

Computer Vision CITS4240

School of Computer Science & Software Engineering
The University of Western Australia

Edge Detection

Classical Feature Detection

It seems clear, both from biological and computational evidence, that some form of data compression occurs at a very early stage in image processing. Moreover, there is much physiological evidence suggesting that one form of this compression involves finding edges and other information-high features in images. Edges often occur at points where there is a large variation in the luminance values in an image, and consequently they often indicate the edges, or occluding boundaries, of the objects in a scene. However, large luminance changes can also correspond to surface markings on objects. Points of tangent discontinuity in the luminance signal (rather than simple discontinuity) can also signal an object boundary in the scene.

So the first problem encountered with modelling this biological process is that of defining, precisely, what an *edge* might be. The usual approach is to simply define edges as step discontinuities in the image signal. The method of localising these discontinuities often then becomes one of finding local maxima in the derivative of the signal, or zero-crossings in the second derivative of the signal. This idea was first suggested to the AI community, both biologically and computationally, by Marr [5], and later developed by Marr and Hildreth [6], Canny [1, 2], and many others [3, 4].

In computer vision, edge detection is traditionally implemented by *convolving* the signal with some form of linear filter, usually a filter that approximates a first or second derivative operator. An odd symmetric filter will approximate a first derivative, and peaks in the convolution output will correspond to edges (luminance discontinuities) in the image. An even symmetric filter will approximate a second derivative operator. Zero-crossings in the output of convolution with an even symmetric filter will correspond to edges; maxima in the output of this operator will correspond to tangent discontinuities, often referred to as *bars*, or *lines*.

In this lecture we will begin by considering the classical approaches to detecting edges in images: first derivative techniques, second derivative techniques, and surface fitting. We will then consider how features other than edges might be detected.

First order differential methods of edge detection

Most edge detection methods work on the assumption that an edge occurs where there is a discontinuity in the intensity function or a very steep intensity gradient in the image (see Figure 1). Using this assumption, if we take the derivative of the intensity values across the image and find points where the derivative is a maximum, we will have marked our edges. In a discrete image of pixels we can calculate the gradient by simply taking

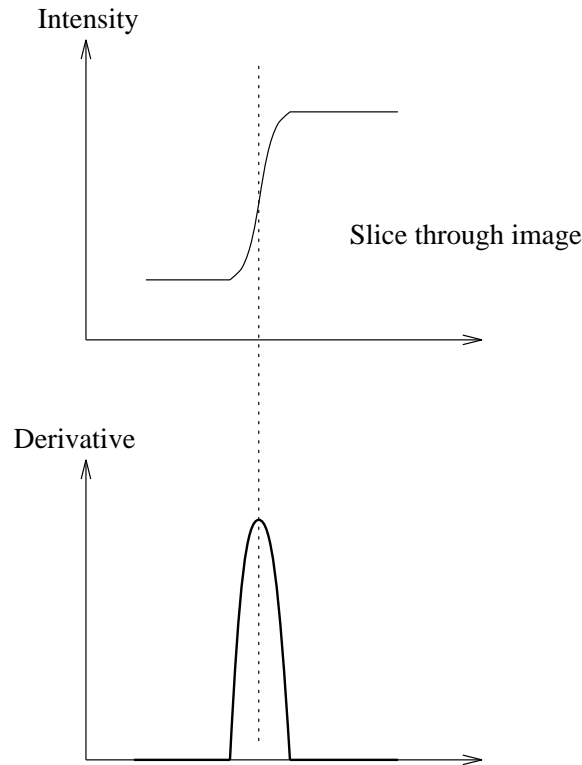


Figure 1: An ideal step edge and its derivative profile.

the difference of grey values between adjacent pixels (see Figure 2). This is equivalent to convolving the image with the mask $[-1,1]$. Note that we have a problem in determining where we place the result of the convolution. Ideally we would like to place it between the two pixels. By using a mask that spans an odd number of pixels we end up with a middle pixel to which we can assign the convolution result.

The *gradient* of the image function I is given by the vector

$$\nabla I = \begin{bmatrix} \frac{\partial I}{\partial x} \\ \frac{\partial I}{\partial y} \end{bmatrix}.$$

The *magnitude* of this gradient is given by $\sqrt{\left(\frac{\partial I}{\partial x}\right)^2 + \left(\frac{\partial I}{\partial y}\right)^2}$ and its *direction* by $\tan^{-1}\left(\frac{\partial I}{\partial y} / \frac{\partial I}{\partial x}\right)$. Note, one can use any pair of orthogonal directions to compute this gradient, although it is common to use the x and y directions.

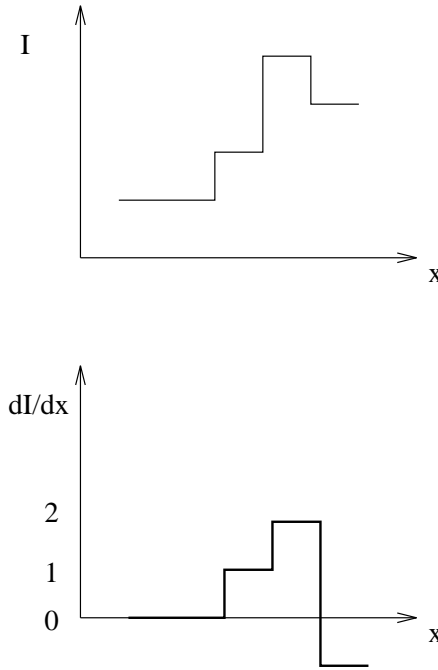


Figure 2: A discrete intensity profile and its derivative profile.

The simplest gradient operator is the *Robert's Cross operator* and it uses the masks

$$\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}.$$

Thus the Robert's Cross operator uses the diagonal directions to calculate the gradient vector.

A 3×3 approximation to $\frac{\partial I}{\partial x}$ is given by the convolution mask

-1	0	1
-1	0	1
-1	0	1

This defines $\frac{\partial I}{\partial x}$ for the *Prewitt* operator, and it detects vertical edges. The *Sobel* operator is a variation on this theme giving more emphasis to the centre cell. The Sobel approximation to $\frac{\partial I}{\partial x}$ is given by

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

Similar masks are constructed to approximate $\frac{\partial I}{\partial y}$, thus detecting the horizontal component of any edges. Both the Prewitt and Sobel edge detection algorithms convolve with masks to detect both the horizontal and vertical edge components of the gradient map. The magnitude of the gradient map is calculated to find the points of high gradient.

This magnitude array will have large values where the image gradient is large, but that is not sufficient to localise the edges. The broad ridges in the magnitude array must be thinned so that only the magnitudes at the points of greatest local change remain. This is known as *non-maxima suppression*, and it results in thinned edges.

Non-maxima suppression thins the ridges of the gradient magnitude by suppressing all values along the line of the gradient that are not peak values of the ridge. A simple non-maxima suppression algorithm begins by reducing the angle of the gradient to one of four sectors: $[0 \pm 22.5^\circ]$, $[45^\circ \pm 22.5^\circ]$, $[90^\circ \pm 22.5^\circ]$ and $[135^\circ \pm 22.5^\circ]$, all modulo π .

The algorithm passes a 3×3 neighbourhood across the magnitude array $M[i, j]$. At each point, the centre element $M[i, j]$ of the neighbourhood is compared with its two neighbours along the line of the gradient given by the sector value at the centre of the neighbourhood. If the magnitude array value $M[i, j]$ at the centre is not greater than both of the neighbour magnitudes along the gradient line then $M[i, j]$ is set to zero. This process thins the broad ridges of gradient magnitude into ridges that are only one pixel wide. The values for the height of the ridge are retained.

The resulting map of local maxima is then thresholded (small local maxima will result from noise in the signal) to produce the final edge map. It is the non-maxima suppression and thresholding that introduce non-linearities into this edge detection scheme. Moreover, the output of the thresholding stage is extremely sensitive and there are no automatic procedures for satisfactorily determining thresholds that work for all images. This is evident from the edge maps of the mandrill image (Figure 3) shown in Figure 4. Another example is shown in Figure 5.

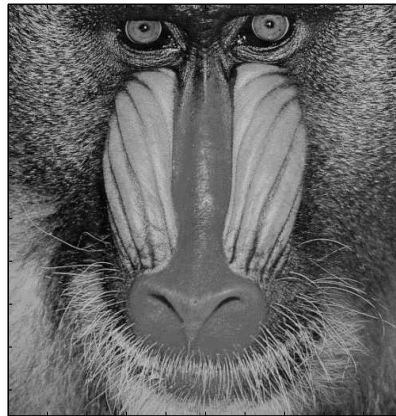


Figure 3: The mandrill image.

The main problem one has to deal with in differential edge detection schemes is noise. The spikes in the derivative from the noise can mask the real maxima that indicate edges. If we smooth the image then the effects of noise can be reduced (see Figure 6). A procedure we could use is

1. convolve the image with a Gaussian mask to smooth it, then
2. calculate derivatives of the smoothed image.

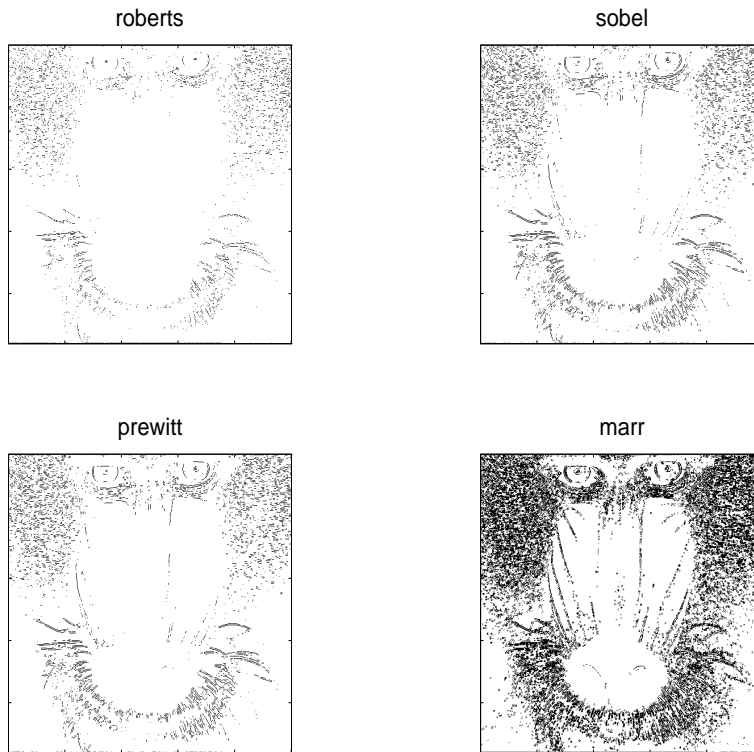


Figure 4: The output of various edge detectors on the mandrill image. Note that the use of default thresholds renders output that is next to useless.

Both these operations are linear, and we can combine them in a different, more computationally efficient way. If we calculate the derivative of the Gaussian mask and convolve the image with this mask we compute the derivative of the smoothed image in one hit. That is

$$(I \otimes G)' = I \otimes G',$$

where I is the image and G is the Gaussian mask.

Second order derivative operators

A maximum of the first derivative will occur at a zero crossing of the second derivative.

To get both horizontal and vertical edges we look at second derivatives in both the x and y directions. This is the *Laplacian* of I

$$\nabla^2 I = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}.$$

The Laplacian is linear and rotationally symmetric.

Thus, we search for the zero crossings of the image that is first smoothed with a Gaussian mask and then the second derivative is calculated; or we can convolve the image with the Laplacian of the Gaussian, also known as the LoG operator (see Figure 7):

$$\nabla^2(G \otimes I) = \nabla^2 G \otimes I.$$

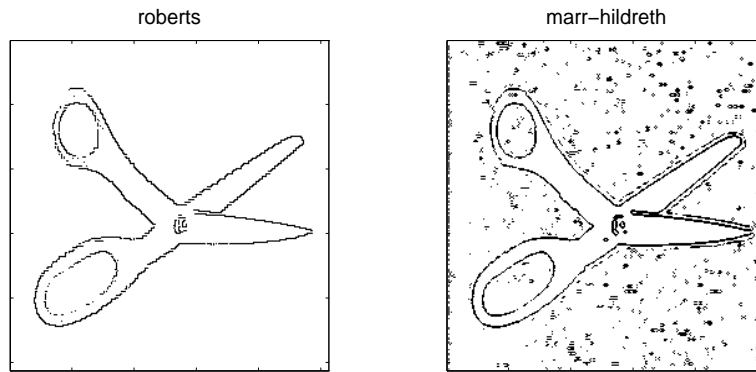


Figure 5: The output of the Roberts and the Marr-Hildreth operators on the scissors image. Note that edge contours are closed in the latter.

This defines the *Marr-Hildreth* operator. One can also get a shape similar to G'' by taking the difference of two Gaussians having different standard deviations. A ratio of standard deviations of 1:1.6 will give a close approximation to $\nabla^2 G$. This is known as the DoG operator (Difference of Gaussians), or the *Mexican Hat Operator*.

The Marr-Hildreth operator became widely used for the following reasons:

1. Marr had considerable reputation.
2. Researchers found receptive fields in the eyes of animals (usually cats and macaque monkeys) that behaved just like this operator.
3. The operator is symmetric. Edges are found in all orientations, unlike the first derivative operators which are directional.
4. Zero crossings of the second derivative are simpler to determine than are maxima in the first derivative; all that needs to be done is to look for a sign change in the signal. Moreover, the zero crossings of a signal always form closed contours. This is nice if one is trying to separate out objects in the scene.

There are some problems, however.

1. Being a second derivative operator the influence of noise is considerable.
2. Animals have lots of other types of receptive fields, so this is not the whole story.
3. An edge detector that always generates closed contours is not realistic.
4. The Marr-Hildreth operator will mark edges at some locations that are not edges.

The Canny edge detector

The current standard in edge detection that is widely used around the world is the Canny edge detector. This arose from the work John Canny did for his Masters degree at MIT in 1983. He treated edge detection as a signal processing problem and aimed to design

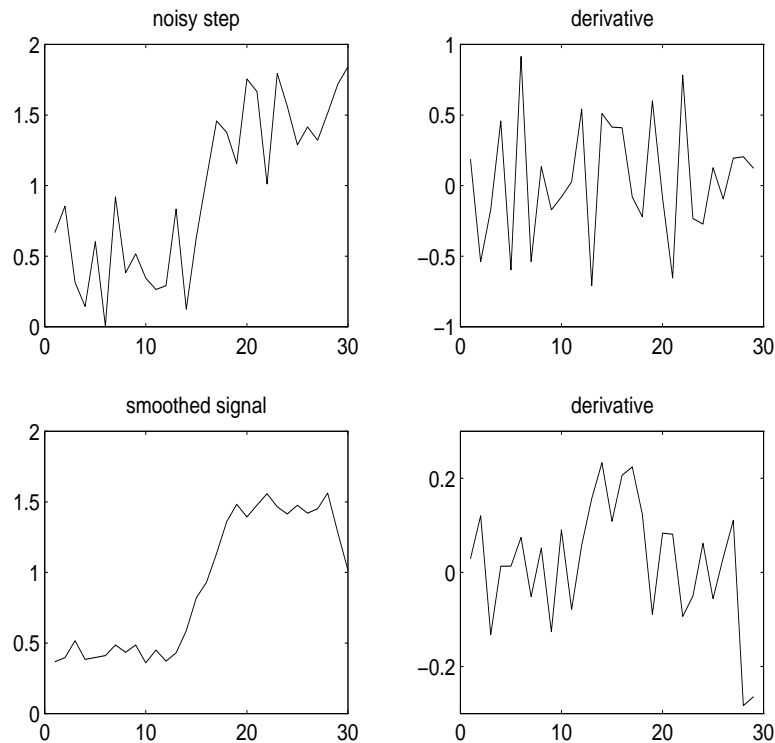


Figure 6: A noisy step edge and its derivative (top); the smoothed signal and its derivative (bottom).

the ‘optimal’ edge detector. He formally specified an objective function to be optimised and used this to design the operator.

The objective function was designed to achieve the following optimisation constraints:

1. Maximise the signal to noise ratio to give good detection. This favours the marking of true positives.
2. Achieve good localisation to accurately mark edges.
3. Minimise the number of responses to a single edge. This favours the identification of true negatives, that is, non-edges are not marked.

Note a difference-of-boxes operator will maximise the signal to noise ratio with arbitrarily good localisation, but will give several responses to a single edge. The difference-of-boxes operator is, in fact, the optimal Wiener filter for the step edge.

After some complicated analysis Canny came up with a function that optimised his three criteria and that was the sum of 4 exponential terms. However, this function looked very much like a first derivative of a Gaussian! So this is what ended up being used.

In the simplest form of his edge detection technique the procedure was as follows:

1. Convolve the image with a two dimensional Gaussian filter to smooth it.
2. Differentiate the image in two orthogonal directions.

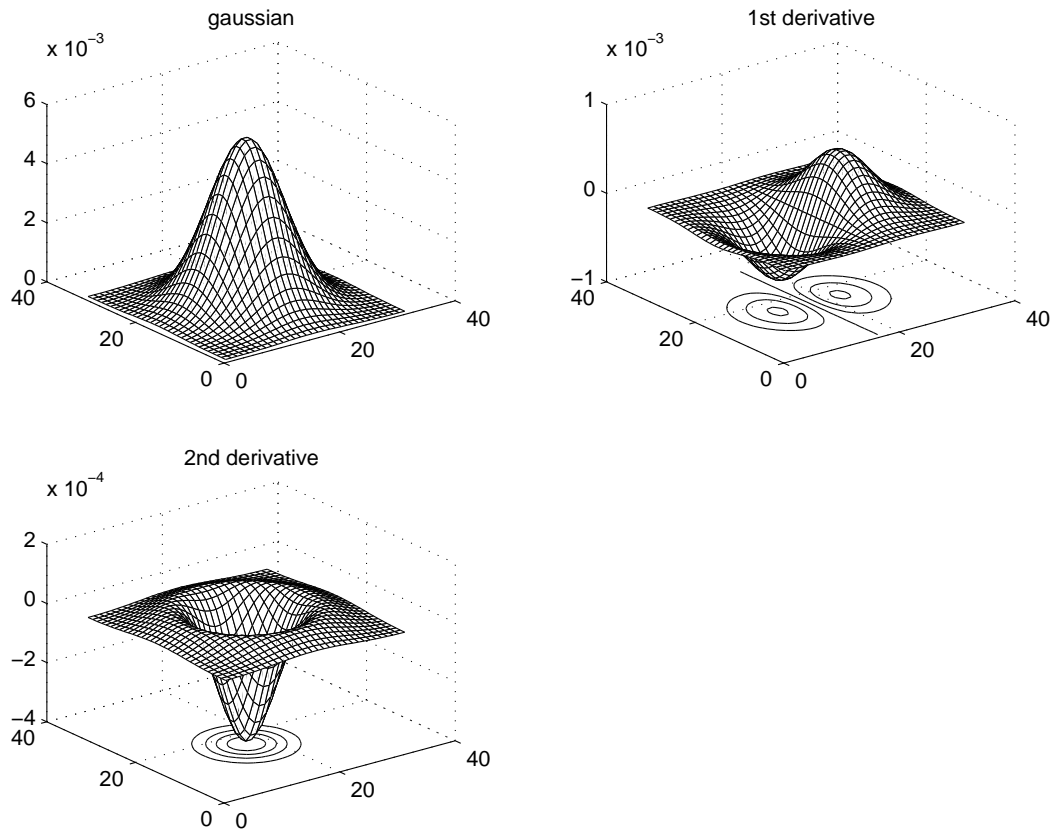


Figure 7: A Gaussian and its first and second derivatives. Note that the 1st derivative is directionally sensitive.

3. Calculate the gradient amplitude and direction.
4. Perform non-maximal suppression.
5. Perform non-maximal suppression. Any gradient value that is not a local peak is set to zero. The gradient direction is used in this process.
6. Threshold these edges to eliminate ‘insignificant’ edges.

In its more elaborate form filters in multiple directions and widths would be applied at each point in the image, with the direction and width of the filter being chosen to maximise his objective function.

Many implementations of the Canny edge detector do something halfway between the two. The image is convolved with 4 masks, calculating horizontal, vertical and diagonal gradients (the masks used are similar to the Prewitt or Sobel masks). The direction producing the largest result at each pixel is used to determine magnitude and direction of the gradient.

Concerning the final step of thresholding, Canny introduced the idea of *thresholding hysteresis*. This involves having two different threshold values, usually the higher threshold being 2–3 times the lower. The output T_2 that is formed with the higher threshold will contain fewer false edges, but there might also be gaps in the contours (that is, too many

false negatives). Thus the edges in T_2 are linked into contours as follows: when the edge of a contour in T_2 is found, the algorithm looks in T_1 , the output of the thresholding at the lower threshold value, at the locations of the 8 neighbours of the pixel under consideration to see if there are edge pixels that can be linked to the contour. The algorithm continues to gather edges from T_1 until the gap has been bridged to an edge in T_2 .

Thus, any pixel in an edge list that has a gradient greater than the higher threshold value is classed as a valid edge point. Any pixels *connected* to these valid edge points that have a gradient value above the lower threshold value are also classed as edge points. That is, once you have started an edge you don't stop it until the gradient on the edge has dropped considerably.

An example showing the performance of the Canny edge detector is given in Figure 8.

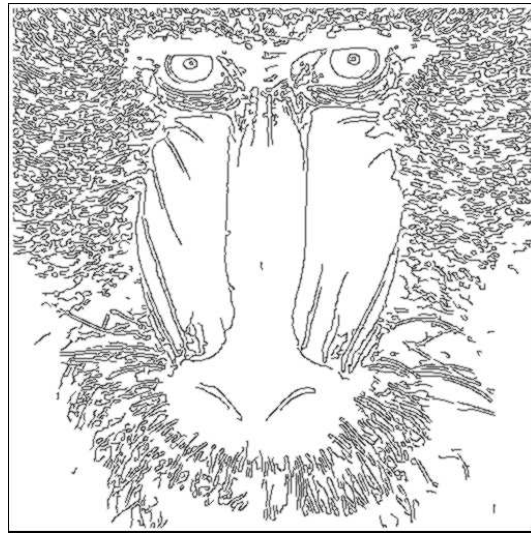


Figure 8: The output of the Canny edge detector on the mandrill image. Note that double edges are marked for line features.

Problems with gradient-based edge detectors

Gradient-based edge detection schemes suffer from a number of problems, but they are still the most commonly used by the computer vision community. Some of these problems include the following:

1. You have to choose threshold values and the width of your masks. This problem is common to all gradient based methods. Note that if you double the size of an image leaving its grey values unchanged all the gradients will be halved. This complicates the setting of any threshold. An additional problem is that the width of the mask (and hence the degree of smoothing) affects the positions of zero crossings and maximum intensity gradients in the image. Ideally, the estimated position of any edge should not be affected by the size of the convolution mask.

2. Corners are often missed because the 1D gradient at corners is usually small. This can cause considerable difficulties for line labelling schemes because they rely on having corners and junctions being marked properly.
3. First derivative operators will find only step-like features. If you want to find lines you need a different operator. For example, to find bar features you could look for *peaks* (not zero crossings) from a second derivative operator. Canny did design an operator for finding lines.

Thus, differential edge detection schemes suffer both from false positives and false negatives.

Edge detector performance

A number of researchers have considered the problem of measuring edge detector performance. In fact, it is difficult since we don't really know what the underlying features are that we wish to detect. However, if we assume that they are step edges corrupted by Gaussian noise, then some criteria can be set for evaluating performance. Such criteria are usually the following:

- the probability of false edges;
- the probability of missing edges;
- the error in estimating the edge angle;
- the mean square distance of the edge estimate from the true edge; and
- the algorithm's tolerance to distorted edges and other features such as corners and junctions.

The first two criteria relate to edge detection, the second two to edge localisation, and the last to tolerance to departures from the ideal edge model. Pratt [8] introduced a *figure of merit* function FM for measuring quantitatively the performance of various edge detectors. His measure is

$$FM = \frac{1}{\max(I_A, I_I)} \sum_{i=1}^{I_A} \frac{1}{1 + d_i \alpha^2},$$

where I_A, I_I, d , and α are respectively the detected edges, the ideal edges, the distance between the actual and the ideal edges, and a design constant used to penalise displaced edges.

Corner detection

For a number of computer vision tasks we are interested in finding so called 'corner' points. These are points in the image that have a significant 2D component to them, typically corners of objects. These points are of interest because they can be uniquely matched

and tracked over a sequence of images (whereas a point along an edge can be matched with any number of other points on the edge in a second image). The ability to match points between images facilitates 3D reconstruction from stereo and motion.

The corner detector devised by Harris and Stephens in 1988 (the ‘Harris Detector’) remains the most widely used operator for this purpose. This operator considers the minimum and maximum eigenvalues, α and β , of the image *gradient* covariance matrix. The gradient covariance matrix is given by

$$G = \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

where I_x and I_y denote the image gradients in the x and y directions. If the underlying feature is a straight edge, so that the gradient structure is one-dimensional, one of the eigenvalues will tend towards 0. When the two eigenvalues are large and similar in magnitude it indicates the gradient has a two-dimensional structure, and a ‘corner’ is said to have occurred.

To avoid an explicit eigenvalue decomposition Harris and Stephens devise a measure using the determinant and trace of the gradient covariance matrix

$$R = \det(G) - k(\text{tr}(G))^2 ,$$

where $\det(G) = \alpha\beta$ and $\text{tr}(G) = \alpha + \beta$, the parameter k is traditionally set to 0.04. This produces a measure that is large when both α and β are large. However we have the problem of determining what is large. Noting that elements of the image gradient covariance matrix have units of intensity gradient squared we can see that the determinant, and hence the measure R will have units of intensity gradient to the fourth. This explains why the Harris operator is highly sensitive to image contrast variations which, in turn, makes the setting of thresholds exceedingly difficult. Some kind of sensitivity to image contrast is common to all corner operators that are based on the local autocorrelation of image intensity values and/or image gradient values.

The use of the parameter k in the equation above, and the value it should be set to, is troubling to some people. Noble (1989) presents an alternative measure, involving the ratio of the determinant and trace, which I find preferable

$$R = \det(G)/\text{tr}(G) .$$

If one of the eigenvalues tends to zero $\det(G)$ tends to zero. If the eigenvalues are similar but also small the result tends to zero as well, but if they are similar and large $\det(G)$ will be much greater than $\text{tr}(G)$ and the result will be large.

The Hough Transform

The Hough transform is a method that, in theory, can be used to find features of any shape in an image. In practice it is generally used for finding only straight lines or circles. The computational complexity of the method grows rapidly with more complex shapes.

Assume we have some data points in an image which are perhaps the result of an edge detection process, or boundary points of a binary blob. We wish to recognise the points

that form a straight line. The idea of the Hough transform is that each of these points votes for several combinations of parameters; the parameters that win a majority of votes are declared the winners, that is, they are the parameters that describe best the line given by these points.

Consider a point (x_i, y_i) in the image. The general equation of a line is

$$y = ax + b.$$

Here x and y are the observed points and a and b are the parameters. There are infinitely many lines that pass through this point, but they all satisfy the condition

$$y_i = ax_i + b$$

for varying a and b .

We can rewrite this equation as

$$b = -x_i a + y_i,$$

and plot the variation of a and b .

If we divide parameter space into a number of discrete accumulator cells we can collect ‘votes’ in ab space from each data point in xy space. Peaks in ab space will mark the equations of lines of co-linear points in xy space.

However, we have a problem with using $y = ax + b$ to represent lines when the line is vertical. In that case $a = \infty$ and our parameter space is unbounded (we would need a very large computer to store our parameter accumulator array!)

An alternative representation of a line is given by

$$x \cos \theta + y \sin \theta = r,$$

where r is the distance of the line from the origin and θ is the angle between this perpendicular and the x-axis. Our parameter space is now in θ and r , where $0 \leq \theta \leq 2\pi$ and r is limited by the size of the image. As before, peaks in the accumulator array mark the equations of significant lines.

The Hough transform does not require prior grouping or linking of the edge points, and the edge points that lie along the curve of interest may constitute a small fraction of the edges in the image. Moreover, if the peak in the parameter space covers more than one accumulator, then the centroid of the region that contains the peak provides an estimate of the parameters.

In theory any kind of curve can be detected if you can express it as a function of the form

$$f(a_1, a_2, \dots, a_n, x, y) = 0.$$

For example a circle can be represented as

$$(x - a)^2 + (y - b)^2 - r^2 = 0.$$

This model has three parameters: two parameters for the centre of the circle and one parameter for the radius of the circle. If the gradient angle for the edges is available,

then this provides a constraint that reduces the number of degrees of freedom and hence the required size of the parameter space. The direction of the vector from the centre of the circle to each edge point is determined by the gradient angle, leaving the value of the radius as the only unknown parameter.

Thus, the parametric equations for a circle in polar coordinates are

$$x = a + r \cos \theta$$

and

$$y = b + r \sin \theta.$$

Solving for the parameters of the circle we obtain the equations

$$a = x - r \cos \theta$$

and

$$b = y - r \sin \theta.$$

Now given the gradient angle θ at an edge point (x, y) , we can compute $\cos \theta$ and $\sin \theta$. Note that these quantities may already be available as a by-product of edge detection. We can eliminate the radius from the pair of equations above to yield

$$b = a \tan \theta - x \tan \theta + y.$$

Then the Hough Transform algorithm for circle fitting can be described as follows.

Circle Fitting Algorithm

1. Quantise the parameter space for the parameters a and b .
2. Zero the accumulator array $M(a, b)$.
3. Compute the gradient magnitude $G(x, y)$ and angle $\theta(x, y)$.
4. For each edge point in $G(x, y)$, increment all points in the accumulator array $M(a, b)$ along the line

$$b = a \tan \theta - x \tan \theta + y.$$

5. Local maxima in the accumulator array correspond to centres of circles in the image.

References

- [1] J. F. Canny. Finding edges and lines in images. Master's thesis, MIT. AI Lab. TR-720, 1983.
- [2] J. F. Canny. A computational approach to edge detection. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 8:679-698, 1986.
- [3] Rachid Deriche. Using Canny's criteria to derive an optimal edge detector recursively implemented. *The International Journal of Computer Vision*, 1:167-187, April 1987.

- [4] Margaret M. Fleck. Multiple widths yield reliable finite differences. *IEEE Transactions PAMI*, 14(4): 412-429, April 1992.
- [5] D. Marr. *Vision*. Freeman, 1982.
- [6] D. Marr and E. C. Hildreth. Theory of edge detection. *Proceedings of the Royal Society, London B*, 207:187-217, 1980
- [7] Vishvjit S. Nalwa. *A Guided Tour of Computer Vision*. Addison-Wesley Publishing Company, 1993.
- [8] W. K. Pratt. *Digital Image Processing*. Wiley, New York, second edition, 1991.
- [9] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing*. Addison-Wesley Publishing Company, 1992, chapter 7.
- [10] Chris Harris and Mike Stephens. A Combined Corner and Edge Detector. *Proceedings of The Fourth Alvey Vision Conference*, Manchester, pp 147-151. 1988
- [11] Alison Noble, "Descriptions of Image Surfaces", PhD thesis, Department of Engineering Science, Oxford University 1989.