ELSEVIER

Image and Vision Computing 14 (1996) 71–83

Segmentation of planar curves into circular arcs and line segments

Jen-Ming Chen[a,1], Jose A. Ventura[b,2], Chih-Hang Wu[b]

[a]*Department of Information Management, The National Central University, Chung-Li 32054, Taiwan*
[b]*Department of Industrial and Management Systems Engineering, 207 Hammond Building, The Pennsylvania State University, University Park, PA 16802, USA*

Received 22 September 1993; revised 25 April 1995

Abstract

In many applications, like shape analysis, it is necessary to decompose an object contour into straight-line segments and circular arcs, because many man-made objects (especially machined parts) are composed of these two types of geometric entities. This paper presents a procedure for segmenting a planar curve into lines and arcs, in which the number of entities (or break points) of the curve is given. This procedure can be divided into two stages: (1) to obtain a starting set of break points, and determine the approximation functions (lines and arcs) for the data intervals that are separated by the break points; and (2) to adjust the break points until the error norm is locally minimized. The first stage is based on the detection of significant changes in curvature using the chain-code and differential chain-code techniques, and the second stage is an optimization curve/line fitting scheme. A computational comparison with a modified dynamic programming (MDP) approach shows that the proposed procedure obtains near optimal solutions (relative errors less than 1%) for all the test problems, and requires less than 1.2% of the computational time needed by the MDP approach.

Keywords: Planar curves; Circular arcs; Line segments; Segmentation

1. Introduction

Segmentation of digitized planar curves is one of the most important elements in early image processing, because a segmented curve can describe the object profile in a meaningful and compact form to facilitate higher level vision processing, such as pattern recognition and shape analysis. Many curve segmentation algorithms have been developed in the past two decades, most of which use straight lines to approximate the edge pixels in an image (see Teh and Chin [1] and Ventura and Chen [2], and their references). However, piecewise linear approximation of digitized curves is scarcely useful for further image analysis of man-made objects that are composed of quadric arcs. In industrial applications, for instance, more than 90% of the machined parts are bounded by circular arcs and straight line segments, because these two types of geometric elements are the most frequently used in engineering design and manufacturing [3]. Hence, we are interested in developing a procedure for segmenting digitized curves by dividing the edge pixels of an image into intervals where the data can be approximated by either circular arcs or straight-line segments.

The problem of finding the best piecewise higher order (degree at least 2) polynomial approximation can be defined in two ways [4]: (1) to find a minimal number of intervals, provided that the error norm does not exceed a preset tolerance; and (2) to minimize the error norm for a given number of intervals (n). The first type of problem has received a large amount of attention in vision research, because it can provide fundamental image processing functions, such as data compression, noise filtering and feature extraction, for a wide range of computer vision applications. Several algorithms have been developed using the first definition [5–8]. The procedure developed in this paper belongs to the second problem domain. This type of problem is particularly important to the production part inspection using machine vision systems, because most industrial vision systems are model-based [9–11], in which the inspection involves matching the segmented image with a predefined model of the object. Therefore, the number of

[1] email: jmchen@im.mgt.ncu.edu.tw
[2] email: javie@engr.psu.edu

geometric primitives (n) of an object is the *a priori* knowledge, based on which the subsequent shape analysis can be performed in a more reliable manner. In addition, to meet the requirements of mass production and enhanced quality control in computer integrated manufacturing environments, the computational speed and accuracy are two crucial performance measures of the segmentation algorithms. The proposed technique is a two-stage procedure which has been designed to preserve both speed and accuracy properties. Initially, a set of break points is identified in a fast computation fashion. Next, an optimization approach is applied to adjust these break points in order to improve the overall accuracy.

The problem of interest is formally described as follows. For a given number of primitives (n), which can be arcs or line segments, the objective is to partition the set of boundary data (arranged in counterclockwise sequence):

$$S = \{p_j = (x_j, y_j) | j = 1, 2, \ldots, m\} \tag{1}$$

into n data subsets:

$$S_i = \{(x_{i,j}, y_{i,j}) | j = 1, 2, \ldots, m_i\}, \quad i = 1, 2, \ldots, n \tag{2}$$

so that the sum of squares of the Euclidean distances between the points in each subset S_i, $i = 1, 2, \ldots, n$, and the corresponding primitives is minimized. Note that p_j is a neighbour of p_{j-1} and p_{j+1} (modulo m), and let $u_i = (x_{i,1}, y_{i,1})$ be the first point of S_i, representing the break point between S_i and S_{i-1} (modulo n). In addition, each subset of data can be approximated by a line ($\mathcal{L}_i$) or an arc ($\mathcal{C}_i$), i.e.:

$$S_i = \begin{cases} \mathcal{L}_i, & \text{for } i \in \mathcal{I}_l, \\ \mathcal{C}_i, & \text{for } i \in \mathcal{I}_c, \end{cases} \tag{3}$$

where $\mathcal{I}_l$ and $\mathcal{I}_c$ are the sets of indices of the straight-line segments and circular arcs, respectively. Let $\mathcal{I} = \mathcal{I}_l \cup \mathcal{I}_c$, so that:

$$n_l + n_c = n \tag{4}$$

where n_l, n_c and n are the cardinalities of sets $\mathcal{I}_l$, $\mathcal{I}_c$ and $\mathcal{I}$, respectively. It is noteworthy to mention that:

$$S = S_1 \cup S_2 \cup \ldots \cup S_n \tag{5}$$

$$m = m_1 + m_2 + \ldots + m_n \tag{6}$$

and

$$S_k \cap S_l = \emptyset, \quad \text{for all } (k, l),\ k \neq l \tag{7}$$

where equations (5) and (6) indicate that the union of all subsets of data is equal to the set of all boundary data, and the system of equations (7) shows that all the subsets are mutually exclusive. Hence, the objective of this problem is equivalent to identifying the best set of break points, u_i, $i = 1, 2, \ldots, n$, and classifying each data subset S_i into $\mathcal{L}_i$ or $\mathcal{C}_i$ for $i \in \mathcal{I}$, so that the defined error norm is minimized.

In the case that the segmented output is to be used for model-based matching, the matcher will know how many straight-line segments and circular arcs to expect. Thus, both n_l and n_c are given as the *a priori* knowledge. The segmentation can be performed more accurately and reliably by choosing the best line segment set and arc set with cardinalities u_l and u_c, respectively.

The remainder of this paper is organized as follows. The next section deals with the detection of break points, using the chain-code and differential chain-code techniques. The data intervals divided by the n break points are then classified into arcs and lines. An optimization data fitting scheme which is used to adjust the set of break points, until the optimal solution is achieved is then presented. Experimental results are then presented, in which a modified dynamic programming (MDP) approach is used for comparison. The MDP is a global optimization approach which can be used to verify the performance of our algorithm. Concluding remarks are provided.

2. Detection of break points

The objectives of the first stage of the proposed procedure are to obtain a starting set of break points, and to determine the approximation functions (lines and arcs) for the data intervals that are separated by the break points. The results of this stage is used as the input data for the subsequent procedure of break point adjustment (see Fig. 1).

The break points on a closed curve that is composed of arcs and line segments represent the points of significant changes in curvature. Asada and Brady [12] classified the isolated curvature changes into two types: the corner and the smooth join. Their mathematical properties are described below:

- *Corner*: is an isolated curvature change, for which the tangent to the contour is discontinuous; and
- *Smooth join*: is an isolated curvature change, for which the tangent is continuous, but the curvature is discontinuous.

Fig. 2 shows examples of corners and smooth joins on a curve. Hence, the detection of break points is equivalent to the detection of corners and smooth joins on the curve. Based on this classification, we propose the differential chain-code technique to identify the corners, and use the chain-code representation to linearize the arcs, so that the smooth joins on the curve are transformed into corners which can be detected by using a piecewise linear approximation procedure. After the breakpoints are completely identified, the resulting data intervals can be classified into arcs and lines, by estimating the slopes of the approximation segments in the chain-code domain.

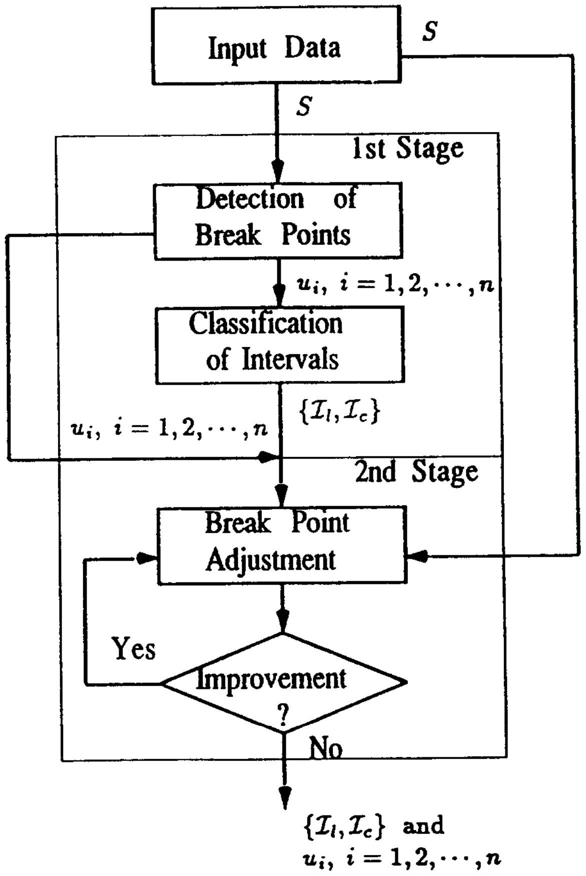
Input Data
S
S
1st Stage
Detection of Break Points
$u_i,\ i=1,2,\cdots,n$
Classification of Intervals
$u_i,\ i=1,2,\cdots,n$
$\{\mathcal{I}_l,\mathcal{I}_c\}$
2nd Stage
Break Point Adjustment
Yes
Improvement ?
No
$\{\mathcal{I}_l,\mathcal{I}_c\}$ and $u_i,\ i=1,2,\cdots,n$

Fig. 1. Overview of the two-stage procedure.

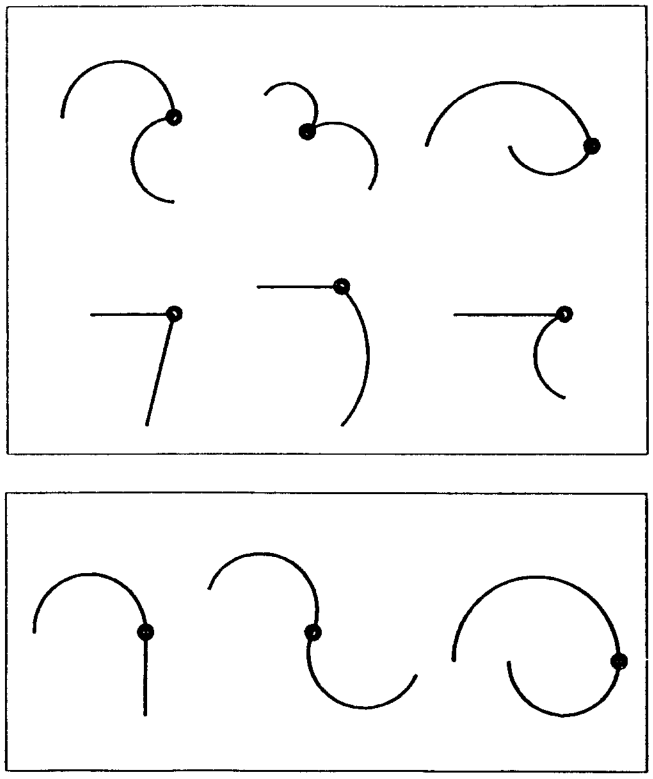

Fig. 2. Two types of isolated curvature changes: (a) corners; (b) smooth joins.

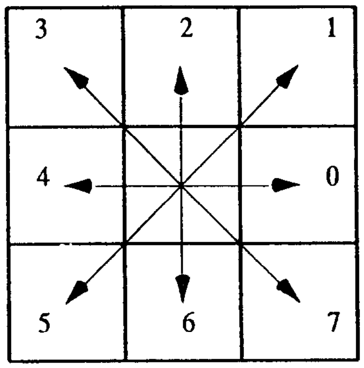

Fig. 3. The 8-directional code.

2.1. Shape coding schemes

In the chain-code representation, a 'code' that represents the tangent direction is assigned to each point on the curve, and hence the differential chain-code is the change of tangent directions between two adjacent points. The chain-code and differential chain-code representation schemes have been used by Sirjani and Cross [13] and Baruch and Loew [6], respectively, for segmenting digitized planar curves. Both approaches use the 8-directional code to represent the tangent at a point, as shown in Fig. 3, based on which a closed contour is partitioned into a minimal number of intervals, provided that the defined error is less than or equal to a preset tolerance. Further remarks and investigations of these coding schemes are given elsewhere [14,15].

Using these coding schemes for curve segmentation offers several merits, including simplicity, speed and reduction in memory requirements. However, there are generally two limitations associated with this technique: (1) a limited number of directions are used to represent the tangent at a point; the 8-directional code is the most common; and (2) they are very sensitive to the noise (perturbation of boundary data). To overcome these drawbacks, we use noise-filtering techniques to smooth the data, and represent each code by a larger number of directions.

The techniques for noise-filtering can be operated either in the spatial domain (e.g. neighbourhood averaging) or in the frequency domain (e.g. low-pass/band-reject filtering). Noise-filtering in the spatial domain uses neighbourhood spatial coherence and neighbourhood pixel value homogeneity as its basis. It detects lack of coherence and either replaces the incoherent pixel value by something more spatially coherent by using some or all of the pixels in a neighbourhood containing the given pixel, or they average or smooth the pixel value with others in an appropriate neighbourhood. The neighbourhood averaging is the simplest approach of noise-filter in the spatial domain. It generates a smoothed edge point (p_j) whose coordinates (x_j, y_j) are obtained by averaging the coordinates of a predefined neighbourhood of the point, i.e.:

$$x_j = \frac{1}{2h+1} \sum_{j-h \le l \le j+h} x_l$$
$$y_j = \frac{1}{2h+1} \sum_{j-h \le l \le j+h} y_l \tag{8}$$

where h is the predefined half-width of the neighbourhood at p_j.

The application of neighbourhood averaging has the effect of defocusing the boundary. Significant curvature changes become attenuated. All other things being equal, the larger the neighbourhood of the filter, the greater the defocusing action. The proper determination of neighbourhood size depends upon the correct balance between the better job of noise removal that larger neighbourhood operators can provide and the loss of detail that this entails. Determination of the correct balance depends upon having a model that relates the observed neighbourhood pixel values to the true underlying neighbourhood pixel values and the effect the noise has on the latter.

The boundary perturbations and the significant curvature changes on a digitized curve contribute heavily to the mid- and high-frequency contents, respectively, of its Fourier transform. Since the significant curvature changes are the desired features, the curve therefore can also be smoothed, yet keep the features, via the frequency domain by attenuating a specified range of frequency components in the transform of the closed contour.

The number of directions used to represent the code depends on the supporting length (k) used to compute the tangent at each point. For example, the 8-directional code shown in Fig. 3 has a supporting length $k = 1$. In the proposed procedure, the following coding schemes have been employed:

- k chain-code: let the sequence of m data points (S) describe a closed curve. The chain-code of S consists of the sequence:

$$\zeta_{j,k} = \tan^{-1}\left[\frac{y_j - y_{j-k}}{x_j - x_{j-k}}\right], \quad j = 1, 2, \ldots, m \tag{9}$$

  where $\zeta_{j,k}$ represents the tangent angle (modulo 2π) at p_j with a supporting length k.
- k differential chain-code: using the definition above, the differential chain-code of S consists of the sequence:

$$\delta_{j,k} = \tan^{-1}\left[\frac{y_{j+k} - y_j}{x_{j+k} - x_j}\right] - \tan^{-1}\left[\frac{y_j - y_{j-k}}{x_j - x_{j-k}}\right], \quad j = 1, 2, \ldots, m \tag{10}$$

  where $\delta_{j,k}$ is the change of tangent angles (modulo 2π) at p_j with a supporting length k.

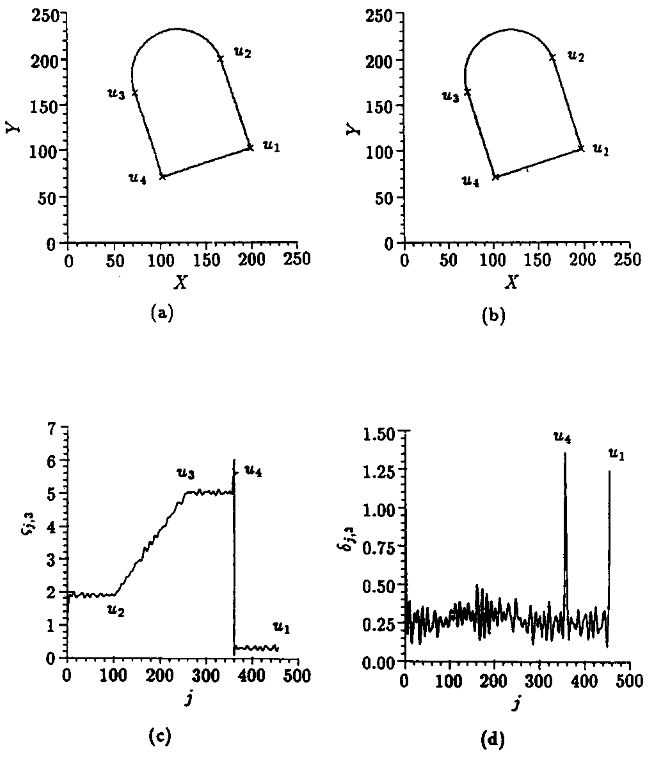

Fig. 4. A mouse-like shape. (a) original data; (b) smoother data; (c) k chain-code ($\zeta_{j,k}$); and (d) k differential chain-code ($\delta_{j,k}$), where $k = 3$, $n = 4$, $m = 458$, u_i, $i = 1,2,3,4$, are the break points, and u_1 is the starting point of the closed contour.

The mouse-like shape of Fig. 4 illustrates the effect of data smoothing through the band-reject filtering, and the sequences derived by using the proposed coding schemes, where (a) displays the original data, (b) presents the smoothed data, and (c) and (d) show the chain-code and differential chain-code sequences, respectively, with $k = 3$. The four break points of the shape are denoted by u_i, $i = 1,2,3,4$, where u_1 and u_4 are corners, u_2 and u_3 are smooth joins, and the starting point is u_1. In Fig. 4b, the boundary perturbations of the shape have been smoothed, based on which the $\zeta_{j,k}$ and $\delta_{j,k}$ sequences are derived. It is noteworthy to mention that the two spikes in the $\delta_{j,k}$ sequence of Fig. 4d represent the corners, which correspond to the discontinuities in the $\zeta_{j,k}$ sequence of Fig. 4c. In addition, the circular arc of the original data is linearized in the $\zeta_{j,k}$ domain, represented by the inclined segment $\overline{u_2u_3}$, and hence the original smooth joins become the two end-points of the inclined segment (i.e. the corner points between two adjacent segments of the $\zeta_{j,k}$ waveform, which have different inclined angles).

2.2. Detection of corners

To establish a robust scheme for identifying the corner points, a statistical approach is employed. First, the mean (μ_δ) of the sequence $\delta_{j,k}$ is computed, so that

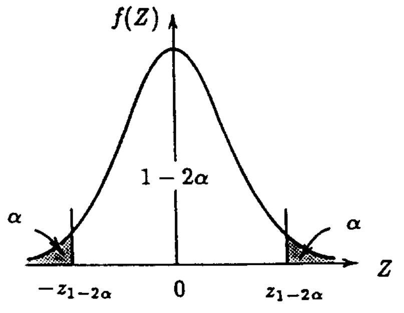

Fig. 5. The $100(1-2\alpha)\%$ confidence interval bounded by $-z_{1-2\alpha}$ and $z_{1-2\alpha}$ for the standard normal distribution.

each $\delta_{j,k}$ can be normalized by the equation:

$$\bar{\delta}_{j,k} = |\delta_{j,k} - \mu_\delta|, \quad j = 1, 2, \ldots, m \tag{11}$$

which represents the absolute value of the deviation from the mean. The cut-off (thresholding) value is the measure with respect to its standard deviation $\sigma_{\bar{\delta}}$:

$$T_{\bar{\delta}} = z_{1-2\alpha}\sigma_{\bar{\delta}} \tag{12}$$

where $z_{1-2\alpha}$ is the critical point for the standard normal distribution which has a $100(1-2\alpha)\%$ confidence interval, as illustrated in Fig. 5, i.e.:

$$P(-z_{1-2\alpha} \le Z \le z_{1-2\alpha}) = 1 - 2\alpha \tag{13}$$

where Z is the standard normal random variable (see Rohatgi [16].

Therefore, the cut-off value ($T_{\bar{\delta}}$) depends upon the series standard deviation ($\sigma_{\bar{\delta}}$) and α, which is defined as follows:

$$\alpha = (2k+1)\frac{n}{m} \tag{14}$$

where n/m is the ratio of the number of break points and the total number of data, and $(2k+1)$ is the weight reflecting the effect of the supporting length. In the mouse-like shape of Fig. 4, $n = 4$, $m = 458$ and $k = 3$. From equation (14) we obtain $\alpha = 0.061$, and hence $z_{1-2\alpha} = 1.56$ is the desired critical point obtained from the look-up table. Detection of the corners is established by the criteria:

$$\bar{\delta}_{j,k} \ge T_{\bar{\delta}} \tag{15}$$

and:

$$\bar{\delta}_{j,k} > \bar{\delta}_{l,k}, \quad \text{for } j-k \le l \le j+k, \quad \text{and } l \ne j \tag{16}$$

where equation (15) is a thresholding operation to identify the spikes in the $\bar{\delta}_{j,k}$ sequence, and equation (16) is the non-maximum suppression which restricts the identified spikes (i.e. corner points) to be the neighbourhood maximum at the predefined region $(j-k,\ j+k)$.

2.3. Detection of smooth joins

The second type of curvature changes, smooth joins, can be detected in the chain-code domain. The $\zeta_{j,k}$ waveform and four break points of the mouse-like shape are shown in Fig. 4c, where the two corner points and two discontinuities on the $\zeta_{j,k}$ waveform (modulo m) represent the smooth joins and corners, respectively, of the original shape. Since the discontinuities on the $\zeta_{j,k}$ curve have been detected in the $\delta_{j,k}$ domain by localizing the spikes, the objective of this section is to detect the corners (i.e. u_2 and u_3) in the $\zeta_{j,k}$ domain. Hence, an ordinary piecewise linear approximation of the curve (modulo m) can be employed to detect the corners, given that the discontinuities on the curve are known and used as the starting points. The waveform approximation scheme presented in this section can be found in many other similar works [4,17,18].

To stabilize the subsequent operations, the scales in the X- and Y-directions are first normalized by multiplying a scale factor to the tangent $\zeta_{j,k}$:

$$\bar{\zeta}_{j,k} = \frac{m}{2\pi}\zeta_{j,k}, \quad j = 1, 2, \ldots, m \tag{17}$$

so that both directions have approximately the same scale. The normalized $\bar{\zeta}_{j,k}$ is shown in Fig. 6a. The key issue associated with this procedure is to define the criterion for the selection of split points (i.e. break points) for the piecewise approximation. The maximum error between the data and the approximation function is intuitively a good criterion for detecting the split points.

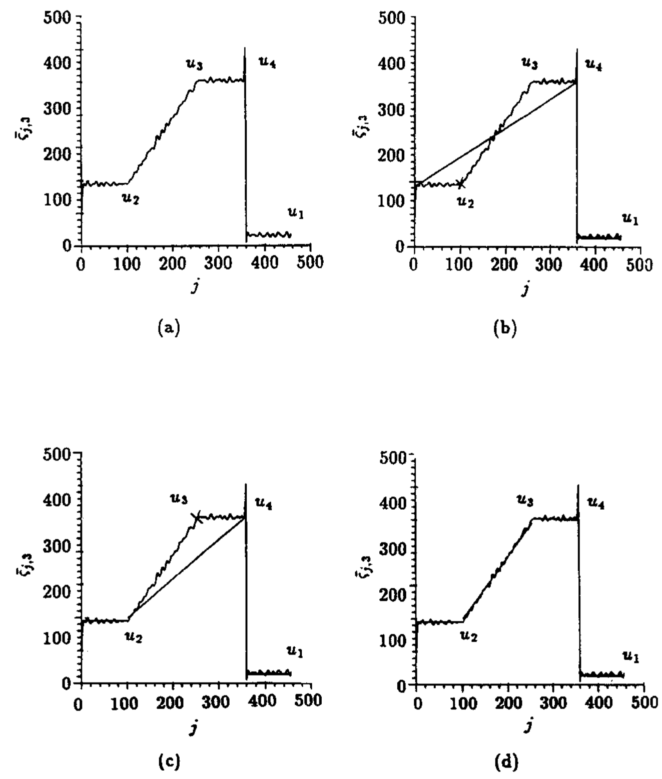

Fig. 6. Step-by-step performance of the piecewise linear approximation method, where the split points are marked with ×, and the starting break points are u_1 and u_4.

The maximum error for the data interval i is defined by:

$$E_\infty(u_1, u_{i+1}) = \max_j \epsilon_{i,j} \tag{18}$$

where $\epsilon_{i,j}$ is the Euclidean distance between the jth point of interval i and its approximation function, and (u_i, u_{i+1}) are the two end-points of the interval. Since the waveform is a piecewise linear function (without arcs), the approximation function is a straight line generated by connecting the two end-points of the interval, so that the split point can be determined by selecting the maximum chord-to-arc error. In addition, to eliminate the effect of the supporting length in the chain-code domain, the chord of data interval i is generated by connecting the k neighbours of the break points, i.e.:

$$E_\infty(u_i + k, u_{i+1} - k) = \max_j \epsilon_{i,j} \tag{19}$$

Therefore, the split operation is performed iteratively by detecting the maximum chord-to-arc error defined in equation (19), until the total number of break points is equal to n. This procedure is illustrated in Fig. 6, where the split points, marked with $\times$, are equivalent to the break points, and the chords in Fig. 6b, for example, are $\overline{(u_1 + k)(u_4 - k)}$ and $\overline{(u_4 + k)(u_1 - k)}$. After the n break points are detected, a split-and-merge method proposed by Ventura and Chen [2] can be employed to improve these points. This refinement is accomplished by iteratively performing the split and merge operations simultaneously in the chain-code domain using the maximum chord-to-arc error.

In certain cases, there are no corners available on the curve, which means that we do not have any starting point to initiate this procedure. In this case, we can select the starting n break points by equally dividing the data set S. On the other hand, if there are more than n break points detected on the $\bar{\delta}_{j,k}$ curve, the points having the smaller $\bar{\delta}_{j,k}$ values are suppressed, and those points having the largest $n\bar{\delta}_{j,k}$ values are retained. Improvement of these n points is then accomplished by using the split-and-merge method in the $\bar{\zeta}_{j,k}$ domain.

2.4. Detection of lines and arcs

The last step is to identify the approximation functions (lines and arcs) for the data intervals which are divided by the n break points. This operation is best performed in the $\bar{\zeta}_{j,k}$ domain (see Fig. 6d), based on which the arcs and line segments can be determined as follows:

- *Circular arc*: the approximation segment is not parallel to the X-axis (the slope is not zero), and
- *Line segment*: the approximation segment is parallel to the X-axis (the slope is zero).

This criterion classifies the n data sets into arcs ($i \in \mathcal{I}_c$) and lines ($i \in \mathcal{I}_l$), which provides the necessary information for the subsequent data fitting procedure.

If both n_l and n_c are known as in the case of model-based segmentation and matching, this step can be made more robust by classifying n_c data intervals with the steepest approximation segment in the $\bar{\zeta}_{j,k}$ domain into $\mathcal{I}_c$, and the remaining n_l data intervals into $\mathcal{I}_l$.

3. Adjustment of break points

The break points obtained by using the proposed shape coding schemes may not provide the best solution, because there is no evidence to support its optimality. Therefore, we would like to adjust each break point to the left or to the right, point-by-point until the error norm is minimized. This refinement is an optimization data fitting approach, where the error norm is defined by the sum of the squares of the Euclidean distances between the approximation functions and the data.

Most of the existing methods for the higher order polynomial approximation either use a heuristic approach to estimate the parameters of the function, or present a closed form solution based on an approximation distance measure. For example, Rosin and West [7] and West and Rosin [8] developed heuristic algorithms for segmentation of planar curves into arcs and lines to minimize the sum of the Euclidean least-squares errors. Their algorithms assume that the centre of each arc is equi-distant from the two end-points of the arc, which may not be true in a grid image plane, especially in the circumstance where the data is perturbed. Albano [5] used an approximate Euclidean distance to measure the goodness-of-fit for the planar quadric curves. In his paper, the error function is defined by:

$$e_j^2 = (f(x_j, y_j))^2 \tag{20}$$

where:

$$f(x, y) = Ax^2 + Bxy + Cy^2 + Dx + Ey + F = 0 \tag{21}$$

represents the approximating curve, and (x_j, y_j) is a boundary point. The error defined in equation (20) is not only significantly biased from the true integral squared distance, but also variant to the scale (see Bolle and Vemuri [19]). Safaee-Red et al. [20] used a modified form of the above error function (20) for the estimation of elliptical shape parameters from a set of edge points. In their work, the contribution of each data point to the error function is normalized by applying a weighting factor to it. Based on a geometric interpretation of the error function (20), the weighting factor is defined as a function of the position of each individual data point.

3.1. Error measure

Since we are concerned with developing a segmentation procedure exclusively for industrial applications, such as machined part inspection, using an optimization approach based on the exact Euclidean distance is necessary to achieve the desired accuracy, although it may require more computational time. The error of an interval of the segmented object is defined by:

$$E_2(u_i, u_{i+1}) = \sum_{j=u_i}^{u_{i+1}} \epsilon_{i,j}^2 \tag{22}$$

where (u_i, u_{i+1}) are the indices of the starting and ending points of the data interval S_i, and $\epsilon_{i,j}$ is the Euclidean distance between the point $(x_{i,j}, y_{i,j})$ and the approximation function for S_i. For the linear fitting, the error is:

$$\epsilon_{i,j}^2 = (x_{i,j} \cos \theta_i + y_{i,j} \sin \theta_i - d_i)^2, \quad \text{for} \quad j = 1, 2, \ldots, m_i, \quad \text{and } i \in \mathcal{I}_l \tag{23}$$

where d_i is the normal distance from the origin to the approximation line for the data interval S_i, and θ_i is the angle between the normal line to the approximation line and the X axis. For the circular fitting, the error becomes:

$$e_{i,j}^2 = \left(((x_{i,j} - a_i)^2 + (y_{i,j} - b_i)^2)^{1/2} - r_i\right)^2, \quad \text{for} \quad j = 1, 2, \ldots, m_i, \quad \text{and } i \in \mathcal{I}_c \tag{24}$$

where (a_i, b_i) and r_i are the coordinates of the centre and radius, respectively, of the approximation arc for S_i. The optimal (linear and circular) fitting is accomplished by finding a solution to the system of equations generated by setting the partial derivatives of equation (22) with respect to each parameter to zero. In the linear case, there is a closed form solution available for this problem. However, there is no closed form solution for the circular fitting using the defined error norm, due to the non-linearity of the resulting system of equations. To solve this problem, an iterative algorithm developed by Yeralan and Ventura [21] is employed, where the estimated circle is arbitrarily close to the least-squares circle.

3.2. Algorithm

After the first stage of the segmentation, an initial set of break points, $\{u_i,\ i = 1, 2, \ldots, n\}$, is determined, and each interval is classified as linear ($i \in \mathcal{I}_l$) or circular ($i \in \mathcal{I}_c$). This information is used as the input data for the break-point adjustment algorithm, denoted by *BA-1*, presented below.

BA-1

```
Step 0. Set the step-size $D = 1$.
Step 1. For $i = 1$ to $n$, do:
  1.1. Compute
      $\epsilon'_0 = (E_2(u_{i-1}, u_i - 1) + E_2(u_i, u_{i+1} - 1))$,
      $\epsilon'_1 = (E_2(u_{i-1}, u_i - 1 - D) + E_2(u_i - D, u_{i+1} - 1))$, and
      $\epsilon'_2 = (E_2(u_{i-1}, u_i - 1 + D) + E_2(u_i + D_i, u_{i+1} - 1))$.
  1.2. If $\min\{\epsilon'_1, \epsilon'_2\} < \epsilon'_0$, then:
      if $\epsilon'_2 > \epsilon'_1$, set $u_i = u_i - D$, and go to Step 1.1;
      else set $u_i = u_i + D$, and go to Step 1.1.
Step 2. If any break points have been adjusted, repeat Step 1;
  else stop.
```

Step 0 selects the step-size (D) for each shift. We choose the smallest step-size, $D = 1$. In Step 1, the break points are shifted to the left or to the right point-by-point until the error norm of every pair of adjacent intervals is minimized. Step 2 checks if any improvement has been made in the previous iteration. If the error norm has been reduced, one more iteration is performed; otherwise, the algorithm terminates. Since this is a strictly decreasing procedure, the algorithm cannot cycle. In addition, since there is a finite number of data intervals and a finite number of points in each interval, the algorithm terminates in a finite number of steps.

It is noteworthy to mention that Algorithm *BA-1* may fall in the trap of local minimum, due to a single outlier. Although there are no general methods (except for enumerative algorithms such as dynamic programming) which can guarantee a global optimal in all circumstances, the proposed algorithm can be improved by using multiple step-sizes. For example, we can use a coarse-to-fine approach to generalize the algorithm, as presented below:

BA-2

```
Step 0. For $D = D_{\max}$ down to 1, do
  BA-1.
```

Algorithm *BA-2* performs *BA-1* a preset number of iterations ($D_{\max}$) in a coarse-to-fine manner. In the first iteration, we adjust the break points by the largest step-size $D_{\max}$, which can avoid the shadow of the local minimum, and in the last iteration, the break points are adjusted point-by-point, which can precisely localize the best solution.

4. Experimental results

This section provides background information about the modified dynamic programming (MDP) technique which is used to verify the accuracy of our algorithm, describes the procedure used to generate the test data sets, and presents the results of an extensive computer experimentation. The objective of this experimental study is to check the computational time and accuracy of the proposed approach. The algorithms have been implemented on a VAX/VMS 8550 computer written in single precision using the standard C compiler and

IMSL (Version 10.0) Fortran subroutines. The execution times reported for the test problems have been recorded in CPU seconds, and do not include the preprocessing and input/output times.

4.1. Review of MDP approach

Dynamic programming (DP) was the first approach to be used in the approximation of two-dimensional continuous functional curves [22,23]. For a given number of line segments, this approach guarantees to reach an optimal piecewise linear approximation based on the principle of optimality [24]. However, the computational time grows very fast with the number of data points and the number of segments, which makes this approach impractical in many applications.

Therefore, the DP approach is modified in two aspects. First, we extend the linear approximation to a known combination of linear and circular approximations for the data; that is, the two sets of indices, $\mathcal{I}_l$ and $\mathcal{I}_c$, for the n data intervals, are given. Second, we only search all possible combinations of the points that are within a given range with respect to an initial set of break points, instead of performing complete enumeration. This initial set of break points, $\{u_i,\ i = 1, 2, \ldots, n\}$, is the solution obtained from our procedure. Hence, the possible sets of solutions are generated by varying each u_i within the range of $(u_i - h,\ u_i + h)$, where h is the half-width of a preset range. In the mathematical form, for a given set of initial break points, $\{u_i,\ i = 1, 2, \ldots, n\}$, and the sets of indices of the approximation functions for data intervals, $\mathcal{I}_c$ and $\mathcal{I}_l$, and the half-width of the search window, h, we want to search for an optimal solution, $\{v_i,\ i = 1, 2, \ldots, n\}$, so that the error norm is minimized:

$$\begin{aligned} &\text{Minimize} \quad \sum_{i=1}^{n} E_2(v_i,\ v_{i+1}) \\ &\text{Subject to} \quad u_i - h \le v_i \le u_i + h, \quad \text{for } i = 1, 2, \ldots, n \end{aligned} \tag{25}$$

4.2. Generation of test data

Both synthetic data and real image data were generated to test the proposed procedure. Two types of synthetic closed contours were created: the mouse-like shape of Fig. 4, and the curve shown in Fig. 7. The first synthetic shape is a combination of three straight lines and one circular arc, consisting of two corners and two smooth joins. The edge points were generated from the analytic (line and circle) equations at equal spacing, and rounded into the nearest integers, so that the data would resemble the error due to the spatial resolution of the image plane. The mouse-like shape was selected to illustrate the step-by-step performance

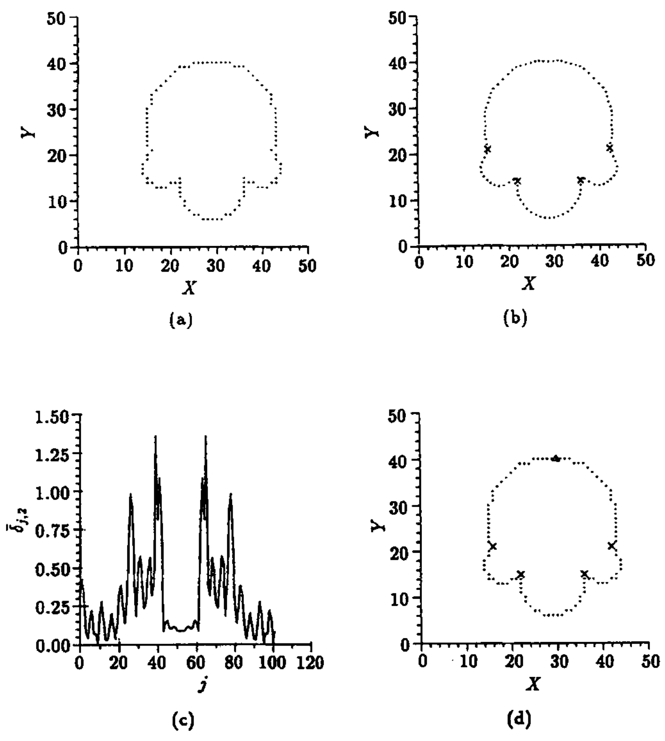

Fig. 7. A curve consisting of four semi-circles with multiple scales: (a) original data (reproduced from Teh and Chin [1]); (b) smoothed data and the break points after the 1st stage; (c) normalized differential chain-code; and (d) original data and the break points after the 2nd stage, where $k = 2$, $D_{\max} = 1$, $n = 4$, $m = 102$, and the starting point is marked with Δ.

of our procedure. The second synthetic figure is reproduced from Teh and Chin [1], consisting of four semi-circles having different radii, which represents a typical example of a curve which has features of multiple scales. The reason for choosing this figure for the experiment is mainly to compare the results produced by our procedure with those from the dominant-point detection methods presented by Teh and Chin [1].

The real images used in this experiment are two typical machined parts: a latch and a cutting tool, as shown in Fig. 8. The generation of the edge points from these two shapes includes three steps: image acquisition, preprocessing and edge tracking. After these processes, two sets of boundary data for the latch and the cutting tool were obtained and arranged in CCW sequence as shown in Figs. 9d and 10d, respectively. They were used as the input data (S) for the proposed segmentation algorithm.

The contour of the latch is composed of five line segments and one arc, consisting two smooth joins and four corners, two of which are nearby. Asada and Brady [12] defined two nearby corners as a compound curvature change, which is identified by a single knot point. In our work, the two nearby corners on the latch contour are considered as two separate (isolated) features, and hence are detected by two break points. The contour of the cutting tool is composed of five line segments and two

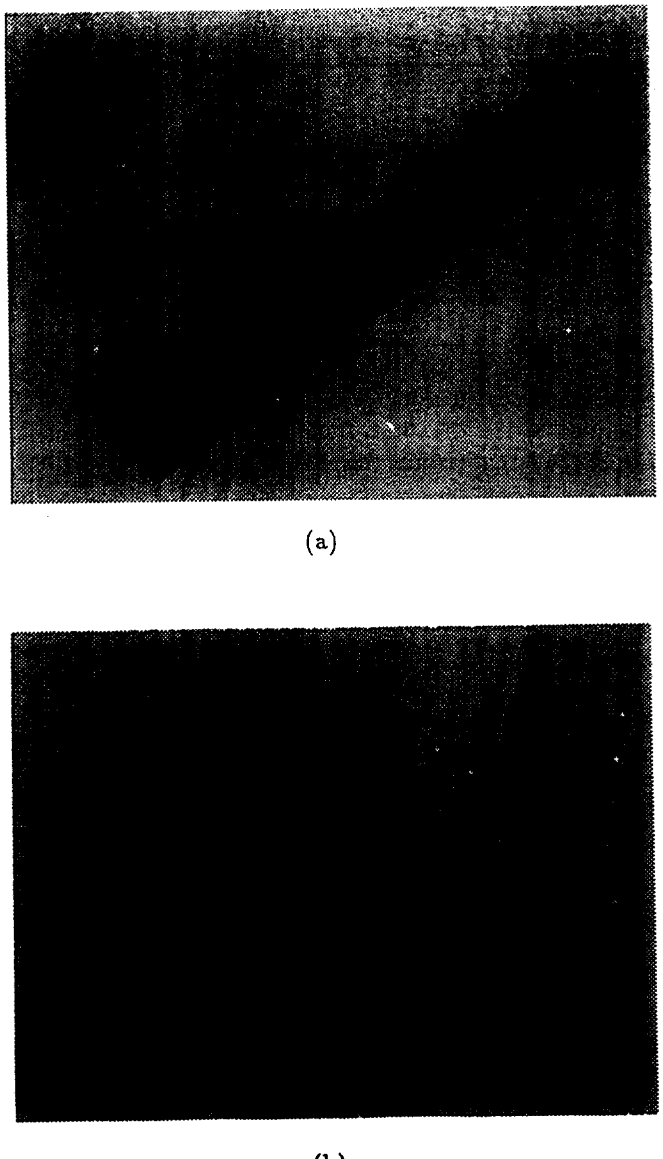

Fig. 8. Two machined parts. (a) Latch; (b) cutting tool.

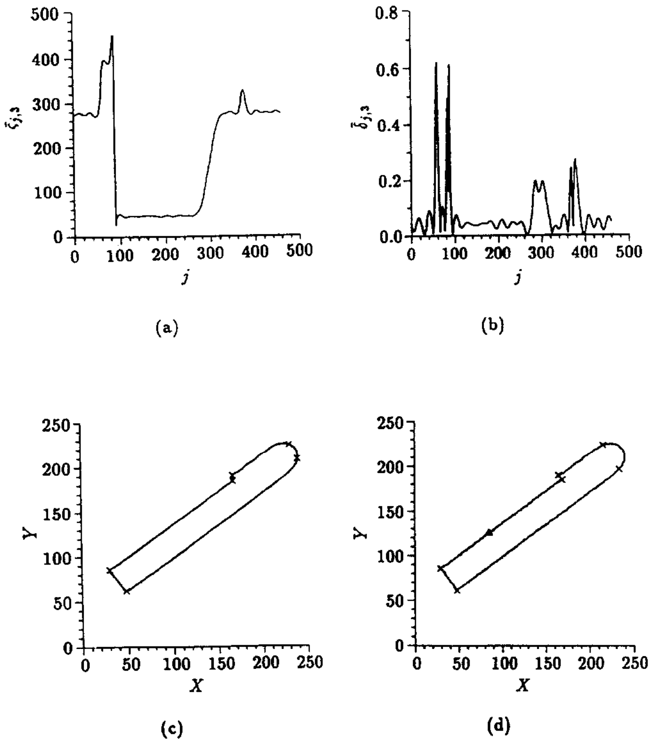

Fig. 9. The contour of the latch extracted from Fig. 8a. (a) Normalized chain-code; (b) normalized differential chain-code; (c) smoothed data and the break points after the 1st stage; and (d) original data and the break points after the 2nd stage, where $k = 3$, $D_{\max} = 1$, $n = 6$, $m = 460$, and the starting point is marked with Δ.

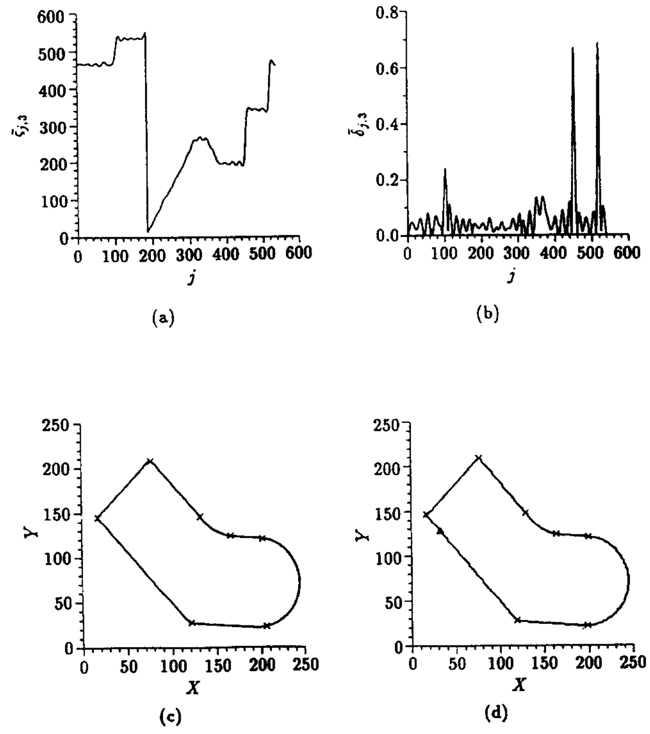

Fig. 10. The contour of the cutting tool extracted from Fig. 8b. (a) Normalized chain-code; (b) normalized differential chain-code; (c) smoothed data and the break points after the 1st stage; and (d) original data and the break points after the 2nd stage, where $k = 3$, $D_{\max} = 1$, $n = 7$, $m = 540$, and the starting point is marked with Δ.

arcs, having three corners and four smooth joins. The four smooth joins, representing the end points of the two arcs, do not have significant curvature changes on the curve, and are difficult to precisely identify.

4.3. Results and discussions

The created data sets were solved using the proposed two-stage algorithm and the MDP method. The results obtained by applying the proposed algorithm to the four test problems are graphically illustrated in Figs. 4, 7, 9 and 10, where we used a supporting length $k = 3$ (except for Fig. 7, where $k = 2$ due to its small number of data points in each interval), and $D_{\max} = 1$ for all the test problems. In these four figures, the obtained break points are marked with $\times$, and the starting points on the curves are

Table 1
Approximation errors

Shapes			1st stage		2nd stage		Relative errors	
Fig.	n	m	E_2^1	E_∞^1	$E_2^2 = E_2^*$	$E_\infty^2 = E_\infty^*$	E_2^r	E_∞^r
4	4	457	38.95	0.644	38.77	0.648	0.46	−0.62
7	4	102	6.60	0.646	5.76	0.511	14.57	26.42
9	6	460	931.72	11.452	49.85	1.053	1769.01	978.06
10	7	540	76.21	1.589	68.94	1.139	10.99	39.51
Average			–	–	–	–	448.65	260.84

indicated by Δ. It should be noted that the four corner points on the curve in Fig. 7 were all detected in the differential chain-code domain, and hence the chain-code sequence is not shown in the figure.

Two performance measures were used to investigate the accuracy and speed of our algorithm: (1) the resulting error norm; and (2) CPU time in seconds. Although the algorithm was implemented using the integral squares error, the maximum error is also reported for a more robust comparison. These two errors are computed by the following equations:

$$E_2 = \sum_{i=1}^{n} E_2(u_i, u_{i+1}) \qquad E_\infty = \max_{1 \le i \le n} E_\infty(u_i, u_{i+1}) \tag{26}$$

where $E_\infty(u_i, u_{i+1})$ and $E_2(u_i, u_{i+1})$ are defined by equations (18) and (22), respectively.

The approximation errors for the test problems are summarized in Table 1, where the first three columns describe the parameters of the input data, such as the physical figure, number of data intervals (n) and number of points (m). Columns 4 and 5 provide the resulting errors after the first stage (denoted by E_2^1 and E_∞^1), and Columns 6 and 7 are the final errors after the second stage (denoted by E_2^2 and E_∞^2). The relative errors (E_2^r and E_∞^r) given in the last two columns are obtained by the following equations:

$$E_2^r = \frac{E_2^1 - E_2^*}{E_2^*} \times 100 \qquad E_\infty^r = \frac{E_\infty^1 - E_\infty^*}{E_\infty^*} \times 100 \tag{27}$$

which represent the degree of inaccuracy of the solution obtained from the first stage of the algorithm with respect to that from the MDP approach described earlier. The relative errors show that using the coding scheme alone for segmenting digitized planar curves is not stable. For example, the relative integral errors (E_2^r) for the object in Figs. 4 and 9 are 0.46% and 1769.01%, respectively, one of which is very close to the final result, and the other severely deviates from the result. On average, the relative integral error and relative maximum error are 448.65% and 260.84%, respectively.

The overall accuracy of the algorithm is verified by comparing the final solutions generated from the second stage of our algorithm with those from the MDP approach. The minimal errors obtained from the MDP approach with a half-width $h = 3$ (except for Fig. 10, where $h = 2$ due to the large amount of data and break points on the curve) are shown in Columns 6 and 7 of Table 1 (denoted by E_2^* and E_∞^*). The computational results show that the proposed algorithm and MDP approach reach the same solutions for all the test problems, which empirically supports the optimality of the break point adjustment algorithm (although it can not be assumed that the BA algorithm is generally valid for finding the optimal solution).

The computational time required to solve the test problems using the proposed method is given in Table 2, which shows the CPU time in seconds required by each stage and the percentage of the time for each stage with respect to the total time. On average, the first and second stages take 9.85% and 90.15% of the total time, respectively, indicating that the shape coding schemes are

Table 2
Computational time in CPU seconds (VAX 8550)

Shapes			1st stage		2nd stage		Total
Fig.	n	m	CPU	%	CPU	%	CPU
4	4	457	0.12	13.48	0.77	86.52	0.89
7	4	102	0.04	5.19	0.73	94.81	0.77
9	6	460	0.12	9.45	1.15	90.55	1.27
10	7	540	0.15	10.34	1.30	89.66	1.45
Average			0.11	9.85	0.99	90.15	1.10

Table 3
Comparison of CPU times with the MDP approach (VAX 8550)

Fig.	CC-BA		MDP	
	CPU	%	CPU	h
4	0.89	0.74	119.96	3
7	0.77	0.53	143.97	3
9	1.27	0.05	2528.64	3
10	1.45	0.03	5273.71	2
Avg.	1.10	0.34	–	–

superior to the data fitting method in computational speed. The average CPU time required to solve the problems is 1.10 s.

Table 3 provides a comparison of the CPU times needed by our algorithm with those needed by the MDP approach. This comparison shows that the proposed two-stage method (denoted by CC-BA; chain-code and break-point adjustment) requires less than 1% of the CPU time needed by the MDP approach for all the test problems. The computational advantage of our method is more significant when the number of segmentation intervals increases, such as in the case of the cutting tool (Fig. 10), where our method takes less than 0.03% of the CPU time required by the MDP approach. As shown in the table, the expensive computational burden makes the MDP approach not applicable for most of the vision applications.

The robustness of the proposed approach was illustrated by testing input shapes having higher degree of noise for the mouse-like shape. Noise was added to the synthetic data shown in Fig. 4 by perturbing the perfect edge points along the normal directions of the boundary contour. For example, the noise added to the point $(x_{i,j}, y_{i,j})$ is:

$$\begin{aligned} x_{i,j} &\leftarrow x_{i,j} + p_i \times l \times U(-0.5, 0.5) \times \cos\theta_{i,j} \\ y_{i,j} &\leftarrow y_{i,j} + p_i \times l \times U(-0.5, 0.5) \times \sin\theta_{i,j} \end{aligned} \tag{28}$$

where p_i is the length of the ith edge, l is the noise level, $U(-0.5, 0.5)$ is a uniformly distributed random variable bounded by -0.5 and 0.5, and $\theta_{i,j}$ is the normal angle of the ith edge at point $(x_{i,j}, y_{i,j})$. After adding the noise, all the sampled points were rounded into the nearest integers.

The approximation errors for various noise levels are summarized in Table 4, where E_2, E_2^r, E_∞, E_∞^r, E_2^* and E_∞^* have the similar meanings as those in Table 1. The generalized BA algorithm (*BA-2*) using $D_{\max} > 1$ is needed for input shapes with higher degree of noise. $D_{\max} = 2$ was chosen due to the satisfactory results. $D_{\max} = 2$ was not implimented for noise level equal to 0.5% and 1.0% since the second stage with $D_{\max} = 1$ reaches the same solutions as those from the MDP approach for these two noise levels. The computational results show that the proposed algorithm reaches the near optimal solutions ($E_2^r < 1\%$) for noise level less than or equal to 2.5%.

The comparison of the CPU times needed by the proposed method with those needed by the MDP approach for the noisy test problems is presented in Table 5. This comparison shows that even though the proposed two-stage method requires more CPU time when $D_{\max}$ is increased, CC-BA ($D_{\max} = 2$) still requires less than 1.2% of the CPU time needed by the MDP approach for noise level less than or equal to 2.5%.

Furthermore, we use the curve of Fig. 7 to demonstrate the inappropriate segmentation of quadric curves into a variable number of line segments. This is largely due to the inaccurate approximation of the curves by

Table 4
Approximation errors for the mouse-like shape with various noise levels

Noise Level (%)	1st stage		2nd stage ($D_{\max} = 1$)		2nd stage ($D_{\max} = 2$)		MDP E_2^* (E_∞^*)
	E_2 (E_∞)	E_2^r (E_∞^r)	E_2 (E_∞)	E_2^r (E_∞^r)	E_2 (E_∞)	E_2^r (E_∞^r)	
0.5	49.40 (0.904)	1.83 (−2.59)	48.51 (0.928)	0.00 (0.00)			48.51 (0.928)
1.0	78.44 (1.064)	1.83 (0.38)	77.03 (1.060)	0.00 (0.00)			77.03 (1.060)
1.5	117.64 (1.292)	1.31 (1.73)	117.35 (1.292)	1.06 (1.73)	117.21 (1.274)	0.94 (0.31)	116.12 (1.270)
2.0	675.90 (6.305)	264.94 (268.93)	198.06 (2.174)	6.94 (27.21)	186.05 (1.735)	0.45 (1.52)	185.21 (1.709)
2.5	816.34 (6.833)	218.62 (272.78)	290.11 (3.219)	13.23 (75.61)	258.34 (1.829)	0.83 (−0.22)	256.21 (1.833)

Table 5
Comparison of CPU times with the MDP approach for the mouse-like shape with various noise levels (VAX 8550)

Noise level (%)	1st stage CPU	CC-BA ($D_{max} = 1$)		CC-BA ($D_{max} = 2$)		MDP CPU
		CPU*	%	CPU*	%	
0.5	0.13	0.68	0.54			125.83
1.0	0.13	0.67	0.54			123.51
1.5	0.14	0.67	0.40	1.41	0.83	169.54
2.0	0.14	1.98	1.04	2.22	1.17	189.81
2.5	0.14	1.35	0.68	2.28	1.15	197.90

* CPU time for both stage 1 and stage 2.

Table 6
Results of the curve in Fig. 7 (reproduced from Teh and Chin [1])

Algorithm	n	E_2	E_∞	CPU
CC-BA	4	5.76	0.51	0.77†
Rosenfeld-Johnston	30	8.85	0.74	8.09‡
Rosenfeld-Weszka	34	15.40	1.00	10.13‡
Freeman-Davis	19	23.31	1.41	9.46‡
Sankar-Sharma	10	769.53	8.00	35.04‡
Anderson-Bezdek	29	6.43	1.18	12.04‡
Teh-Chin	22	20.61	1.00	9.40‡

† VAX 8550 (in C); ‡ VAX 750 (in Pascal).

using straight-lines alone. The solution generated from our method is compared with those from the six dominant-point detection algorithms presented by Teh and Chin [1]. Table 6 shows the results obtained from these algorithms. The table includes the final number of dominant points (break points), integral squares error, maximum error, and CPU time in seconds. The results in Table 6 are directly reproduced from Teh and Chin [1] (Table 4), where the six algorithms were written in Pascal and implemented on a VAX 750 computer. (Note that the computing speed of VAX 8550 is approximately 8–10 times faster than VAX 750). Table 6 indicates that our method uses the least number of data intervals to approximate the curve, and obtains the smallest E_2 and E_∞ errors among the algorithms. This example shows the need of segmenting digitized curves into higher order polynomial functions.

5. Concluding remarks

In this paper, a procedure for segmenting a planar curve into straight line segments and circular arcs has been presented, in which the number of data intervals (or break points) is assumed to be known. This procedure is divided into two stages. First, a starting set of break points is obtained by using the chain-code and differential chain-code schemes. The approximation functions (lines and arcs) for the data intervals that are separated by the break points are determined in the chain-code domain. Next, the break points are adjusted to the left or to the right using an optimization data fitting technique.

This procedure has been implemented to solve a set of test problems, where the mouse-like shape was chosen to illustrate the step-by-step performance of this procedure, and the curve consisting of four semi-circles was selected to compare the solutions generated from our method with those from the six dominant-point detection methods presented by Teh and Chin [1]. Two real image contours were also used to test the performance of our algorithm. The first real image is a latch which contains two nearby corners, and the second image is a cutting tool with smooth joins that are difficult to detect. In addition, the robustness of the proposed method was illustrated by testing inputs having higher degree of noise for the mouse-like shape. The solutions generated from our procedure were compared to those from the modified dynamic programming technique. The results have shown that the proposed procedure obtains near optimal solutions (relative errors less than 1%) for all the test problems, and requires less than 1.2% of the CPU time needed by the MDP approach.

References

[1] C.H. Teh and R.T. Chin, On the detection of dominant points on digital curves, *IEEE Trans. PAMI, 11* (1989) 859–872.

[2] J.A. Ventura and J.M. Chen, Segmentation of two-dimensional curve contours, *Patt. Recogn. 25* (1992) 1129–1140.

[3] F.T. Farago, Handbook of Dimensional Measurement, 2nd Ed, Industrial Press Inc., New York (1982).

[4] T. Pavlidis and S.L. Horowitz, 'Segmentation of plane curves', *IEEE Trans. Comput., 23* (1974) 860–870.

[5] A. Albano, Representation of digitized contours in terms of conic arcs and straight-line segments, *Comput. Graph., Image Process., 3* (1974) 23–33.

[6] O. Baruch and M.H. Loew, Segmentation of two-dimensional boundaries using the chain code, *Patt. Recogn., 21* (1988) 581–589.

[7] P.L. Rosin and G.A.W. West, Segmentation of edges into lines and arcs, *Image & Vision Comput., 7* (1989) 109–114.

[8] G.A.W. West and P.L. Rosin, Techniques for segmenting image curves into meaningful descriptions, *Patt. Recogn., 24* (1991) 643–652.

[9] R.T. Chin and A.R. Dyer, Model-based recognition in robot vision, *ACM Comput. Surv., 18* (1986) 67–108.

[10] R.T. Chin and C.A. Harlow, Automated visual inspection: a survey, *IEEE Trans. PAMI, 4* (1982) 557–573.

[11] R.T. Chin, Survey of automated visual inspection: 1981 to 1987, *Comput. Vision Graphics Image Process., 41* (1988) 346–381.

[12] H. Asada and M. Brady, The curvature primal sketch, *IEEE Trans. PAMI, 8* (1986) 2–14.

[13] A. Sirjani and G.R. Cross, An algorithm for polygonal approximation of a digital object, *Patt. Recogn. Lett., 7* (1988) 299–303.

[14] W. Kim and R.H. Park, Contour coding based on the decomposition of line segments, *Patt. Recogn. Lett., 11* (1990) 191–195.

[15] R. Sriraman, J. Koplowitz and S. Mohan, Tree search chain coding for subpixel reconstruction of planar curves, *IEEE Trans. PAMI, 11* (1989) 95–104.

[16] V.K. Rohatgi, Statistical Inference, Wiley, New York (1984).

[17] T. Pavlidis, Waveform segmentation through functional approximation, *IEEE Trans. Comput., 22* (1973) 689–697.

[18] U. Ramer, An iterative procedure for the polygonal approximation of plane curves, *Comput. Vision Graph., Image Process. 1* (1972) 244–256.

[19] R.M. Bolle and B.C. Vemuri, On three-dimensional surface reconstruction methods, *IEEE Trans. PAMI, 13* (1991) 1–13.

[20] R. Safaee-Red, K.C. Smith and B. Benhabib, Accurate estimation of elliptical shape parameters from a grey-level image, *Proc. Int. Conf. Patt. Recogn., 2* (1990) 20–26.

[21] S. Yeralan and J.A. Ventura, Computerized roundness inspection, *Int. J. Prod. Res., 26* (1988) 1921–1935.

[22] R. Bellman, On the approximation of curves by line segments using dynamic programming, *Comm. ACM, 4* (1961) 284.

[23] B. Gluss, Further remarks on line segment curve-fitting using dynamic programming, *Comm. ACM, 5* (1962) 441–443.

[24] S.E. Dreyfus and A.M. Law, The Art and Theory of Dynamic Programming, Academic Press, New York (1977).