DETECTION OF CIRCULAR ARCS IN IMAGES

Rosin P.L.[1] and West G.A.W.[2]

[1]Department of Neurology,
Guys Hospital,
London Bridge,
London SE1 9RT.

[2]Measurement and Instrument Centre,
School of Electrical Engineering and Applied Physics,
City University,
Northampton Square,
London EC1V OHB.

A long standing problem in computer vision is the extraction of meaningful features from images. This paper describes a method of detecting circular arcs in images based on a recursive algorithm that first analyses lists of connected edge points and finds a polygonal description, and then analyses this description fitting arcs to groups of connected lines. The result is a description of image edges consisting of circular arcs and lines. The algorithm uses no thresholding. Instead the best option is chosen at each decision stage.

INTRODUCTION

This paper describes a method for finding circular arcs in images. It is an extension of a method proposed by Lowe[1] which analyses arbitrary curves and produces a high level straight line description. The curves are described by a connected set of pixels obtained from an image after application of the Marr-Hildreth[2] edge detector and a zero-crossing detection algorithm. Lowe's algorithm segments each curve by recursively splitting it at the maximum deviation from the approximating straight line. At each level a decision is made as to whether the single straight line is better than the representation at a lower level. The lower level representation will consist of two or more approximating straight lines. The measure of goodness of fit is termed the significance and is defined as the ratio of maximum deviation from the straight line to the length of the straight line.

To detect circular arcs, the straight line description (not the original data) is processed using a similar algorithm to find the best fit of arcs and straight lines to the data. The difference is that at each level of recursion, a decision is taken as to whether an arc is a better fit than the lower level description. This results in a description of the arbitrary curve consisting of straight lines and circular arcs.

Extending the algorithm to deal with space curves increases the complexity since arcs, ellipses etc. need more parameters for their description than the straight line. This means that there are more degrees of freedom when fitting the curve to the data. However, a circular arc fitting algorithm has been developed that determines the best fit (in a least mean square error sense) of an arc to the data using the constraint that the arc must pass through the end points of the data (so as to connect with the adjoining arcs or lines).

An interesting property of the algorithm is that no thresholding is required in any stage of the processing used to segment the original image. Instead of using thresholds (that normally have to be tuned to the particular application) the processing is organised so that at each decision stage, the best option is taken, e.g. whether an arc or a combination of straight lines and arcs is best. It should therefore be general purpose and useful for any application. In fact, any connected set of points in 2-d space can be segmented by this process, and chain coded boundaries from binary images have also been processed by the authors. In the case of images of zero-crossings, the only constraints are that the image must be reasonably clean, and real edges in the image represented by connected zero-crossing points. If there is too much disconnectedness, then the algorithm produces poor results although these may be improved at a later stage.

The algorithm has been used to determine high level descriptions of images from industrial scenes as well as natural scenes. These are being used for further processing in the fields of object recognition and automatic inspection.

PRE-PROCESSING

The grey scale image is first processed using the Marr-Hildreth edge detector, after which the zero-crossings are detected, e.g. figure 1. For this particular example, the image is 512 x 512 pixels and a sigma value of 1.8 was used for the edge detector. The edge pixels are stored in an image as pixels of value 255 on a background of value 0. The image is converted to a number of lists of linked edge pixels. In

AVC 1988 doi:10.5244/C.2.40

general, a real edge in the image will be represented by one or more lists. Since the edges of the resulting image are only one pixel wide, thinning is unnecessary. Theoretically, the space curves should form closed loops. However, this is only possible where the zero-crossings are detected to sub-pixel accuracy. In our applications, this is unnecessary, so zero-crossings are placed at the nearest pixel. This causes some edge pixels to be combined and others ignored.

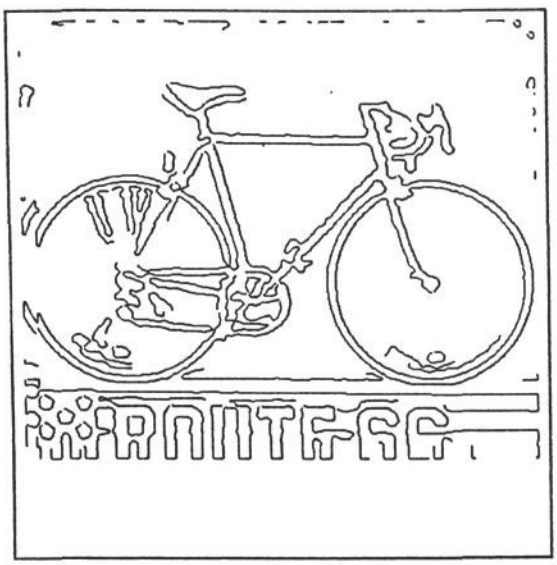

Figure 1. Image showing zero-crossings.

To allow fast processing of the edge data, the image is scanned and each space curve is transformed into a linked-list of pixel coordinates. These are either open or closed lists representing open and closed curves respectively. Open lists are detected first, followed by closed lists. In the case of open lists, the algorithm looks for a pixel on a space curve that has only one neighbour (which is therefore at the end of the curve) and then follows the curve, generating the list until it reaches the other end. During this process all pixels that are added to the lists are deleted from the image, thereby removing them from the search space. After all open lists have been processed, the remaining pixels are searched for closed lists. A closed list is acquired by locating a pixel on a closed curve (one with two neighbours) and then following the curve until it returns to the start pixel. As with the open curves, as each pixel is examined it is deleted from the image.

LINE DETECTION

The next stage is to find a polygonal approximation of the lines. An improvement is made on the method described by Lowe[1]. In Lowe's work, a list of edge pixels is hypothesised as being a straight line passing through its end points. This is segmented into two lists at the point of maximum deviation, and the process repeated recursively on each of the two lists. The recursive process is halted when a line segment is smaller than four pixels long or the deviation is less that three pixels. A three pixel length line is the smallest line that can contain deviations. Even so, all lines are given a minimum non-zero deviation to avoid small, perfect line segments preventing the formation of longer, non-perfect lines. The result of the recursive process is a multi-level tree where the description of the list of edge pixels at each level is a finer approximation of the level above. As the recursion unwinds, the tree is traversed back up to the root. At each level, if any of the line segments passed up from the lower level are more significant than the line segment at the current level, they are retained and passed up to the next higher level as candidate line segments. If this is not the case, the line segment at the current level is passed up. Once the recursion has completely unwound, a description of the space curve is obtained made up of a polygonal approximation. The decision is based on a pseudo-psychological measure of perceptual significance. Figure 2 shows an example of line fitting for three levels of recursion.

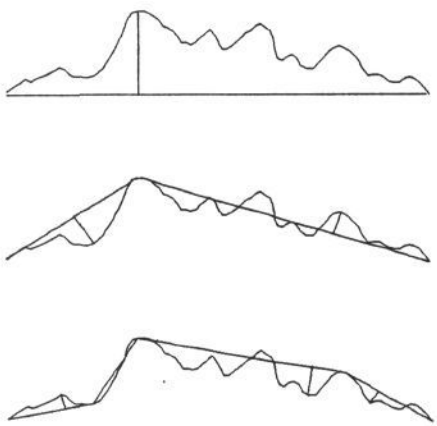

Figure 2. Three levels in straight line fitting (after Lowe[1]).

Lowe's measure of significance is determined by the ratio of the line segment length divided by the maximum deviation of the space curve from the straight line segment. However, when all the pixels lie directly on the straight line (which occurs regularly with short lines), the significance is infinite. To avoid divide-by-zero errors, the significance measure has been redefined as the ratio of the maximum deviation divided by the line segment length. Thus, the lower the significance value, the more significant the line. The procedure is weighted in favour of long (i.e. highly significant) line segments. The longer the line, the greater the deviations that will be tolerated.

This results in a high quality, general purpose polygonal approximation. No arbitrary error threshold is required. Instead, the most appropriate values are chosen dynamically throughout the procedure. For instance, curves of different sizes give very similar approximations, but at different scales. The use of a fixed threshold would approximate the smaller curves crudely, and larger curves with unnecessary detail. Figure 3 shows the result of detecting line segments in a real image.

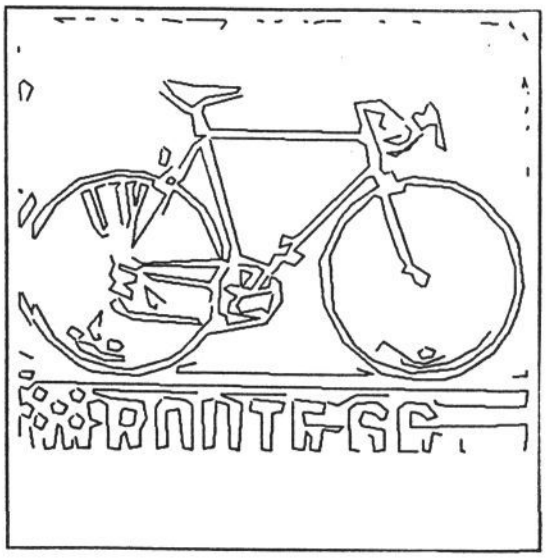

Figure 3. Result of straight line approximation for image of figure 1.

ARC DETECTION

Extending Lowe's work, the same approach can be applied to curve detection. To determine a line description, sequences of pixels are hypothesised as line segments, and the hypotheses recursively refined by subdividing at the point of maximum deviation. In a similar manner, sequences of line segments, obtained from the polygonal approximation procedure, can be hypothesised as curves, and refined by subdividing at the point of maximum deviation, as shown in figure 4.

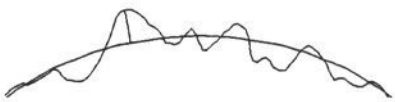

Figure 4. Example of fitting an arc to the data of figure 2.

Circular arc detection is more complicated than line detection since the specification of an arc requires more parameters as it has more degrees of freedom than a line. To fit a circular arc to a space curve requires the determination of the centre, start and end coordinates, and the radius. Algorithms for performing this procedure have been proposed by Landau[3] and Albano[4]. However the process can be simplified by applying the same constraint that is used for fitting straight lines described in the section above. Since space curves are described by a combination of connected lines and arcs, arcs are constrained to start and finish at the two end points of the space curve of interest. Given the two end points of the arc, three parameters need to be determined, namely the radius and the coordinates of the centre of the circle forming the arc. Minimising the error for the three parameters is a non- trivial task, but the problem can be simplified as the end points of the arc are known. As the centre point of the circle must be equidistant from the end points of the arc, it is constrained to lie on a straight line that is normal to the line joining the two end points as shown in figure 5.

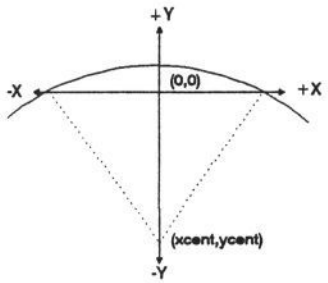

Figure 5. Arc finding space.

Thus a method of finding the best or optimum arc is to search along this line in some intelligent way. In our algorithm, the measure of match between the data and the arc is determined using a least squared error criterion. The error for each vertex is the shortest distance to the circle. This is the difference between the radius and the distance from the vertex to the centre as shown in figure 6. A recursive binary split algorithm rapidly searches for the minimum error using the gradient descent method.

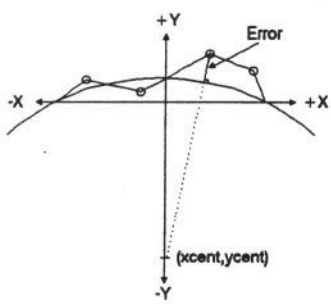

Figure 6. Error for one vertex when fitting an arc to a polygonal approximation.

The algorithm has the following stages:

1/ Transform the coordinates of the vertices so that the centre of the line joining the two outer points lies at the

origin and the two outer points lie on the x-axis. This means that the centre of the circle will lie along the y-axis (figure 5).

2/ Compute the errors with the centre of the hypothesised circle centred at the origin (0,0) and either side of the x-axis (0,1) and (0,-1). This enables the gradient of the error function to be determined at the origin. The required minimum will lie on the side of the x-axis indicated by the direction of negative gradient. The rule is:

if error(0,1) > error(0,-1) then direction = -1
else direction = 1

3/ Compute the mean y coordinate of all the vertices. This determines whether the arc is small (less that 180 degrees) or large (greater than or equal to 180 degrees) using the following rules:

if direction = 1 and sum < 0.0 then small arc
if direction = -1 and sum > 0.0 then small arc
if direction = 1 and sum > = 0.0 then large arc
if direction = -1 and sum < = 0.0 then large arc

4/ Set up the search space in which to search for the minimum error. At this point the maximum radius has to be decided. If the arc is small, then a reasonable maximum radius is 16383 pixels. Thus for a small arc the search space is 0 to 16383. For a large arc the centre will be somewhere between the origin and the maximum y value of the data.

5/ Search for the minimum error. This uses a recursive binary splitting algorithm to find the minimum quickly. The maximum number of recursions required is for a small arc with a search space of 0 to 16383 requiring 14 levels of recursion. For large arcs this is usually much lower.

6/ Compute the parameters of the arc, namely centre, radius, length of the arc and polygonal approximation. Also required is the maximum deviation of the polygonal data from the arc. This is carried out by walking around the arc computing the minimum deviation at each point and saving the maximum of these. This stage is necessary to accommodate situations such as those shown in figure 7.

7/ Compute the significance as for line segments, i.e. the ratio of maximum deviation to feature length where in this case, the feature length is the length of the arc.

The recursive procedure repeatedly subdivides arcs into subarcs, stopping when the sequence of line segments to be approximated contains fewer than three line segments which is analogous to the three pixel limit in the polygonal line approximation. Obviously a single line segment should not be replaced by a single arc. Two line segments contain three points and would always be replaced by a perfect fit. This would prevent its replacement by higher level matches since they could never improve on the lower level match.

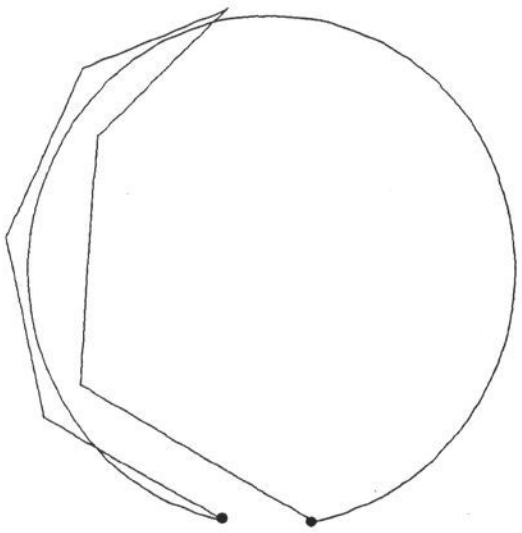

Figure 7. Good least mean square fit requiring stage 6 for detection of poor match.

The deviation between the hypothesised arc and the line segments was initially calculated by finding the minimum distances between the line segment endpoints and midpoints and the arc, and returning the maximum. However, certain non-circular arrangements of lines returned low deviations. A more robust measure of deviation was provided by stepping around the arc, finding the minimum distance from the line segments, and returning the maximum. Additional degenerate arcs are trapped by checking that the length of the arc is approximately equal to the length of its polygonal approximation.

Another difference between line and arc detection is that whereas all pixels should be replaced by straight line segments, in many cases lines are more appropriate than arcs. The decision to replace lines by arcs can be uniformly incorporated without the need for an arbitrary threshold. The significance of an arc is calculated using a similar manner to the significance of a line, namely the ratio of the maximum deviation divided by the length of the arc. Therefore, since their significances are compatible, lines are only replaced by an arc if the arc is more significant than all the lines. It is only because the previous procedure results in such good polygonal approximations that the arc significances are compatible. If it did not dynamically determine the most appropriate error threshold, then small curves would be crudely approximated, producing arcs with large deviations and thus poor significances.

The algorithm has been tried on numerous images ranging from manufactured components to outdoor scenes containing man made artifacts e.g. vehicles. Figure 8 shows the result of processing the bicycle (compare with figure 3) in which arcs describing the wheels are evident.

Figure 8. Result of finding arcs in image of figure 3.

SUMMARY

This paper has described an algorithm for detecting circular arcs in arbitrary space curves that result from edge detection in images. It essentially extends the line fitting algorithm proposed by Lowe[1] so that the curves are described as a combination of straight lines and circular arcs. An example was given of grey level processing with zero-crossings detected in an image after convolution with the Marr-Hildreth edge detector. The algorithm is not restricted to this image type, and has been used on chain-coded binary images for boundary descriptions.

A fast algorithm for determining the minimum error match of a circular arc with the data is used. The computational burden is reduced over other curve fitting methods by the constraint that the curve must be bounded by two defined end points.

Extending this work to detect other curve types e.g. ellipses, conics etc. from the arc description would be rewarding as a higher level description would be obtained, for example a combination of arcs replaced by a single ellipse.

An interesting property of this algorithm is that no thresholding is necessary at any stage from the edge detection through line and then arc detection. At each stage of curve fitting, the best description is chosen i.e. that description with the best significance or closest fit to the data. The lack of thresholding should make the algorithm general purpose.

REFERENCES

1. **Lowe D.G.**, "Three-dimensional object recognition from single two-dimensional images", *Artificial Intelligence*, Vol. 31, pp. 355-395, 1987.
2. **Marr D. & Hildreth E.**, "Theory of edge detection", *Proc. Royal Society London B.*, Vol 207, pp 187-217, 1980.
3. **Landau U.**,"Estimation of a circular arc centre and its radius", *Computer Vision, Graphics and Image Processing*, Vol. 38, pp. 317-326, 1987.
4. **Albano A.**, "Representation of digitised contours in terms of conic arcs and straight-line segments", *Computer Vision, Graphics and Image Processing*, Vol. 3, pp. 23-33, 1974.