Hindawi
Mathematical Problems in Engineering
Volume 2021, Article ID 8881187, 11 pages
https://doi.org/10.1155/2021/8881187

Research Article

Segmentation of Online Freehand Sketching Based on Speed Feature

Guanfeng Wang, Shouxia Wang, Jingjing Kang, and Shuxia Wang

School of Mechanical Engineering, Northwestern Polytechnical University, Xi'an, China

Correspondence should be addressed to Shuxia Wang; 2008wangshuxia@163.com

Received 9 September 2020; Revised 14 May 2021; Accepted 23 May 2021; Published 2 June 2021

Academic Editor: Maria Patrizia Pera

We present a novel method to extract speed feature points for segmenting hand-drawn strokes into geometric primitives. The method consists of three steps. Firstly, the input strokes are classified into uniform and nonuniform speed strokes, representing a stroke drawn at relatively constant or uneven speeds, respectively. Then, a sharpening filter is used to enhance the peak features of the uniform speed strokes. Finally, a three-threshold technique that uses the average speed of the pen and its upper and lower deviations is used to extract speed feature points of strokes. We integrate the proposed method into our freehand sketch recognition (FSR) system to improve its robustness to support multiprimitive strokes. Through a user study with 8 participants, we demonstrate that the proposed method achieves higher segmentation efficiency in finding speed feature points than the existing method based on a single speed threshold.

1. Introduction

Freehand sketching is a natural and intuitive tool for designers to communicate and visualize their ideas during the conceptual stage of product design. Generally, concept designers use paper and pencil to share, communicate, and record new ideas. However, the paper becomes clutter with all sorts of changes. With the development of tablet computers and electronic pencils, pen-based interfaces have become more and more popular. In recent years, sketch-based user interfaces have been widely studied to enable users to interact with computers with sketches. Sketch-based interfaces have the potential to combine the benefits of the creative and freed nature of pen-and-paper drawing and the processing power of computers [1].

Sketch recognition is the core technology of sketch-based user interfaces, aiming to capture the users' design intention. In interactive sketch systems, dynamic sketching information such as coordinates, pen speeds, and pressures is recorded when the user draws a stroke, which can help understand graphic objects. However, the fuzzy characteristics of sketches and the differences in user's drawing habits have brought about great difficulties to sketch understanding. Geometric primitives are the basic elements of technical drawings, such as circuit diagrams, mechanical drawings, and chemical molecular structure diagrams. In many cases, strokes drawn by users contain multiple primitives, which increases the difficulty of sketch recognition. Sketch-based interfaces should support multi-primitive strokes to avoid placing constraints on the drawing process.

Stroke segmentation, also known as corner detection, aims at dividing the original strokes into lines, arcs, circles, and other geometric primitives. It is the primary step of sketch regularization and recognition. Existing stroke segmentation methods are mainly based on shape features, such as concavity, curvature, or other dynamic information obtained as the pen moves, such as speed and pressure. Most existing stroke segmentation methods start with establishing a set of candidate feature points. Then the valid segmentation points are determined based on heuristic approximation methods [2–5] or machine learning methods [6, 7]. Speed-based feature points are essential parts of the segmentation point set. Using the pen speed information to

segment strokes is based on the observation that users often slow the pen at the corner point. The correct detection of speed feature points can improve the efficiency of stroke segmentation. Many sketch-based interfaces [4, 6, 7] used the method of Sezgin et al. [2] to extract speed feature points, which locates segment points at speed minima that are slower than a fixed threshold. When drawing strokes at a relatively constant speed, the method may result in extra segmentation points, leading to recognition errors.

We propose a novel method to extract speed feature points for segmenting hand-drawn strokes into geometric primitives. For an input stroke, we first classify it into a uniform speed or nonuniform speed stroke. Then, a sharpening filter is used to sharpen the speed curves of the stroke to enhance its peak features. Finally, we adopt a three-threshold segmentation technique, which uses the average speed of the pen and its upper and lower deviations to group the sampling points of the strokes; and the speed minimum of each group is defined as a speed feature point.

We have implemented the proposed speed feature point extraction method in our freehand sketch recognition (FSR) system and performed a user study with 8 participants to test the method's effectiveness. The results showed that the speed feature points extracted by the proposed method are closer to the valid segmentation points than those of the existing speed feature extraction method based on a fixed speed threshold.

The paper is organized as follows: Section 2 introduces the related work on stroke segmentation approaches; Section 3 describes the proposed method for segmenting strokes using speed feature; Section 4 reports the user study; Section 5 gives the experiment and analysis; Section 6 is the conclusion.

2. Related Work

As the core content of stroke segmentation, many researchers have studied the detection of feature points such as the corner points and tangent points of digital strokes. These methods have evolved from heuristic methods to machine learning methods.

2.1. Feature Points Detection. Some segmentation methods only use strokes' geometric features such as concavity and curvature to find corners. Wolin et al. [8] presented ShortStraw, which introduces the concept of "straws" and uses a small window to detect continuous segments of a polyline. Our previous method [9] divides strokes into convex and concave strokes according to whether the vector direction of preprocessing points changes. The method is challenging to determine tangent points. The main problem of segmenting strokes based only on geometric features is that they are susceptible to local noise and the local structure of the sketched curve. Feng and Viard-Gaudin [10] presented a stroke segmentation method based on multilevel contexts. The method combines geometry features and Hidden Markov Model to locate the segmentation point and determine the primitive type so that smooth strokes can be effectively processed. However, due to the inconsistency between the continuity of curvature calculation and the discontinuity of curvature between primitives, some short segments in the dashed line will be recognized as arcs, which will reduce the recognition rate of lines.

Some segmentation methods combine the shape features and other dynamic information of the strokes, such as the speed, time, and pen pressure [3, 4, 7–10]. Qin et al. [11] proposed a heuristic knowledge-based corner point detection algorithm that uses drawing speed, acceleration, and linearity. Pen speed information is used to influence its adaptive segmentation threshold. Kim and Kim [12] segmented strokes by detecting the points of high curvature and proposed a curvature estimation method, which only considers local shape information such as convexity and monotonicity. Pen speed was also used to help determine the final segmentation points, but it was not used as the dominant feature. However, the method can cause false positives in curves with sharp changes in direction and monotonicity. Xiong and LaViola [13] developed IStraw, a corner-finding method based on ShortStraw [8] to segment strokes containing curves and arcs. Pen speed information is used to influence the curvature-dominated segmentation threshold. Sezgin et al. [2] used stroke curvature and pen speed to infer the user's segmentation intention. The speed minima below a threshold computed from the average pen speed along the stroke are added to the candidate segment point set. The set also includes curvature maxima where the pen speed is again below a threshold. The method solves the problem of speed feature extraction to a certain degree and has been widely referenced in many sketch research. However, the method has some inherent deficiencies: (1) local noise can cause grouping error; (2) the relatively fixed threshold may result in a noise-sensitive algorithm to strokes segmentation.

2.2. Heuristic Methods. Many stroke segmentation researches use heuristic methods to segment or merge the initial stroke segments to improve the corner detection results [2–5]. Sezgin et al. [2] refined the initial segmentation by iteratively adding segment points until the error of fit between the line segments and the raw stroke is less than a threshold. The method does not use the segment merge process. SpeedSeg [4] uses an improved heuristic method to merge and split the initial segments. However, it depends on preset parameters. MergeCF [3] eliminates false positive corners by continuously merging smaller stroke segments with similar larger stroke segments. Albert et al. [5] used parametric cubic curves approximation to detect corner vertices and tangent points in strokes. They later improved the method by establishing a discriminative function rather than a heuristic threshold when detecting corners and introducing some key parameters independent of the scale to obtain curves [14]. Chieppa et al. [15] presented a multilevel sketch segmentation algorithm unifying the wavelet approach. The method starts from a cubic B-spline curve approximation, and it is dedicated to defining the complex curves' control points. However, the segment results can be

sensitive to the threshold they used. Wang and Hu presented an approach to detect tangent vertices based on the curve type of substroke after corners extraction. DPFrag [16] uses a globally optimal segmentation method that learns segmentation parameters from data and generates segments by combining primitive recognizers in a dynamic programming framework. The advantage of this method is that there is no need to adjust the parameters manually. Wang et al. proposed a stroke segmentation method called quick penalty dynamic programming, which extended the dynamic programming framework with a customizable penalty function. With the help of the penalty function, the method can segment strokes without knowing the number or type of fragments contained in the stroke in advance. These approaches rely on heuristics and empirical parameters, which may limit their extensibility and reduce their adaptability to different users and devices.

2.3. Machine Learning Methods. Some stroke segmentation methods use machine learning techniques to determine the true corner points of a stroke from a set of candidate points [6, 7, 17]. ClassySeg [6, 7] is a typical machine learning-based stroke segmentation technique to split hand-drawn pen strokes into lines and arcs. The technique uses multiple stroke features from existing methods to train a statistical classifier to identify which candidate points are correct. It can be easily tuned for specific users, specific shapes, and specific drawing hardware. However, ClassySeg relies on local decisions to construct a corner set, and it misses finding a globally optimal solution. RankFrag [17] uses machine learning technique to find corner points in hand-drawn strokes. The method groups stroke points by iteratively extracting them from a candidate corner set. The points extracted in the last iterations are considered to be corner points. Zheng et al. [18] proposed a corner and tangent point detection method for sketched strokes with a deep learning approach. The method improves stroke shapes and biased datasets through multiscaled point contexts and a vote scheme. Therefore, it is robust for users. Machine learning-based stroke segmentation methods are more general and extensible compared to heuristic-based approaches. Our stroke segmentation method aims to obtain more accurate speed feature points. These points can be used as part of the candidate segmentation point set and then are identified as true segmentation points by machine learning methods.

3. Speed-Based Stroke Segmentation Method

Our freehand sketch recognition system (FSR system) input is a set of digital strokes drawn by the user with a tablet pen or mouse. Through stroke preprocessing, primitive recognition, stroke segmentation, and parameter fitting, a sketch is represented as a line drawing. The stroke processing flow of the FSR system is shown in Figure 1. Firstly, the stroke is preprocessed to obtain its polyline approximation. Secondly, a primitive recognition method based on fuzzy theory [19] is used to recognize the geometric type of the stroke. The stroke will be directly fitted to a parameter curve if it is recognized as a single primitive. Otherwise, it is divided into a set of continuous geometric primitives in the segmentation step.

The speed-based stroke segmentation consists of three steps: (1) classify the stroke into a uniform and nonuniform speed stroke, representing a stroke drawn at relatively constant or uneven speeds; (2) sharpen the speed waveform if it is a uniform speed stroke; and (3) extract the speed feature points based on the three-threshold technique. These three steps are described in detail in the following subsections.

3.1. Stroke Definition and Classification. In the FSR system, a stroke, denoted as S, is stored as a list of time-stamped coordinates sampled along the trajectory of the stylus; that is, $S = \left\{\overrightarrow{P}_i(x_i, y_i, t_i); 0 \le i < n\right\}$, where n is the number of the sampling points of the stroke S. The pen speed at the i^{th} sampling point, P_i, is defined as

$$v_i = \begin{cases} \dfrac{\sqrt{\left|\overrightarrow{P}_i - \overrightarrow{P}_{i-1}\right|}}{t_i - t_{i-1}}, & 0 < i < n-1, \\ 0, & i = 0; i = n-1. \end{cases} \tag{1}$$

Therefore, we obtain a speed value list $V = \{v_i; 0 \le i < n\}$ corresponding to the sampling point list of the stroke S. As described in [2], the average speed is calculated as the length of the stroke, L, divided by the stroke execution time, t, where the stroke length is defined as the sum of the distance between every two consecutive points:

$$v_{\text{avg}} = \frac{L}{t} = \frac{\sum_{i=1}^{n-1}\left|\overrightarrow{P}_i - \overrightarrow{P}_{i-1}\right|}{t}. \tag{2}$$

According to the change characteristics of the speed values of the sampling points along the stroke, we divide the strokes into uniform and nonuniform speed strokes, respectively, representing strokes drawn at relatively uniform and uneven speeds. Then, we use different filtering and segmentation methods for different types of strokes.

For an input stroke, if the ratio of the number of its resampling points to its length is not greater than a given value ξ_{thr} (default value: 0.15) and the average stroke speed v_{avg} is larger than a set value v_{thr} (default value: 200), the stroke is defined as a uniform speed stroke. Otherwise, it is a nonuniform speed stroke.

$$\begin{cases} \dfrac{n}{L} \le \xi_{\text{thr}}, \\ v_{\text{avg}} > v_{\text{thr}}. \end{cases} \tag{3}$$

3.2. Three-Threshold Segmentation Method for Nonuniform Speed Strokes. We use a three-threshold segmentation method to identify speed feature points for nonuniform speed strokes, which consists of four steps.

Firstly, we average the speed value of each sampling point with the speed values of its two neighboring points to decrease the noise in the speed data. The speed values of the

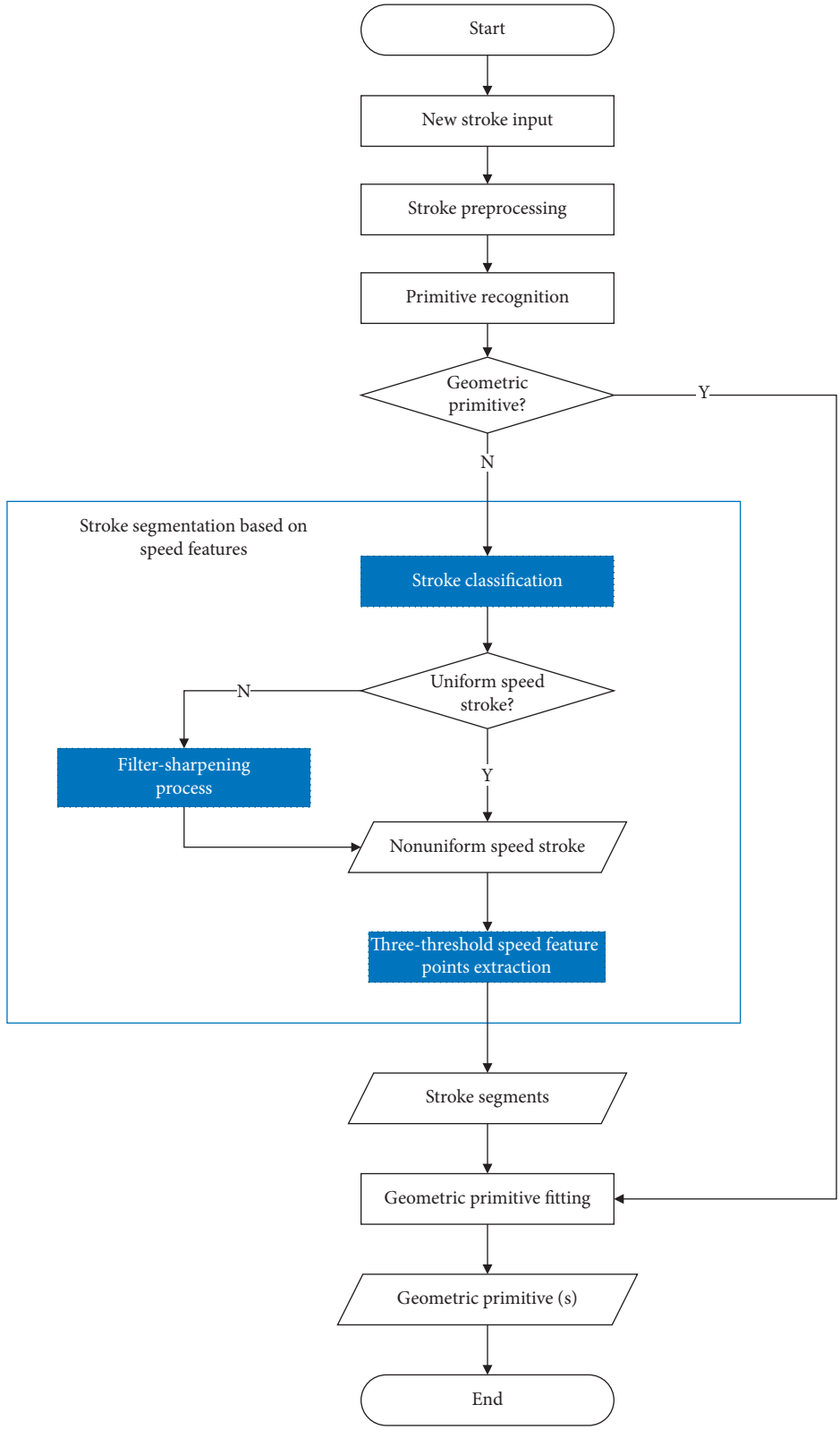


Figure 1: Sketch processing flow of the FSR system.

first and last resampling points of the stroke are set to 0. Moreover, the speed values of the second point and the second-to-last point of the stroke are, respectively, set equal to the speed of the third point and the third-to-last point.

Secondly, we define three speed thresholds as the baselines for speed feature point computation. The first threshold is the average stroke speed v_{avg}, which classifies the stroke points into low-speed points $(v_i < v_{\text{avg}})$ and high-

speed points ($v_i > v_{\text{avg}}$). The second and third thresholds are, respectively, defined as the average of the speed values of all high-speed points and all low-speed points, and they are named as high-speed threshold, v_{H}, and low-speed threshold, v_{L}.

As shown in Figure 2, the speed curve of a stroke is divided into several curve segments by its intersections with the isoline v_{L}. We define $L_j\,(0 \le j < m)$ as the projections of the intersection points on the abscissa axis, where L_0 and L_{m-1} are, respectively, the projections of the start point and end point of the speed curve on the abscissa axis. The curve segments between two consecutive projection points $L_{j,j+1}$ are classified as lower-speed curve segments and higher-speed curve segments. The characteristics of these speed curve segments include the following: (1) the first and last curve segments are both low-speed curves, (2) lower speed curve segments and higher speed curve segment appear alternately, and (3) each curve segment contains at least one speed value. In previous studies [2, 4], researchers generally believe that there should be at least one speed feature point in each low-speed curve segment. However, through experiments, we found that when the user's degree of freedom of drawing increases, there may be no speed feature points in the lower-speed curve segment. For this reason, we set v_{H} as the supplementary baseline to revise the lower-speed curve segments, and j is initially set to 1:

(a) If the jth higher-speed curve segment and the isoline v_H intersect, go to step (d)

(b) If the high-speed curve segment and the isoline v_{avg} intersect and the number of the intersection points is more than a set value (default value: 3), go to step (d)

(c) Combine the higher-speed stroke segment with its previous and subsequent low-speed segments to form a new lower-speed curve segment

(d) Perform the above three steps on the next higher-speed curve segment $(j = j + 1)$.

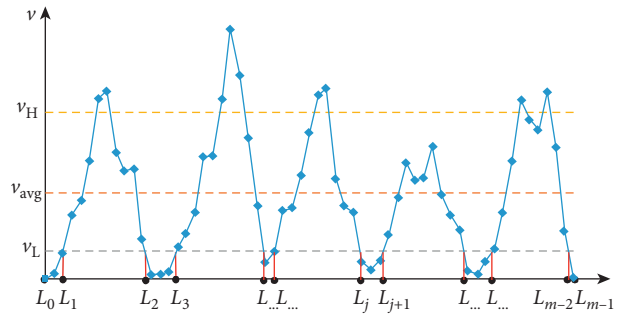


FIGURE 2: Three speed thresholds: average stroke speed, v_{avg}, high-speed threshold, v_{H}, and low-speed threshold, v_{L}.

Finally, we select the point with the smallest speed value in each revised lower speed curve segment as the speed feature points of the stroke$S$.

3.3. Filter-Sharpening for Uniform Speed Strokes. We use a filter-sharpening processing method to reduce the number of sampling points of the uniform speed strokes and enhance the peak features of their speed waveforms. Thereby we can transform the uniform speed stroke into the nonuniform speed stroke and use the three-threshold segmentation method to process the stroke. The filter-sharpening method for a uniform speed stroke S consists of two steps.

Firstly, we smooth the speed data by averaging the speed value at every sampling point with its neighboring T_d points. T_d is computed as equation (4), where k_s represents the control coefficient (default value: 0.04).

$$T_d = \frac{n}{L * (\xi_{\max} - k_s)}. \tag{4}$$

Secondly, we sharpen the smoothed speed waveform from the former step to enhance its peak characteristics using equation (5) to get the revised speed value list of the stroke S, which is stored as $\{v'_j; 0 \le j \le (n-1)/T_d + 2\}$:

$$v'_j = \begin{cases} -v_{\text{avg}} * T_v, & i = 0; i = \left(\dfrac{n}{T_d}\right) + 2, \\ \dfrac{T_v}{T_d} \displaystyle\sum_{j=(i-1)*(T_d-1)+1}^{(T_d-1)*i+1} \left(v_j - v_{\text{avg}}\right), & 0 < i < \left(\dfrac{n}{T_d}\right) + 2, \end{cases} \tag{5}$$

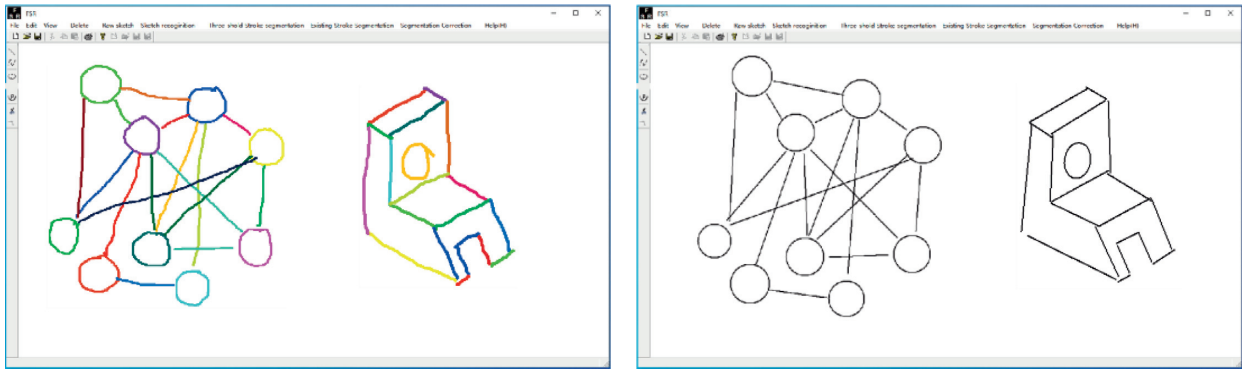
where T_v is the linear sharpening coefficient determined by the following equation:

$$T_v = \frac{v_{\max}}{v_{\text{avg}}}. \tag{6}$$

Because the endpoints of the stroke are defined as the speed segment points, when the residual point is less than T_d, there is no need to reduce the span of the smooth filtering to calculate the remaining speed feature points.

4. User Study

4.1. Prototype. We developed a freehand sketch recognition (FSR) system written in C++ using Microsoft Foundation Class (MFC) in Visual C++ 6.0. The FSR system integrates modules such as sketch input, stroke preprocessing, sketch recognition, and primitive fitting. The interface of the FSR system is shown in Figure 3. It uses a mouse or tablet as an input tool, allowing users to sketch as easily as using a pencil and paper (see Figure 3(a)). The input sketch strokes are first



(a) (b)

Figure 3: Interface of the FSR system. (a) Input strokes. (b) Recognized parametric curves.

recognized as straight lines or conic curves and then are fitted as parametric curves (see Figure 3(b)). We integrated the proposed speed-based stroke segmentation approach with the FSR system to improve its robustness to support multiprimitive strokes. To test the segmentation efficiency of the proposed method, we conducted a user study using several shapes drawn repeatedly by multiple users. The input device used in the user study was a Wacom pen display, as shown in Figure 4.

Figure 4: Wacom display with its digital stylus used in the user study.

4.2. Testing Examples. As shown in Figure 5, the shapes we used in the user study were selected from the existing stroke segmentation research [4, 6, 7, 10]. These shapes are composed of geometric primitives such as line segments and circular arcs. The small red circles indicate the real desired segmentation points. The start point and the last point of the stroke were automatically treated as the true speed feature points and are not included in the evaluation. Shapes 1–7 were used to test the efficiency of our method for segmenting shapes composed of one or more primitive types. Shapes 8–9 were used to test the performance of our method when processing the same strokes drawn by the same person at different speeds.

4.3. Procedure. We recruited 8 participants (P1–P8) from the local university. They were all majored in mechanical engineering and had experiences in hand-drawn mechanical drawings; and two of them had experiences in drawing with a tablet. Before the experiment, we first introduced the usage of the FSR system and then gave them 10–15 minutes to practice drawing digital sketches on the system's interface through the pen display shown in Figure 4. The screen resolution of the digital pen display was set to 1024×768 pixels. During the experiment, we first asked 7 of the 8 participants (P1–P7) to draw shapes 1–7 shown in Figure 5. Each shape was drawn discontinuously five times, using the standard shape of about 3 cm displayed on the system's interface as a reference. We did not limit the position of the starting point and drawing direction of the stroke. The order in which each participant draws the shapes was counterbalanced using a Latin-Square design. The system only displayed the original strokes, not the segmentation results, to prevent them from adjusting their drawing strategies based on the feedback. We recorded 245 strokes from P1–P7 and discarded some strokes that are far away from the reference shapes. Finally, we got 220 valid strokes for evaluation. In addition, we asked the eighth participant (P8) to draw shapes 8–9 of Figure 5 at uniform and ununiform speeds, respectively. We took the first uniform speed stroke and the first ununiform speed stroke drawn by the participant as examples to analyze the segmentation results.

4.4. Measurements. We used processing time and accuracy to evaluate the effectiveness of the stroke segmentation algorithm. The accuracy was computed from the number of missing and extra segmentation points, as described in previous stroke segmentation research [4, 6, 7, 10]. The three metrics are precision, recall, and F-measure, which are defined by formulas (7)–(9). Precision, P, is the ratio of the accepted segmentation points that are true segmentation points. Recall, R, is the ratio of the true segmentation points that are correctly identified. F-measure, F, is the simple harmonic mean value of precision and recall.

$$P = \frac{n_{\text{accepted true segmentation points}}}{n_{\text{all accepted segmenting points}}}, \tag{7}$$

$$R = \frac{n_{\text{accepted true segmentation points}}}{n_{\text{all true segmentation points}}}, \tag{8}$$

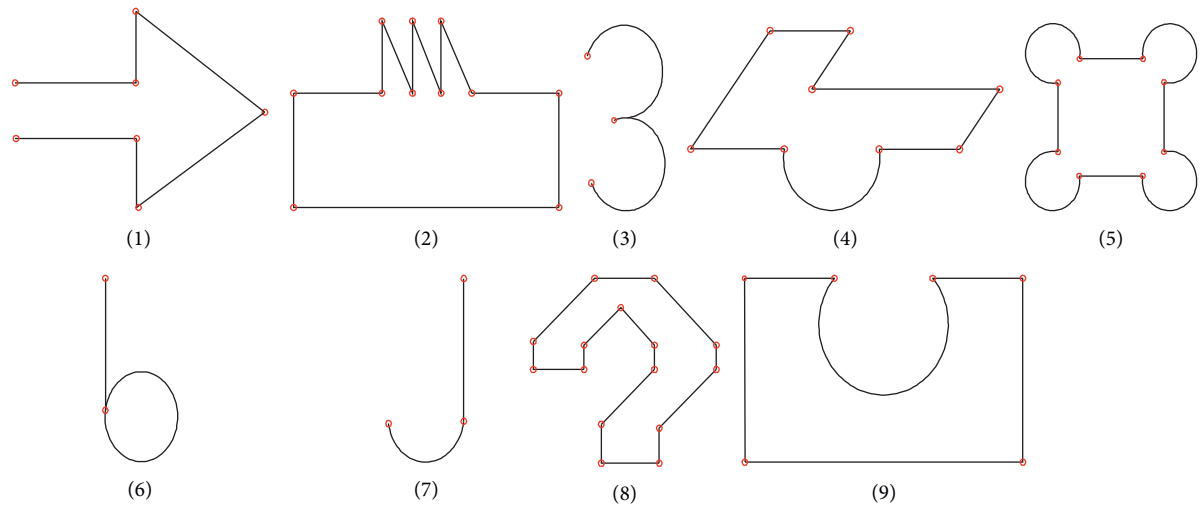


Figure 5: Nine shapes used in the user study.

$$F = \frac{2PR}{(P+R)}. \tag{9}$$

5. Results and Discussion

5.1. Results. Figures 6–12 show the results of using the proposed algorithm to extract the speed feature points of the stroke from the user study. We also implemented the speed feature point extraction method of Sezgin et al. [2] in the FSR system, using a threshold value of 90% of the average speed. The results of the method in [2] are also shown in Figures 6–10 to facilitate an intuitive comparison with the proposed method.

Figures 6–9 show the results of processing the strokes drawn by P1–P7 in the user study. Figure 6 shows the average processing time of the strokes of each shape, while Figures 7–9 show the average accuracy (precision, recall, and F-measure) of the strokes of each shape.

(i) *Processing Time*. It can be seen from Figure 6 that the proposed method is slower than the method of Sezgin et al. [2] when dealing with all the seven shapes. However, the processing time of the proposed method is all within 1 ms, so the users would not feel the delay. It can be concluded that the proposed method can quickly extract speed feature points from input digital freehand strokes.

(ii) *Accuracy*. It can be seen from Figure 7 that our method achieved much higher precision than the method of Sezgin et al. [2] for all the seven shapes. From Figure 8, however, we can see that the recall values of the proposed method are lower than those of the method in [2]. This means that the proposed method failed to detect some of the true segmentation points. Combining the precision and recall values, we can see from Figure 9 that the proposed method still achieved much higher F-measure values compared to the method in [2] for all the seven shapes. This means that, compared with the method in [2], the speed feature points computed by the proposed method are closer to the true segmentation points; and the accuracy of using the speed feature points as the true split points exceeded 70%.

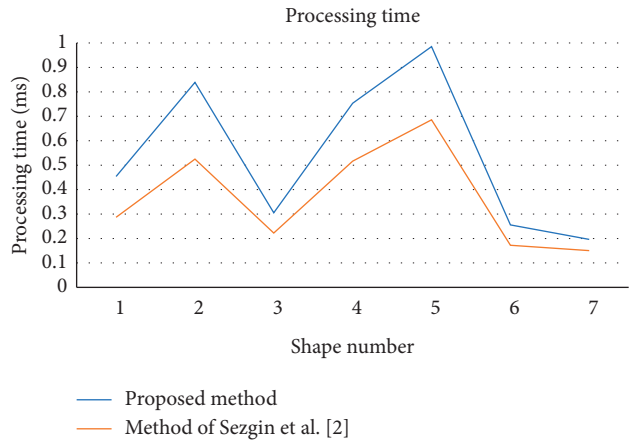


Figure 6: Processing time.

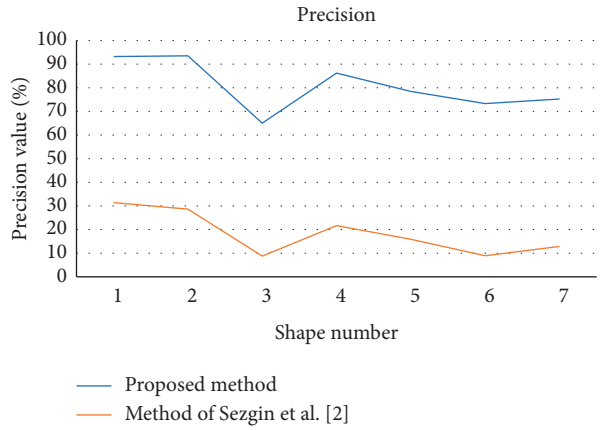


Figure 7: Precision accuracy for the user study.

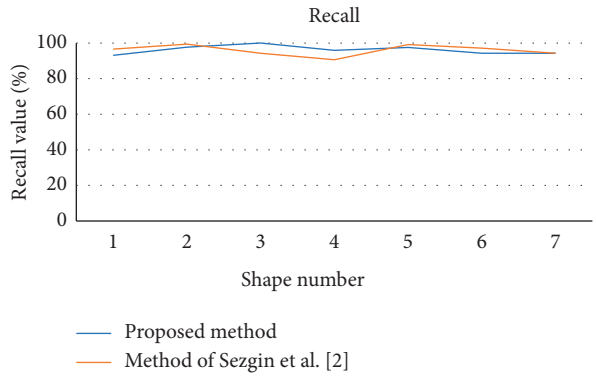


Figure 8: Recall accuracy for the user study.

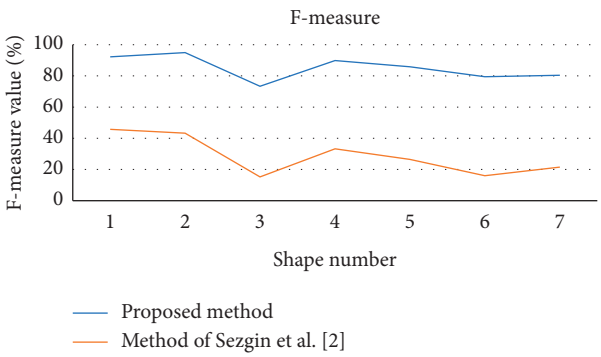


Figure 9: F-measure accuracy for the user study.

(iii) *Examples of Strokes Drawn at Different Speeds.* Figure 10 shows the example strokes of shapes 8 and 9 in Figure 5 drawn by P8 at uniform and nonuniform speeds. The nonuniform speed stroke is shown in the upper half, and the uniform speed stroke is shown in the lower half. The speed feature point extraction results of the proposed method and those of the method in [2] are shown in the left and right columns, respectively. Black points indicate the detected speed feature points. Figure 11 shows the speed waveforms of the example strokes of shape 9, where the horizontal axis represents the serial number of the sampling points, and the vertical axis represents their speed values. Figure 11(a) shows the revised speed waveform after the filter-sharpening process for one stroke drawn at uniform speeds. Figure 11(b) shows the original speed waveform for the other stroke drawn at uniform speeds. Some key parameters such as average speed and its upper deviation and lower deviation for speed points extraction of the proposed method are expressed in the form of isolines. More specific values such as the sampling points number, true segmentation points number, and the precision of speed feature point extraction are given in Table 1.

(iv) Through Figure 10, we can see that the proposed method achieved high segmentation accuracy for all the example strokes drawn at either nonuniform or uniform speeds. We can see from Table 1 that the precision values of the method in [2] for both shapes 8 and 9 are lower than those of the proposed method, particularly for the example strokes drawn at uniform speeds.

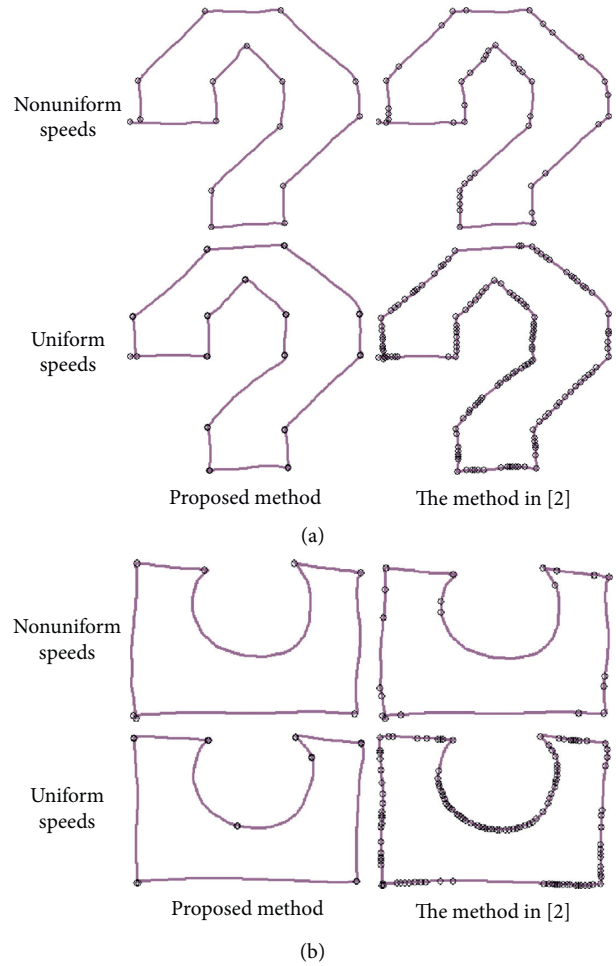


Figure 10: Results of the example strokes drawn at different speeds. (a) Example stroke of shape 8. (b) Example stroke of shape 9.

5.2. Discussion. Through the experiment, we can conclude that the proposed method is more efficient than the method of Sezgin et al. [2] to find the valid speed feature points. The method in [2] regarded too many sampling points as the speed feature points, which led to the low precision of stroke segmentation for all the shapes shown in Figure 5. By comparing the speed waveforms of the same stroke drawn at different speeds shown in Figure 11, we can see that the sampling points of the uniform speed stroke are more densely distributed, and the speed curve is smoother. Therefore, using a single speed threshold as in [2] may overgroup the speed data and result in too many candidate segmentation points. However, we effectively reduce the number of redundant points by first filtering and sharpening the speed data of uniform speed strokes and then extracting speed feature points by the proposed three-threshold

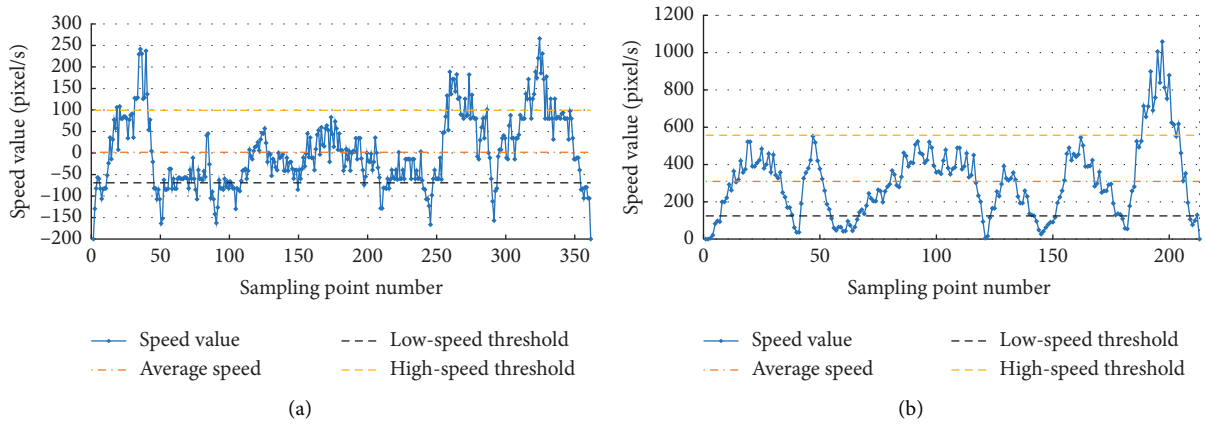


Figure 11: Speed graphs for the example strokes of shape 9 in Figure 5. (a) Uniform speeds. (b) Nonuniform speeds.

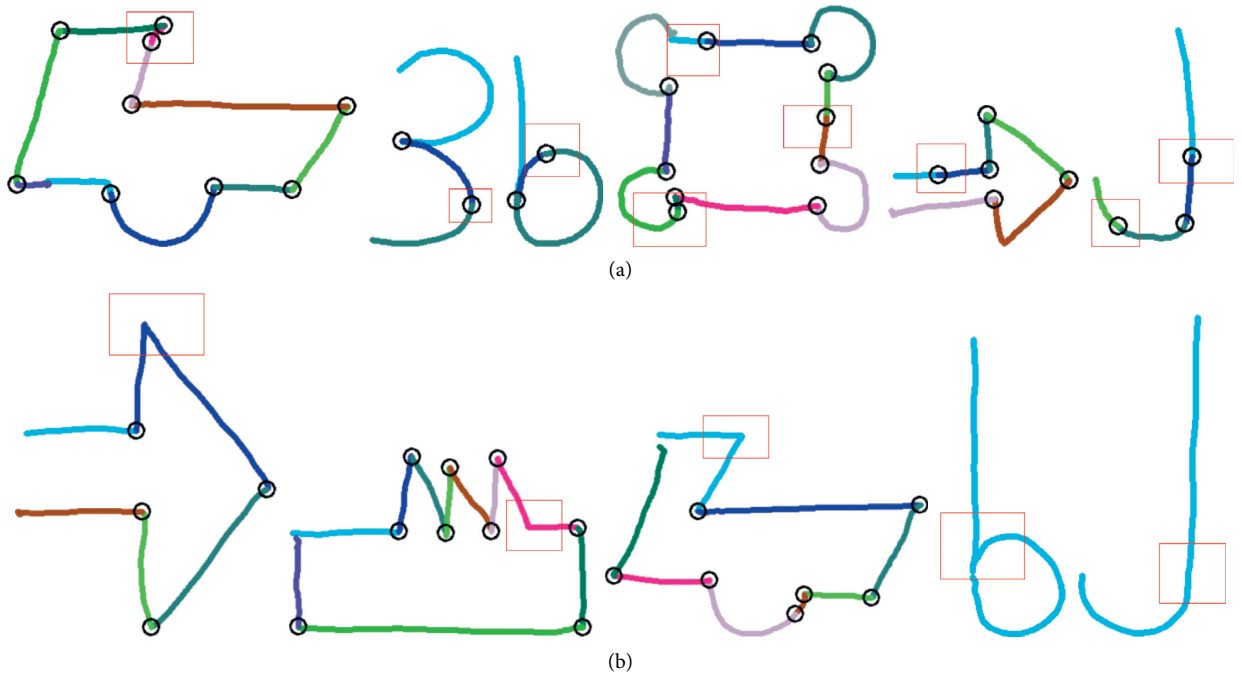


Figure 12: Examples of the wrong segmentation results produced by the proposed method. (a) Oversegmentation cases. (b) Undersegmentation cases.

Table 1: Specific results of the example strokes drawn at different speeds.

Shape	Speed feature	n_t	Algorithm	v_{avg}	v_L	v_H	n_a	n_{at}	Precision (%)
8	Uniform speeds	14	The method in [2]	208.833	187.950	—	14	43	32.558
			Proposed method	208.823	92.518	385.072	14	14	100
	Nonuniform speeds		The method in [2]	98.975	89.078	—	14	138	10.145
			Proposed method	−1.726	−76.951	79.358	14	14	100
9	Uniform speeds	5	The method in [2]	309.687	278.718	—	5	18	27.778
			Proposed method	309.393	124.520	556.437	5	5	100
	Nonuniform speeds		The method in [2]	91.188	82.069	—	5	124	4.032
			Proposed method	1.733	−69.218	99.361	5	7	71.429

v_{avg} = average speed; v_L = low-speed threshold; v_H = high-speed threshold; n_a = the number of all accepted true segmentation points; n_t = the number of true segmentation points; n_{at} = the number of accepted true segmentation points.

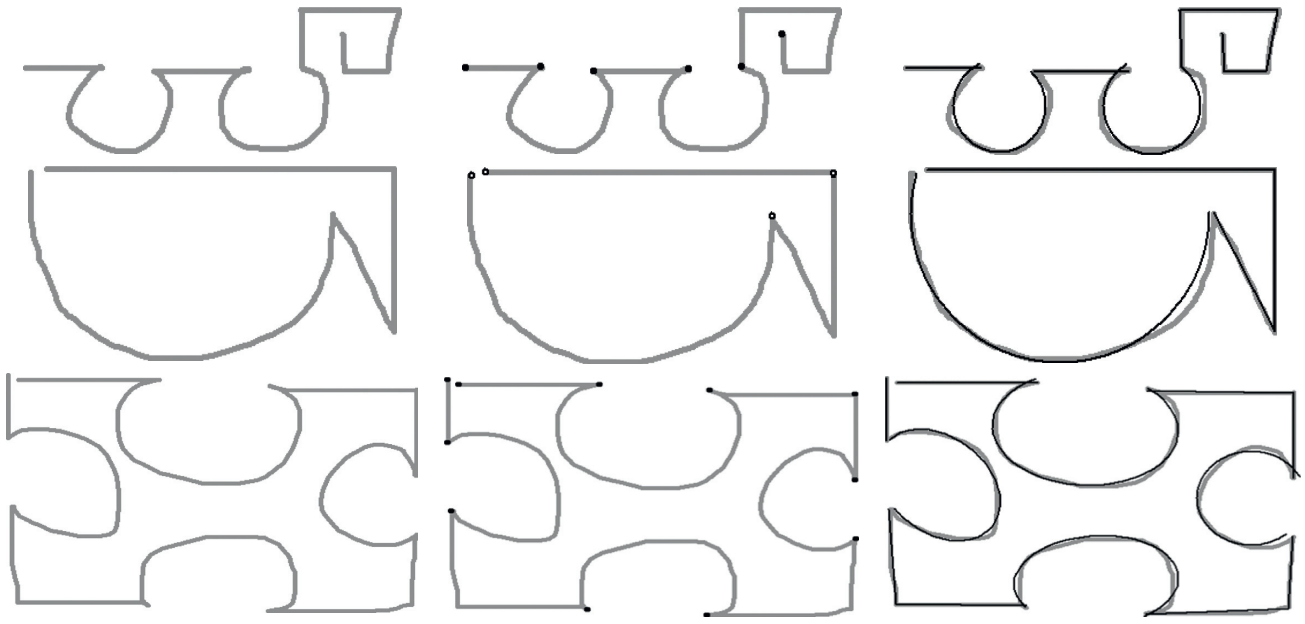


Figure 13: Recognition results of several input complex strokes of the FSR system. (a) Input stroke. (b) Segmentation results. (c) Primitive recognition.

segmentation technique. Figure 13 shows that the effective segmentation of strokes increases the FSR system's support for flexible input and can reduce the burden of recognizing complex strokes.

However, it was also found that only using the speed information to segment pen strokes can cause undesired results such as missing points and extra points. Some of the failed cases are shown in Figure 12, where the detected speed feature points are indicated as black circles and the problematic areas are framed by red rectangles. This may be due to the use of empirically determined parameter values. In future work, we would like to use machine learning techniques to optimize those parametric values to adapt to different users and devices. Other stroke information, such as geometric features, pen pressures, and primitive recognition results, should be combined with speed features to get more accurate segmentation points, which is a worthwhile direction for further research.

6. Conclusion

In this paper, we presented a method to extract speed feature points for segmenting digital freehand strokes into primitives. The method is characterized by its ability to divide strokes drawn at uniform speeds. Firstly, a sharpening filter is used to enhance the peak features of the speed data and reduce the sampling points number. Then, a three-threshold segmentation method is used to extract the final speed feature points. We integrated the method into a freehand sketch recognition (FSR) system to enhance its robustness of supporting multiprimitive strokes. We tested the effectiveness of the proposed method by a user study with eight participants. The results show that the method achieves higher segmentation efficiency in finding valid speed feature points than the existing method based on a single speed threshold. Future work is to combine other stroke information such as pen pressures and geometric features with speed features to improve the accuracy of segmenting strokes.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was partly supported by the National Key R&D Program of China (Grant no. 2019YFB1703800), Fundamental Research Funds for the Central Universities (Grant no. 3102020gxb003), Natural Science Basic Research Plan in Shaanxi Province of China (Grant no. 2016JM6054), and the Programme of Introducing Talents of Discipline to Universities (111 Project), China (Grant no. B13044).

References

[1] A. Bonnici, "Sketch-based interaction and modeling: where do we stand?" *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, vol. 33, no. 4, pp. 370–388, 2019.

[2] T. M. Sezgin, T. Stahovich, and R. Davis, "Sketch based interfaces: early processing for sketch understanding," in *Proceedings of the ACM's International Conference Proceedings Series*, p. 6, Heidelberg, Germany, November 2001.

[3] A. Wolin, B. Paulson, and T. Hammond, "Sort, merge, repeat: an algorithm for effectively finding corners in hand-sketched strokes," in *Proceedings of the 6th Eurographics Symposium on Sketch-Based Interfaces and Modeling*, pp. 93–99, New Orleans, LA, USA, August 2009.

[4] J. Herold and T. F. Stahovich, "SpeedSeg: a technique for segmenting pen strokes using pen speed," *Computers & Graphics*, vol. 35, no. 2, pp. 250–264, 2011.

[5] F. Albert, D. G. Fernández-Pacheco, and N. Aleixos, "New method to find corner and tangent vertices in sketches using parametric cubic curves approximation," *Pattern Recognition*, vol. 46, no. 5, pp. 1433–1448, 2013.

[6] J. Herold and T. F. Stahovich, "ClassySeg: a machine learning approach to automatic stroke segmentation," in *Proceedings of the Eighth Eurographics Symposium on Sketch-Based Interfaces and Modeling*, pp. 109–116, New York, NY, USA, August 2011.

[7] J. Herold and T. F. Stahovich, "A machine learning approach to automatic stroke segmentation," *Computers & Graphics*, vol. 38, pp. 357–364, 2014.

[8] A. Wolin, B. Eoff, and T. Hammond, "Shortstraw: A simple and effective corner finder for polylines," in *Proceedings of the Fifth Eurographics Conference on Sketch-Based Interfaces and Modeling*, pp. 33–40, Lisbon, Portugal, May 2008.

[9] S. Wang, G. Wang, and S. Qin, "Online sketch recognition using geometric-based classifiers," in *Proceedings of the ICAC 12—2012 18th International Conference on Automation and Computing (ICAC 2012)*, pp. 183–186, Loughborough, UK, September 2012.

[10] G. Feng and C. Viard-Gaudin, "Stroke fragmentation based on geometry features and HMM," 2008, https://arxiv.org/abs/0812.0874.

[11] S.-F. Qin, D. K. Wright, and I. N. Jordanov, "On-line segmentation of freehand sketches by knowledge-based nonlinear thresholding operations," *Pattern Recognition*, vol. 34, no. 10, pp. 1885–1893, 2001.

[12] D. H. Kim and M.-J. Kim, "A curvature estimation for pen input segmentation in sketch-based modeling," *Computer-Aided Design*, vol. 38, no. 3, pp. 238–248, 2006.

[13] Y. Xiong and J. J. LaViola, "A ShortStraw-based algorithm for corner finding in sketch-based interfaces," *Computers & Graphics*, vol. 34, no. 5, pp. 513–527, 2010.

[14] F. Albert and N. Aleixos, "Improvements to the TCVD method to segment hand-drawn sketches," *Pattern Recognition*, vol. 63, pp. 416–426, 2017.

[15] L. Chieppa, M. Fiorentino, A. E. Uva, and G. Monno, "Unified interactive wavelet approach for 2D sketch segmentation and editing," *International Journal of Shape Modeling*, vol. 16, pp. 39–56, 2010.

[16] R. S. Tumen and T. M. Sezgin, "DPFrag: trainable stroke fragmentation based on dynamic programming," *IEEE Computer Graphics and Applications*, vol. 33, no. 5, pp. 59–67, 2013.

[17] G. Costagliola, M. De Rosa, V. Fortino, and V. Fuccella, "RankFrag: A machine learning-based technique for finding corners in hand-drawn digital curves," in *Proceedings of the 21st International Conference on Distributed Multimedia Systems*, pp. 29–38, Vancouver, Canada, September 2015.

[18] L. Zeng, Z. K. Dong, and Y. F. Xu, "Robust corner and tangent point detection for strokes with deep learning approach," 2019, https://arxiv.org/abs/1903.00899.

[19] S. Wang, G. Wang, M. Gao, and S. Yu, "Using fuzzy hybrid features to classify strokes in interactive sketches," *Advances in Mechanical Engineering*, vol. 5, Article ID 259152, 2013.