

Northumbria Research Link

Citation: Wang, Shuxia, Wang, Shouxia, He, Weiping and Qin, Sheng-feng (2020) Tolerance Zone-Based Grouping Method for Online Multiple Overtracing Freehand Sketches. *Mathematical Problems in Engineering*, 2020. pp. 1-12. ISSN 1024-123X

Published by: Hindawi

URL: <http://dx.doi.org/10.1155/2020/7393846> <<http://dx.doi.org/10.1155/2020/7393846>>

This version was downloaded from Northumbria Research Link:
<http://nrl.northumbria.ac.uk/id/eprint/44043/>

Northumbria University has developed Northumbria Research Link (NRL) to enable users to access the University's research output. Copyright © and moral rights for items on NRL are retained by the individual author(s) and/or other copyright owners. Single copies of full items can be reproduced, displayed or performed, and given to third parties in any format or medium for personal research or study, educational, or not-for-profit purposes without prior permission or charge, provided the authors, title and full bibliographic details are given, as well as a hyperlink and/or URL to the original metadata page. The content must not be changed in any way. Full items must not be sold commercially in any format or medium without formal permission of the copyright holder. The full policy is available online: <http://nrl.northumbria.ac.uk/policies.html>

This document may differ from the final, published version of the research and has been made available online in accordance with publisher policies. To read and/or cite from the published version of the research, please visit the publisher's website (a subscription may be required.)



UniversityLibrary



Northumbria
University
NEWCASTLE

Research Article

Tolerance Zone-Based Grouping Method for Online Multiple Overtracing Freehand Sketches

Shuxia Wang ¹, Shouxia Wang, ¹ Weiping He, ¹ and Shengfeng Qin ²

¹School of Mechanical Engineering, Northwestern Polytechnical University, Xi'an, China

²School of Design, Northumbria University, Newcastle, London, UK

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

Received 12 November 2019; Accepted 27 February 2020; Published 14 April 2020

Academic Editor: Jerzy Baranowski

Copyright © 2020 Shuxia Wang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Multiple overtracing strokes are common drawing behaviors in freehand sketching; that is, additional strokes are often drawn repeatedly over the existing ones to add more details. This paper proposes a method based on stroke-tolerance zones to group multiple overtraced strokes which are drawn to express a 2D primitive, aiming to convert online freehand sketches into 2D line drawings, which is a base for further 3D reconstruction. Firstly, after the user inputs a new stroke, a tolerance zone around the stroke is constructed by reference to its polygonal approximation points obtained from the stroke preprocessing. Then, the input strokes are divided into stroke groups, each representing a primitive through the stroke grouping process based on the overtraced ratio of two strokes. At last, each stroke group is fitted into one or more 2D geometric primitives including line segments, polylines, ellipses, and arcs. The proposed method groups two strokes together based on their screen-space proximity directly instead of classifying and fitting them firstly, so that it can group strokes of arbitrary shapes. A sketch-recognition prototype system has been implemented to test the effectiveness of the proposed method. The results showed that the proposed method could support online multiple overtracing freehand sketching with no limitation on drawing sequence, but it only deals with strokes with relatively high overtraced ratio.

1. Introduction

Generally, concept designers use paper and pencil to share, communicate, and record new ideas. However, the paper becomes cluttered with all sorts of changes. Freehand sketching in a real-time 3D styling system presents a different way to create a 3D model from the traditional CAD system [1]. However, there is a wide gap between the designer's intentions and the abilities of computers to understand that [2]. To better explain the design concept and to integrate it more closely with the back-end engineering design, a large number of studies have been conducted in this field to make computers correctly catch the designer purposes from conceptual design sketches [3].

In the field of mechanical design, sketching is often used in its conceptual design stage before the depicted design is converted into a 3D model. Current high-end commercial CAD modeling systems, such as Maya, SolidWorks, and

CATIA, do help tremendously in the detailed design phase, but they do not robustly support the creation of 3D objects directly from 2D freehand sketches and hence are not well suitable for conceptual design. In recent years, there is considerable research on sketch-based modeling (SBM), the goal of which is twofold [4]. One is to support rapid geometric modeling either in 2D or 3D, and the other is to support freehand sketching activity with a more intuitive interface for the user. SBM systems obtain freehand sketches with pencil and paper or digital tools, which are referred to as offline or online freehand sketches, respectively [5]. Real pencil-and-paper is a vibrant medium for communication, but a model reconstruction from an offline scanning of a sketch has been a challenge, and improved information obtained from sketching is reduced to a batch of data points representing lines [5]. On the other hand, online-based sketching systems with figure drawing or other intuitive drawing devices can not only support the drawing activity

with the same or very similar user experience as drawing with paper and pencil, but also obtain the instant digital creation of design artifacts. Comparing with the image-based recognition of offline sketches, online sketches recognition benefits from more information such as drawing sequence, speed, and pressure to the application and constant feedback to the user [6]. The locations of vertices, corners, and edges of the object can be determined when drawing the strokes. Therefore, more research has focused on online 2D and 3D reconstruction technology towards AI-based design [7]. Sketch-based modeling techniques generally start with an axonometric drawing of a model, which is a kind of stereoscopic graph made by two-dimensional lines and is applied extensively in engineering because of its three-dimensional effect. Extraction clean vector drawings from rough and messy freehand sketches is a crucial step towards sketch-based modeling. Currently, in the field of reconstruction of 3D models from 2D sketches, sketch interpretation is usually based on primitives. Such methods addresses the uniqueness problem of the sketch interpretation, which involves sketch segmentation [6,8], stroke grouping [9,10], line fitting [11], endpoint fusion, and 2D regularity enhancement [12].

Overtracing is a common phenomenon or drawing behavior in freehand sketching; that is, additional strokes are often drawn repeatedly over the existing ones to add more information. It is a quick and effective way to correct a sketch visually and to add more details. Overtracing makes it harder to create line drawings from sketches towards geometrical reconstruction. For an online non-overtraced sketch, strokes are typically segmented into sets of smaller substrokes [13], and then each substroke is fitted with a geometric primitive; finally, the fitted lines are beautified to form a connected line drawing. However, for an offline overtraced sketch, the strokes that express a primitive are not inputted in sequence. Therefore, the strokes need to be automatically grouped by the sketch system to represent the related primitives. For giving the drawing freedom, an interactive sketch-based design system should allow designers to draw initial sketches with overtraced strokes. The behavior further increases the number of possible interpretations of a sketch, which in turn increases the risk of misrecognition. Most of the current online sketch systems do not support overtraced stroke primitives or add additional constraints on users. Supporting overtraced input is one of the challenges in online sketch recognition.

As one of the critical steps in both online sketch recognition and artistic sketch simplification, stroke grouping has attracted many researchers' attention over the past few years. Gestalt psychologists [14] provide the laws of perceptual organization to explain how people understand visual scenes. Gestalt laws consist of proximity, similarity, closure, continuation, and common fate. In a stroke-based interface, strokes are usually grouped in a bottom-up greedy way according to the geometric relationships between them, such as proximity, continuity, and parallelism. Sketches in different fields may contain different line types, for example, technical drawings usually composed of geometric primitives such as straight lines and circular arcs, while free-form

splines are more common in fashion design sketches and cartoon images. Stroke grouping method should be closely connected with specific-domain knowledge.

This paper proposes a tolerance zone-based method for grouping multiple overtraced strokes into 2D primitives to transform online freehand sketches into clean line drawings. Firstly, each stroke is represented as a polyline by a polygonal approximation approach, and then the tolerance zone around the stroke is constructed through its related polyline. Next, the overtraced ratio of two strokes is obtained by computing the percentage of sample points of the smaller stroke falling into the tolerance zone of the other stroke; then this overtraced ratio is used to decide whether the two strokes should be grouped together or not. In the last step, each group of strokes is fitted as one or more simple geometric primitives such as line segment, polyline, ellipse, ellipse arc, and so on. The proposed stroke grouping method can group strokes with arbitrary shapes and can overcome the limitation that only strokes of the same types can be grouped in some sketching systems which support overtracing inputs [9, 10]. A prototype system named FSR-MOS has been developed to check the validity and feasibility of the proposed method. The work presented here provides a foundation for further line drawing reconstruction. In FSR-MOS system, our previous fuzzy classification method [11] is used to recognize primitives as straight lines, polylines, ellipses, elliptical arcs, circles, circular arcs, parabolas, or hyperbolas. Among them, ellipses, elliptical arcs, circles, circular arcs, parabolas, and hyperbolas are collectively called conic curves in this paper.

The structure of this paper is as follows. Section 2, introduces the related work on multiple overtracing sketch recognition. Section 3 details the proposed stroke tolerance zone-based grouping method. Section 4 reports the implementation of the method and shows some examples before the conclusion is drawn in Section 5.

2. Related Work

Sketch-based interface is an up-and-coming technology that will release users from a maze of menus, toolbars, and many complicated commands [15]. Lot of work has investigated the use of sketch-based interfaces, but strict restrictions on the drawing manner do not allow users to sketch naturally. Most of the recognizers assume that any primitive shape will be drawn with a single stroke [16, 17]. However, there are many instances in which a user chooses to draw a primitive shape with more than one stroke [18]. Multistroke overtracing sketches should be allowed to support free-form sketching in many applications. Interpretation of such sketches requires grouping strokes that belong to a perceptual curve together. Although human vision can understand graphics from a large number of disconnected, overtraced, or disordered strokes intuitively, it is difficult for computers.

In recent years, some researches have done to allow overtracing or multistroke input to sketch-based user interfaces, strokes are grouped automatically and then they are converted into 2D line drawings. Some sketching systems

placed additional constraints on users that restricted their freedom of rapid design expression. Fonseca and Jorge [19] presented a method based on fuzzy logic for recognizing multistroke sketches. Their system processes one or more strokes, which are drawn within a specified time, and then they are classified into one of several predefined shape types. Calhoun et al. [20] utilized simple button clicks to designate when a shape is finished and ready to be recognized. Some sketch-based interfaces allow overtraced input and grouped strokes into geometric lines automatically to generate line drawings for 3D reconstruction. SMARTPAPER [21] simplified sketches composed exclusively of straight line segments by replacing strokes sharing similar slopes and proximate endpoints with an average straight line. Chansri and Koomsap [5] created line drawing from a paper-based overtraced freehand sketch by image processing techniques such as dilation and thinning, but it is still limited to the polyhedron sketch composed of straight line segments. Ku et al. [10] and Wang et al. [9] both have studied the grouping problem of online overtraced sketches. They both can process overtraced line segments and curves, while the later can also deal with overtraced polylines and fit them as a whole polyline to avoid losing connective information due to segmenting a polyline into line segments. Both the methods in [9, 10] first classified input strokes into 2D geometric primitives and then grouped strokes based on their fitting characteristics, so only strokes of the same geometric types are grouped. In their way, the case that a primitive is drawn with strokes of different geometric types cannot be solved, which is common in users' sketches. Moreover, the curve grouping method in [10] may result in over-group cases because curves are grouped if there is an overlap between their bounding boxes.

There exists considerable research dedicated to line drawing simplification. Much work has focused on the problem of simplifying artistic line drawings. While such methods also face the challenge of merging strokes to form long curves, the additional knowledge provided by the shape and orientation of the input strokes greatly facilitates proper handling of junctions [22]. Barla et al. [23] used the proximity of the strokes as the primary constraint to combine them for the simplification of line drawings. The method is based on a "E-line" that has a user-specified width covering the strokes to be grouped. However, the "E-line" cannot fold onto itself and hence is not applicable to the folding or self-intersecting curves. Pusch et al. [24] used hierarchical space partitioning to divide many strokes into locally orientable segments and then fitted a B-spline curve passing through the segments. The method was more suitable for nonintersecting curves. Shesh and Chen [25] merged strokes according to proximity, color, local gradient, and extent of overlap. The method works for both contour strokes and hatching strokes. But their concept of overlapping can be weak because the contour curves may not overlap so much. Orbay and Kara [26] proposed a method that simplifies freehand sketches into free-form curves through a trainable stroke clustering algorithm that learns the rules for stroke grouping from the users' sketches. The method can handle self-intersecting and bifurcating curves that are often difficult to distinguish using a purely local analysis. Chien et al. [27] proposed

an algorithm to simplify line drawing sketches. It firstly used a low-pass filter to assign a weight to every stroke. Then strokes are moved to the position of the higher weight. Finally, strokes are paired based on the endpoints' position and tangent. However, the method cannot simplify the strokes locally. Liu et al. [28] made the first attempt in incorporating the law of closure into the semantic analysis of the sketches to simplify sketch drawings. The method combined stroke grouping with region interpretation based on the notion of regions formed by strokes. However, the method did not explicitly handle closed strokes and only deals with contour lines. Ogawa et al. [29] proposed a method to convert a rough sketch into a line drawing using a machine learning approach such as in [26]. But it can be trained without the use of manual annotations. These line-drawing simplification methods cannot directly deal with long and tortuous strokes, because they were designed to calculate the geometric relationship between short smooth strokes. Moreover, the stroke segmentation step broke the original information of strokes such as closure and geometric types, and stroke clustering process may not restore the original data. Therefore, they are not entirely applicable to the multistroke primitive grouping in mechanical drawings where contour lines are made up of normal curves.

Several approaches have been proposed to extract simplified drawings from scanning pencil sketches [5, 22, 30–34]. Such approaches are built directly upon overtraced strokes for line drawing creation because strokes are blended "for free" as the user draws [19]. Kara and Stahovich [35] presented a sketch recognizer which allows overtracing symbol input, but it relies entirely on the symbol libraries for the recognition where the sketch is examined in pixel, and hence no work has been done on individual stroke interpretation. Sezgin and Davis [36] applied moving least squares to move a circular window along overtraced strokes of non-self-intersecting polylines to obtain several small segments before linking to form a single line. Chansri and Koomsap [5, 37] proposed a method to extract single-line drawings from binary images, for their 3D reconstruction system. The method starts from using image processing techniques to cluster strokes to form a thick-line sketch, then contour expanding and shrinking concept is applied for identifying segments from every pair of extracted contours that are said to be neighbors. A single-line drawing consisted of line segments and polylines is created after all junctions were identified. Bonnici et al. [31, 34] described approaches based on Gabor and Kalman filtering to convert rough strokes into vectorized representation. SimoSerra et al. [30, 32] proposed approaches based on Convolutional Neural Networks to directly simplify rough raster sketches. They are able to tackle multiple lines that have to be collapsed into a single line by using their dataset for large-scale learning of sketch simplification. Favreau et al. [22] proposed the first vectorization algorithm that explicitly balances fidelity to the input bitmap with simplicity output, as measured by the number of curves and their degree. Chen et al. [33] presented an improved topology extraction approach for vectorization of crude line drawings. It was proved to be efficient and robust but lacked high-level understanding of the line drawing.

Our tolerance-based stroke grouping method overcomes the limitations of the algorithms [9, 10] that only grouped the strokes of the same types. It groups the original strokes directly without fitting them first and can group the different types of strokes together. Our approach allows users to draw a primitive with any strokes without considering their shapes, thus giving users more freedom and making the sketch system more robust. Additionally, the proposed approach can group overtracing self-intersecting strokes; overtracing strokes of self-intersecting polylines that have free shapes can be fitted as a whole polyline.

3. Tolerance Zone-Based Grouping Method

The input of our online multistroke sketch recognition system is a set of digital strokes drawn by the user via a tablet stylus or a mouse. A stroke is a time-ordered sequence of 2D points that are sampled along the trajectory of the stylus. The flow chart of our system is presented in Figure 1. The proposed approach that converts an online multiple overtraced sketch into a line drawing consists of four main steps: (1) preprocess an input stroke as a polyline by polygonal approximation; (2) build a tolerance zone around the input stroke, which is then used to judge the overlapping ratio of two strokes; (3) after the completion of the whole sketch, conduct the grouping process to cluster multiple overtraced strokes into groups together; (4) fit geometric primitives to the stroke groups obtained from the grouping step in sequence. The details of all steps are discussed in the following subsections. The explanations of symbols used to introduce the approach are shown in Table 1.

3.1. Preprocessing an Input Stroke. In our sketch system, a stroke is defined as a sequence of mouse positions which is sampled along the trajectory of the stroke. The mouse positions are measured in the screen coordinate system with the upper-left corner as origin, one unit being a screen pixel with system time. The split and merge algorithm of polygonal approximation [38] is used to represent a stroke as a polyline. The algorithm depends on a given threshold which is expressed by a measure function as described in [39].

Assume a stroke S is represented by a linked list of sampling points $\{p_i = (x_i, y_i, t_i); 0 \leq i \leq n\}$, where p_i is the coordinate and time of the i th point in the sampling point list. Through stroke preprocessing, a set of polygonal approximation points $\{z_i; 0 \leq i \leq u < n\}$ are obtained, where $\{z_i\} \in \{p_i\}$. The polygonal approximation points are linked orderly to create a polyline to represent the stroke, shown in red line in Figure 2, where the polygonal approximation points are labeled in small blue circles.

3.2. Building a Tolerance Zone Around a Stroke. The tolerance zone of a stroke is described as a region around the stroke and is formed by several pairs of equidistant curves, as shown in Figure 3. It is obtained by the polygonal approximate points of the stroke $\{z_i; 0 \leq i \leq u\}$ as follows:

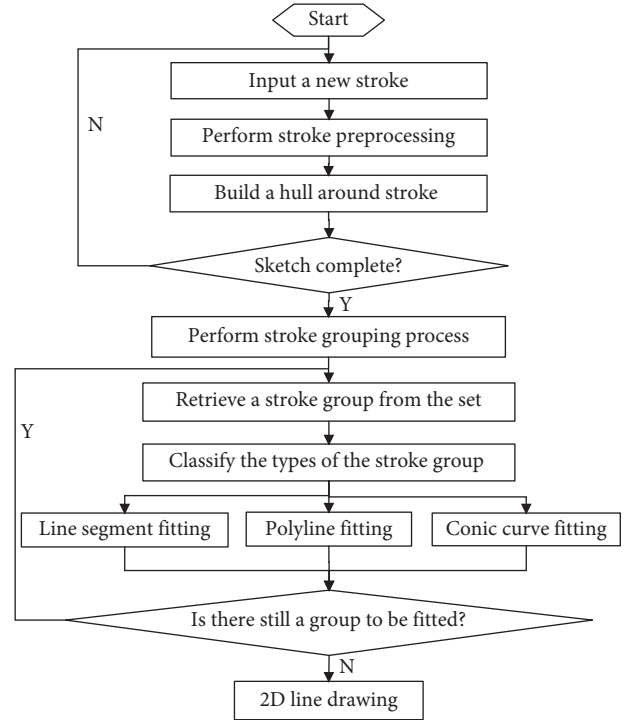


FIGURE 1: Flowchart of line drawing creation from multiple overtracing sketches in FSR-MOS system.

Step 1: Compute the bounding box of the stroke, which reflects the scale of the stroke. Two steps compute it. First, Graham's convex hull algorithm [40] is used to find the convex hull of the sampling points of the stroke; then the bounding box of the convex hull is obtained by the extremal method.

Step 2: Create a rectangle R_i of width w between every two adjacent polygon approximate points z_i, z_{i+1} in sequence, where z_i and z_{i+1} are located in one of the center lines of R_i , as shown in Figure 4(a). Thus, a set of rectangles along the stroke $\{R_i; 0 \leq i < u\}$ are obtained.

Step 3: Create a circle C_i of radius $w/2$ centered on each polygon approximate point z_i in turn, as shown in Figure 4(b). Thus, a set of circles along the stroke $\{C_i; 0 \leq i \leq u\}$ are obtained.

Step 4: The tolerance zone of a stroke built by the above steps is represented as a region formed by several rectangles $\{R_i\}$ and circles $\{C_i\}$ which are overlapped with each other. It can be further simplified by computing the union of $\{C_i\}$ and $\{R_i\}$, as shown in Figure 4(c). However, such simplification is useless for the next grouping process. The width of the rectangle R_i is considered to be the width of the tolerance zone of the stroke. Instead of using absolute width of the stroke tolerance zone w , we make the value adaptive to the scale of the stroke. A series of tests were conducted to determine the adaptive function as in (1) based on subjective decisions on whether two strokes should be grouped together.

TABLE 1: Explanation of symbols.

| Symbol | Description |
|----------------|---|
| P | Sampling points captured along the trajectory of the mouse, with drawing positions and drawing times, $p = (x, y, t)$ |
| S | Stroke formed by a set of sampling point, $S = \{p_i\}$ |
| Z | Polygonal approximation points of a stroke |
| W | Width of the tolerance zone of a stroke |
| R | Rectangle created between two adjacent approximation points |
| C | Circle centered on a polygonal approximation point |
| C | Perimeter of the bounding box of a stroke |
| B | Pen width of a stroke |
| G | Stroke group represented by a linked list of strokes, $g = \{S_i\}$ |
| Sc | Linked list of input strokes of the whole sketch, $Sc = \{S_i\}$ |
| r_{S_1, S_2} | Overtrace ratio of two strokes S_1 and S_2 |
| δ | Overtrace ratio threshold to decide whether two strokes should be grouped together |

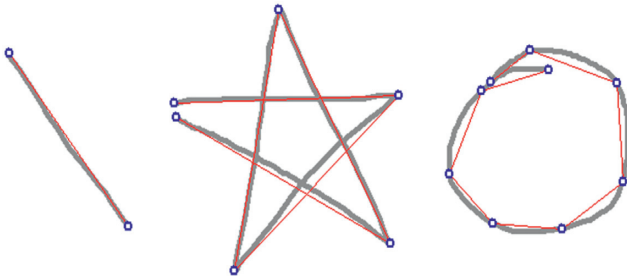


FIGURE 2: The polygonal approximation points of the stroke.

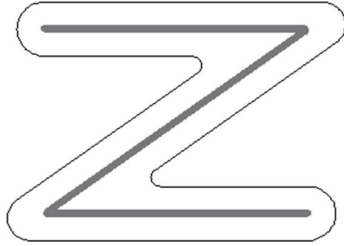


FIGURE 3: Stroke tolerance zone.

$$w = \frac{2}{3\sqrt{c}} + 2b, \quad (1)$$

where c and b are the perimeter of bounding box and the width of the stroke respectively, b is set to 5 pixels in this paper.

3.3. Grouping Multiple Overtraced Strokes. Given a sketch $Sc = \{S_i; 0 \leq i < u\}$, which is represented by a linked list of input strokes, where u is the stroke number of Sc . The proposed stroke grouping algorithm is used to cluster overtraced strokes into groups together, finally the input strokes are divided into several stroke groups. During the stroke grouping process, our system iteratively chooses two strokes to be grouped together if their overtraced ratio is greater than a given threshold value δ (this paper takes 0.5).

3.3.1. Overtraced Ratio of Two Strokes. Given two strokes S_1 and S_2 , the overtraced ratio r_{S_1, S_2} of them is defined as the percentage of the sample points of the smaller stroke (compared by perimeter of their bounding box), which fall into the other's tolerance zone, as shown in Figure 5. The overtraced ratio computing algorithm is as follows:

Step 1: Compare the perimeters of bounding boxes of the two strokes. The stroke with shorter perimeter is denoted by S_{\min} , and the other is denoted by S_{\max} . The number of the sampling points of S_{\min} is denoted by n_{\min} .

Step 2: Count the number n_{fall} of the sampling points of S_{\min} which fall into the tolerance zone of S_{\max} . This requires judging whether a point is inside the tolerance zone of a stroke, since a series of rectangles and circles form the tolerance zone of the stroke, it only needs to judge whether the point is inside one of these rectangles or circles.

Step 3: The overtraced ratio r_{S_1, S_2} of S_1 and S_2 is the ratio of n_{fall} to n_{\min} ; that is $r_{S_1, S_2} = n_{\text{fall}}/n_{\min}$.

3.3.2. Stroke Grouping Algorithm. Grouping strokes is an iterative process that creates new empty stroke groups and removes the strokes from the linked list of input strokes Sc to them. During the process, two strokes of higher overtracing ratio are selected to be merged in priority. A stroke group is also a linked list of strokes, which is represented as $g = \{S_i; 0 \leq i < t\}$, where t is the number of the strokes in it. Each iteration consists of four main steps: (1) create a new empty stroke group g ; (2) move the first stroke from the input stroke list Sc to the new group g ; (3) compare a stroke in the group g with the remaining strokes of the input stroke list Sc in turn, if the overtraced ratio of two strokes is greater than a threshold δ (it is set to 0.5 in this paper), then the stroke of Sc is moved to the group g ; (4) repeat Step 3 until there is no stroke in the group g is overtraced with the remaining strokes of Sc . After the grouping process, the input strokes are divided into several groups that include one or multiple strokes representing a primitive alone or collectively. The pseudocode of the proposed stroke grouping algorithm is shown as follows.

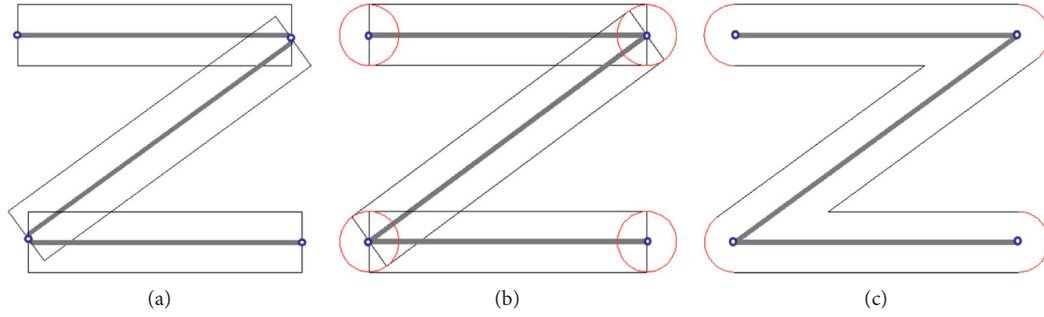


FIGURE 4: Process of building the tolerance zone of a stroke: (a) rectangles created by polygon approximate points; (b) circles by polygon approximate points; (c) the combined region.

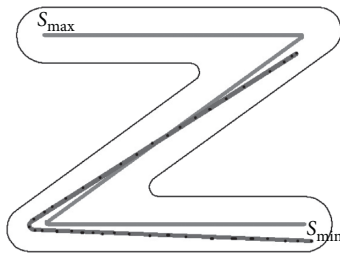


FIGURE 5: Two overtraced strokes.

Input: a linked list of input strokes $Sc = \{S_i; 0 \leq i < u\}$, where u is the stroke number of Sc .

Output: a linked list of stroke groups g_j , where $g_j = \{S_k; 0 \leq k < t_j \leq u\}$, t_j is the stroke number of g_j .

Step 1: Initialize parameters, set $j = t_j = 0, x = y = 1$.

Step 2: Create a new stroke list g_j .

Step 3: Remove the first stroke S_0 from Sc and insert it at the end of g_j , then $t_j = t_j + 1, u = u - 1$; if $u = 0$, go to Step 7.

Step 4: Let $x = y, y = t_j$. Assume that g_p is the list of strokes that were added into g_j in the previous step and are the $(x - 1)$ th to the y th strokes of g_j ; that is, $g_p = \{S_k; x - 1 \leq k < y\}$. Compare the strokes of g_p with the remaining strokes of the input stroke list Sc in turn; if the overtraced ratio of two strokes that belong to g_p and Sc , respectively, is greater than a threshold δ (it is set to 0.5), then move the stroke S_k in Sc to the group g_j .

Step 5: If $x = t_j$ or $u = 0$, then go to Step 6; otherwise go to Step 4.

Step 6: If $u > 0$, then $j = j + 1$, go to Step 2.

Step 7: Over.

3.3.3. Fitting Group of Strokes. The stroke group fitting algorithm is used to fit geometric primitives to stroke groups. A stroke group from the grouping process may contain one or more geometric primitives. Before the fitting process, a fuzzy classification method in [11] is used to recognize each stroke of the group as a line segment, a polyline or a conic curve.

When a group is merely composed of line segment strokes, it would be fitted into a line segment if the whole group is recognized as a line segment by comparing the length and width of its bounding box [11]. A line segment is obtained from connecting the two furthest polygonal approximate points of the group. Otherwise, we use an interpolation method to iteratively simplify pairs of strokes into one longer stroke, as shown in Figure 6.

When a group is mixed with polylines and line segments, not only the connection information of the vertexes but also the fork structure formed by two partly overtraced strokes should be taken into consideration. Taking Figure 7 as an example, such stroke group is simplified as follows. Firstly, each polyline is divided into several substrokes at its approximate polygonal points. Then, the above stroke grouping method is used to group these substrokes and original line segments into groups. Next, each stroke group is fitted into a line segment according to the two furthest polygonal approximate points. Thirdly, the two most adjacent line segments is linked by their intersections in order, and the minimum distance of their endpoints must be smaller than a threshold (this paper takes 20). Finally, repeat the third step until there is no line segments can be linked with others.

When a group is made up merely of conic curves, it is fitted into a conic curve by quadratic curve fitting method, where the fitted data comes from all approximate polygonal points of all strokes of the stroke group. Then the closure detection and endpoint determination process presented in [9] was performed to determine the endpoints of the fitting line.

When a group consists of strokes of different types, the fitting result is determined by the type of the longer strokes of the group, or a human-computer interactive dialog if desired. Firstly, find the stroke of the largest perimeter of bounding box c_{max} . Then, compare the perimeter of the bounding box of each stroke with c_{max} and record the stroke as a dominant stroke if the difference was smaller than $c_{max}/4$. If the dominant strokes are all conic curves, as shown in Figure 8(b), then we fit the stroke group into a conic curve. Otherwise, if the dominant strokes are polylines or include both conic curves and polylines, as shown in Figures 8(b) and 8(c) separately, the system highlights the strokes of the group and pops up a dialog box for users to select the fitting

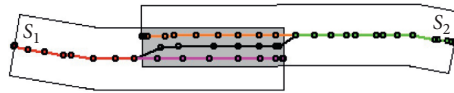


FIGURE 6: The simplification of two overtracing strokes.

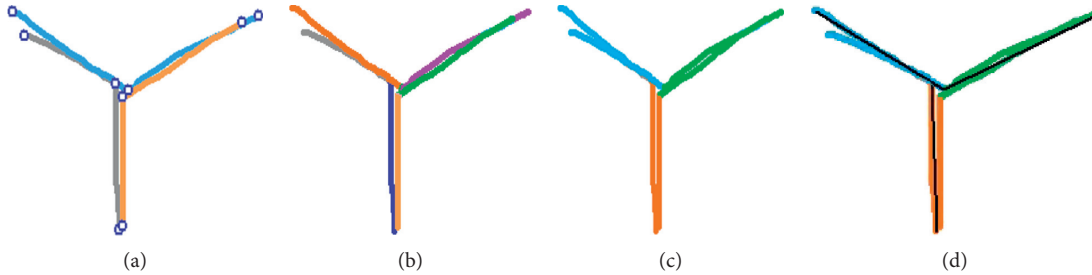


FIGURE 7: Process of fitting a polyline group: (a) input sketch with polygonal approximate points; (b) segmenting strokes into substrokes; (c) grouping substrokes into groups and fitting a line segment to each substroke group; (e) connecting adjacent line segments in sequence.

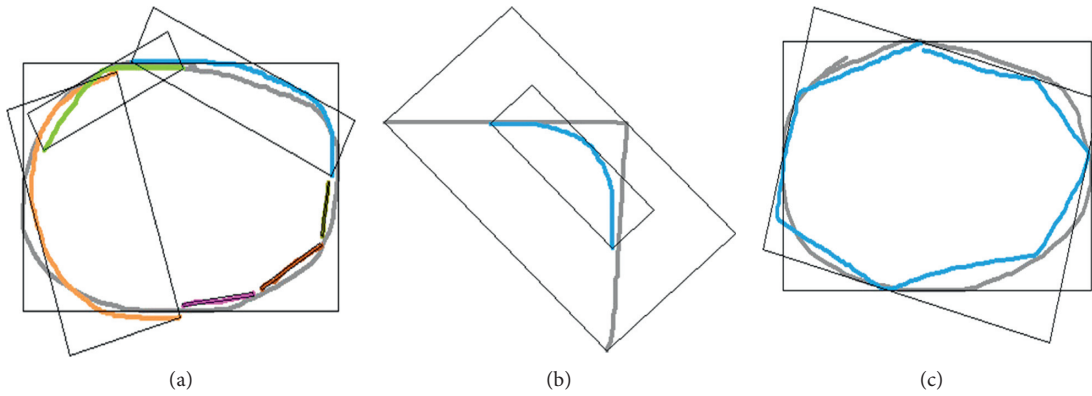


FIGURE 8: The bounding box of strokes: (a) the dominant strokes are conic curves; (b) the dominant stroke is a polyline; (c) the dominant strokes include both conic curve and polyline.

results: (1) fit the stroke group as a conic curve; (2) filter all conic curves of the group, and fit the remainder polylines or line segments as a whole polyline; (3) fit all conic curves of the stroke group as a conic curve, and fit the remainder polylines or line segments as a polyline.

4. Implementation and Examples

A freehand sketch-recognition system FSR-MOS for multiple overtraced sketches has been implemented on Win7 by using Visual C++. The input device could be a mouse or a tablet stylus. In order to test the grouped strokes and their fitting results of the proposed method, a number of test cases were carried out and some of them are shown in Figures 9–11, where the strokes are shown in different colors, and the fitted line segments, polylines, and conic curves are shown in blue, green, and red separately. We also compare our method to the existing grouping methods [9, 10], the results of the comparison are shown in Figures 10, 12, and 13. All our experiments are conducted on a 3.5 GHz computer with 10 GB of RAM.

Some fitting examples of various kinds of stroke groups that consist of multiple overtraced strokes are given in

Figure 9 to test the performance of the proposed stroke group fitting method. Figure 9(a) shows 5 stroke groups. Figure 9(b) shows the primitive recognition and fitting results of every stroke. Figure 9(c) shows the fitting results of stroke groups. Figure 9(d) shows the comparison of the original stroke groups and their fitting results, which are respectively shown in gray and black. Figure 9(e) gives the composition of the 6 stroke groups, where n_L , n_P , and n_C , respectively, denote the number of line segments, polylines, and conic curves.

Each of the five examples shown in Figure 9(a) represents one type of stroke groups: (1) a multi-overtraced line segment merely composed of line segments; (2) a multi-overtraced ellipse merely composed of line segments; (3) a multi-overtraced ellipse merely composed of line segments; (3) a group composed of polylines and line segments; (4) a multi-overtraced circle merely composed of conic curves; (5) a group composed of polylines, line segments, and conic curves and is fitted into a conic curve directly by the system; (6) a group consisting of three type of strokes and is fitted into different results from artificial selection. From Figure 9, it can be concluded that the proposed method can fit multiple overtraced strokes of various cases into three types

| Sequence | 1 | 2 | 3 | 4 | 5 | 6 |
|---------------------------|-----------|------------|--------------------|-----------|-----------------------------|--------------------|
| (a) Stroke group | | | | | | |
| (b) Single stroke fitting | | | | | | |
| (c) Group fitted result | | | | | | |
| (d) Comparison | | | | | | |
| (e) Composition | $n_L = 3$ | $n_L = 21$ | $n_p = 2, n_L = 1$ | $n_C = 3$ | $n_L = 1, n_C = 3, n_p = 2$ | $n_C = 1, n_p = 2$ |

FIGURE 9: Examples of fitting overtraced strokes.

of basic primitives including line segments, conic curves, or self-intersecting polylines. From the fitting results of sketches 2, 5, and 6, it can be seen that the proposed method can not only fit primitives into a group composed of different types of strokes but also group one type of strokes but fit them into another type of geometry.

Figure 10 shows some examples of creating line drawings from multiple overtracing sketches. Figure 10(a) shows 6 sketches with their stroke numbers ns . Sketch 1 is composed merely of line segments; sketch 2 is mixed up with line segments and polylines. Besides overtracing line segments and polylines, sketches 3–6 also include overtracing conic curves. Figure 10(b) shows the primitive recognition and fitting results of each stroke. Figure 10(c) shows the grouped results and the numbers of stroke groups ng , and different groups are shown in different colors. Figure 10(d) shows the fitting results of each stroke group. To compare the effectiveness of the proposed approach with the methods [9,10], the 6 sketches are also processed using the two methods, and the results are shown in Figures 10(e) and 10(f).

As shown in Figure 10(c), the overtraced strokes of the 6 sketches are grouped together properly. As shown in Figure 10(d), the resulted stroke groups are also fitted into right primitives. It can be concluded that the proposed method can deal with sketches containing multiple overtracing line segments, polylines, and conic curves, and convert the overtracing sketches into 2D line drawing effectively. In sketches 1–3, all the overtraced strokes are of the same types: line segments are only overtraced with line segments, conic curves are only overtraced with conic curves, and polylines are overtraced with polylines or line segments. In sketches 4–6, some overtraced strokes are of the different types. From sketches 1–4, all overtraced strokes are automatically grouped. From sketches 4–6, when fitting stroke groups that contain conic curves and polylines, the system pops up a dialog to let users choose the fitting results

manually. It can be concluded that the proposed method can not only group overtraced strokes of the same types, but also group strokes of different types together, and fit the overtraced strokes with correct primitives.

From Figures 10(e) and 10(f), the results of [10] composed of line segments and conic curves, while the results of [9] also contain polylines. Comparing the results of the proposed method with the methods [9, 10], it can be seen that all the three methods yield the same results for sketch 1, and the proposed method yields the same results as the method [9] in sketches 2–3. But our results are different with the results of [9, 10] in sketches 3–6, the differences are mainly manifested in overtraced strokes of different types: instead of handling the overtraced strokes as a whole, the methods [9, 10] divided them into groups that composed of the same types of strokes. This is because the two methods both classified and fitted each stroke first, then grouped fitted lines by types, so only the same types of strokes are grouped together. Such approaches cannot resolve the cases that a geometric primitive is drawn with strokes of different types, which are common in users' sketches. However, the proposed method can give correct results; it allows users draw a primitive with any strokes without considering their shapes. So, it gives the users more freedom to complete a sketch and makes the system more robust.

Besides, for the conic curves grouping, in theory, the method [10] may result in over-group cases shown in Figure 12, because it grouped curves when there is an overlap between the bounding boxes of the two curves. But the proposed method can group such cases correctly, because it clusters two strokes based on the relationship between the tolerance zone of one stroke and the sampling points of the other stroke, which are both generated along the stroke trajectory. And for this reason, in the proposed method, although each stroke is classified into line segment, polyline, or conic curve before the grouping process, it still can group

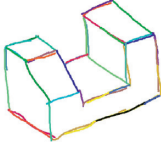

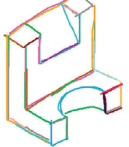
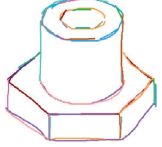
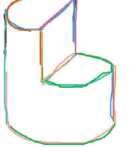
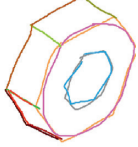
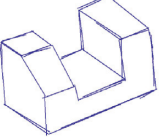
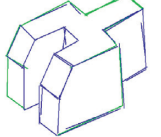
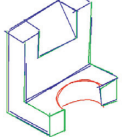
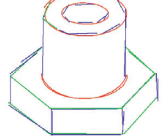
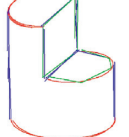
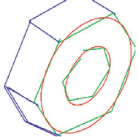
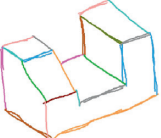
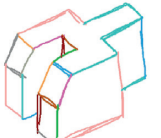
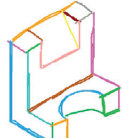
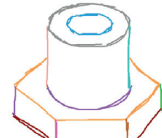


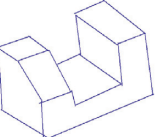
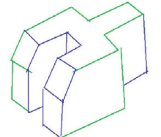
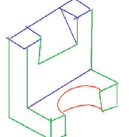
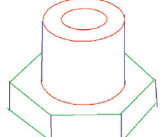
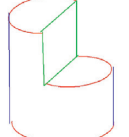
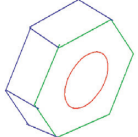
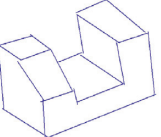
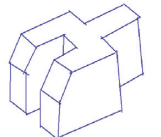
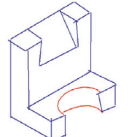
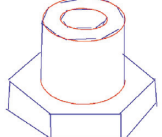
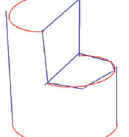
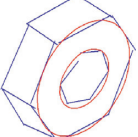
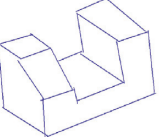
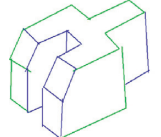
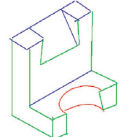
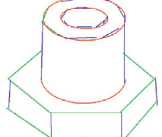
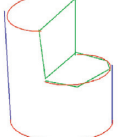
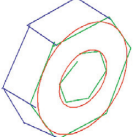
| Sequence | 1 | 2 | 3 | 4 | 5 | 6 |
|--------------------------------------|---|---|---|--|---|---|
| (a) Input sketch |  $n_S = 42$ |  $n_S = 42$ |  $n_S = 41$ |  $n_S = 37$ |  $n_S = 24$ |  $n_S = 14$ |
| (b) Single-stroke fitting |  |  |  |  |  |  |
| (c) Stroke groups ($\delta = 0.5$) |  $n_g = 21$ |  $n_g = 21$ |  $n_g = 20$ |  $n_g = 11$ |  $n_g = 7$ |  $n_g = 9$ |
| (d) Fitted stroke groups |  |  |  |  |  |  |
| (e) Day [2] |  |  |  |  |  |  |
| (f) Wang [1] |  |  |  |  |  |  |

FIGURE 10: Examples of grouping and fitting overtraced sketches in FSR_MOS and comparison with methods [9, 10].

strokes of arbitrary shapes. As shown in Figure 11, the input sketch is formed of 30 free-form strokes. The tolerance zones of the strokes are shown in Figure 11(b). Through the grouping process, the input strokes are divided into eight groups, as shown in Figure 11(c). However, the fitting results of groups consisting of strokes of arbitrary shapes are not given because it is beyond the scope of this paper.

The overtrace ratio threshold δ of the proposed grouping algorithm is used globally. It has great impact on the effect of grouped results, as shown in Figure 14. In Figure 14(b), the sketch is over-grouped when δ is set to 0.35, this is mainly because that a short stroke may be grouped with another stroke with a wide tolerance zone, although it represents a useful edge. In Figure 14(c), the overtraced strokes are clustered into right groups when δ is set to 0.5. In Figures 14(d) and 14(e), the sketch is undergrouped when δ

is set to 0.7 and 0.85. In this paper, the overtrace ratio threshold δ is taken as 0.5. However, the correct setting of δ which is generalizable to the variation in drawings is not constant. Figure 13 shows the grouping and fitting results of a sketch composed of overtracing line segments using the method [9] and the proposed method. It can be seen that some overtraced strokes are not grouped together by the proposed method, but are grouped correctly by the method [9]. This is because that the method [9] grouped line segments based on endpoints locations and slopes, so strokes which are not overtraced that much can be grouped together. Besides, the method [9] used time stamps to group short strokes that are close in drawing time and distance to dashed lines. However, the line drawing that is undergrouped generated from this paper can be further extended to the method [9] to group strokes of the same types.

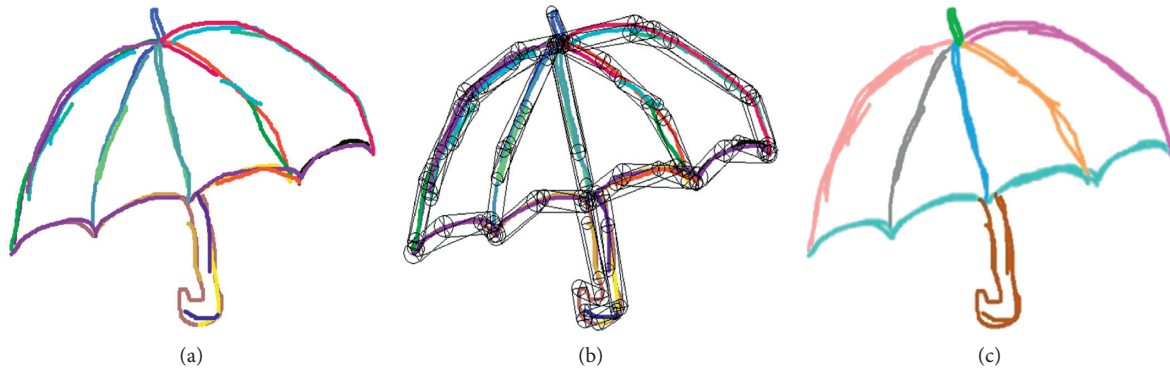


FIGURE 11: Grouping of free-form strokes: (a) input sketch; (b) the tolerance zone of the strokes; (c) grouped result.

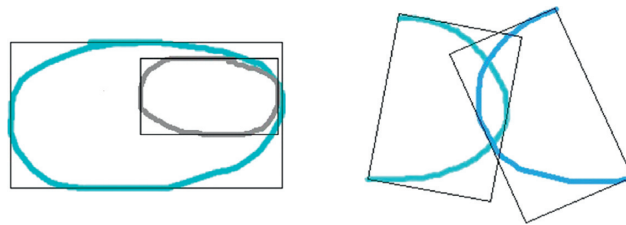


FIGURE 12: The over-group cases in [10].

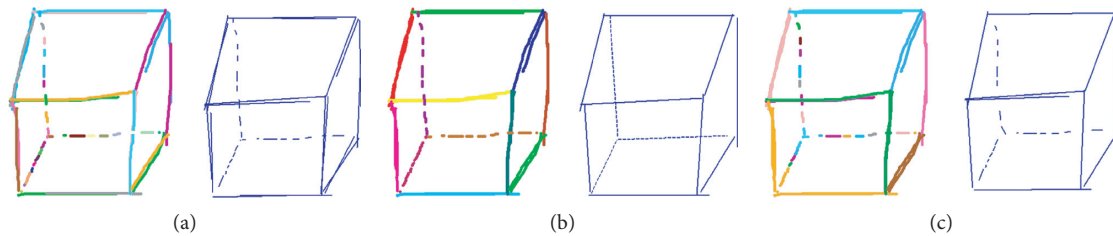


FIGURE 13: Comparison with the method [9]: (a) input sketch; (b) and (c) grouping and fitting result of the method [9] and the proposed method.

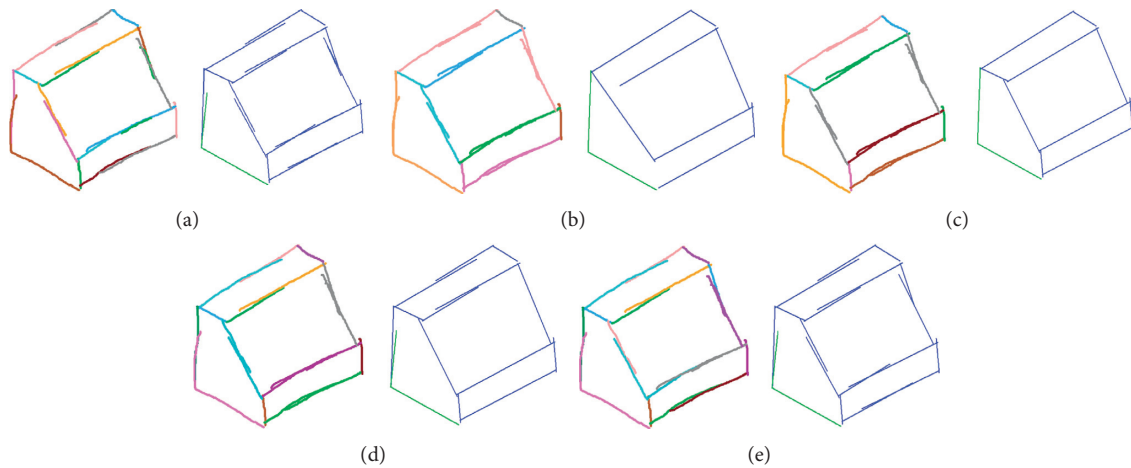


FIGURE 14: Grouping examples with different overtrace ratio threshold δ : (a) input sketch and single stroke fitting results; (b) $\delta = 0.35$; (c) $\delta = 0.5$; (d) $\delta = 0.7$; (e) $\delta = 0.85$.

5. Conclusion

This paper proposes a tolerance zone-based method for transforming online multiple overtraced sketches into 2D line drawings. It consists of three parts including stroke preprocessing, stroke grouping, and group fitting. The proposed method is suitable for online conceptual design. The user can draw over the existing drawing to complete, correct, or enhance an edge. All strokes are considered as part of the sketch, and multiple overtraced strokes are automatically grouped into geometric primitives. The stroke grouping method presented does not like [9, 10] which rely on fitted parameters of the stroke; it can group strokes that represent arbitrary geometric primitives, such as line segments, polylines, and conic curves. The work presented here is only part of our final sketched-based 3D modeling system, but it could provide a good base point for 3D geometric recognition from line drawing. This paper only studies the grouping methods for strokes with relatively high overtraced ratio. The future work is to research how to process strokes that are not overtracing so much.

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 National Key R&D Program of China (Grant no. 2019YFB1703800), Natural Science Basic Research Plan in Shaanxi Province of China (Grant no. 2016JM6054), the Programme of Introducing Talents of Discipline to Universities (111 Project), China (Grant no. B13044).

References

- [1] Q. Li, C. Yang, Z. Zhang, X. Wang, Y. Gong, and E. Xuan, "Features from a single vector pressure stroke," *Mathematical Problems in Engineering*, vol. 2017, Article ID 5636913, 11 pages, 2017.
- [2] C. Alvarado and R. Davis, "Dynamically constructed bayes nets for multi-domain sketch understanding," in *Proceedings of the ACM SIGGRAPH 2007 Courses*, San Diego, CA, USA, August 2007.
- [3] Z. Lu, H. Yuan, W. Bi, and L. Tang, "Research on sketch-based design," in *Proceedings of the 2009 IEEE 10th International Conference on Computer-Aided Industrial Design & Conceptual Design*, pp. 1170–1175, IEEE, Wenzhou, China, November 2009.
- [4] D. C. Ku, S. F. Qin, and D. K. Wright, *A Sketching Interface for 3D Modeling of Polyhedron*, Eurographics, Aire-la-Ville, Switzerland, 2006.
- [5] N. Chansri and P. Koomsap, "Automatic single-line drawing creation from a paper-based overtraced freehand sketch," *International Journal of Advanced Manufacturing Technology*, vol. 59, no. 1–4, pp. 221–242, 2012.
- [6] T. M. Sezgin, T. Stahovich, and R. Davis, "Sketch based interfaces: early processing for sketch understanding," in *Proceedings of the ACM SIGGRAPH 2007 Courses*, pp. 37–es, San Diego, CA, USA, August 2007.
- [7] H. Tian and S. Qin, "Exploring local regularities for 3D object recognition," *Chinese Journal of Mechanical Engineering*, vol. 29, no. 6, pp. 1104–1113, 2016.
- [8] T. M. Sezgin and R. Davis, "HMM-based efficient sketch recognition," in *Proceedings of the 10th International Conference on Intelligent User Interfaces*, pp. 281–283, San Diego, CA, USA, January 2005.
- [9] S. Wang, S. Qin, and M. Gao, "New grouping and fitting methods for interactive overtraced sketches," *The Visual Computer*, vol. 30, no. 3, pp. 285–297, 2014.
- [10] D. C. Ku, S.-F. Qin, and D. K. Wright, "Interpretation of overtracing freehand sketching for geometric shapes," in *Proceedings of the 14th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision 2006, WSCG'2006 - In Co-operation with EUROGRAPHICS*, pp. 263–270, Vaclav Skala Union Agency, Plzen, Czech Republic, January-2006.
- [11] 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, p. 259152, 2013.
- [12] S. Qin and H. Tian, "On the algorithm for reconstruction of polyhedral objects from a single line drawing," in *Proceedings of the 21st Automation and Computing (ICAC)*, pp. 1–6, IEEE, Glasgow, UK, September 2015.
- [13] 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.
- [14] R. Arnheim, "Untersuchungen zur Lehre von der Gestalt," *Psychologische Forschung*, vol. 11, no. 1, pp. 2–119, 1928.
- [15] B. Yu and S. Cai, "A domain-independent system for sketch recognition," in *Proceedings of the 1st International Conference on Computer Graphics and Interactive Techniques in Australasia and South East Asia*, pp. 141–146, Australasia, 2003.
- [16] D. J. Kang, M. Masry, and H. Lipson, "Reconstruction of a 3D object from a main axis system," in *Proceedings of the AAAI fall symposium series: Making pen-based interaction intelligent and natural*, vol. 3, Arlington, VA, USA, October 2004.
- [17] M. Masry, D. Kang, and H. Lipson, "A freehand sketching interface for progressive construction of 3D objects," *Computers & Graphics*, vol. 29, no. 4, pp. 563–575, 2005.
- [18] T. Hammond and B. Paulson, "Recognizing sketched multistroke primitives," *ACM Transactions on Interactive Intelligent Systems (TiiS)*, vol. 1, no. 1, pp. 1–34, 2011.
- [19] M. J. Fonseca and J. A. Jorge, "Using fuzzy logic to recognize geometric shapes interactively," in *Proceedings of the Ninth IEEE International Conference on Fuzzy Systems. FUZZ-IEEE 2000 (Cat. No. 00CH37063)*, vol. 1, pp. 291–296, IEEE, San Antonio, TX, USA, May 2000.
- [20] C. Calhoun, T. F. Stahovich, T. Kurtoglu, and L. B. Kara, "Recognizing multi-stroke symbols," in *Proceedings of the AAAI Spring Symposium on Sketch Understanding*, pp. 15–23, Palo Alto, CA, USA, March 2002.
- [21] A. Shesh and B. Chen, "Smartpaper: an interactive and user friendly sketching system," *Computer Graphics Forum*, vol. 23, no. 3, pp. 301–310, 2004.
- [22] J.-D. Favreau, F. Lafarge, and A. Bousseau, "Fidelity vs. simplicity," *ACM Transactions on Graphics*, vol. 35, no. 4, pp. 1–10, 2016.

- [23] P. Barla, J. Thollot, and F. X. Sillion, "Geometric clustering for line drawing simplification," in *Proceedings of the ACM SIGGRAPH 2005 Sketches*, Los Angeles, CA, USA, 2005.
- [24] R. Pusch, F. Samavati, A. Nasri, and B. Wyvill, "Improving the sketch-based interface - forming curves from many small strokes," *Visual Computer*, vol. 23, no. 9–11, pp. 955–962, 2007.
- [25] A. Shesh and B. Chen, "Efficient and dynamic simplification of line drawings," *Computer Graphics Forum*, vol. 27, no. 2, pp. 537–545, 2008.
- [26] G. Orbay, L. B. Kara, and C. Graphics, "Beautification of design sketches using trainable stroke clustering and curve fitting," *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 5, pp. 694–708, 2011.
- [27] Y. Chien, W. C. Lin, T. S. Huang, and J. H. Chuang, "Line drawing simplification by stroke translation and combination," in *Proceedings of the Fifth International Conference on Graphic and Image Processing (ICGIP 2013)*, vol. 9069, Hong Kong, China, October 2014.
- [28] X. Liu, T.-T. Wong, and P.-A. Heng, "Closure-aware sketch simplification," *ACM Transactions on Graphics (TOG)*, vol. 34, no. 6, p. 168, 2015.
- [29] T. Ogawa, Y. Matsui, T. Yamasaki, and K. Aizawa, "Sketch simplification by classifying strokes," in *Proceedings of the 2016 23rd International Conference on Pattern Recognition (ICPR)*, pp. 1065–1070, IEEE, Cancun, Mexico, December 2016.
- [30] E. Simo-Serra, S. Iizuka, K. Sasaki, and H. Ishikawa, "Learning to simplify: fully convolutional networks for rough sketch cleanup," *ACM Transactions on Graphics*, vol. 35, no. 4, p. 121, 2016.
- [31] A. Bonnici and K. P. Camilleri, "Scribble vectorization using concentric sampling circles," in *Proceedings of the 2009 Third International Conference on Advanced Engineering Computing and Applications in Sciences*, pp. 89–94, IEEE, Sliema, Malta, October 2009.
- [32] E. Simo-Serra, S. Iizuka, and H. Ishikawa, "Mastering sketching: adversarial augmentation for structured prediction," *ACM Transactions on Graphics*, vol. 37, no. 1, pp. 1–13, 2017.
- [33] J. Chen, M. Du, X. Qin, and Y. Miao, "An improved topology extraction approach for vectorization of sketchy line drawings," *The Visual Computer*, vol. 34, no. 12, pp. 1633–1644, 2018.
- [34] A. Bartolo, K. P. Camilleri, S. G. Fabri, J. C. Borg, and P. J. Farrugia, "Scribbles to vectors: preparation of scribble drawings for CAD interpretation," in *Proceedings of the 4th Eurographics Workshop on Sketch-Based Interfaces and Modeling*, pp. 123–130, ACM, New York, NY, USA, August 2007.
- [35] L. B. Kara and T. F. Stahovich, "An image-based, trainable symbol recognizer for hand-drawn sketches," *Computers & Graphics*, vol. 29, no. 4, pp. 501–517, 2005.
- [36] T. M. Sezgin and R. Davis, "Handling overtraced strokes in hand-drawn sketches," *Making Pen-Based Interaction Intelligent and Natural*, vol. 2, 2004.
- [37] N. Chansri and P. Koomsap, "Sketch-based modeling from a paper-based overtraced freehand sketch," *The International Journal of Advanced Manufacturing Technology*, vol. 75, no. 5–8, pp. 705–729, 2014.
- [38] P. L. Rosin, "Techniques for assessing polygonal approximations of curves," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 6, pp. 659–666, 1997.
- [39] S.-X. Wang, M.-T. Gao, and L.-H. Qi, "Freehand sketching interfaces: early processing for sketch recognition," *Human-Computer Interaction: Interaction Platforms and Techniques*, pp. 161–170, Springer, Berlin, Germany, 2007.
- [40] M. D. Berg, O. Cheong, M. V. Kreveld, and M. Overmars, "Computational geometry," *ACM Computing Surveys*, vol. 28, no. 1, pp. 27–31, 2000.