Online stroke segmentation by quick penalty-based dynamic programming

Liu Wenyin\(^1\), Tong Lu\(^2\), Yu Yajie\(^3\), Shuang Liang\(^4\), Rui Zhang\(^3\)

\(^1\)College of Computer Science and Technology, Shanghai University of Electric Power, Shanghai, People’s Republic of China
\(^2\)State Key Laboratory of Novel Software Technology, Nanjing University, Nanjing 210046, People’s Republic of China
\(^3\)Department of Computer Science, City University of Hong Kong, HKSAR, People’s Republic of China
\(^4\)School of Software Engineering, Tongji University, Shanghai 210804, People’s Republic of China

E-mail: lutong@nju.edu.cn

Abstract: A stroke segmentation method named quick penalty-based dynamic programming is proposed for splitting a sketchy stroke into several regular primitive shapes, such as line segments and elliptical arcs. The authors extend the dynamic programming framework with a customisable penalty function, which measures the correctness of splitting a stroke at a particular point. With the help of the penalty function, the proposed dynamic programming framework can finish the stroke segmentation process without any prior knowledge of the number and/or the type of segments contained in the sketchy stroke. Its response time is sufficiently short for online applications, even for long strokes. Experiments show that the proposed method is robust for strokes with arbitrary shape and size.

1 Introduction

Sketching on paper or a pen-input display such as table PCs and smart phones is a natural way to quickly externalise ideas, and it is often used in the early prototyping stages to express and record design ideas and can be adopted by many domains including mechanical engineering [1], software design [2, 3], information architecture [4, 5] and pen-based recognition and simulation systems [6]. As a recent trend against traditional window, icon, menu and pointer paradigm, sketching plays a growing important role in two-dimensional (2D)/three-dimensional (3D) modelling and human–computer interface [7].

Stroke segmentation is a key problem in sketch-based applications [8]. Given a pen-marking point chain \(P_N = \{p_i\}_{i=0}^{N}\) drawn on an input display, stroke segmentation is defined as first extracting the segmentation points having relatively high curvatures and then splitting \(p\) into a group of fitted primitive shape results. Stroke segmentation can provide the following advantages against the format of sketch point chains: (i) small fluctuations brought by human hands can be effectively corrected, (ii) sketching shapes can be restricted to a group of meaningful strokes, better reflecting human drawing processes and (iii) geometrical or topological constraints inside a segmented and fitted sketch are helpful for further shape analysis or semantic interpretation tasks.

In this paper, we propose a novel stroke segmentation algorithm quick penalty-based dynamic programming (QPBDP), based on our previous research [9], to automatically segment sketches without any prior knowledge on the number and/or the type of target segments. QPBDP reformulates the stroke segmentation task as an improved DP problem and simultaneously uses penalty functions to evaluate local fitting errors. Comparing the existing stroke segmentation algorithms, the contributions of our method are as follows. First, a novel ‘global + local’ DP-based framework is proposed for automatic stroke segmentation. The overall perceptual error for stroke segmentation can be efficiently minimised in our method without any prior knowledge. Second, compared with our previous method PBDP [9], the computational cost is reduced significantly. Moreover, we give a benchmark database for performance evaluation in stroke segmentation. Experiments on our data set show that our proposed method is accurate and efficient for strokes of various sizes.

The rest of the paper is as follows. Section 2 gives a detailed literature review for existing stroke segmentation approaches and their difficulties in automatic stroke segmentation. Then, Section 3 introduces our QPBDP framework for stroke segmentation. Sections 4 and 5 give the detailed calculation of the penalty and error function. Section 6 presents the benchmark database and the performance evaluation method. Experimental results and discussion are illustrated in Section 7. Finally, we conclude our method in Section 8.

2 Related research

To present our problem clearly, we first define the following terms (see Fig. 1):
Fig. 1 Example to describe the defined terms

- ‘Stroke’ is a unit of a user’s original sketch input. It is the pen trajectory on the screen between each pair of pen-down and pen-up operations.
- ‘Segmentation point’ is a point on a stroke where the stroke is split apart, which is also called break point, such as points 15, 33, 54, ….
- ‘Fragment’ is a piece of an original stroke, separated by its segmentation points, such as the cursive line segment in black colour between points 33 and 54.
- ‘Segment’ is a fitting (regular) curve (primitive shape) of a fragment, such as straight line segment in grey (or blue in colour) between points 33 and 54.
- ‘Fitting error’ is the error of fitting a fragment to a segment of certain type, which is usually estimated by integral squared error or maximal single error.
- ‘Perceptual error’ is the error of fitting a stroke to a series of segments. It is based on the fitting error, but may not be equal to the total or average fitting error of all fragments.

Then, we categorise the existing approaches for stroke segmentation into two classes: ‘locally optimal approach’ and ‘globally optimal’ approach.

2.1 Locally optimal approach

Locally optimal methods detect segmentation points according to local geometric properties around each point in a sketch [10, 11]. Most of these approaches first determine a supporting region $h_i$ for each point $P_i$. Then, they calculate the significance of a point $P_i$ in a sketch and use it as an indicator to identify segmentation points within the supporting region. For instance, Rosenfeld and Johnston [10] detect significant maxima in average grey-level gradient by using a variable degree of smoothing. The curvature value at each point is calculated to detect the segmentation points. However, these methods can lead to incorrect results when edges are too close to each other [11]. The curvature calculation method is further improved in [12].

In locally optimal methods, how to decide accurate supporting regions is important to obtain correct segmentation results. In [13], the supporting region for a point is dynamically determined according to the length of a chord and the perpendicular distance from the point to the chord. A segmentation point is decided only when its curvature reaches the local maximum. However, unwanted segmentation points may be produced in the presence of noise. Ray and Pandyan [14] propose an adaptive method based on ‘variance of curvature’ to detect segmentation points in. In general, different regions of a curve have different roughness and so different amount of smoothing is required. Gaussian filters with different window sizes can be used to determine the best window size that the filter should have at each point of the curve. As a result, segmentation points are those points whose variances with the best window size reach local maximum.

Dynamic information, such as pen speed and curvature [15, 16], can also be used during the online segmentation procedure. Both the shape of a stroke and the motion of the pen tip are considered to provide a more customised sketch interface. The advantages of these approaches are that they are usually very fast and sometimes even parameter free. However, their accuracy is limited because of the local optimality and cannot be applied in off-line stroke segmentation applications.

2.2 Globally optimal approach

Globally optimal approach is also referred to as edge approximation. DP algorithms are proved to be helpful in stroke segmentation [15, 17–19]. Generally speaking, the existing DP-based methods for stroke segmentation have the following three phases: primitive shapes fitting, perceptual error measuring and DP to obtain stroke segmentation results. However, they face the following two limitations.

2.2.1 First limitation: explicit specification on the number or type for each segment: The stroke segmentation problem using a DP framework can be formulated as follows: given a digital curve $L$ composed of $N$-ordered points $P_N = \{p_1, p_2, \ldots, p_N\}$, approximate $L$-ordered fragments of predefined regular geometric segments $Q(Mq_1, q_2, \ldots, q_M)$, separated by $M-1$ segmentation points $S_{M-1} = \{s_1, s_2, \ldots, s_{M-1}\}$, with the following minimal perceptual error $e$

$$e = \sum_{i=1}^{M} f(s_{i-1}, s_i, q_i) \quad (1)$$

where $s_0 = p_1$ and $s_M = P_N$. Note that $f(s_{i-1}, s_i, q_j)$ is the fitting error generated when fitting points between $s_{i-1}$ and $s_i$ with a segment $q_j$. It is the error of fitting a fragment to a segment of certain type, which is usually estimated by integral squared error or maximal single error.

Then solution of (1) can be described as follows

$$E_p(i, j) = \min_{j \leq k < j+1} \left( E_p(k, j-1) + e_p(k, i) \right) \quad (j \in (1, M), i \in (j, N)) \quad (2)$$

where $e_p(k, i)$ is the minimal fitting error during fitting the point chain $P_{i+k} = \{p_i, p_{i+k-2}, \ldots, p_k\}$, $E_p(k, j)$ is the minimal perceptual error of fitting the point chain $P_i = \{p_1, p_2, \ldots, p_i\}$ with $j$ segments. $N$ is the number of points contained in the stroke and $M$ is a parameter that specifies the number of segments that the stroke contains.

Obviously, the discussed DP framework requires an explicit specification on the number of segments $(M)$ contained in $P_N$ in (2), and also the type of each target fitted stroke segment [20]. It limits the usage of the existing DP-based methods.

2.2.2 Second limitation: $O(N^2)$ computation complexity for error tolerance: Considering error
tolerance in DP-based stroke segmentation, the problem may be reformulated as: given an error tolerance \( \varepsilon \) and a stroke denoted by the point chain \( P_N = \{p_1, p_2, \ldots, p_N\} \), the set of resulting segment is \( Q \), and the segmented points denoted by \( S_Q = \{s_0, \ldots, s_{Q-1}, s_Q\} \), the problem of stroke segmentation with an error tolerance is modelled as follows

\[
\text{minimise } |Q| \text{ subject to } \sum_{i=1}^{Q} f(s_{i-1}, s_i, q_i) \leq \varepsilon
\]

(3)

where \( s_0 = p_1, s_Q = p_N, q_i \) is a segment in \( Q \) and \( f(s_{i-1}, s_i, q_i) \) is the fitting error of approximating the points between \( s_{i-1} \) and \( s_i/q_i \).

Solution to the optimisation problem of (3) has the following two steps:

**Step 1:** Compute the minimal perceptual error of fitting the first \( i \) points with \( j \) segments by recursively applying the following function

\[
E_p(i, j) = \min_{j \leq k \leq 1} \left( E_p(k, j-1) + c_p(k, i) \right)
\]

(4)

\[1 \leq j < i \leq N\]

The process starts with \( E_p(i, 1) = c_p(1, i) \) for \( i \in (2, N) \) and terminates at \( E_p(N, N-1) \). Clearly, the time complexity for this step is \( O(N^3) \). It is quite high for online applications. The execution of the framework is shown in Fig. 2.

**Step 2:** Arrange \( E_N = \{E_p(N, i) | i \in (1, N-1)\} \) in monotonic decreasing order. Obviously, the result is as follows

\[E_p(N, 1) \geq E_p(N, 2) \geq \cdots \geq E_p(N, N-1) = 0\]

(5)

As \( |Q| \) is the least number of segments needed to approximate \( P_N \) with the perceptual error less than \( \varepsilon \), then \( E_p(N, |Q|) \leq \varepsilon \). Therefore within \( O(N) \) operations, we can find

\[|Q| = \arg \min_k \{E_p(N, k) \leq \varepsilon\}, \quad 1 \leq k \leq N-1\]

(6)

Unfortunately, it is rather difficult to find a global error tolerance value that is suitable in all situations. For example, the approximating errors of fitting Figs. 3a and b with elliptical arcs are very close. As a result, a smaller tolerance value will result in more segments and a larger tolerance value will result in less segments.

### 3 QPBDP framework for stroke segmentation

Stroke segmentation techniques approximate a user’s sketchy input with regular geometric shapes (or segments), so that the presentation of the sketch is simplified and meaningful. As a result, the storage, retrieval and further analysis of the sketch can be eased. Therefore we can summarise two requirements for stroke segmentation:

**Requirement 1:** The fitting error should be as small as possible.

**Requirement 2:** The number/type of the segments should be as few/simple as possible.

In practice, the two requirements are contradicting. The more/less segments used to approximate a sketch and the smaller/larger the approximation error is. Existing DP frameworks only meet the first requirement by minimising the perceptual error defined in (1), but violate the second requirement. Moreover, the existing DP approaches require knowing the number of fragments before segmentation. Otherwise, it can only obtain the ‘best’ but meaningless result, which is a curve consisting of \( N-1 \) line segments.

To solve this problem, the following three approaches can be considered: user interaction, improved global properties or novel comprehensive properties. Actually, it is very hard to ask a user to determine the number of fragments, and the mentioned difficulties in Section 2.2.2 indicate that it is not sufficient to automatically determine segmentation number contained in a stroke by considering global properties (or perceptual error) only. However, if local properties such as curvature \( \text{CUR}(p) \) or significance \( S(p) \) of a point are simultaneously considered, it is possible to distinguish Figs. 3a and b as every point in Fig. 3a is of almost the same importance, whereas in Fig. 3b three points at the corner are more important than others.

Inspired by these, we propose a novel approach, QPBDP, to effectively and efficiently segment sketch strokes, simultaneously combining global and local properties in our DP framework. The segmentation accuracy is greatly improved in our method without any prior knowledge about segmentation number or type; moreover, the time complexity of the method is efficiently reduced to \( O(N^2) \).

#### 3.1 Problem reformulation

Given the point chain \( P_N \), we need to find a set of segments \( Q_M \) and corresponding segmentation points \( S_M = \{s_0, \ldots, s_M\} \),
where \( s_M \) such that the perceptual error \( e_M \) is minimised among \( e_m \) for \( m = 1, \ldots, N - 1 \). Moreover, we try to meet both the two mentioned requirements in stroke segmentation in our algorithm. Therefore the perceptual error \( e_M \), considering both fitting errors and penalties caused by segmenting the stroke at certain points, can be defined as follows

\[
e_M = \sum_{i=1}^{m} (f(s_{i-1}, s_i, q_i) + T_p(s_i))
\]  

where \( s_0 = p_1, s_M = p_N \) and \( T_p(s) \) is a penalty term measuring the incorrectness of segmenting the stroke at point \( s \). The illustration of the point chain \( P_N \), the segmentation points \( s_M \) and the segments \( Q_M \) are shown in Fig. 4.

As a result, the problem of stroke segmentation with QPBDP is mathematically reformulated as follows

\[
M = \arg \min_{m} \{ e_m \}, \quad 1 \leq m < N
\]

### 3.2 QPBDP-based stroke segmentation

QPBDP finds the optimal solution to the problem of approximating the point chain \( P_N \) with segments \( Q \) by computing the following function iteratively

\[
PB_p(i) = \min_{1 \leq k \leq i} \left( PB_p(k) + e_p(k, i) + T_p(k, \bullet) \right)
\]

where \( PB_p(i) \) is the minimal perceptual error for fitting the point chain \( P_i = \{ p_1, p_2, \ldots, p_i \} \) with segments. \( T_p(k, \bullet) \) measures the rationality of segmenting at \( p_k \) and is independent of the optimal solution to the subproblem \( PB_p(k) \). Fig. 5 shows the framework of our proposed method, which will be further explained in the next subsection.

QPBDP framework achieves the satisfying tradeoff between the approximate error and the segment number through the penalty function. In QPBDP, penalty function is used to produce perceptual error when a new segmentation point is introduced in the DP process, which suppresses the trends to increase globally perceptual error by introducing more fragments. Unless the perceptual error of fitting the points to the regular geometric segment is extremely large, our approach will generate longer and rational fragments.

### 3.3 Proposed framework

As our approach is based on a DP strategy, it is a bottom–up process for calculating the minimal error and a top–down process for obtaining the optimisation solution. Given the point chain \( P_N = \{ p_1, p_2, \ldots, p_N \} \), the process of solving the above optimisation problem is formulated as computing \( PB_p(N) \) from \( PB_p(i) \), \( T_p(i) \) and \( e_p(i, j) \) with \( 1 \leq i < j \leq N \).

\[
T_p(i) \text{ measures the rationality of splitting at } p_i \text{ and is independent of the optimal solution to the subproblem } PB_p(i).
\]

The detailed calculation is given as follows:

1. Calculate \( T_p(i) \): for each point \( p_i \), we calculate its penalty degree, which measures the rationality of segmenting at the point.
2. Calculate \( e_p(i, j) \): calculate the minimal fitting error and corresponding fitting type (straight line or elliptical arc) for each subpoint chain \( P_0(1 \leq i < j \leq N) \).
3. Calculate \( PB_p(N) \): according to (9), calculate the minimal fitting error for whole point chain in the bottom–up fashion; and
4. Construct the optimal solution from the computed \( T_p(i) \), \( e_p(i, j) \) and \( PB_p(N) \).

In Sections 4 and 5, we will give the detailed calculation of \( T_p \) and \( e_p \) in (9), respectively.

### 4 Penalty function selection

The penalty function \( T_p \) in QPBDP is the key to the success and should be carefully designed. As shown in Fig. 6, it is reasonable to segment at point \( B \) therefore the penalty should be small at point \( B \). However, considering the point \( D \) on segment \( AB \), the penalty should be large. Therefore we define the penalty function as

\[
T_p = F_L \times F_\theta
\]

where \( F_L \) is derived from the size of the stroke and \( F_\theta \) is derived from the angle information of the stroke.
4.1 Definition of $F_L$

Generally speaking, the value of $F_L$ is based on the context of a sketchy stroke because a suitable value of $F_L$ for a stroke of small size may not be suitable for a stroke of large size. Considering a simple stroke with three points $P=(A,B,C)$, as shown in Fig. 7. If $P$ is recognised as a line segment $AC$, then the fitting error is $e=l\times\cos(\theta/2)$ and no penalty is generated. If $P$ is divided at point $B$, then the penalty is $T_p=F_L \times F_\theta$ and no fitting error is generated. Intuitively, point chain $P$ should be a line segment if $\theta$ is sufficiently close to $\pi$ (e.g. $\theta=\delta=\pi/9$, which is a threshold) or should be split at $B$ otherwise. As a result, the following inequality holds:

$$
\begin{align*}
\left\{ \begin{array}{ll}
l \times \cos(\theta/2) & > F_L \times F_\theta \quad \text{if } \theta < \delta \\
l \times \cos(\theta/2) & < F_L \times F_\theta \quad \text{if } \theta > \delta 
\end{array} \right. 
\end{align*}
$$

From (11), we can obtain

$$
F_L = l \times \frac{\cos(\delta/2)}{F_\theta} \approx l
$$

It is difficult to exactly evaluate the value of (12), but as assumed in (10), $F_L$ is a function of the size of the stroke; it should depend on $l$ only. From (12), $F_L$ is a linear function of $l$. Therefore we define

$$
F_L(i) = \alpha \times l(i)
$$

where $\text{Dis}(p_i, p_{i+1})$ is the Euclidean distance between pixel points $p_i$ and $p_{i+1}$ and $\alpha$ is a constant. In our method, $\alpha$ is set to 1. If we consider the noise contained in the stroke,

![Fig. 6](image-url)  
*Fig. 6* Penalty generated at point $D$ should be larger than the penalty generated at point $B$

we can smooth the noise using a mask of $w$ and revise (13) as

$$
l(i, w) = \frac{\sum_{j=i-w}^{i+w-1} \text{Dis}(p_j, p_{j+1})}{2 \times w}
$$

From (14), we can obtain

$$
F_L(i, w) = \alpha \times l(i, w)
$$

$F_L(i, w)$ is set to infinity at points $\{p_i | i \in (1, w) \cup (N-w+1, N)\}$ because these points are either close to $p_1$ (start point) or $p_N$ (end point). If $\{p_i | i \in (1, w) \cup (N-w+1, N)\}$ were accepted as segmentation points, short segments would be generated. From our experiments, $w=5$ balances the smoothness and the complexity for most cases. Therefore we use the smoothing factor $w=5$ throughout this paper. Fig. 8 shows an example case for noise smoothness by masking the point chain. The noise point is shown in grey.

![Fig. 7](image-url)  
*Fig. 7* Calculation of $F_L$: $\theta = \angle ABC$ and $l = |AB| = |BC|

4.2 Definition of $F_\theta$

As it is more reasonable to separate a stroke at an acute corner than at a flat corner, which is shown in Figs. 9a and b, we can simply define $F_\theta$ as

$$
F_\theta(i) = \sin\left(\frac{\angle p_{i-1}p_ip_{i+1}}{2}\right)
$$

However, (15) is not sufficient. Consider Figs. 9b and c, although the angle at point $i$ is the same, the penalty $|T_\theta(i)|$ in (b) should be smaller than the penalty $|T_\theta(i)|$ in (c). Therefore we define $\Delta_\theta$ as

$$
\Delta_\theta(p_i, p_j) = \frac{|\angle p_{i-1}p_ip_{i+1} - \angle p_{j-1}p_jp_{j+1}|}{2}
$$

$$
F_\theta(i) = \frac{1}{\sin(\Delta_\theta(p_i, p_{i-1})) + \sin(\Delta_\theta(p_i, p_{i+1}))}
$$

![Fig. 8](image-url)  
*Fig. 8* Noise smoothness by masking the point chain. The noise point is shown in grey

![Fig. 9](image-url)  
*Fig. 9* Calculation of $F_\theta$ for point $i
The definition of \( \theta(w) \) is based on the difference between adjacent angles. The idea is originated from the basic observation that within a segment (an elliptical arc or a straight line), the angle at adjacent points changes smoothly. A rapid change between adjacent angles is a good indicator of a new segmentation point. As shown in Fig. 9c, angles change smoothly; thus large angle penalty [\( \theta(w) \)] will be generated if the stroke is split at any point. Although in Figs. 9a and b, the change at point \( p_i \) (segmentation point) is larger than any other points; thus the angle penalty is the smallest, which is consistent with our expectation. If we consider the noises contained in the stroke, we can use the same mask used in (14) and revise (16) as (see (17))

4.3 Definition of \( T_p \)

From the discussions in Sections 4.1 and 4.2, we define \( T_p \) as follows

\[
T_p(i, w) = \begin{cases} 
F_\theta(i, w) \times F_L(i, w) \times F_i(i, w) < \delta_{\max} \\
+\infty & \text{otherwise} 
\end{cases} 
\]  

(18)

where \( \delta_{\max} \) is a threshold. If the calculated value of \( F_\theta(i, w) \times F_L(i, w) \) exceeds the threshold, it means that it is absolutely unreasonable to split the stroke at point \( i \). Therefore the penalty is set to positive infinity to avoid recognising the point as a segmentation point.

5 Fragment fitting

As there are two possible common types (and more can be extended to) of curves to be used in fragment fitting in our method, the perceptual errors of both line segment fitting and elliptical arc fitting are computed and compared to obtain the minimum value

\[
e_k(i, j) = \min \{ \eta_L \times \epsilon_L(i, j), \eta_A \times \epsilon_A(i, j) \} 
\]  

(19)

where \( \eta_L \) is the type weight for line segment and \( \epsilon_L(i, j) \) is the error of fitting point chain \( P_{i+1} = \{ p_n, p_{n+1}, \ldots, p_t \} \) to a line, \( \eta_A \) is the type weight for arc segment and \( \epsilon_A(i, j) \) is the error of fitting \( P_{i+1} = \{ p_n, p_{n+1}, \ldots, p_t \} \) to an arc.

As a line segment is preferred over an elliptical arc if their fitting errors are close, and moreover the ratio of memory space required to store the parameters of a line segment and an elliptical arc is 1:2, we set \( \eta_L \) to 1 and \( \eta_A \) to 2 in this paper. \( e_k(i, j) \) is given by

\[
e_k(i, j) = \sqrt{\sum_{k=1}^{j} d^2(p_k, f_k)} 
\]  

(20)

where \( f_k \) is the line passing through points \( p_i \) and \( p_j \) and \( d(p_k, f_k) \) is the distance from point \( p_k \) to line \( f_k \). \( \epsilon_A(i, j) \) is given by

\[
\epsilon_A(i, j) = \sqrt{\sum_{k=1}^{j} f_k^2(p_k)} 
\]  

(21)

where \( f_A \) is the ellipse that fits \( P_{ij} \) best [21].

6 Benchmark database and performance evaluation

In order to evaluate our methods, we have built a benchmark data set for stroke segmentation, including line segment, ellipse, elliptical arc and their combinations. Some example curves are illustrated in Fig. 10.

We developed a simple tool for users to draw a stroke and specify the segmentation points and the fitting results for each segment. A number of users were asked to draw each stroke in the sample list (which contains 45 standard types of strokes) and specify the segmentation results. The segmentation results, known as the ground truths, together with the original sketchy curve, are saved to an InkML file for each stroke. In total, we have collected 6154 = 270 user-sketching strokes and their ground truths, which are available at http://www.cs.cityu.edu.hk/~liuwy/QuickDiagram/StrokeSegmentation/.

In such benchmark data set, each InkML file contains an original sketchy stroke, which is represented as a chain of points, labelled as \( P \), and its corresponding ground truth information, including:

![](Image)

**Fig. 10** Example curves

\[
p_i(w) = \frac{\sum_{j=1}^{w} p_{i+\text{sign}(w)xj}}{|w|} \\
\Delta(p_i(w), p_j(w)) = \frac{-p_i(-w)p_j(w) - p_j(-w)p_i(w)}{2} \\
F_\theta(p_i(w), p_{i+1}(w)) = \begin{cases} 
1, & i = 1 \text{ or } N \\
\sin(\Delta(p_i(w), p_{i-1}(w))) + \sin(\Delta(p_i(w), p_{i+1}(w))), & i \in (w, N - w) \\
+\infty, & \text{otherwise} 
\end{cases} 
\]  

(17)
• Segment types (line or arc).
• Segmentation points (including the start point and the end point).
• Segment attributes/features.

The attributes of a line segment are its start and end points (represented as their point indices in the original stroke). The attributes of an arc segment include the centre, major axis, minor axis, rotate angle of the major axis and the start and end angles of the elliptic arc. Especially, TopLeftPoint and BottomRightPoint in the InkML file are used to calculate the centre and the lengths of the major axis and the minor axis of the ellipse.

Then, we evaluate the stroke segmentation results as follows. We use \( S \) to denote the segmentation points of a stroke \( P \) generated by a stroke segmentation algorithm, and use \( T \) to denote the segmentation points in the ground truth of \( P \). A mapping from \( S \) to \( T \) is created if the size of \( S \) is smaller than \( T \). Otherwise, a mapping from \( T \) to \( S \) is created. Assume the size of \( S \) is smaller than \( T \) and the mapping function is \( M(p) = q(p \in S, q \in T) \). \( M(\bullet) \) should be a one-to-one and order-preserving mapping. That is, for any \( M(p_i) = q_r \), \( M(p_j) = q_l \) and \( l < r \).

For each mapping \( M(\bullet) \), we define the following distance

\[
Ds_M(S, T) = \sum_{p \in S} \text{Dis}(p, M(p))
\]

(22)

where \( \text{Dis}(p, q) \) is the Euclidean distance from point \( p \) to point \( q \). During all of the mappings, we need to find a mapping \( M_{\text{min}}(\bullet) \) that could minimise the value \( Ds_M(S, T) \). Defining \( S_i = \{p_i | i \leq k, p_i \in S\} \) and \( T_i = \{q_i | i \leq k, q_i \in T\} \).

The following algorithm finds \( M_{\text{min}}(\bullet) \) for \( S \) and \( T \) (we just consider the situation when the size of \( S \) is smaller than \( T \)) in \( O(N^2) \) time complexity

\[
D(i, j) = \begin{cases} 
  \text{if } i > j & \text{if } i > j \\
  \min \{D(i, j + 1), D(i + 1, j + 1) + \text{Dis}(p_i, q_j)\} & \text{else}
\end{cases}
\]

(23)

where \( D(i, j) \) is the minimal distance of \( Ds M(S_i, T_j) \).

After obtaining the bipartite matching result, we regard the segmentation for a stroke as correct ‘if’ for each \( p_i \in S \), its matching \( p_j \in T \) exists and the distance between them does not exceed a predefined error threshold (which is five in our benchmark data set).

7 Experiments and discussion

The proposed stroke segmentation method was implemented in Visual Studio C++ on a Pentium(R) 4 central processing unit 3.00 GHZ, 512 MB of random access memory computer.

7.1 Effectiveness of QPBDP

We first test our QPBDP method on the benchmark database and the accuracies of QPBDP are 92% on the average, where 248 out of the 270 testing samples are correctly segmented. In the remaining incorrect 22 samples, the error is mainly caused by a very ambiguous input. In some situations, the segments are connected too smoothly, so that even human users have difficulties detecting it, illustrated as the red split point in the left example of Fig. 11. Sometimes the stroke is drawn with extra sharp turns that an additional split point is detected, such as the middle green split point in the right example of Fig. 11. By incorporating both global and local error constraints, we achieve or maintain the high segmentation accuracy (or with little loss of accuracy), whereas at the same time reduce the time complexity for online applications.

Then, we compared our method QPBDP with the Rosenfeld–Johnston angle detection procedure [10], the adaptive method of corner detection [14] and show some results in Fig. 12. The first row of Fig. 12 shows the segmentation results by the Rosenfeld–Johnston angle detection procedure. It is easy to see that the method misses some segmentation points at the corners and on the elliptical arc. For example, in Fig. 12b, an unwanted segmentation point is generated. Moreover, the location of some detected segmentation points is just close to but not exactly at the corner. The second row shows the segmentation result by the adaptive method of corner detection. The result is much better than the previous method as it does not miss any segmentation point.

Fig. 12 Segmentation results

\( a \) and \( b \) Using the Rosenfeld–Johnston angle detection procedure [10]
\( c \) and \( d \) Using the adaptive method of corner detection [14]
\( e \) and \( f \) Using QPBDP, respectively
However, it also generates an unwanted segmentation point on the elliptical arc, as shown in Fig. 12d. The last row shows the segmentation result by our QPBDP method and we can see that the result is exactly as expected. Therefore our QPBDP method is robust in stroke segmentation and outperforms the other two methods.

7.2 Improvement of QPBDP compared with our previous work

Next, we evaluate the segmentation accuracies of QPBDP with our previous work [9] on this benchmark data set. Liu et al. [9] calculate the following function iteratively

\[ E_p(i, j) = \min_{1 < k < j - 1} \left( E_p(k, j - 1) + e_p(k, i) + T_p(k, i, j) \right) \quad (1 \leq j < i \leq N) \]

(24)

where \( T_p(k, i, j) \) is defined as follows

\[ T_p(k, i, j) = \delta_p \times \sin \left( \frac{\text{angle}(q_{k, j-1}, q(k, i))}{2} \right) \]

(25)

where \( \delta_p \) is the basic segmentation penalty; \( q_{k, j-1} \) is the last segment in the optimal solution that approximates point chain \( P_k \) with \( j-1 \) segments, \( q(k, i) \) is the segment used to approximate point chain \( P_{k-1}(p_0, p_{k-1}, \ldots, p_j) \) and angle \( \{q_{k, j-1}, q(k, i)\} \) is the angle formed by segment \( q_{k, j-1} \) and \( q(k, i) \).

Comparing with [9], QPBDP reduces the computational complexity from \( O(N^2) \) to \( O(N^3) \). As mentioned in Sections 3.2 and 3.3, the computation of \( T_p \) in our approach is independent of the optimal solution to the subproblem. However, the method proposed in [9] is not. Table 1 shows the processing speed comparison results. We can see that QPBDP is fast enough for online applications, which usually deal with strokes containing less than 200 points.

Moreover, QPBDP finds an optimal solution compared with [9]: since the penalty function \( T_p(k, i, j) \) defined in (25) depends on \( q_{k, j-1} \), it is possible that the \( P_k \) can be approximated by a set of \( j-1 \) segments \( Q_{j-1}(q_1', q_2', \ldots, q_{j-1}') \) with a perceptual error \( E_p(k, Q_{j-1}') \) that satisfies

\[ E_p(k, Q_{j-1}') \geq E_p(k, j - 1) \]

\[ T_p(k, i, j) > T_p(k, i, j) \]

\[ E_p(k, Q_{j-1}') + T_p(k, i, j) < E_p(k, j - 1) + T_p(k, i, j) \]

(26)

Therefore \( Q_{j-1}(q_1', q_2', \ldots, q_{j-1}') \) is not the optimal solution to approximate \( P_k \) with \( j-1 \) segments. In other words, the minimal perceptual error \( E_p(i, j) \) is not necessarily originated between \( E_p(k, j - 1) \) and (24) actually gives a suboptimal solution.

The optimisation problem in (3) can also be solved by another DP framework, which is directly from the formulation of the problem

\[ PN(e) = \begin{cases} 
1 & \text{if } e_p(1, i) \leq e \\
\min_{1 < k < i} \{ PN(k, e - e_p(k, i)) + 1 \} & \text{if } e_p(1, i) > e 
\end{cases} \]

(27)

where \( PN(e) \) is the least number of segments needed to approximate \( P_N \) is \( PN(N, e) \). The fitting error when approximating point chain \( P_{j-1}=(p_0, p_{j-1}, \ldots, p_i) \) with segment \( q \) is denoted as \( e_p(i, j) \). \( e_p(i, j) \) is calculated by

\[ e_p(i, j) = \sum_{k=i}^{j} d(p_k) \]

(28)

where \( d(p_k) \) is used to measure the distance from point \( p_k \) to segment \( q \). The calculation of \( PN(e) \) is quite similar to the calculation of \( e_p(i, j) \), which is shown in Fig. 2. However, as \( e \) is not an integer, it should be rounded up to (in the unit of pixel)

\[ e_p(i, j) = \left\lceil \sum_{k=i}^{j} d(p_k) \right\rceil \]

(29)

Equation (27) indicates that the least number of segments needed to approximate \( P_N \) is \( PN(N, e) \), which could be calculated within \( O(eN^2) \) operations. We set \( e = \delta N \), which means that the average fitting error for each point is limited to \( \delta \). Therefore the overall time complexity for (27) is \( O(\delta N^3) = O(N^3) \).

The first/second row of Fig. 13 shows the segmentation results of the algorithm given by (27) and (4), respectively. The segmentation points are marked by small rectangles. Comparing Fig. 13a with Fig. 13b, we may find that smaller \( \delta \) (which is defined after (29) as the threshold of the average fitting error at each point) favours strokes consisting of multiple lines, whereas larger \( \delta \) favours elliptical curves. As discussed in Section 2.2.2, it is difficult to find a suitable threshold for both cases. On the contrary, penalty functions can help segment strokes correctly, as shown in Fig. 13c (QPBDP) and Fig. 13f [9].

7.3 Performance of QPBDP with complex symbols

The QPBDP approach that we propose is generic and can be applied to multiple domains of hand-drawn symbols. To further verify the performance of the proposed method, we show some experimental results of complicated symbol segmentation using QPBDP in Fig. 11. Here, occlusion, distortion and non-isotropic scaling exist frequently as in real cases of hand-drawn symbols. Five arbitrary hand-drawn strokes are shown. It is easy to see that all corner points are detected as split points. Furthermore, all the segmentation results are still reasonable and acceptable.
stroke segmentation. Experiments also show that QPBDP is accurate and efficient for strokes of various size/number/type of segments in the segmentation result while the overall perceptual error is minimised.

9 Acknowledgments

The work described in this paper was fully supported by the Natural Science Foundation of China under grant nos. 61272218 and 61021062, the 973 Programme of China under grant no. 2010CB327903, Shanghai Pujiang program under grant no. 13PJ1408200 and the Fundamental Research Funds for the Central Universities under grant no. 2100219038.

10 References


Fig. 13 Segmentation results of the algorithm

a Segmentation result for (27) with δ = 0.05
b Segmentation result for (27) with δ = 0.5
c Segmentation result for (4) with δ = 0.3
d Segmentation result for (4) with δ = 0.5

Therefore our QPBDP method is also robust in segmenting complicated symbols and can still give suitable results.

8 Conclusions

We have investigated the possibility of developing a globally optimal approach to stroke segmentation which does not rely on the number and/or the type of segments in advance. For this purpose, we have tried to use the DP framework for automatic stroke segmentation and propose the following two possible extensions: the error tolerance extension and the penalty-based extension. Through theoretical analysis and empirical evaluation, we found the proposed QPBDP outperforms other possible solutions. QPBDP is a new framework which considers local geometric properties to measure the correctness of splitting a stroke at a particular point and automatically detect the number/type of segments in the segmentation result while the overall perceptual error is minimised. Hence, QPBDP combines together the locally optimal approaches and globally optimal approaches to