ELSEVIER

Computer-Aided Design 38 (2006) 238–248

COMPUTER-AIDED DESIGN

www.elsevier.com/locate/cad

# A curvature estimation for pen input segmentation in sketch-based modeling

Dae Hyun Kim [a,*], Myoung-Jun Kim [b]

[a] *Institute for Graphic Interfaces, Ewha-SK Bldg. 11-1, DaeHyun, Seodaemun, Seoul Korea*
[b] *Division of Digital Media, Ewha Womans University, Ewha-SK Bldg. 11-1, DaeHyun-dong, Seodaemun, Seoul Korea*

Received 27 January 2005; received in revised form 12 October 2005; accepted 16 October 2005

**Abstract**

A proper segmentation of pen marking enhances shape recognition and enables a natural interface for sketch-based modeling from simple line drawing tools to 3D solid modeling applications; user input is otherwise restricted to draw only one segment per one stroke. In general, the pen marking segmentation is achieved by detecting the points of high curvature-called, segmenting points-and splitting the pen marking at those points. This paper presents a curvature estimation method, which considers only local shape information. The proposed method can therefore estimate curvature on-the-fly while user is drawing on a pen-input display, such as tablet PCs.


*Keywords:* Curvature; Local shape information; Pen-input displays; Segmentation

## 1. Introduction

Given a pen marking $p=\{p_i|i=0,\ldots,n\}$, drawn on a pen-input display, such as LCD tablets integrated into most Tablet PCs, segmenting points can be defined as the points that have high curvature (i.e. $|c|>\tau$, where $c$ is an estimated curvature and $\tau$ is a given threshold). Detected segmenting points lead to a preliminary segmentation, when the pen marking is split at each of them. Therefore, a proper segmentation owes its success, in most cases, to a proper curvature estimation. The inflection points, zeros of the second derivative, are also regarded as segmenting points. However, if needed, it is more advantageous to process them during curve fitting for each segment after the preliminary segmentation; otherwise small fluctuations as shown in Fig. 12 will produce many unnecessary inflection points [10].

Before developing a curvature estimation algorithm for pen-input displays, one needs to know input style; for example, user's drawing skill and the mechanical properties of the input devices. In this paper, the following input data assumption has been considered in the design of a new method to estimate curvature: *Input data is taken from a pen-based input device; therefore, it contains no noise occurring because the input device cannot follow user's drawing.* Therefore, all sampled points are regarded as genuine data, and their coordinates are taken as given. For example, wiggly lines, as those in Fig. 1, are caused by user intention, but not by the noise added performing the pen marking.

*Previous works*: We review two classes of approaches for detecting segmenting points which are directly related to our work; one specialized aspect of our work is that the method should be applicable for pen input devices—for a broader survey of segmenting point detection we refer the reader to [3].

The first class of approaches is to use scale space decomposition; it decomposes the pen marking into multi-level representations by progressively smoothing it with different smoothing functions and picks up one level among them, which can represent user's intention. There are two approaches available within this class: either curvature is evaluated in a selected scale after transforming the input data to the Gaussian scale space [7,8] or estimated curvature is transformed into scale space and one scale is selected [2,10]. One advantage of using scale space decomposition is that it is suitable for noisy data as shown in Fig. 1. However, a long computing time is needed to construct Gaussian scale space. Considering the input assumption that we have made above,

* Corresponding author. Fax: +82 2 3277 3893.
*E-mail addresses:* daek@acm.org (D.H. Kim), mjkim@ewha.ac.kr (M.-J. Kim).

0010-4485//$ - see front matter © 2005 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cad.2005.10.006

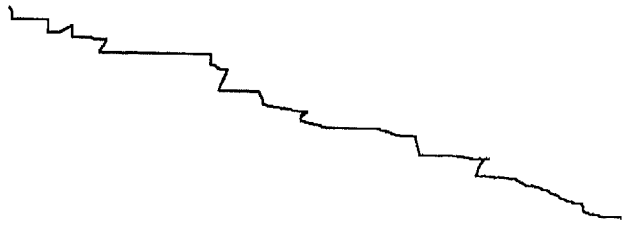


Fig. 1. A wiggly pen marking, which may come from noisy input or from user intention.

this high computation has been considered unnecessary for our computational model.

The second class of approaches is to estimate curvature directly from input data [3,9,11,12]; it examines the $k$-nearest neighbors (region of support) to estimate curvature since the definition of curvature for the mathematical curves does not hold for a digital curve. Teh and Chin [11] claimed that a precise determination of region of support is most important to a proper curvature estimation.

Rosenfeld's method [9] takes one parameter, the support size, which needs to be assigned initially [9,12]. Fu et al. [3] proved that this method produces redundant points or misses some points. Teh and Chin [11] proposed a method to dynamically change the support size. However, Fu et al. showed that the criteria of Teh and Chin to select the support size at each point sometimes do not work correctly [3]. Fu et al. suggested a new method where a global support size is searched for ahead such that in every support only one local maximum is found for a height function. Because this method has to scan the point sequence several times to find a proper support size, it is less efficient than ours.

*Contributions*: For freehand pen marking, this paper proposes new criteria to determine region of support: local shape information such as local convexity and local monotonicity. They can be effectively tested only by checking signed direction. Since they are defined locally, curvature estimation can be done on the fly while scanning the input sequence. Furthermore, to let the programmers test the characteristics of pen input devices, we provide tolerance models as well. This paper also helps understand the characteristics of freehand pen marking, to develop sketch-based applications.

*Paper organization*: In Section 2, we define the terms and notations that will be used throughout the paper. Section 3 describes our curvature estimation method. Based on the curvature and an additional interaction feature, pen-speed, which can be provided only in interactive applications, segmentation rule is presented. Furthermore, through an example will be explained how to use obtained segmentation information in the process of recognizing pen markings. In Section 4, we evaluate our algorithm and similar methods by analyzing their behaviors. Furthermore, for more quantitative evaluation we conduct user tests to see to which extent the algorithm is regarded as efficient for different drawing styles as its theoretical ground. In Section 5, we discuss the limitations of our approach.

## 2. Definitions

Let us define direction at a point, which will be used throughout the paper to explain our approach.

**Definition 1.** (Direction at a point)

Let $A$, $B$, and $C$ be three points in the plane. Let $L=B\text{-}A$ and $Q=C\text{-}B$ be the vectors from $A$ to $B$ and $B$ to $C$, respectively. The angle $d=\angle(L,Q)$ from the direction of $L$ to that of $Q$ is called the 'direction change at $B$', or simply 'direction at $B$'.

See also Fig. 2 for the direction change at a point of a pen marking; subsequently, direction change at point $p_i$ is denoted by $d_i$.

If input points are acquired in pixel space, direction data can have at most eight values. Two approaches deal with this problem of the narrow angle space; one approach is to resample the point set and the other is to use a sliding window [10]. Resampling means picking up the point that first comes beyond a given distance from the last sample; in our test, the distance value, 5.0, has been used. The other approach, called the sliding window method, computes $d_i$ with two lines that have least square distance to $k+1$ preceding points $\{p_{i-k},\ldots,p_i\}$ and $k+1$ succeeding points $\{p_i,\ldots,p_{i+k}\}$, respectively. To obtain a wider range of direction value efficiently, we resample the input data. For the rest of this paper, by input data we mean already re-sampled points unless stated otherwise. The re-sampling is performed based on the distance criterion such that the points of pen marking are almost equally distanced; this will simplify computations as well as explanations on some notions used later.

**Definition 2.** (Support for curvature estimation at $p_i$)

A set of adjacent points that are used for the curvature estimation at $p_i$ is called *support*. When the number of the preceding and succeeding points are both $k$, it is called $k$-symmetric support.

The curvature at point $p_i$ is denoted by $c_i$. When it depends on the support size '$k$', we denote it by $c_i^k$.

Before estimating curvature, let us see what happens while re-sampling the input points, from Fig. 3. The resampling, which are performed before curvature estimation, smoothes out sharp angles at some points, so that some features are lost. Therefore, by necessity, the support size should be increased for each point; in other words, curvature estimation should be further generalized as $c_i^k$, including the $k$-symmetric support.

## 3. Curvature estimation

We first derive the curvature for the pen marking $p$ (i.e. digital curve) from differential geometry. Although we do not

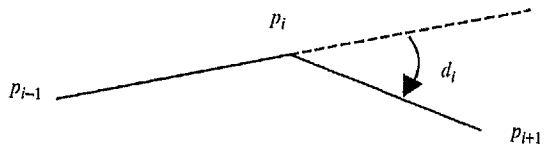



Fig. 2. Defining the direction at point $p_i$.

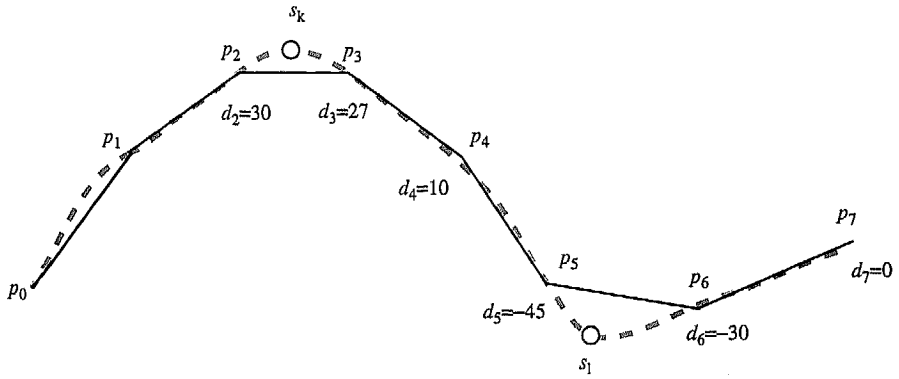



Fig. 3. Sharp angle around $s_1$ and $s_2$ has been smoothed out; smoothing by re-sampling cannot be avoided. Original pen marking $s=\{s_i\}$, compared with re-sampled one, is shaded in gray.

believe the curvature estimation for the digital curves does not need to be always based on the second derivative, most other approaches ultimately come from the definition of the second derivative. We show that curvature estimation based on angle formed by three consecutive points is basically the same as the infinitesimal behavior of the curvature estimation (i.e. second derivative) on continuous curves. Let $\gamma(s)$:$R\rightarrow R^2$ be a planar curve, twice differentiable and arc length parameterized. Then, the curvature for the continuous planar curve is defined as [1]:

$$\|\gamma''(s)\| = \left\| \lim_{\Delta s\rightarrow 0} \frac{\gamma'(s+\Delta s)-\gamma'(s)}{\Delta s} \right\| \tag{1}$$

On the right side of Eq. (1), $\gamma'(s)$ represents the unit tangent vector at the curve point $\gamma(s)$. The difference of the two tangent directions, $\gamma'(s+\Delta s)-\gamma'(s)$, can be more intuitively represented by the angle. Therefore, when $\gamma(s)$ is approximated by a digital curve (i.e. a pen marking, $p$) the curvature at a point $p_i$ can be simplified into the following

$$c_i = \frac{d_i}{\Delta s_i}, \tag{2}$$

where $d_i$ is defined as in Definition 1, and $\Delta s_i = \|p_{i+1}-p_i\|$. Since the *re-sampling* on the pen marking turns $\Delta s_i$ into nearly a constant, $\Delta s_i$ can be omitted from the Eq. (2):

$$c_i = d_i. \tag{3}$$

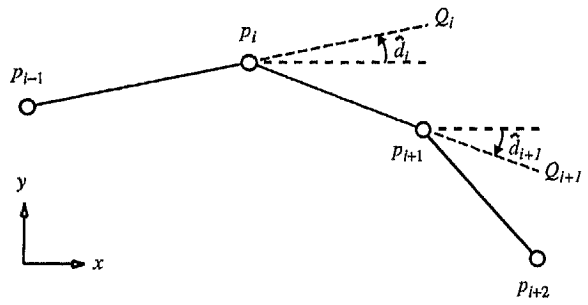



Fig. 4. Direction in [10].

Therefore, we set $c_i^1 = d_i$ as first level curvature estimation, with the support size one. Meanwhile, the omitted $\Delta s_i$ is reflected into the support size $k$.

Our definition of direction change is different from that of Sezgin's [10]. Let $\hat{d}_i$ and $\hat{c}_i$ be Sezgin's direction data and curvature data, respectively, then $\hat{d}_i$ is defined as the angle between the $x$-axis and the vector $Q_i=p_i-p_{i-1}$ (see Fig. 4), and $\hat{c}_i = (\hat{d}_{i+1}-\hat{d}_i)/\Delta s_i$. Since $d_i = (\hat{d}_{i+1}-\hat{d}_i)$, $\hat{c}_i = d_i/\Delta s_i$, and $\Delta s_i$ is presumed nearly constant, our direction change at $p_i$ is as informative as Sezgin's curvature estimation.

Given the direction data as a preliminary approximation of the curvature, we define a new curvature measure considering two local shape information: Local convexity and local monotonicity (Fig. 5).

**Definition 3.** (*Local convexity at* $p_j$ with respect to $p_i$) A polygon is locally convex at $p_j$ with respect to $p_j (i\neq j)$, if $d_i$ and $d_j$ have the same sign.

A naive approach for curvature estimation considering only local convexity information can be formulated as follows: extending the support to each side of $p_i$, if the current point $p_j$ ($j=i\pm l$, $l=1,\ldots,k$) is locally convex with respect to $p_i$, the point $p_j$ is added to the adaptive support at $p_i$. Direction values within the support are added to yield the curvature $c_i^k$. The curvature estimator can be described procedurally as the algorithm in Fig. 5.

In Algorithm 1, $k$ no longer indicates actual support size, but rather the maximum support size. For example, on the lefthand side of Fig. 6 with $k=1$, local area searched for spans from $p_{i-1}$ to $p_{i+1}$; however, actual support is set to cover only

**Curvature estimation with local convexity: $C_i^k$**

*If* $|d_i| < u$ *Return 0;*
// For each side of $p_i$, add up the neighbor direction values
For ( $j = 1$; $sgn(d_i) = sgn(d_{i-j})$, $j \leq k$; $j=j+1$) $c = c+d_{i-j}$;
For ( $j = 1$; $sgn(d_i) = sgn(d_{i+j})$, $j \leq k$; $j=j+1$) $c = c+d_{i+j}$;
*Return* $c+d_i$;

Fig. 5. Algorithm 1: Curvature estimation only with local convexity.

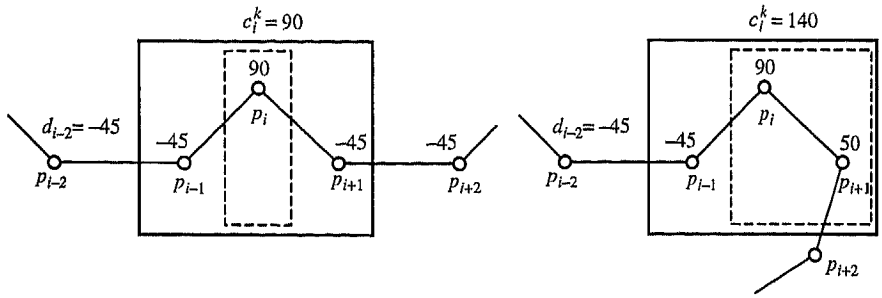



Fig. 6. The effect of the adaptively changing support with $k=1$. The actual support is denoted by the dotted rectangle. Without convexity criterion, however, actual support is set to the solid-line rectangle.

the point $p_i$ according to the local convexity criterion. Meanwhile, for the right hand figure the actual support for the point $p_i$ includes $p_i$ and $p_{i+1}$ because the latter point is locally convex with respect to $p_i$.

As mentioned in Section 2, sharp segmenting points are often blurred out and lost while re-sampling. Such features can be recovered by enlarging the support size. Another effect can be understood by comparing it with other methods; in Fig. 6, our estimate $c_i^k$ does not change as $k$ goes from 1 to 2, but curvature estimated by the circumscribing circle decreases. In contrast, bending value increases. Because our method considered local convexity in deciding actual support, curvature did not change.

Although Algorithm 1 can be used for some applications without further modification, at times it makes actual features indistinguishable. For example, Fig. 7 illustrates a counter example: Computing $c_{i-1}^k$, $c_i^k$, and $c_{i+1}^k$, all neighbor direction values within the support are added and wind up 180 degree because they are all locally convex with respect to each other. To mend this problem, the local monotonicity property is considered in the curvature estimation.

From the example of Fig. 7, we derive the local monotonicity property; although $d_i$ has a local minimum at $p_i$, applying Definition 5 makes all three points $p_{i-1}$, $p_i$, and $p_{i+1}$ have the same curvature value, 180. As a result, segmentation is confused in choosing appropriate segmenting points; such confusion will require complicated heuristics, as in [3,12,10], to remove unnecessary features. Therefore, it is more desirable to make the sequence, $\{c_i^{k>1} | i=0,\ldots,n\}$, as distinctive as the sequence of the direction value $\{d_i = |i = 0,\ldots,n\}$. Relations (i.e. <and>) between $d_i$ and $d_{i+1}$ will be preserved between $c_i^k$ and $c_{i+1}^k$ as well (Fig. 8).

**Lemma 4.** Local Monotonicity

Given a sequence of non-negative numbers, $\{\ldots, s_{i-n}, \ldots, s_i, \ldots, s_{i+n}, \ldots\}$, define $S_i^l$ to be the monotonically decreasing subsequence of maximum length $k+1$, extending from $s_i$ to the left. Define $S_i^r$ to be the monotonically decreasing subsequence of maximum length $k+1$, extending from $s_i$ to the right. Further,

$$a_i = \sum S_i^l + \sum S_i^r - s_i. \tag{4}$$

*If* $s_i < s_{i+1}$, $a_i < a_{i+1}$. *If* $s_i > s_{i+1}$, $a_i > a_{i+1}$.

**Proof.** : Assume that $s_i < s_{i+1}$. We show $a_i > a_{i+1}$.

Let us denote the last elements of $S_i^l$ and $S_{i+1}^r$ by $\alpha$ and $\beta$, respectively. By their definition then

$$\sum S_{i+1}^l = \sum S_i^l - \alpha + s_{i+1} \qquad \sum S_{i+1}^r = \sum S_i^r - \beta - s_i$$

Similarly to Eq. (4), $a_{i+1}$ can be represented by

$$a_{i+1} = \sum S_{i+1}^l + \sum S_{i+1}^r - s_{i+1} = a_i + \beta - \alpha.$$

Therefore, $a_{i+1} - a_i = \beta - \alpha$. We now have $a_{i+1} > a_i$ since $\beta > \alpha$.

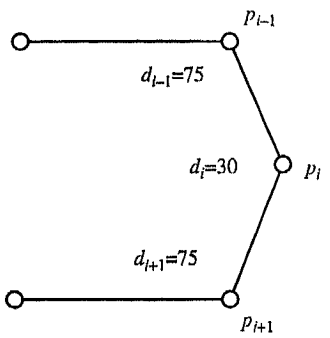



Fig. 7. An example to explain local monotonicity. Curvatures only with local convexity at $p_{i-1}$, $p_i$, and $p_{i+1}$ are all 180. Curvature with both local convexity and monotonicity are 105, 30, and 105, respectively.

**Curvature estimation with local shape information : $C_i^k$**

```
If |d_i | < u  Return 0;
min = |d_i |;
For( j = 1; sgn(d_i) = sgn(d_i-j), j £ k; j=j++){   // for the left side of pi
  if(|d_i-j| £ min) {                  //p_i-j is locally monotonous w.r.t. p_i?
      c = c + d_i-j;                   //add the current direction to the curvature
      min = |d_i-j|;                                //set the min value
  } else break;
}
// Analogously, process the right side of p_i
Return c+d_i;
```

Fig. 8. Algorithm 2: estimate curvature $c_i^k$ at point $p_i$.

The case of $s_i > s_{i+1}$ can be analogously proved. □

The sequences stated in Lemma. 4, $S_i^r$ and $S_i^l$, are said to have local monotonicity property with respect to $s_i$. We apply Lemma 4 to the adaptively changing support, leading to the new curvature estimation of Algorithm 2; it considers both local convexity and local monotonicity. The resulting curvature sequence, $\{c_i^k\}$, will not only preserve the ordering information of direction data, but also provide a kind of enhanced view to the local shape of the pen marking. To compute the curvature $c_i^k$, our method adds the neighbor direction values, which are locally convex and monotonous with respect to $p_i$. We express this curvature estimation in a procedure as the algorithm depicted in Fig. 8.

Interpretation on the support size $k$ and re-sampling distance $\ell$ follows. Since the pen marking was resampled such that distance between adjacent sample points is not larger than $\ell$, size of the view into the local shape becomes $2\ell k$. On this basis, the practical number of $k$ can be chosen; throughout our experiments, $k=3$ and $\ell=5$. More importantly, we can easily incorporate tolerances that allow small fluctuations in convexity and monotonicity property, by slightly modifying the *min* value and *sgn* function. For example, sgn($x$) function can be modified to return zero for $-\varepsilon \le x \le \varepsilon$. Likewise, *min* can be modified to have a tolerance zone.

As seen from the two curvature estimators defined above and the calculation of direction value at each point (see Definition 1), scan the pen marking only once to compute curvature at all the input points. Except for computing direction value our approach introduces no noticeable complex computation. Estimated curvature values tell us whether the curve portion is convex or concave, as well as its magnitude by angle. Another property of our approach is that it is invariant under the rotation and reordering of pen markings, e.g. back to front or front to back order; since the direction data and the actual support at each point are invariant under the change of the scanning order, so is the resulting curvature. For example, on the right side of Fig. 6 scanning from the left to the right and the other way around do not affect the result. Its computational complexity is $O(n)$ because it scans pen marking once.

For segmentation, local maximum for positive $c_i^k$ and local minimum for negative curvatures are recognized as segmenting points. Then the absolute value of the curvature under the given threshold value are recognized as segmenting points. However, the way specifying the threshold affects the overall performance of the pen-based interface; too low threshold may result in unnecessary segmenting points and thus wrong recognition. To resolve this problem, we add a characteristic feature, pen speed; at lower speed use smaller threshold. Note, however, that pen speed has not been used as the dominant feature, rather it is referenced to additionally detect almost invisible features. The principle that the user interface should not restrict the user's behavior is behind this priority strategy. The shape features extracted from the segmentation results are summarized in Fig. 9. In this figure, classification of segments into spline, line, and circular arc has been done using standard least squares fitting algorithms as in [5]; for example, after line fitting if error is larger than expected it is no longer regarded as a line, instead perform circular arc fitting.

If the segmentation results are reliable, for a sketch-based application, shape recognition improves as well. Recently, Hammond and Davis [4] proposed a general language to describe shapes; we followed their approach. To briefly explain how the recognition works with the segmentation results, we show an example procedure that recognizes a pen marking shown in the middle of Fig. 10 as a *rotational-arrow* by which the user can invoke rotation commands:

(1) Four segments were found—$c_1$, $l_2$, $l_3$, and $l_4$.
(2) $c_1$ has been classified as circular arc.
(3) $l_2$, $l_3$, and $l_4$ have been classified as straight lines.
(4) $c_1$ and $l_3$ intersect.
(5) $\angle c_1 s_2$, $\angle l_2 s l_3$, and $\angle l_3 l_4$ have the same sign.
(6) Each of their absolute values is larger than 110.
(7) $l_4$'s end point approximately coincides with $c_1$'s end point.
(8) The first acute angle between first two adjacent segments occurs after half the length of the pen marking.

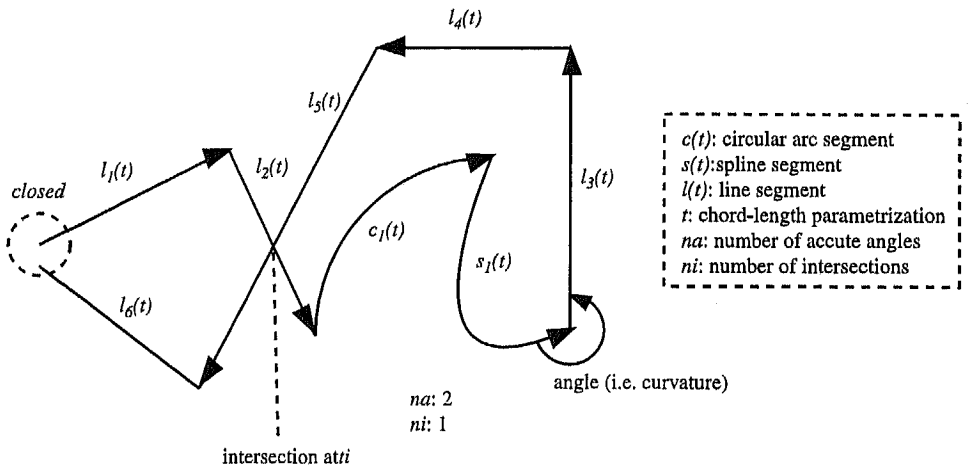



Fig. 9. Shape features used for shape recognition.

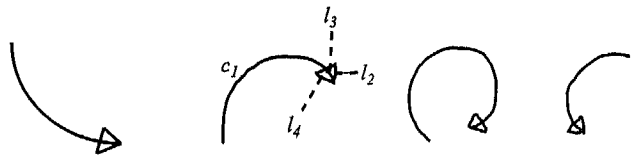



Fig. 10. The three pen markings on the right, which are similar to the shape of the left image, will be recognized as a *rotating-arrow*.

## 4. Evaluation of the Algorithm

In this section we quantitatively analyze our method as well as other similar methods for comparison. For more quantitative discussion, we conduct user tests and show the results as well.

### 4.1. Analysis and comparison of curvature estimators

We compare our method with two curvature estimation methods which have been regarded standard—*circumscribing circle* and *bending function*—and verifies our method through the comparison. The first method for estimating the curvature at the point $p_i$ is to compute the reciprocal of the radius of a circle that passes through three consecutive points $p_{i-1}$, $p_i$, and $p_{i+1}$ [5]:

$$K_i = \frac{4\Delta i}{||L_i||||L_{i+1}||||Q_i||}, \tag{5}$$

where $L_i = p_i - p_{i-1}$, $Q_i = p_{i+1} - p_{i-1}$, and $\Delta i = \text{area}(p_{i-1}, p_i, p_{i+1})$. The quantity $K_i$ has a negative sign when $(p_{i-1}, p_i, p_{i+1})$ are in clockwise order, and a positive sign if they are in counter-clockwise order. We call the above measure *circumscribing circle curvature* (for illustration, see Fig. 11).

The second well-known method is to use the curve *bending function*, which is represented by the angle:

$$A_i = \angle p_{i-1} p_i p_i', \tag{6}$$

where $p_i'$ is a projection of $p_i$ onto the line segment $p_{i-1}p_{i+1}$ [3,12].

*Smoothing effect*: Regardless of the estimation methods, both resampling and sliding window, which are performed before curvature estimation, smooth out sharp angles at some points, as depicted in Fig. 3, so that some features are lost. Therefore, we need to see the phenomenon in a wider view; support must be enlarged at some points, as we did dynamically within $k$ given.

To compute circumscribing circle and bending function (Eqs. (5) and (6)) with the enlarged support size $k > 1$, three ordered points, $(p_{i-k}, p_i, p_{i+k})$, are used instead of $(p_{i-1}, p_i, p_{i+1})$.

*Shape description*: For example, in Fig. 3, $c_3^3$ of $(p_0, p_3, p_6)$ from both methods reflects the curvature at $p_3$ no better than $c_3^2$ of $(p_1, p_3, p_5)$, because a *locally concave* part at $p_5$ with respect to $p_3$ influences $c_3^3$. Therefore, local convexity can be regarded as a local shape descriptor and should be considered in calculating curvature.

Consider another example involving a larger $k$, as shown in Fig. 7. The support at $p_i$ does not need to extend beyond $p_{i-1}$

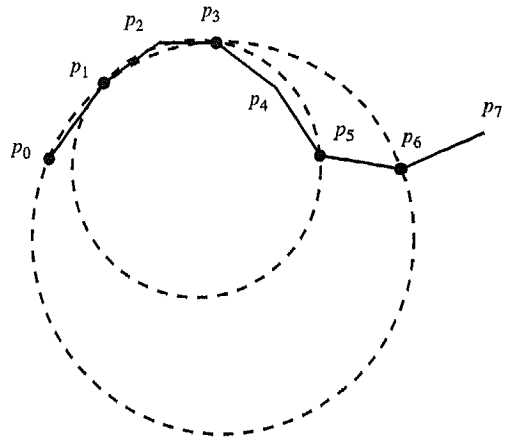



Fig. 11. $c_3^2$ of the circumscribing-circle method better reflects curvature at $p_3$ than $c_3^3$.

since it will increment curvature at $p_i$ as $k \to 2$ with the two previous methods. In this example, information that 'there is large curvature nearby' can be better encoded at points $p_{i-1}$ and $p_{i+1}$ than encoded at $p_i$. This has been expressed by local monotonicity property (Fig. 11).

*Efficiency*: Fu et al. [3] determines $k$ globally such that every support for $c_i^k$, $i=0,\ldots,n$, has only one local maximum of a height function $h^k(j)$, which computes the minimum distance between the line segment $p_{i-k}p_{i+k}$ and $p_j$, $i-k<j<i+k$. More specifically, if a sequence $\{h^k(i-k), h^k(i-k+1), \ldots, h^k(i+k)\}$ within the $k$-symmetric support of any point $p_i$, $i=1,\ldots n$, has more than one local maximum, the algorithm changes $k$ to look for a right one until the requirement is satisfied. However, if the pen marking contains a small fluctuation as in Fig. 12, $k$ will be no bigger than 1. Unfortunately, such minor fluctuations frequently occur even after being smoothed. Moreover, since the line segment $p_{i-k}p_{i+k}$ changes as $k$ changes, searching for the right $k$ requires at least $(n+1)\sum 2k-1$ evaluations of $h^k(j)$; above all, it will require $k$-pass scans of the pen-marking and at each pass the angle values must be computed. In our approach, small fluctuations are ignored (e.g. see the first step of Algorithm 2) and actual support size changes dynamically using the local shape information.

Comparably, our curvature estimator defined above and the calculation of direction value at each point, scan the pen marking only once. Except for computing direction value our approach introduces no noticeable complex computation.

*Correct angle measurement*: Although the bending function is represented by angle, it often incorrectly reflects the dynamic

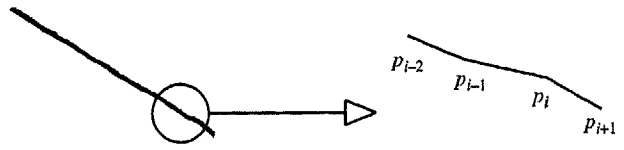



Fig. 12. Small fluctuations: a pen marking (left) and a zoom-in (right).

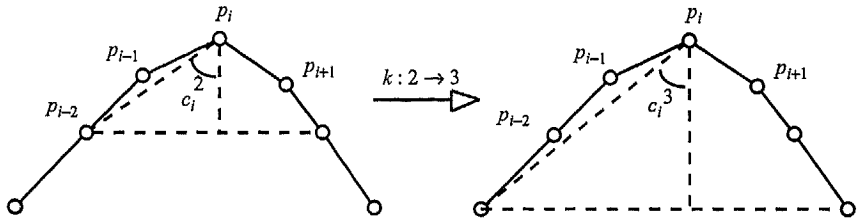



Fig. 13. A problem from bending function: as $k$:2→3, although direction at $p_{i-2}$ is zero, $c_i^3$ ends up with a higher value than $c_i^2$.

change of support size. For example, in Fig. 13, although the direction at $p_{i-2}$, $d_{i-2}$, is zero, $c_i^k$ of the bending function changes as $k$:2→3. In other words, the bending function insensitively reflects point-wise change of geometry unlike ours.

Now, the above analysis justifies our curvature estimation with adaptively changing support reflecting local shape information.

### 4.2. Evaluation tests

In this section we perform three evaluation tests to evaluate our algorithm:

- For a given standard drawings on a test sheet, test how our algorithm performs in case of different drawing styles of different users.
- Compare four approaches: curvature estimation with only direction data, only with local convexity, with both local convexity and monotonicity, and with [3]'s bending function.
- Analyze our algorithm along with a few special test cases.

First, we have conducted an evaluation test with seven graduate students studying at the department of computer science. We gave them 21 figures to draw; these figures have been selected from other papers [2,10], and Microsoft Powerpoint basic shapes—see Fig. 14. The subjects drew the figures twice with a Logitech optic mouse and twice with a tablet PC. No time to practice their drawing has been given to the subjects, but the purpose of the test has been explained to them; so, the user can fill in the form with the number of segments found. Interestingly, when a drawing is far different from the given shape, the users ignored the result and redrew it. This implies the assumption we have made in Section 1 for input data is plausible.

In this test, the average success rate of our approach was about 95 percent; drawings are considered successful when the feature points shown in the test sheet (Fig. 14) are correctly produced from user's drawing. Lengthy figures in Fig. 14 have shown higher errors.

Second, we perform an evaluation test by comparing four algorithms for the same input: one from the curvature estimation just adding neighbor direction values within $k=3$ support (See Fig. 15), one from the curvature estimation considering local convexity (see Fig. 16 and Algorithm 1), one from the curvature estimation considering both local convexity and monotonicity (see Fig. 17 and Algorithm 2). Indistinguishable features disappear as more shape

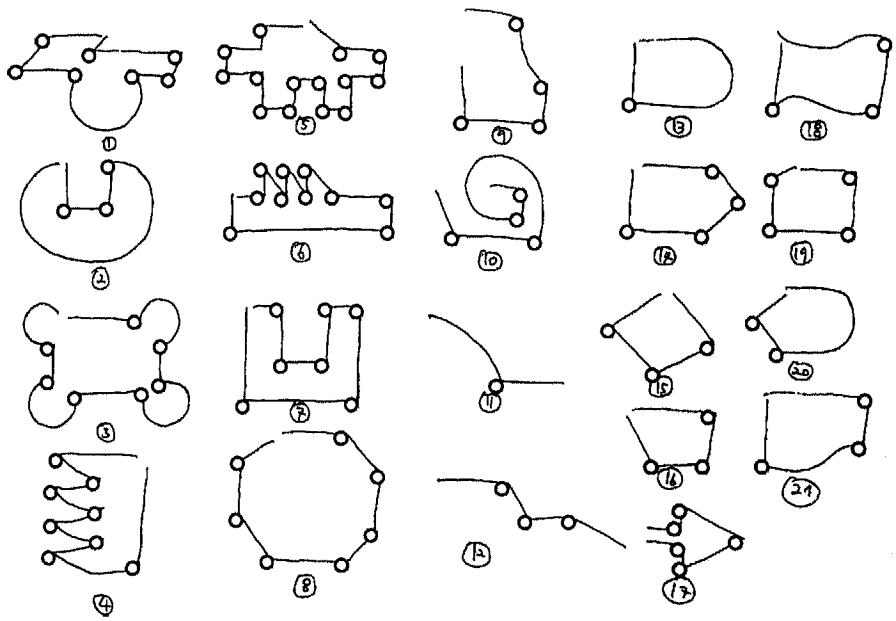


Fig. 14. The 22 figures. that the subjects for the user test are instructed to draw. Circles mark the intended feature points.

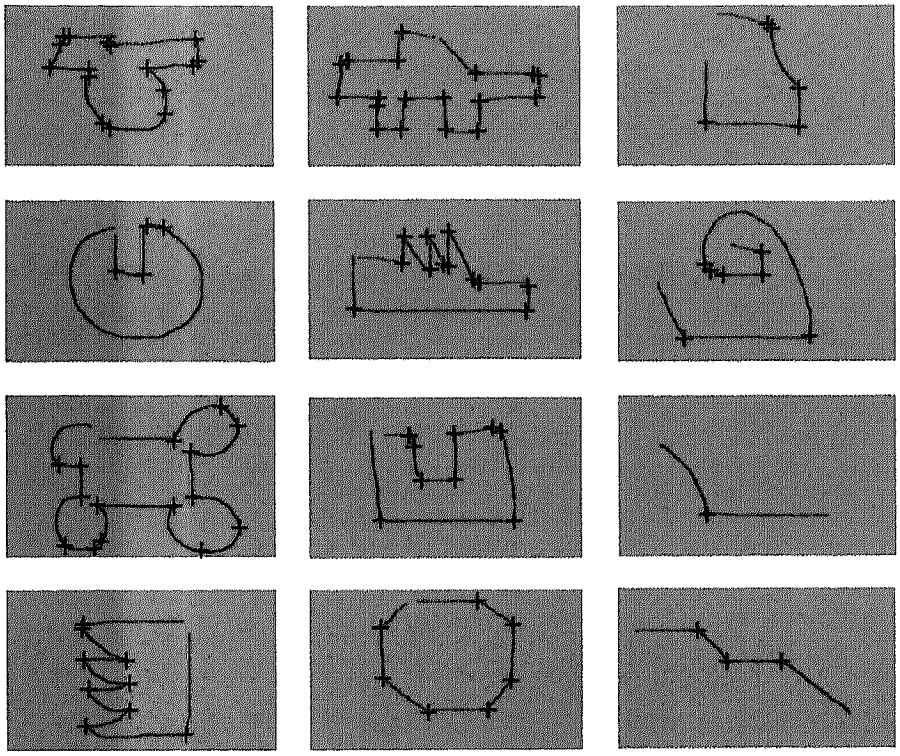


Fig. 15. Feature point detection by adding neighbor directions.

descriptors are used. With the method in [3] which uses bending function with the globally determined support size $k$, the result is shown in Fig. 18; for each figures corresponding $k$ value is shown. As mentioned in Section 4.1, because of small fluctuations from user hand drawing, the support size is sometimes set too small (e.g. second figures in the first column and third in the second column), thus losing some segmenting points.

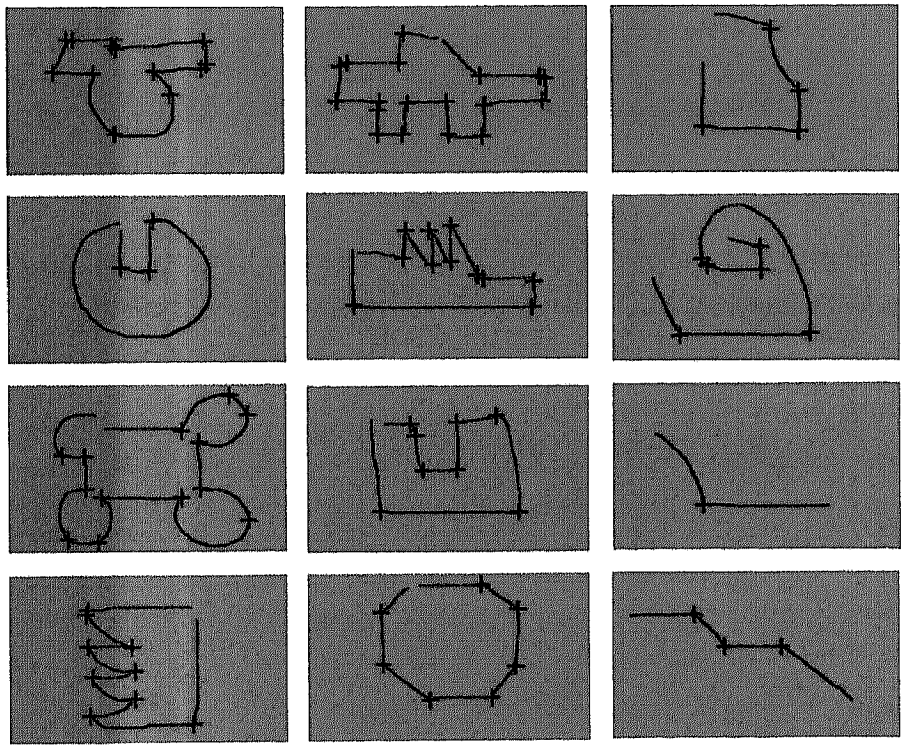


Fig. 16. Feature point detection only with convexity information.

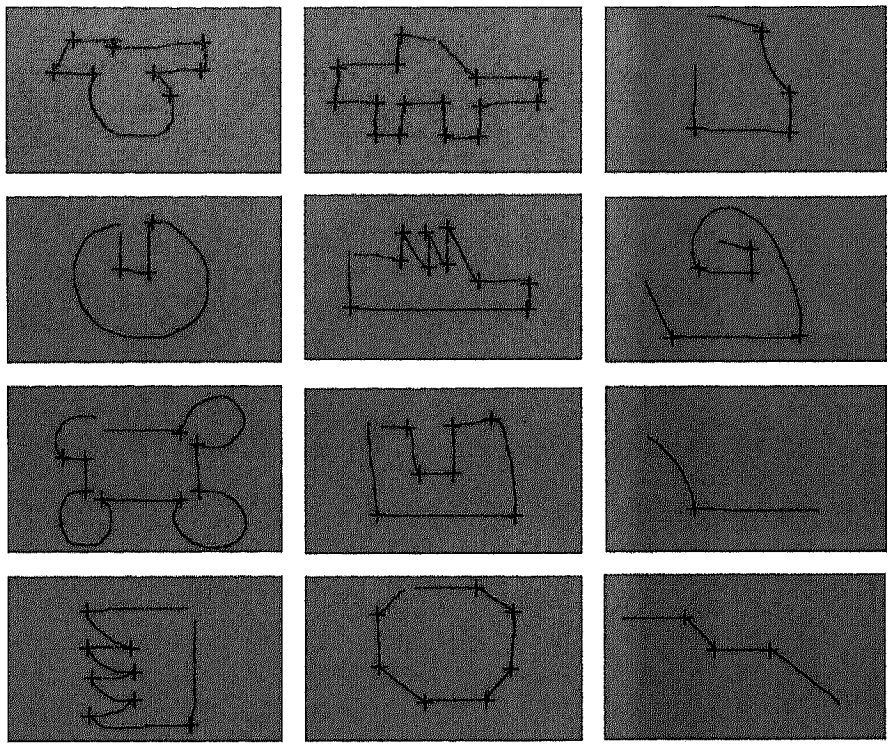


Fig. 17. Feature point detection with both local shape descriptors: local convexity and local monotonicity.

Finally, we show a series of examples to demonstrate the segmentation methods in Fig. 19. Points marked by a small circle are detected as segmentation points by curvature data only. Squares mark segmentation points that are detected after referencing speed data. In Fig. 19(d), e.g. one segmenting point was lost due to the curvature estimation; at this point, estimated curvature was 21, which is under the threshold value. Thus, it slipped out of the curvature test. Comparing Fig. 19(e) with (f),

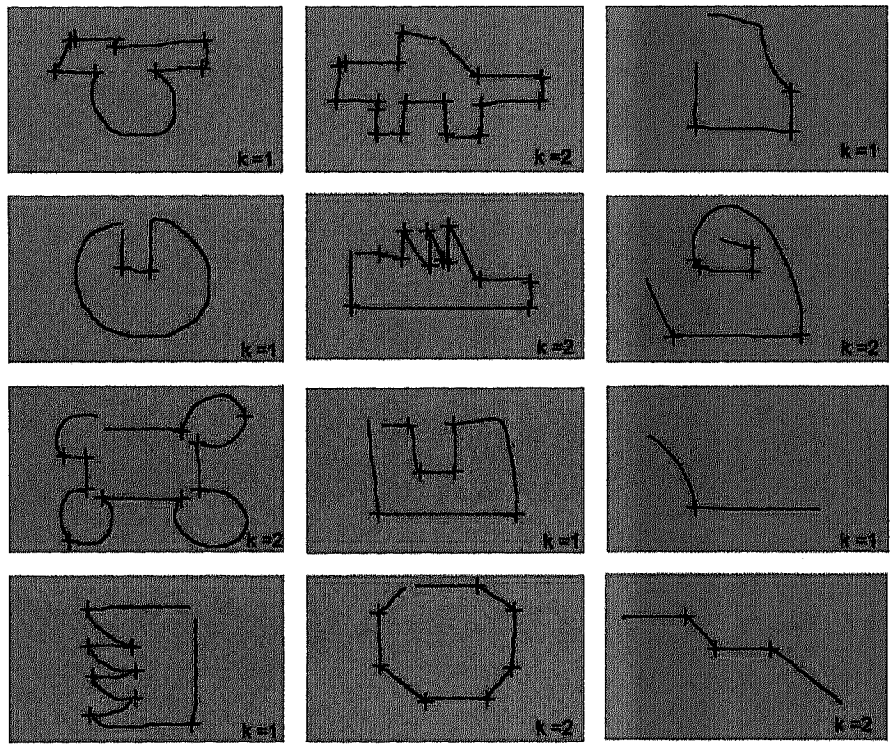



Fig. 18. Segmenting point detection with Fu's method [3].

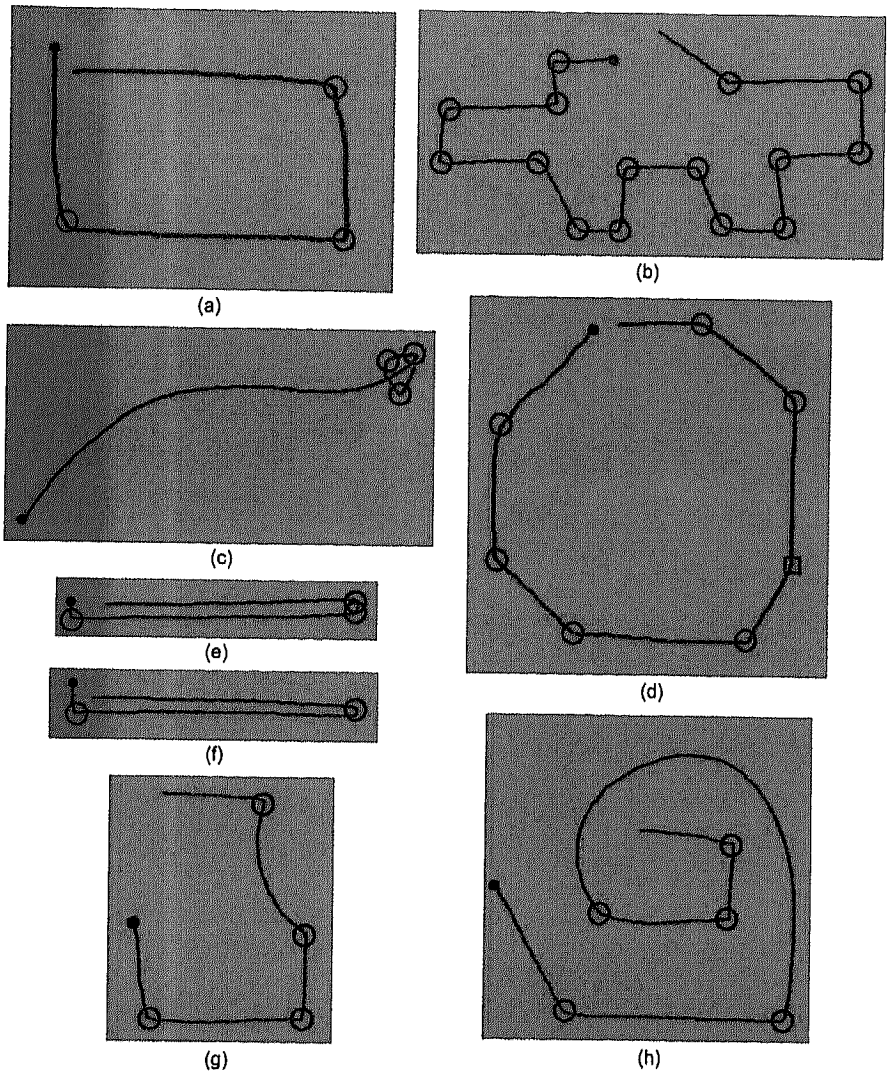



Fig. 19. (a) Four line segments were detected. The period marks the start of the pen marking and the circles the segmenting points. (b) Sixteen straight-line segments were detected. (c) Arrow shaped marking: one *spline* and three straight lines were found. (d) Using only curvature information, six straight lines and one spline were found. The fifth segment (containing the square mark) was classified as a spline, since the curvature estimation lost the feature. After relating pen speed data, we could get eight straight-line segments (octagon). (e) A thin rectangle was recognized. (f) Two adjacent points that form two corners of a rectangle are merged because segmentation selects only local extreme. (g) Five segments were found including one freeform curve. (h) Four straight lines and one spline were detected.

we can derive one property of our approach: when the two adjacent points have high curvature only one of them can be selected out as a segmenting point. In other words, *the minimum arch length of two adjacent segmenting points is 3M*, where $M$ is the minimum distance between two adjacent points.

## 5. Discussion

Digital curves drawn on the pen-input display are stored in pixel space, therefore their final shape and interpretations in pixel space are limited unlike just drawing on a paper. For example, Fig. 20 can be drawn in any scale on a paper but not in the pixel space when it is scaled down. For the drawing on a paper, between any two consecutive sample points in the figures exist infinitely many points. In that case, minimum distance $M$ between two adjacent sample points does not need to be as large as in the case of the pixel space. Then direction

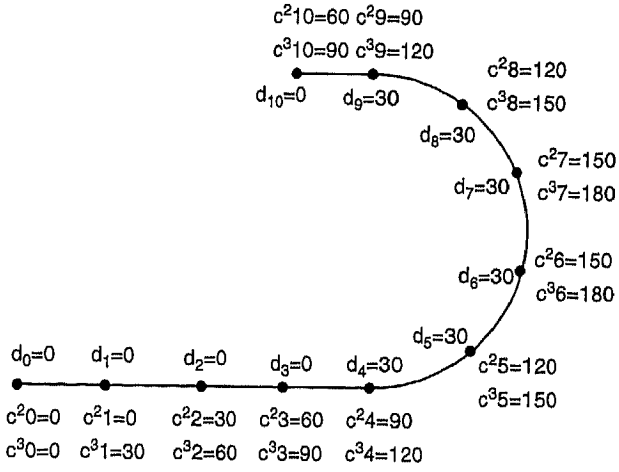



Fig. 20. Pixel space limits drawing style.

values will be far smaller than shown in Fig. 20, resulting in no segmenting points detected. Remember that for such resampling to happen the figure must be drawn in about $30\times20$ pixels. From this resolution on downwards, it is unclear from the user drawing whether, for example, $p_8p_9$ is part of a circular arc or a straight line and the final curve is a polyline or one smooth curve. This is the restriction of our approach and other segmentation algorithms, too, working in pixel space.

## 6. Conclusion

We have presented a new curvature estimation method that operates simply in angle space, considering only local shape information: *local convexity* and *local monotonicity*. We also showed, how this estimate can be applied to segment pen markings. Since this method yields curvature in angle, estimated curvatures can directly be used for shape recognition.

For future research, we suggest testing this method with locally smoothed data by feature-preserving smoothing [6]. Another future line of research is to test our idea on analytical shapes as in [3]. Since the shape information is derived from mathematical definitions, this should work fine as well.

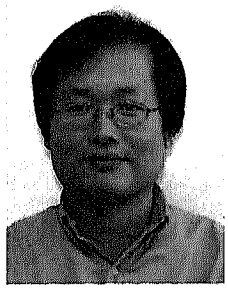
## Supplementary data

Supplementary data associated with this article can be found, in the online version, at doi:10.1016/j.cad.2005.10.006

## References

[1] Carmo M. Differential geometry of curves and surfaces. Englewood, Cliffs, NJ: Prentice- Hall; 1976.
[2] Davis R. Position statement and overview: sketch recognition at mit. In: American association for artificial intelligence spring symposium on sketch recognition; 2002.
[3] Fu AMN, Yan H, Huang K. A curve bend function based method to characterize contour shapes. Pattern Recognit 1997;30(10):1661–71.
[4] Hammond T, Davis R. Ladder: a language to describe drawing, display, and editing in sketch recognition. In: Proceedings of IJCAI (International joint conference on artificial intelligence); August 2003.
[5] Hoschek J, Lasser D.Fundamentals of computer aided geometric design. A.K. Peters, Ltd; 1989.
[6] Jones TR, Durand F, Desbrun M. Non-iterative, feature-preserving mesh smoothing. In: ACM SIGGRAPH 2003. ACM; 2002.
[7] Moktarian F, Mackworth K. A theory of multiscale-based shape representation for planar curves. IEEE Trans Pattern Anal Mach Intell 1992;14(8):789–805.
[8] Ratterangsi A, Chin T. Scale-based detection of corners of planar curves. IEEE Tran Pattern Anal Mach Intell 1992;14(4):430–49.
[9] Rosenfeld A, Weszka S. An improved method of angle detection on digital curves. IEEE Trans Comput 1975;C-24:940–1.
[10] Sezgin TM. Feature point detection and curve approximation for early processing of free-hand sketches. PhD Thesis. EECS of UC Berkeley; 2001.
[11] Teh CH, Chin T. On the detection of dominant points on digital curves. IEEE Trans Pattern Anal Mach Intell 1989;11(8):859–72.
[12] Wang MJ, Wu W, Huang L, Wang D. Corner detection using bending value. Pattern Recognit Lett 1995;16:575–83.

Dae Hyun Kim received his BS degree in 1994 in computer science from Seoul National University of Technology, Korea, his MS in 1996 in computer science from Korea University, his PhD in 2004 from Bremen Universitaet, Germany. His PhD supervisor is Prof. Frieder Nake, who was one of the founders of Computer Art in 1960s. Since his PhD, he had been a post-doctoral researcher under the supervision of Prof. Myoung-Jun Kim, Ewha Womans University, Korea. From June 2004 he has been working for the Institute for Graphic Interfaces as a senior researcher and the leader of the Tiled Display Team. He also worked for the Korea Electronics and Telecommunications Research Institute (ETRI) from 1996 to 1999 as a researcher for the computer graphics team. His research focuses on geometric modeling, sketch-based modeling, and, recently, appearance modeling.



Myoung-Jun Kim received his BS degree in 1989 in computer science from the Korea Institute of Technology, Korea, his MS in 1992 and his PhD in 1996, both in computer science, from the Korea Advanced Institute of Science and Technology (KAIST). Since his PhD, he has been a post-doctoral researcher at the University of Washington in 1996. He has also worked for the Korea Electronics and Telecommunications Research Institute (ETRI) from 1997 to 2000 as a senior researcher for the Computer Graphics Team. From 2001, he has been an assistant professor at Ewha Womans University. His technical research and experience include mathematical modeling, computer graphics, and, recently, digital appearance modeling.