COMPUTER VISION AND IMAGE UNDERSTANDING
Vol. 63, No. 2, March, pp. 334–343, 1996
ARTICLE NO. 0023

Recognition of Partial Circular Shapes from Segmented Contours

FILIBERTO PLA

Department of Computer Science, University Jaume I, 12071 Castellón, Spain

Received February 22, 1994; accepted December 21, 1994

This paper presents a circle-finding method to deal with partially occluded circles using segmented contours. Contours are segmented using a criterion based on the derivative of the curvature. Constant curvature segments are primitives inputted into a clustering algorithm which brings out the relationships among contour segments which are likely to represent the same circles. A minimization criterion is used to find the circle parameters, providing an accurate parameter estimation. The method allows recovery of the contour segments used to estimate circle parameters, providing useful information in some practical applications. © 1996 Academic Press, Inc.

INTRODUCTION

Circular contour recognition has received much attention since the earliest days in image analysis research, mainly because a wide range of round and circular objects are present in many machine vision applications in industrial and other environments. In these applications, the location of circular or round objects allows automatic manipulation or automatic inspection for grading or sorting. In particular, the work we are presenting here has been aimed at some applications in the agricultural and food industries to locate fruits more accurately for robotic harvesting and for sorting of food products.

The classical approach to recognizing geometrical patterns has been the Hough transform [1] which, due to its simple but exhaustive approach in calculating the three parameters that define a circular arc (center coordinates and radius), leads to a heavy computational burden in time and memory. In order to reduce this complexity, later approaches to the original Hough transform have been developed to recognize circular patterns. A comparative study of these techniques was made by Yuen *et al.* [2]. Even the most memory efficient method requires twice the image storage (the two-stage Hough transform [3, 4]) to achieve at least one pixel accuracy in the calculation of the parameters.

Some problems in Hough transform techniques are the choice of a threshold and finding peaks in the accumulator space, although for finding peaks, the Gerig and Klein [5] method gives satisfactory results. Finding peaks is a more difficult task in the radius histogram in the two-stage Hough transform [3, 4], which is sensitive to inaccuracy in the first stage. A particular deficiency of the Gerig Hough transform method [5] is that it cannot find concentric circles in its simplest implementation. Other versions of the Hough transform method [6] can perform quickly, but only to estimate the centers of the circles, and suffer from imprecision in estimating the center when the missing parts of a contour are significant.

On the other hand, the accuracy of parameter estimation for the Hough transform methods depends on the quantization of the accumulator space. Applications which require more accuracy in parameter estimation usually use an initial estimation of the circle parameters by a Hough transform technique and subsequently perform a least-squares fit on points which fall in a template around the initial estimated circle [7]. Alternatively, robust estimators are used in an iterative way starting from this initial estimation to avoid the sensitivity to outliers [8].

In the work presented here, the problem of circle finding is dealt with from another direction. The motivation was to deal with the problems of circular contours in several degrees of partial occlusion, bringing out their circle parameters in an accurate way and also to identify the proportion of missing (or present) circular contours of each object. Knowing the proportion of missing circular contour allows one, for example, to reject broken items in a grading process.

Our approach was to look at the problem with a more structured representation. Instead of using points as primitives to find the relations among them without taking into account other close-by feature points, curve segments were used as initial inputs, and their relationships were used to look for segments which define or belong to the same circle. Once curves have been segmented into constant curvature segments, an interpretation process looks for relationships between the curve segments and groups them into different sets. Segments in the same set are likely to represent a single circle whose parameters (radius and center coordinates) and proportion of contour may be subsequently found.

1077-3142/96 $18.00

CONTOUR SEGMENTATION

The classical approach to segment curves is to split them at some control points. This is a general methodology which includes the use of curvature extrema [9], zero-crossing points [10], or intersecting points in transformed curves obtained from curvature scaling [11]. The other general methodology to segment curves is based on grouping points that satisfy some homogeneity criterion, for instance, linear piecewise segmentation [12, 13], circular arc segmentation [14], or elliptical arc segmentation [15, 16]. These methods are often based on higher level concepts using procedures with more complexity than the simpler control point-based approaches.

The objective of curve segmentation in this paper is to provide an initial stage where the curve segments may be arcs of a higher level circumference structure. It is not important if the segmentation splits a real circumference arc into several arcs, but it is desirable that segments are as long as possible. The subsequent interpretation process will bring out and decide which segments belong to the same arc.

Given the characteristics of the problem, it is appropriate to use a homogeneity segmentation criterion. The curvature function is well known and widely used as a powerful method to characterize curve points. Therefore, including in the same segment points with the same constant curvature value means that points in the segment are likely to belong to a single circle.

Wuescher and Boyer [17] proposed an algorithm to segment curves based on a constant curvature criterion. As they pointed out, the advantage of constant curvature primitives over straight line approximation is that linear piecewise segmentation always breaks circular arcs into multiple line segments, resulting in a larger number of primitives. Moreover, their algorithm tends to occasionally place a break point to either side of corners instead of placing a single break point at the true corner.

Since one of the objectives of this work is to assess the portion of circular contour from which the circle has been recognized, it is desirable to achieve a more accurate segmentation by correctly placing break points at true corners, thus maintaining the constant curvature criterion. This is achieved by combining the concepts of curvature extrema and constant curvature in the same segmentation procedure.

Tangent Calculation

As described in the subsequent sections, tangent at a point of a curve is used for circle parameter estimation. In order to obtain accurate tangent values, a method based on the geometrical interpretation of the derivative is proposed. In addition, the contour segmentation criterion used is based on the derivative of the curvature, and these values will be calculated directly from tangent angle values.

Let $\alpha(s) = (x(s), y(s))$ represent a regular curve where $s \in [0, L]$ is measured along the arc. Denote $\theta(s)$ as the angle defined by the tangent at the point $\alpha(s)$. The curvature $k(s)$ of $\alpha(s)$ is defined as $k(s) = d\theta(s)/ds$ [18]. Since $y = y(x)$ is not available from real images, let us look at the problem of calculating θ from another source. Geometrically, the tangent at a point (x_0, y_0) in the curve $y(x)$ with slope $m = dy(x_0)/dx$ is the limit of a succession of chords whose extreme points, $y(x_0 + h)$ and $y(x_0 - h)$ with $h > 0$, tend to $y(x_0)$, that is,

$$m = \lim_{h \to 0} \frac{y(x_0 + h) - y(x_0 - h)}{2h}.$$

Therefore, we can approximate the tangent of a point (x_0, y_0) by the chord defined by two points of the curve (x_l, y_l) and (x_r, y_r), one to either side of (x_0, y_0), where both points are as close as possible to (x_0, y_0). Hence, we can calculate the angle θ of the tangent at (x_0, y_0) as

$$\theta = \tan^{-1}\left(\frac{y_r - y_l}{x_r - x_l}\right).$$

In order to reduce the error due to sampling, instead of taking (x_l, y_l) and (x_r, y_r) as points of the curve, the coordinates of these points are calculated by averaging the coordinates of a group of neighbors to perform a less noise prone resampling [19] where the new coordinates have subpixel resolution.

Let us consider the mid point (x_0, y_0) of n contiguous points in a chain code of a curve, where n is an odd number, and let $p = n/2 + 1$ be the point (x_0, y_0). Thus, the initial point of the chord (x_l, y_l) is calculated from the $n/2 + 1$ previous points as

$$x_l = \frac{\sum_{i=1}^{p} x_i}{n/2 + 1} \text{ and } y_l = \frac{\sum_{i=1}^{p} y_i}{n/2 + 1}$$

and similarly for the end point of the chord (x_r, y_r)

$$x_r = \frac{\sum_{i=p}^{n} x_i}{n/2 + 1} \text{ and } y_r = \frac{\sum_{i=p}^{n} y_i}{n/2 + 1}$$

In the case of the open curves, the end points produce a discontinuity in tangent values. Since the end points do not have contour points on one of their sides, the corresponding extreme points of chords for points near an end point are calculated by taking the end point coordinates as many times as curve points lack on that side to complete the n neighbors. By this approach, tangent values near the

end points maintain their continuity, and every point of the contour, including the end points, has its corresponding tangent value.

In fact, each chord may not pass through each contour point (x_0, y_0), but the slope and angle of these chords are taken as the approximate values of tangent and tangent angles for each point of the curve. Keeping in mind that, in the limit, these chords would be the tangent line at (x_0, y_0) when (x_1, y_1) and (x_r, y_r) tend to $(x_0\ y_0)$.

Since the arctangent function only returns an angle in the range $(-\pi, \pi)$, any angle outside this range is wrapped around, resulting in an artificial discontinuity of the tangent directions. A normalization step [18] is performed, searching for discontinuities between two consecutive points greater than π or less than $-\pi$.

Segmentation Criterion

Let us examine the segmentation method based upon the extraction of constant curvature segments. It is obvious from the properties of differentiable functions that the curvature function $k(s)$ at position s on a constant curvature segment satisfies $dk(s)/ds = d^2\theta(s)/ds^2 = 0$.

To avoid the inherent differentiation noise in discrete signals, it is common to smooth $\theta(s)$ with a Gaussian filter, $G(s) = \exp(-s^2/2\sigma^2)$ [18]. In practice, Gaussian filtering eliminates the high-frequency noise in the digital curve, but low-frequency noise is still present, so causing the value of the above expression to oscillate about zero. Therefore, a margin of tolerance $\varepsilon > 0$ is set,

$$-\varepsilon < \frac{d^2G(s)}{ds^2} * \theta(s) < \varepsilon.$$

Curve points satisfying the above condition are labeled as constant curvature points. Each segment is defined by a set of contiguous points in the chain code representation of the contour with constant curvature. Similarly, the nonconstant curvature segments are defined by grouping contiguous nonconstant curvature points.

A special case in the previous formulation arises through the behavior of $d^2\theta/ds^2$ at corners or points with sudden variation in curvature. This case deals with the well-known rounding effect caused by data smoothing and finite differentiating [20]. Sharp changes in the tangent orientation $\theta(s)$ result in high curvature peaks in $k(s)$ which are zero-crossing points in the function dk/ds, similar to zero-crossings in edge detection theory [21]. These zero-crossing points are sometimes characterized as constant curvature points in the previous formulation, but from segments no longer than two or three points. Zero-crossing points are detected either by identifying these short constant curvature segments, placing the zero-crossing point at the point in this short segment with the nearest value of $d^2\theta/ds^2$ to zero, or from nonconstant curvature segments where values of $d^2\theta/ds^2$ of two consecutive points skip from under $-\varepsilon$ to over $+\varepsilon$, with ε being the above-mentioned tolerance margin.

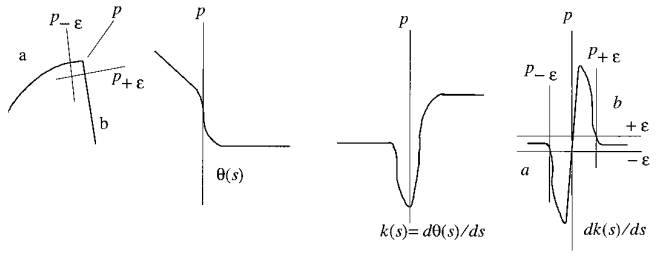


FIG. 1. Behavior of $\theta(s)$, $k(s)$, and $dk(s)/ds$ at corner surrounding.

To avoid the effect of the surrounding zero-crossing points, a break point is placed at the identified zero-crossings. As curvature in the neighborhood has been altered by smoothing, then points near zero-crossings are reconsidered, assigning points at each side of the zero-crossing to the nearest constant curvature segment along the chain code. Looking at Fig. 1, points between the end of segment a and the break point p will be assigned to segment a, and points on the other side of p will be assigned to segment b.

With this approach which combines the zero-crossing points with the constant curvature information, it is possible to avoid the local behavior in the neighborhood of corners which could lead to confused segmentation if only the constant curvature criterion were used. This treatment of points near corners avoids producing any extra small segments, and so produces larger segments resulting in a more compact segmentation.

GROUPING SEGMENTS

The main objective of this work is to recognize circles from segmented contours. Therefore, once segmented contours are available, the next step is to perform some kind of interpretation process, starting from the initial set of segments, to choose those segments that represent parts of the same circumference. The circle parameters can then be derived. The concept of segment grouping has been used for curve segmentation [16] to produce higher levels of contour representation, merging contiguous segments in several stages using a binary search algorithm.

To find the relations between segments from any part of a contour and even between different contours, an algorithm based on agglomerative hierarchical clustering [22] is used. In the algorithm, a criterion decided whether a segment, or group of segments, represents a circle, grouping them together.

The clustering algorithm consists of:

Data: t_i, $i = 1,..N_s$ initial segments produced by the contour segmentation.

Result: $SE = \{s_i, i = 1,..K\}$, where SE is the set of final K groups of segments.
Distance function $d(s_i,s_j)$ between two groups of segments $s_i,s_j \in SE$.
Parameter D: maximum distance between two groups of segments.
Algorithm:
1. Initialization: $SE = \{s_i = t_i , i = 1,...,N_s\}$.
2. Search for s_i, s_j which $d(s_i,s_j)$ is minimized, $i \neq j$.
3. if $d(s_i,s_j) < D$ then
   begin
      Delete s_i,s_j from SE.
      Add to SE the resulting group of the union $s_i \cup s_j$.
      Skip to step 2.
   end
else END.

Starting from the initial set of segments, and initially considering each segment to be the only element of its own group, the algorithm previously described chooses, at each iteration, two group of segments to be merged according to the value of the function $d(s_i, s_j)$. When the termination condition is satisfied, the result is a set of groups of segments where each group may represent part of the same circumference. The information provided in this way is more structured and precise than, for example, the result of backmapping using the Gerig and Klein Hough transform method, which associates a set of points with a circle, but it occasionally introduces noisy points, and points are not organized in segments (see [24] for some examples).

Function $d(s_i, s_j)$ gives a value assessing the possibility that the groups of segments s_i, s_j may be part of the same circle. To work out this value, it is supposed that both groups of segments are part of the same circumference. The parameters of the circle are then estimated using every point of all segments in both candidate groups s_i, s_j. Once the circle parameters are estimated, an error measure is calculated to assess how well the points in the segments fit the circle with the estimated parameters. This error defines the value of the function $d(s_i, s_j)$.

PARAMETER ESTIMATION

As pointed out previously, every time two groups of segments (s_i, s_j) are considered for merging, the parameters of the possible circle they represent are estimated. Least-squares fit is one of the most commonly used methods for this parameter estimation. The disadvantages of this technique are well known given outliers or data with a moderate level of noise. Some authors have proposed the use of robust estimation methods using error metrics to avoid the effect of points which lie far from the curve [8]. Rejection of outliers can also be performed by means of statistical methods, to test which points from the data set are corrupting the parameter estimation [23]. On the other hand, one of the advantages both of least-squares procedures and robust estimation methods is the capability of subpixel accuracy [8].

In the problem we are dealing with here, the purpose of parameter estimation is changed slightly. In the described procedure, parameter estimation is used to calculate how much error is introduced in the distribution of segment points of a group when we try to add another group of segment points to fit the data to a single circular contour.

Therefore, at each iteration of the algorithm only the two groups of segments which introduce a minimum error with respect to the new distribution are merged, rejecting possible mergings which can lead to the introduction of outliers in the distribution. Thus, instead of trying to detect outliers in a given set of data, the algorithm starts from an initial set of assumed noise free data, and at each iteration the data distribution grows from noncorrupted data, rejecting outliers.

To estimate the circle parameters, we propose a method with two steps, in order to simplify the procedure by solving a set of linear equations. In the first step the coordinates of the center are calculated, and in the second step the radius is estimated using the center coordinates which have been found in the previous step.

We use the property that for every point on a circumference, the radius and the tangent are perpendiculars. Thus, a point (x, y) on the circumference of radius r and center (x_c, y_c) and a vector $\mathbf{m} = (m_x, m_y)$ in the direction of the tangent line at this point (x, y) satisfy the condition

$$\mathbf{m} \cdot (x - x_c, y - y_c) = 0,$$

where $\cdot$ denotes the scalar product, and vector $(x - x_c, y - y_c)$ is defined by points (x_c, y_c) and (x, y). Components (m_x, m_y) of vector $\mathbf{m}$ are calculated from points (x_l, y_l) and (x_r, y_r). Thus, $m_x = x_r - x_l$ and $m_y = y_r - y_l$.

If there are a total of N points in the segments of a group, we will have N equations like the previous one. In practice, because experimental data do not exactly fit to the ideal circumference they represent, the result of the above scalar product is usually nonzero,

$$\mathbf{m}_i \cdot (x_i - x_c, y_i - y_c) = f_i, i = 1, \ldots, N,$$

where f_i is a value close to zero for each point.

The point (x_c, y_c) which best approximates the center of the circumference the N data points of the distribution form minimizes the expression

$$E = \frac{1}{N} \sum_{i=1}^{N} (f_i)^2.$$

Using the expression for f_i and applying partial derivatives with respect to x_c and y_c, and setting them to zero, we obtain a linear system of two equations. Solving the unknown variables x_c and y_c, we obtain

$$x_c = \frac{\sum_i m_{x_i} A_i \sum_i m_{y_i}^2 - \sum_i m_{x_i} m_{y_i} \sum_i m_{y_i} A_i}{\sum_i m_{x_i}^2 \sum_i m_{y_i}^2 - \left(\sum_i m_{x_i} m_{y_i}\right)^2}$$

$$y_c = \frac{\sum_i m_{y_i} A_i \sum_i m_{m_{x_i}}^2 - \sum_i m_{x_i} m_{y_i} \sum_i m_{x_i} A_i}{\sum_i m_{x_i}^2 \sum_i m_{y_i}^2 - \left(\sum_i m_{x_i} m_{y_i}\right)^2},$$

where $A_i = x_i m_{x_i} + y_i m_{y_i}$, and $\mathbf{m}_i = (m_{x_i}, m_{y_i})$ is the vector in the direction of the tangent at a point (x_i, y_i), $i = 1, \ldots, N$, and all the summation terms are over the $i = 1, \ldots, N$ points of the distribution.

Once the center of the circumference (x_c, y_c) has been estimated, the radius r can be estimated by the average distance of each point (x_i, y_i) of the distribution to this center, therefore,

$$r = \frac{1}{N} \sum_{i=1}^{N} ((x_i - x_c)^2 + (y_i - y_c)^2)^{1/2}$$

and the error of the fit, d_e, is taken as the variance of the distances of each point to the estimated center. Since the mean radius is the above-mentioned r, the expression of the error d_e is

$$d_e = \frac{1}{N} \sum_{i=1}^{N} \{[(x_i - x_c)^2 + (y_i - y_c)^2]^{1/2} - r\}^2$$

PARTIALLY OCCLUDED CONTOUR ESTIMATION

The result of the procedure described in the previous section is a set of groups of segments where each group represents circularly distributed dot points. In some applications it is useful to know the proportion of present (or occluded) circular contour for each circle found. This proportion p_c can be defined from the simple summation of angles swept by each segment of a group with respect to its center,

$$p_c = \frac{1}{2\pi} \sum_{k=1}^{S} \alpha_k,$$

where S is the number of segments in a given group, and α_k is the angle swept by the points of the segment with respect to the center (x_c, y_c) of the circumference associated to the group (Fig. 2). This approach is more accurate than, for example, taking p_c as $p_c = N/2\pi$, with N being the number of contour points representing the circle of radius r. For instance, if segments represent a complete circle, it is very unlikely that p_c equals 1, and even it can be larger. However, with the above-mentioned approach, p_c is exactly 1.

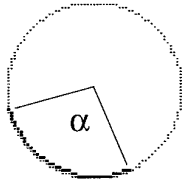


FIG. 2. Angle α swept by a segment with respect to the center.

PRACTICAL CONSIDERATIONS

In order to implement the method, some parameters in the algorithm must be quantified. In the case of the contour segmentation procedure, we must choose:

—σ, the standard deviation of the Gaussian filter. As in all smoothing processes, this parameter is related to the level of detail in the contour segmentation. Small values of σ lead to a very fragmented contour and short segments. Nevertheless, as it is shown later, this parameter can vary within a relatively wide band of values without substantially changing the contour segmentation.

—n, the number of neighbors involved in the tangent calculation. Increasing this parameter produces a smoothing effect, since the n neighborhood is effectively an averaging filter. Decreasing σ and increasing n has a similar effect to increasing σ and decreasing n. Moreover, experiments show that it is advisable to fix n to a small value, for example, 3, in order to maintain the local accuracy of the tangent.

—ε, margin of tolerance of the second derivative around zero. Because this margin depends on the level of noise, it may have some dependence on the Gaussian filter, but, as is shown later, the margin of tolerance can remain constant even if the standard deviation σ of the smoothing filter changes significantly.

From the previous parameters, n and ε can remain constants in a wide range of different situations, and only σ is the main parameter to be set. In the examples which are shown later, the typical values of the above-mentioned parameters were $\sigma = 3.0$, $n = 3$, and $\varepsilon = 0.15$.

For the clustering algorithm, a value for the maximum distance D between two groups of segments must be chosen. The value of D depends on the expected accuracy level of experimental contour data to fit an ideal circle, and this value is independent of the contour segmentation

procedure. A typical value used in the examples in the next section is $D = 2.25$, which means that standard deviation of the distance from each point of the distribution to the center of the circumference is $D^{1/2} = 1.5$ pixels. Given D, different values could lead to slight changes in p_c for the same object, since a lower value of D means more accuracy in segments to fit an ideal circle. Increasing D may occasionally add some other segment to a group, changing slightly the proportion p_c.

Note that the output of the algorithm is a set of groups of segments with their corresponding circle associated. Also associated with each group is a proportion of circular contour p_c from which the circumference has been calculated. This value together with the radius r can be used for further decisions in another level of the classification scheme for the objects these circular contours represent. The thresholds on p_c and r are not parameters of the algorithm. The ranges of r and p_c depend on the application. In the experiments carried out in this paper it was considered that a group of segments represents a circumference if their segments represented at least 30% of the total circular contour. Groups of segments with an associated radius of less than 10 pixels were rejected.

COMPUTATIONAL COST

Memory requirements are clearly proportional to the number of contour points in the image. This memory is used to store the chain code representation of every contour. Computational time is lineal with the number of contour points for the contour segmentation process. Moreover, it is $O(N_s^2)$ because of the clustering structure of the segment grouping algorithm, with N_s equal to the number of segments resulting from the segmentation.

A modification of the clustering algorithm for achieving a lower cost consists of merging the first two groups whose distance is less than the maximum established D, instead of taking the pair of groups whose distance is the minimum of all possible pairs. By this approach computational time in clustering is reduced significantly. Results using the original algorithm and the above-mentioned modification are very similar, although the result with the original form is better when the result of segmentation leads to higher fragmented contours.

EXPERIMENTAL RESULTS

The described method was tested using real images from two different situations in real applications. It was also tested on synthetic images of circles generated by software to check the accuracy in measuring the circle parameters. The algorithm was run on a set of synthetic images of circles with radius between 5 and 50 pixels, incrementing the radius 5 pixels each time.

Due to the procedure utilized to evaluate the circle parameters, results are given in subpixel accuracy. To evaluate this subpixel accuracy first, circles were drawn in the image frame memory from integer values of parameters. In this case the results of the estimated parameters from the digital data had an average error of 0.01 pixel. In a second set of measurements, circle parameters were given in real values to draw the digital circle in the frame memory. In this case the radius and center coordinates were incremented 0.5 pixels from the previous experience. The average error found in this case for the radial measures was 0.04 and for the center coordinates 0.47.

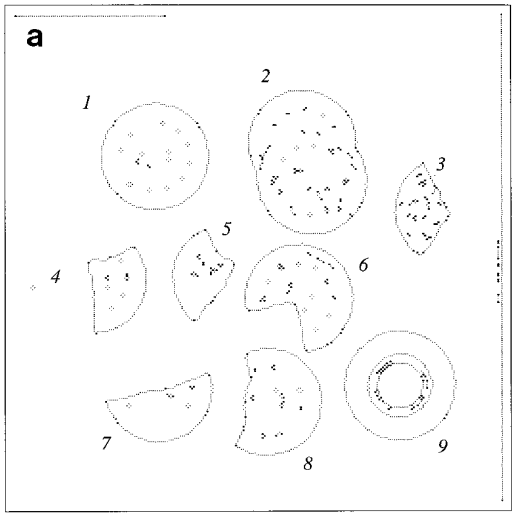
Note that circle parameters are estimated from the values of tangent angles at each point of the contour. This result shows that the definition introduced for the tangent calculation is suitable and sufficiently accurate.

The biscuit images were the first real images to be used. In agro-food industries, one of the main processes is product grading. The aim was to locate and identify broken biscuits in order to reject bad items. Broken biscuits were identified by evaluating the proportion of present circular contour of each object. Circular objects with an estimated proportion less than 95% of present contour were considered broken biscuits. Figure 3a shows the result of contour segmentation of a 256×256 image of biscuits with several degrees of missing circular parts. Contours were extracted with a Canny edge operator, using a single threshold value in the gradient magnitude of 30, and a standard deviation of 0.15 in the Gaussian filter. After contours were stored in a chain code, they were segmented using $\sigma = 3.0$.

Dark points in Figs. 3a, 4–6, and 7b denote break points between segments. Because the segmentation procedure assigns each contour point to some segment, these break points represent the end point of each segment, following the contours clockwise. All end points in open contours also have been marked as break points, since they are always limits of a segment.

Figure 3b is the result of the algorithm showing the corresponding circumference to the circle parameters found from the segments in Fig. 3a. In this figure, darker pixels represent the contour points from which an ideal circle (clearer pixels) has been found. The algorithm can deal with circles having more than 50% occluded or missing contour (items 3, 4, and 5 in Fig. 3a and most of the circles in Fig. 7). The correct treatment for overlapping objects is also shown (item 2 in Fig. 3a, and items 2 and 4 in Fig. 7b), as well as concentric circles from the extra item which has been placed in the scene to test the method for this particular case (item 9 in Fig. 3a).

Occasionally, very small segments of 3 or 4 pixels length in Fig. 3b have not been assigned to the natural corresponding circles. These mistakes are produced in the early iterations of the clustering algorithm because they are usually grouped with other small segments. They can fit any circu-



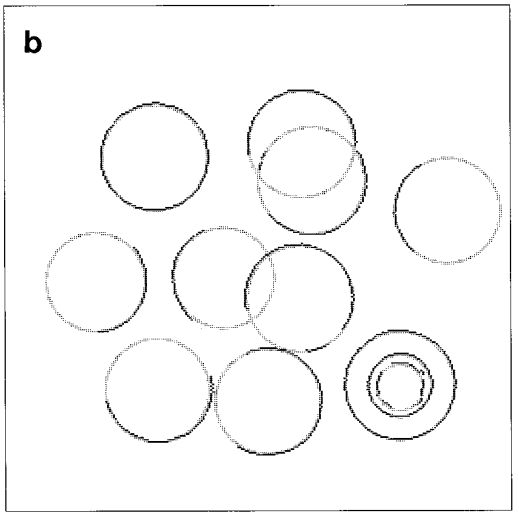


FIG. 3. (a) Contour segmentation using $\sigma = 3.0$ (b) Result of the circle-finding algorithm.

lar representation with very low errors due to their small number of points. These errors could be overcome through an initial stage by grouping the connecting segments into larger primitives before the clustering algorithm, using certain techniques as in West and Rosin [16]. Losing these segments could lead to a poor performance of the algorithm, if contours are excessively fragmented due to an unsuitable election of σ in the filtering.

In respect to the contour segmentation procedure, looking at Figs. 3a and 7b, we note how the proposed segmentation method has correctly separated segments at corners, overcoming the effect of corner surrounding using the zero-crossing criterion together with the constant curvature criterion.

Figure 5 shows the segmented contours of the biscuit image used in Fig. 3; however, contour points missed by

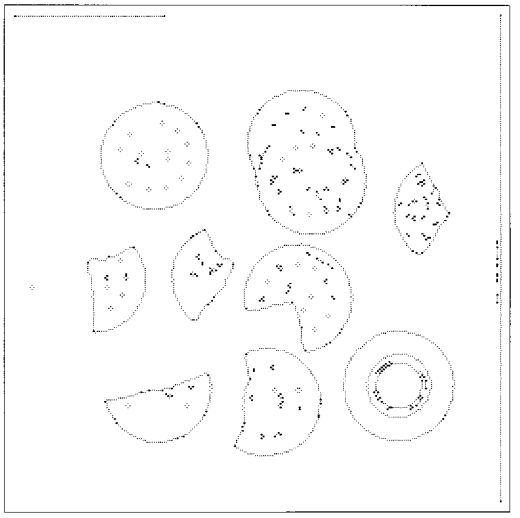


FIG. 4. Contour segmentation of the same image of Fig. 3a but using $\sigma = 2.0$.

the edge detector were replaced manually. Figure 3a has the same biscuit contours but some of them are cut at certain points due to errors in edge detection. Comparing both segmentations, we note how when there are cuts in edges, the segmentation approximately gives the same result except that a breakpoint at each segment cut has been placed, as in Fig. 3a, because of lack of contour connectivity. This result shows the stability of the contour segmentation procedure which remains invariant in cases of error due to edge detection.

Figures 3a, 4, and 6 are segmented results of the same contours but with different degree of smoothing with σ values of 3.0, 2.0, and 4.0 respectively. Note that segments

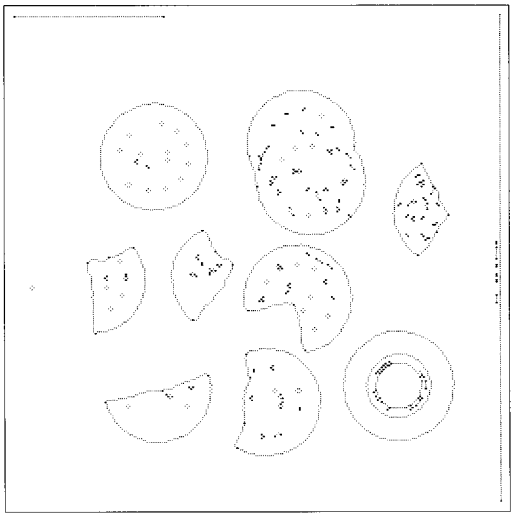


FIG. 5. Contour segmentation of the same image of Fig. 3a but without cuts in contours.

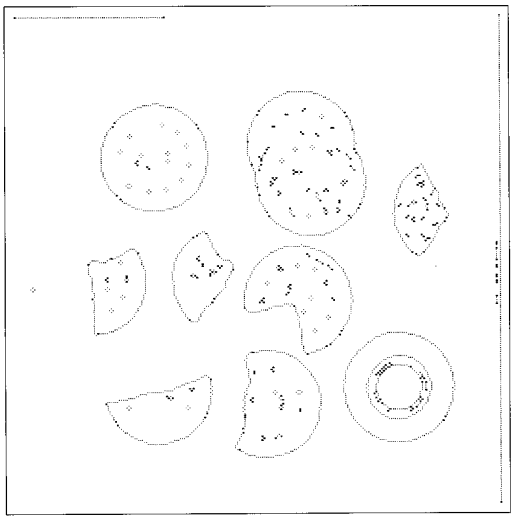


FIG. 6. Contour segmentation of the same image of Fig. 3a but using $\sigma = 4.0$.

extracted using $\sigma = 3.0$ (Fig. 3a) are approximately the same as segmentations of Figs. 4 ($\sigma = 2.0$) and 6 ($\sigma = 4$). This result has been obtained without changing any other parameter of the algorithm, as explained in previous sections. Small changes among the results in Figs. 3a, 4, and 6 can be mainly found at end segment points placed at sharp corners, which have been shifted in Fig. 6 ($\sigma = 4.0$) with respect to Figs. 3a and 4, due to the bigger smoothing effect in Fig. 6. Therefore, the contour segmentation method is robust because significant variations of the smoothing parameter lead to similar segmentations without changing any other parameter of the algorithm.

The other example of real images used was images from orange trees. The aim in this case was to locate the center of partially occluded fruits in order to direct a robotic arm to pick them. The algorithm was applied to contours of regions obtained from color-segmented images. Figure 7a shows a 256×256 gray level image of an orange tree scene. Figure 7b represents the contour segmentation obtained from region borders of the color-segmented image, and Fig. 7c shows the result of the partial circle recognition. Note that the algorithm can deal with noisy contours, since fruits are not exactly round and contours have appreciable noise. Note also how the method can detect centers of overlapping fruits (items 2 and 4) or fruits partially occluded by leaves (item 5).

We must point out that Fig. 7c shows the circles found from contours in Fig. 7b, and each one does not represent a fruit. Note how the contour segments in item 5 corresponding to the leaf border has generated a circle, and that it really fits. The same effect produces the concave part of contour in item 1, due to the overlapping effect with the other fruit. These concave parts have also generated a circle, but it does not correspond to any fruit. In fact, during the application of this algorithm to locate centers of fruits, circles found from concave contour parts were rejected, since they do not represent real fruit borders.

CONCLUSIONS

A method for partial circle recognition using information from segmented contours has been described. The method has been presented in two steps: an initial segmentation of contours, decomposing them into constant curvature segments, and a second step of contour segment grouping into circle candidates.

The main features of the constant curvature segmentation are:

—The combination of a constant curvature criterion with a zero-crossing criterion to overcome the smoothing effects at corners. The combination of both criteria has proven in performance to produce larger primitives or segments, avoiding overfragmentation at corners.

—Stability of the algorithm in cases of error of extracted contours, preserving the segmentation in presence of cuts or missing parts.

—Robustness, because variations in the smoothing parameter do not affect the result significantly without changing any other parameter of the algorithm.

The constant curvature segmentation provides an adequate framework for the segment grouping algorithm to deal with partially circular contours. The algorithm for circle recognition has the following features:

—The use of contour segments as primitives instead of individual points, thus, taking advantage of relationships among them.

—The accurate calculation of circle parameters (center and radius) via a minimization criterion using the tangent at each point.

—The capability to deal with circular contours occluded by more than 50%, thus, maintaining accuracy in parameter estimation.

—The ability to estimate the proportion of contour of the recognized circles, resulting in identifying and locating the circular segments.

—Low memory requirements, with lineal cost in respect to the number of contour points, and an acceptable computing time.

Although the shortcoming of losing some small segments during the clustering algorithm does not affect final results, it could be resolved using an initial step of relating connected segments before the clustering. This improvement would allow more accuracy in estimating the proportion of contour present.

Further extensions of the presented method for recog-

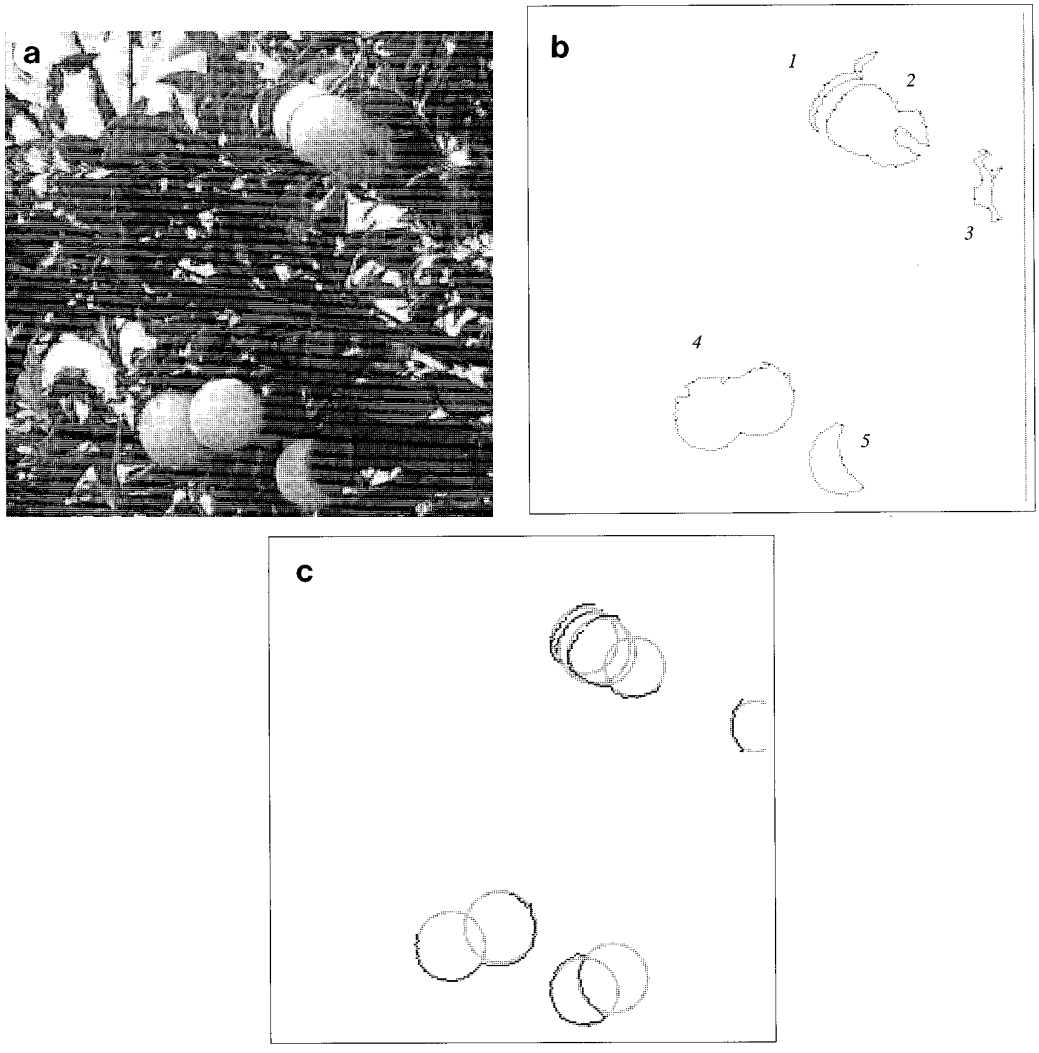


FIG. 7. (a) Gray level image of an orange tree scene. (b) Contour segmentation of orange color regions using $\sigma = 3.0$. (c) Result of the circle-finding algorithm.

nizing straight lines from contour segments, by keeping the structure of the method but changing the criterion function, are being considered. This function would assess if two segments or groups of segments can be merged, and would group those segments which may belong to the same line.

ACKNOWLEDGMENT

This work has been partially supported by the EUREKA-176 Project and INIA-9513 Project.

REFERENCES

1. R. O. Duda and P. E. Hart, Use of the Hough transformation to detect lines and curves in pictures, *Commun. ACM,* **15,** 1972, 204–208.
2. H. K. Yuen, J. Princen, J. Illingworth, and J. Kittler, Comparative study of Hough transform methods for circle finding, *Image Vision Comput.* **8**(1), 1990, 71–77.
3. J. Illingworth and J. Kittler, The adaptive Hough transform, *IEEE Trans. Pattern Anal. Mach. Intell.* **9**(5), 1987, 690–697.
4. E. R. Davies, A modified Hough scheme for general circle location, *Pattern Recognit. Lett.* **7**(1), 37–44.
5. G. Gerig and F. Klein, Fast contour identification through efficient Hough transform and simplified interpretation strategy, in *Proceedings, 8th International Joint Conference Pattern Recognition, Paris, France, 1986,* pp. 498–500.
6. E. R. Davies, A high speed algorithm for circular object detection, *Pattern Recognit. Lett.* **6**(5), 1987, 323–333.
7. A. M. Wallace, Greyscale image processing for industrial applications, *Image Vision Comput.* **1**(4), 1983, 178–188.
8. G. A. Jones, J. Princen, J. Illingworth, and J. Kittler, Robust estimation of shape parameters, in *Proceedings of BMVC'90, Oxford, 1990,* pp. 43–48.
9. M. A. Fischler and R. C. Bolles, Perceptual organization and curve

partitioning, *IEEE Trans. Pattern Anal. Mach. Intell.* **8**(1), 1986, 100–105.

10. F. Mokhtarian and A. Mackworth, Scale-based description and recognition of planar curves and two-dimensional shapes, *IEEE Trans. Pattern Anal. Mach. Intell.* **8**(1), 1986, 34–43.
11. N. Katzir, M. Lindenbaum, and M. Porat, Planar curve segmentation for recognition of partially occluded shapes, in *Proceedings IEEE of CVPR, 1990,* pp. 842–846.
12. J. A. Ventura and J. M. Chen, Segmentation of two-dimensional curve contours, *Pattern Recognit.* **25**(10), 1992, 1129–1140.
13. J. G. Dunham, Optimum uniform piecewise linear approximation of planar curves, *IEEE Trans. Pattern Anal. Mach. Intell.* **8,** 1986, 67–75.
14. P. L. Rosin and G. A. West, Segmentation of edges into lines and arcs, *Image Vision Comput.* **7,** 1989, 109–114.
15. A. Albano, Representation of digitized contours in terms of conic arcs and straight-line segments, *Comput. Vision Graphics Image Process.* **3,** 1974, 23–33.
16. G. A. W. West and P. L. Rosin, Multistage combined ellipse and line detection, in *Proceedings of BMVC'92, 1992,* pp. 197–206.
17. D. M. Wuescher and K. L. Boyer, Robust contour decomposition using a constant curvature criterion, *IEEE Trans. Pattern Anal. Mach. Intell.* **13**(1), 1991, 41–51.
18. H. C. Liu and M. D. Srinath, Partial shape classification using contour matching in distance transformation, *IEEE Trans. Pattern Anal. Mach. Intell.* **12**(11), 1990, 1072–1079.
19. T. F. Knoll and R. C. Jain, Recognizing partially visible objects using feature indexed hypotheses, *IEEE J. Rob. Autom.* **RA-2**(1), 1986, 3–13.
20. L. D. Cai, J. Porrill, S. B. Pollard, J. E. W. Mayhew, and J. P. Frisby, Segmentation of planar curves using local and global behaviour analysis, in *Proceedings of BMVC'90, Oxford, 1990,* pp. 247–252.
21. D. Marr and E. Hildreth, Theory of edge detection, *Proc. R. Soc. London* **207,** 1980, 187–217.
22. R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis,* Wiley, New York, 1973.
23. P. H. S. Torr and D. W. Murray, Statistical detection of independent movement from a moving camera, *Image Vision Comput.* **11**(4), 1993, 180–187.
24. H. K. Yuen, J. Illingworth, and J. Kittler, Detecting partially occluded ellipses using the Hough transform, *Image Vision Comput.* **7**(1), 1989, 31–37.