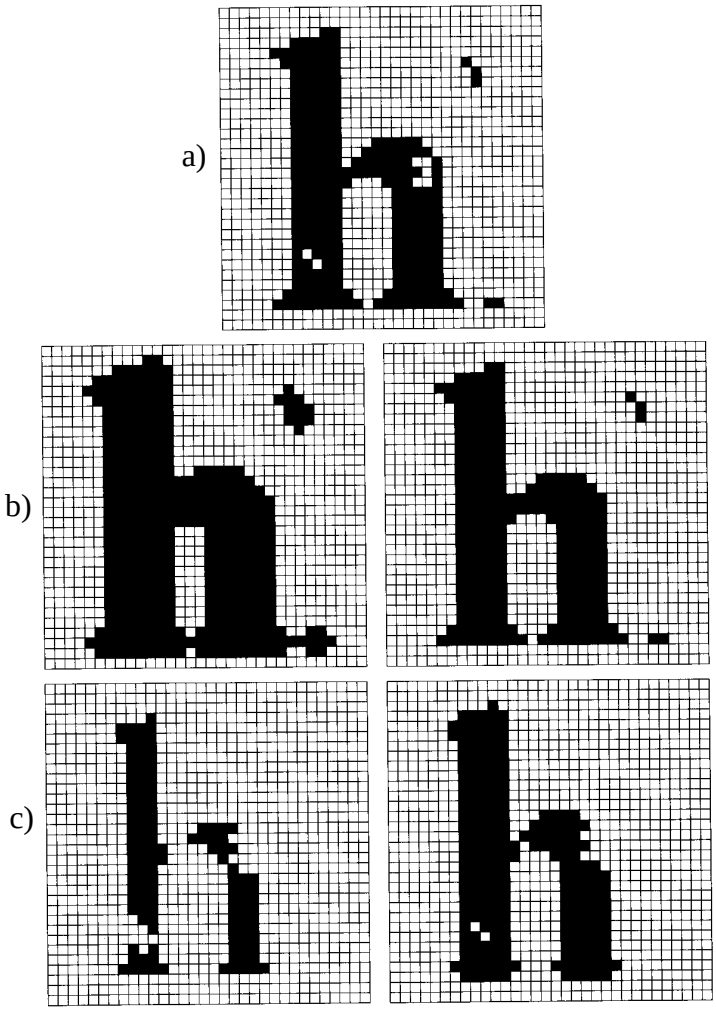


Figure 8.19 (a) Noisy binary image from an optical character recognition application. Note: 1 pixels are shown as black and 0 pixels as white. (b) The results from expanding followed by shrinking. (c) The results from shrinking followed by expanding [1].

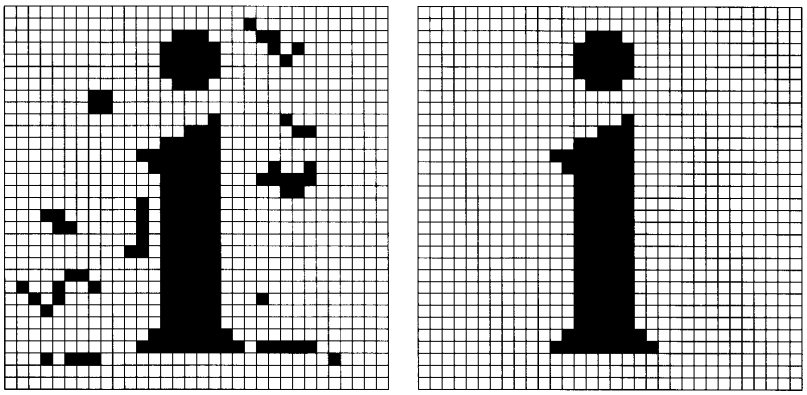


Figure 8.20 Left: Noisy binary image from an optical character recognition application. Right: The result of size filtering with T_S=10. Note: 1 pixels are shown as black and 0 pixels as white.

8.5 Image Analysis Methods

8.5.1 Segmentation

The goal of image analysis is to extract useful information from the processed image. The first crucial step is termed segmentation. With segmentation we are primarily interested in separating the objects in the scene from each other and from the background. We may also want to segment specific object features, such as edges. In fact we have already covered a method for edge detection and this may be classified as a segmentation technique (for separating the edges from the rest of the image).

Since industrial machine vision applications usually involve scenes with a single object, and controlled lighting, we will concentrate on this case. Since the scene has only two portions, the object and the background, a binary image will be the end result of segmentation. If the range of grayscale values for the object is distinct from the range for the background then the computationally simplest segmentation method is the thresholding operation (equation 8.14). If the object is lighter than the background then T_{upper} should be set equal to 255 and T_{lower} set equal to the darkest object pixel. If the object is darker than the background then T_{upper} should be set equal to the lightest object pixel and T_{lower} set equal to 0. If thresholding cannot be used, a second method based on edge detection is an alternative. This only requires that the pixel levels at the outer edges of the object are distinct from their neighbouring background pixels. To segment the image using this edge-based method:

1) Apply edge detection to the image.
2) Remove noise from the binary image if necessary (see section 8.4).
3) Find the outer edge of the object by:
   i) Scanning the image for the first 1 pixel.
   ii) Tracing the edge by following the connected 1 pixels (see Figure 8.34 in Niku [2]).
4) Fill in the inside of the object with 1 pixels.

If neither segmentation method is successful other more sophisticated methods (including those described in Chapter 8 of Niku [2]) are available.

8.5.2 Dimensional Measurements

After the pixels belonging to the object are identified, dimensional measurements may be performed. We will assume that the system has been calibrated such that the conversion from pixels to mm is known. Since the position and orientation of the object are normally not known in advance, it is best if the measurements are not affected by position/orientation changes. The simplest measurement is the area in pixel^2:

$$\text{object area} = \sum_{i=1}^{N_{row}} \sum_{j=1}^{N_{col}} p_{ij} \tag{8.15}$$

Another simple measurement is the length of the perimeter. This is obtained by counting the pixels along the outside edge of the object (see Figure 8.34 in Niku [2]). If the shape but not the size of the object is known, the area can be used to calculate specific dimensions. For example, if the object is circular then:

$$\text{object radius} = \sqrt{\frac{\text{area}}{\pi}} \text{ pixels} \tag{8.16}$$

8.5.3 Locating Objects

Some applications require the location of the object in the image to be identified. Two common reasons are to allow more dimensional measurements to be performed, and for guiding a robot to pickup the object. The simplest approach is based on moments. An XY coordinate frame is setup such that the row number of the pixel is its Y coordinate and the column number is its X coordinate. We now need a measure of the object position. The centre of mass is a good choice since it is not affected by orientation. Assuming the object pixels all have unit mass, we have:

$$\bar{x} = \frac{1}{\text{area}} \sum_{i=1}^{N_{row}} \sum_{j=1}^{N_{col}} j p_{ij} \tag{8.16}$$

$$\bar{y} = \frac{1}{\text{area}} \sum_{i=1}^{N_{row}} \sum_{j=1}^{N_{col}} i p_{ij} \tag{8.17}$$

where $\bar{x}$ and $\bar{y}$ are the coordinates of the centre of mass in pixels. Please see Figure 8.21. The major axis of the object may then be used to define the orientation of the object. Its angle is given by:

$$\theta = \tfrac{1}{2} \text{atan2}(2M_{xy},\ M_{xx} - M_{yy}) \tag{8.18}$$

where

$$M_{xx} = \frac{1}{\text{area}} \sum_{i=1}^{N_{row}} \sum_{j=1}^{N_{col}} (j - \bar{x})^2 p_{ij} \tag{8.19}$$

$$M_{yy} = \frac{1}{\text{area}} \sum_{i=1}^{N_{row}} \sum_{j=1}^{N_{col}} (i - \bar{y})^2 p_{ij} \tag{8.20}$$

$$M_{xy} = \frac{1}{\text{area}} \sum_{i=1}^{N_{row}} \sum_{j=1}^{N_{col}} (j - \bar{x})(i - \bar{y}) p_{ij} \tag{8.21}$$

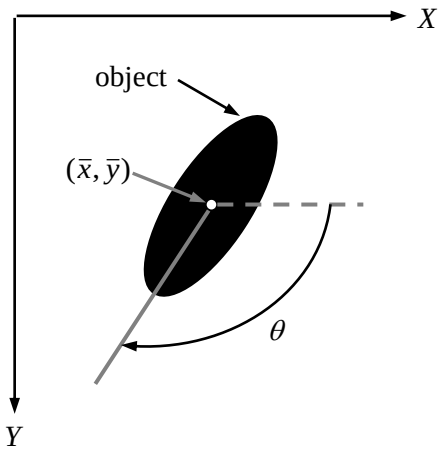


Figure 8.21 Object position and orientation geometry.

The measured position and orientation is then $(\bar{x}, \bar{y}, \theta)$. Note that symmetric objects such as the one shown in Figure 8.21 have two valid solutions for the orientation, θ and $\theta \pm 180°$.

With the location of the object known, more dimensional measurements can be taken. For example the object's dimensions along the major axis (its "length") and minor axis (its "width") may be easily determined.

Example 8.2

Given the 5 × 6 binary image containing a single object:

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{8.22}$$

Determine the position and orientation of the object.

Solution:

$$\text{object area} = \sum_{i=1}^{N_{row}} \sum_{j=1}^{N_{col}} p_{ij} = \sum_{i=1}^{5} \sum_{j=1}^{6} p_{ij} = 8\,\text{pixel}^2$$

$$\bar{x} = \frac{1}{\text{area}} \sum_{i=1}^{N_{row}} \sum_{j=1}^{N_{col}} j p_{ij} = \frac{1}{8} \sum_{i=1}^{5} \sum_{j=1}^{6} j p_{ij} = \frac{1}{8}(4+5+3+4+5+3+4+2) = 3.75\,\text{pixel}$$

$$\bar{y} = \frac{1}{\text{area}} \sum_{i=1}^{N_{row}} \sum_{j=1}^{N_{col}} i p_{ij} = \frac{1}{8} \sum_{i=1}^{5} \sum_{j=1}^{6} i p_{ij} = \frac{1}{8}(2+2+3+3+3+4+4+5) = 3.25\,\text{pixel}$$

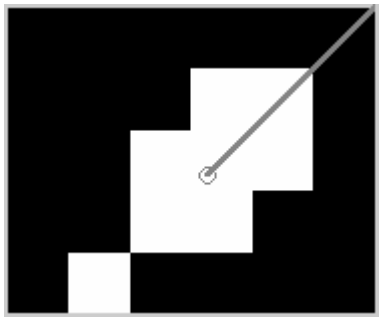
$$M_{xx} = \frac{1}{\text{area}} \sum_{i=1}^{N_{row}} \sum_{j=1}^{N_{col}} (j-\bar{x})^2 p_{ij} = 0.938$$

$$M_{yy} = \frac{1}{\text{area}} \sum_{i=1}^{N_{row}} \sum_{j=1}^{N_{col}} (i-\bar{y})^2 p_{ij} = 0.938$$

$$M_{xy} = \frac{1}{\text{area}} \sum_{i=1}^{N_{row}} \sum_{j=1}^{N_{col}} (j-\bar{x})(i-\bar{y}) p_{ij} = -0.688$$

$$\theta = \tfrac{1}{2}\,\text{atan2}(2M_{xy},\, M_{xx} - M_{yy}) = \tfrac{1}{2}\,\text{atan2}(2(-0.688),\, 0.938 - 0.938) = -45°$$

This is shown graphically below:



8.5.4 Object Identification (also known as object recognition)

Any properties of the image of the object that make it distinct may be used to identify the object. These include:

- Its average, maximum or minimum grayscale pixel levels.
- Its histogram.
- Its area, perimeter, length and width. Ratios or other functions of these values may also be useful.

Many other approaches may also be used (please see section 8.25 in [2] and chapter 15 in [1] if you're interested).

References

1. R. Jain, R. Kasturi and B.G. Schunck, "Machine Vision", McGraw-Hill, 1995.
2. S.B. Niku, "Introduction to Robotics", Pearson Education, 2001.