Received January 17, 2021, accepted January 26, 2021, date of publication January 29, 2021, date of current version February 5, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3055500

A Lightweight Processing Method for Hand-Drawn Pressure-Sensitive Trajectories Oriented Toward Web-Based 3D Modeling

FEIYU ZHAO[1,2], WEI TIAN[1], ZONGXIAO ZHU[1], SHENG LEI[1], AND DELONG KONG[1]
[1]College of Computer Science, South-Central University for Nationalities, Wuhan 430074, China
[2]Hubei Digital Manufacturing Key Laboratory, Wuhan University of Technology, Wuhan 430070, China

Corresponding author: Feiyu Zhao (zhaofeiyu1992@scuec.edu.cn)

This work was supported in part by the Start-up Scientific Research Funds for the Introduction of Talents, South-Central University for Nationalities, under Grant YZZ20002, in part by the National Natural Science Foundation of China under Grant 51977221, in part by the Natural Science Foundation of Hubei Province under Grant 2020CFB174, and in part by the Provincial Teaching and Research Project of Higher Education Institutions in Hubei Province (Special Laboratory Project) under Grant 2018219.

ABSTRACT Digital geometry processing of hand-drawn pressure-sensitive trajectories oriented toward Web-based three-dimensional (3D) modeling becomes attractive problem. Existing practices usually focus on converting raw trajectories into 3D stroke meshes with consistent line width, which the pressure information can not been fully utilized for denoising, simplification and reflecting realistic graphics of 3D stroke mesh. Also, there are little research on the high quality of trajectory with data transmission fluency and light weight in the Web environment. We propose a lightweight processing method for hand-drawn pressure-sensitive trajectories that is oriented toward Web-based 3D modeling. An angle-chord height united trajectory simplification (ACUTS) algorithm is proposed, in which the discrete points are simplified by comprehensively considering the angle deviation and chord height deviation with the sliding window mechanism. In addition, a lightweight reconstruction algorithm based on pressure -sensitivity for discrete points (LRPDP) is also proposed, in which the pressure value can be fully utilized for simplifying the redundant points while ensuring the reconstructed 3D stroke mesh more realistic. Besides, the experimental results show that compared to some other methods, we perform best in terms of data reduction using compression ratio and reduction percentages. We also perform best in terms of high accuracy of 3D stroke mesh using mean stochastic error distance, local maximum error and total length error. Although the insufficient utilization of the pressure-sensitive information limits our research to some extents, the significance of our research is to provide a new strategy to realize Web-based 3D modeling using hand drawing mode.

INDEX TERMS TERMS lightweight, 3D modeling, hand-drawn, pressure-sensitive trajectory, web environment.

I. INTRODUCTION

Three-dimensional (3D) modeling is a key technology that uses computers and graphics equipment to help engineers and technicians in 3D product design work [1]. In the early 1960s, computer graphics (CG) became a specialized subject and gradually matured [2]. Simultaneously, the continuous development of new ideas such as interactive graphics processing technology, has laid a theoretical foundation for enhancing human-computer interaction in 3D modeling [3]. With the development of the Internet, 3D modeling is gradually being developed from client software to Web applications, such as TinkerCAD from Autodesk [4], a Web-based 3D modeling software that is free and easy-to-use. It runs directly on Web browsers such as Chrome and Safari, without the need to install client software in the operating system, which allows designers to perform 3D modeling based on the Web browser anytime and anywhere. Although the professional computer-aided design (CAD) software supports the creation of more precise and complex 3D models such as SolidWorks and Onshape, Web-based 3D modeling is an easy-to-use method with which teachers, kids, hobbyists, and designers

The associate editor coordinating the review of this manuscript and approving it for publication was Jiachen Yang.

can create 3D models using their creativity. It is widely used in 3D modeling technology education and information technology education in middle schools and primary schools.

With the development of human-computer interaction (HCI), 3D modeling software is developing for the implementation of the post windows, icons, menus, pointer (WIMP) interface [5], which provides consumers with multisensory perception and an immersive 3D modeling interactive experience via hand drawing and somatosensory modalities, which effectively simplify 3D modeling and enhance the fun of 3D modeling. One of our main hypotheses is that a tablet would be accessed through the developed Web-based 3D modeling software, which allows users to automatically convert hand-drawn contours into lightweight 3D models in the Web browser, thereby further enhancing the fun of 3D modeling and enabling more teenagers and nonprofessionals to participate in innovative product design anytime and anywhere. Hand-drawn 3D modeling not only reflects the designer's original design thinking [6] but also enables quick expression of the conceptual shape of the digital geometric model, such that the model has both high artistic appeal and individualized customization [7].

Hand-drawn pressure-sensitive trajectories collected by the tablet are a series of discrete points with serial numbers, timestamps, coordinate values, pressure values and directions [8]. Due to the high acquisition frequency of the tablet [9], different user habits and other outside interferences, the collected trajectories have the characteristics of a large amount of data, inconsistent data density, inhomogeneous distribution of points and considerable noise [10], which makes it difficult to satisfy the requirements of high-efficiency digital geometry processing and data lightweight for 3D rendering in the Web environment. Thus, an interesting target problem named "lightweight processing for hand-drawn pressure-sensitive trajectories" is proposed, that is, to simplify the hand-drawn pressure-sensitive trajectories with low loss or lossless, and reconstruct to the 3D stroke mesh in the Web environment. First, it makes the higher execution efficiency of the digital geometry processing algorithm to satisfy the low time complexity of the algorithm in the Web environment. Second, it makes the data lightweight of the reconstructed 3D stroke mesh to satisfy the fluency of 3D rendering in the Web environment. Finally, the collected pressure values should be fully utilized so that the 3D stroke mesh performs with inconsistent widths, which is more realistic and closer to the effect of real hand painting shape.

To solve the abovementioned target problem, research emphasis should be summarized as two aspects: how to efficiently compress the pressure-sensitive trajectory data; how to transform the pressure values into the specific effect after compressing the trajectory data.

For the first question, trajectory compression method can be usually divided into batch compression and online compression. Batch compression achieves the purpose of compression by removing redundant points in the complete trajectory, which makes it easier to generate the global optimal trajectory, but usually takes a long time. Online compression can compress the trajectory in real time, but the time sequence and spatial sequence of the data acquisition process must be considered comprehensively, which is associated with high computational complexity. Thus, an angle-chord height united trajectory simplification (ACUTS) algorithm is proposed. We first merge the trajectory data collection process with the resampling process, i.e., the trajectory data should also be resampled while being acquired. To further simplify the trajectory data, the resampled discrete points are simplified by comprehensively considering the angle deviation and chord height deviation with the sliding window mechanism, which can make the trajectory data lightweight and achieves a high degree of accuracy. To quantitatively evaluate the feasibility and effect of trajectory compression, some metrics are used to measure the data reduction effect which include compression ratio (CR) and whole processing time t, while others are used to measure the accuracy effect which include local maximum error ($E_{\max}$), total length error (E_l) and mean stochastic error distance (SED) [53], [54]. Although the pressure values are not utilized enough to reduce the trajectory data, our method can still ensure not only the light weight of hand-drawn trajectory data to meet the requirements of high-efficiency data transmission and low computational complexity of data processing in the Web environment but also the high accuracy of the simplified trajectory data.

For the second question, although some previously methods can achieve accurate mapping from pressure-sensitive information to specific effects, the established virtual brush or stroke models were relatively complex. Since the calculation complexity is too high to achieve satisfactory visual effects on human eyes, high performance graphics hardware should be used, which made it difficult to meet the high-efficiency requirements of data processing in the Web environment. Thus, a lightweight reconstruction algorithm based on pressure -sensitivity for discrete points (LRPDP) is proposed. We first use the pressure value as the basis for judging whether a local point set needs to be simplified while ensuring that the trajectory is not distorted, i.e., whether the points are greatly affected by their pressure values. Meanwhile, we mapped the pressure values to the adaptive-width hierarchical 3D point cloud data, so that the shape of the reconstructed 3D stroke mesh could be more realistic. To quantitatively evaluate the transformation effect, the reduction percentages of the number of vertices ($\eta_{N'}$), number of triangular facets (η_F), data volume of the 3D stroke mesh (η_S), and 3D reconstruction time (η_T) are calculated, compared with the original 3D stroke mesh. Although we do not perform a quantitative analysis of the fidelity and quality of the 3D stroke mesh based on relevant metrics that contain pressure values, the experimental results still show that our method can not only simulate the real handwriting but also reduce the amount of data to some extent, which is of great significance for generating more realistic and lightweight 3D stroke mesh in the Web environment.

The remainder of this paper is organized as follows. Section II introduces the related work in trajectory compression and sketch-based 3D modeling. Section III introduces the ACUTS algorithm. Section IV introduces LRPDP. Section V shows the experimental tests via the tablet and the browser. Section VI introduces the limitations of our study, and Section VII contains the conclusion.

II. RELATED WORK

To solve the problem of lightweight processing for hand-drawn pressure-sensitive trajectories@comm the work presented in this article covers two literature domains: trajectory compression and pressure-sensitive trajectory processing. The purpose of the trajectory compression is to make the trajectory data achieve a high degree of accuracy and light weight, and the purpose of the pressure-sensitive trajectory processing is to map the collected pressure information into adaptive-width hierarchical point cloud data to ensure that the generated hand-drawn trajectory 3D stroke mesh performs with light weight with inconsistent widths. Therefore, literature review should be carried out from these two categories.

A. TRAJECTORY COMPRESSION

Trajectory compression can be divided into batch compression and online compression. Batch compression achieves the purpose of compression by removing redundant points in the complete trajectory, which makes it easier to generate the global optimal trajectory, but usually takes a long time. Online compression can compress the trajectory in real time, but the time sequence and spatial sequence of the data acquisition process must be considered comprehensively, which is associated with high computational complexity [11].

1) BATCH COMPRESSION METHOD

In the field of batch compression, Douglas and Peucker [12] proposed a trajectory simplification method that is referred to as the Douglas-Peucker (DP) algorithm. The algorithm connects the head and tail points of the trajectory as a reference line segment, calculates the vertical distance from the sampling point to the reference line segment by iteration, judges whether the maximum vertical distance is greater than the given threshold value, determines whether to split the trajectory, and then realizes the trajectory compression. Long *et al.* [13] introduced a polynomial-time algorithm for optimal direction-preserving simplification and another approximate algorithm with a quality guarantee, in which the original trajectory was simplified based on the direction and angle threshold. Abbasi *et al.* [14] presented a polygonal approximation technique using reverse polygonization, in which the trajectory compression was achieved by extracting the trajectory feature points and performing a polygonal approximation, such that the maximal perpendicular distance of an approximating straight line from an original curve is minimized. Liu *et al.* [15] proposed a novel online algorithm for error-bounded trajectory compression named the bounded quadrant system (BQS), which compresses trajectories with extremely small costs in space and time using convex -hulls. This method effectively reduces the time and space complexity of trajectory compression. Zhao and Shi [16] presented a method based on the improved DP algorithm, which can significantly reduce the compression time and exhibits better performance at high compression strengths and compression times.

2) ONLINE COMPRESSION METHOD

In the field of online compression, Muckell *et al.* [17] proposed a new online compression algorithm named SQUISH that offers improved performance across multiple error metrics with small to medium compression ratios. When the queue is full, the point that caused the smallest error is deleted, which causes the algorithm to have low time complexity. Muckell *et al.* [18] proposed an improved online trajectory compression algorithm named SQUISH-E, which can achieve the optimal compression rate within a given error threshold. Deng *et al.* [19] proposed an online direction-preserving simplification method for trajectory streaming data, in which a data structure referred to as BQS is employed and the offline direction-preserving trajectory simplification (DPTS) method is modified as an online algorithm to improve the compression rate. Gao *et al.* [20] proposed an online compression algorithm of automatic identification system (AIS) trajectory data based on an improved sliding window, in which the sensitivities of distance and angle thresholds with the compression rate of the algorithm were analyzed. Han *et al.* [21] presented a parallel online trajectory compression approach named PSQUISH-E, in which multicore and many-core approaches were employed to accelerate SQUISH-E.

Batch compression methods [12]–[16] consider both the high compression rate and high fidelity, and the time complexity can be reduced by the optimization of the algorithm. However, the process relies on the completeness of the trajectory data and is time-consuming, which cannot satisfy the requirements of the efficient transmission and processing of the trajectories in the Web environment. Thus, we merge the trajectory data collection process with the resampling process, i.e., the trajectory data should also be resampled while being acquired. To further simplify the trajectory data, the resampled discrete points are simplified by comprehensively considering the angle deviation and chord height deviation with the sliding window mechanism, which can achieve efficient transmission and processing in the Web environment.

In addition, the trajectories processed by online compression methods [17]–[21] are not acquired by the tablet. Thus, they do not contain pressure-sensitive information and cannot accurately reflect the feature of the uneven widths of the hand-drawn trajectories. Thus, the pressure-sensitive information obtained by the tablet is mapped into adaptive-width hierarchical 3D point cloud data in our method, so that the shape of the 3D stroke mesh performs with inconsistent widths, which can not only simulate the real handwriting but also reduce the amount of data to some extent. The speed and quality of 3D rendering can be effectively improved in the Web environment.

B. PRESSURE-SENSITIVE TRAJECTORY PROCESSING

Regarding the pressure-sensitive trajectory processing, we believe that three main issues need to be studied, namely, transform stroke force into specific effect, simplify trajectory using pressure information, transform original trajectory to pressure-sensitive data. The literature review is conducted from the abovementioned three aspects.

1) TRANSFORM STROKE FORCE INTO SPECIFIC EFFECT

The key issues associated with processing pressure-trajectory data are how to use stroke force and transform stroke force into the specific effect after compressing the original pressure-sensitive trajectories. For instance, Luo *et al.* [22] presented a novel handwriting device that is capable of capturing both the static trajectory of the writing pen and the dynamic handwriting information, particularly the multiaxis handwriting force information. After the original points were filtered, the mean value of these points was employed as the new pressure value to simplify the original trajectory, and the data coupling on the five-dimensional channel was eliminated by the proposed decoupling algorithm. Wang *et al.* [23] proposed a force sensing method that is based on the leverage effect to detect 3D forces between the pen's tip and the paper. A compact and low-cost pressure-sensing assembly was designed, which was integrated by five off-the-shelf one-dimensional (1-D) force sensors. A matrix-based measurement model was established to compute the force signal in the task coordinate system (CS), which was transformed from the force signal in the Sensor CS and the angle signal from a two-dimensional (2-D) angle sensor. Jiao *et al.* [24] described the feature representation of the friction direction in Cartesian coordinates, and a novel system combines the extracted signature features with the characteristic of force magnitude to realize the fusion of distance measures and probability. Guo *et al.* [25] proposed a novel simulation method of the brush stroke. The spring-mass model was applied to construct the brush model, which could realistically simulate the brush morphological changes according to the force exerted on it. According to the deformation of the brush model at a sampling point, the brush footprint between the brush and the paper was calculated in real time. Tamilarasi and Nithya Kalyani [26] developed a deep learning strategy-based smart signature verification system. The extracted pressure values were analyzed to construct the dynamic feature of the signature using integer wavelet transformation, and a neural network was selected for decision-making based on the original and forged signatures. Compared with the conventional system, the error ratio was reduced to 20% using the pressure information. Heckeroth *et al.* [27] used the mean pressure values to construct a statistical multilevel model, which could more precisely verify whether digitally captured signatures and conventional signatures executed with a pen on paper differed in their characteristics. The experimental results showed that there were significant differences between the three signature types: a) with a stylus on a pad, b) with an inking pen on a sticky note attached to a signature pad allowing the simultaneous attainment of a digital and an analogue version, and c) with a pen on paper. It can be seen that the processing of original pressure data using the mean/median or another operator to calculate new pressure values, or to construct new mathematical models for simulation analysis or verification of strokes, was beneficial for analyzing the semantic information contained in the trajectories. For the pressure-sensitive trajectory, stroke force is contained which reflect the direct visualization of the line width. Thus, we construct an easy given function for simulating the handwriting strokes, in which the stroke force should be automatically mapped to the equivalent pressure-sensitive radius of the pressure-sensitive circle. Compared with the literatures [22]–[27], LRPDP performs efficient execution in time complexity because of the easy conversion from the stroke force to the specific effect just using Formula (6) in Section IV.

2) SIMPLIFY TRAJECTORY USING PRESSURE INFORMATION

The pressure data can also simplify the originally collected trajectory data. Gong *et al.* [28] proposed a system structure of a 3D automotive sketching system based on the pressure sensitivity of a digital pen. For the trajectory data that contain pressure information, the points were simplified based on the DP algorithm, and the least squares method, which was based on the parameter polynomial, was then employed for curve fitting to obtain the best simplified points and stroke trends. Guo and Zhang [29] proposed a compression method using curve fitting. A fitting curve of the haptic data was constructed by reversing the process of generating the haptic data, which was defined by piecewise quadratic parametric curves not only with first derivatives that agree at the points where they join but also by approximating the original signals with a quadratic polynomial precision. This approach can significantly reduce the compression distortion while maintaining a high data reduction rate. Meng *et al.* [30] presented a method to evaluate the accuracy by applying a point load at the reference position on the tablet. The coordinates of the point on where the pen tip contacts the plate can be calculated using the stroke force to resample the original trajectories. Li *et al.* [31] treated the handwriting pressure as one of the extracted timing features. The original point set containing handwriting pressure data was resampled in equal time frames. Moreover, the handwriting pressure was divided into 4 levels, and the pressure level corresponding to a randomly selected frame was obtained using the constructed rounding function and normalized to satisfy the requirement of initial simplification. Donati *et al.* [32] created a "simulated" dataset called an inverse dataset. The strokes pressure in this dataset was applied to the start and end portions of each trait (first-last

10%), which was lower to satisfy the requirement of real traits and data lightweight. The scribbles and variable-width strokes can be transformed to clean 1-pixel lines. As shown in the previously mentioned references, the pressure value of a point on the pressure-sensitivity trajectory represents the importance of this point in the trajectory to some extent, which also gave us the idea to study the pressure-based simplification method. Thus, we should determine whether the points in the current sliding window are greatly affected by their pressure values, that is, whether the points need to be simplified based on the pressure values. For us, if the pressure values of these points change slightly, for the painter, these points are not relatively important and do not show semantics, which can be removed according to the curvature change. Compared with the curve fitting methods [28], [29] and the pressure value re-estimation methods [30]–[32], we just use the variance of the pressure values to measure the degree of pressure change in ACUTS, which is presented using Formula (3) in Section III. This strategy not only fully considers the semantic information contained in the pressure value, but also simplifies the trajectory just based on the curvature change rather than the pressure value, which ensures the accuracy of the trajectory after simplification processing.

3) TRANSFORM ORIGINAL TRAJECTORY TO PRESSURE-SENSITIVE DATA

In addition, how to transform the original trajectory to pressure-sensitive data is also an important issue. Xuan *et al.* [33] presented a novel design pattern that is based on the direct conversion of the pressure information to the 3D stoke depth. To transform the trajectories to the pressure-sensitive 3D data, they determined the stroke influence area of the plana or spatial curve with the simplified method of normal pressure estimation on each node of the stroke point, such that the pressure-sensitive information was converted to the depth information of a planar stroke to create a 3D stroke. Wang *et al.* [34] modeled the haptic skill during the writing of different strokes in Chinese characters to achieve haptic rendering with high fidelity and stability. A force model during the handwriting process was established to achieve realistic force simulation. For the constraint force within a paper's surface, different force attraction factors can be established at different feature points to reflect the characteristics of each stroke. Xing and Wu [35] analyzed the force of the stroke based on the midpoint ellipse model. The pressure in the X and Y directions determines the direction of the long and short axes of the ellipse, and the pressure in the Z direction determines the size of the ellipse. Based on the diffusion limit aggregation model (DLA), the ink diffusion simulation on paper was realized, and the stroke pressure information was converted into the width information. Otsuki *et al.* [36] studied a basic model to calculate the strokes pressure named ''reaction force vector'', which changes according to the brush pressure applied to the canvas and the painting direction. The magnitude and direction of this vector could be calculated only through geometric relationships in the basic model. In addition, an extended model containing the frictional force between the canvas surface and brush was also studied to transform the brush location to pressure-sensing information. Xie *et al.* [37] modeled the ink-and-water dispersion dynamics using a discrete square lattice model in 2D surface flow, which was suitable for simulating the real strokes with adaptive -widths. The water-based paint flow physical simulation was adopted, and the quantity of ink and water on the canvas was initialized according to the current sampled brush texture images. Liang *et al.* [38] calculated the stroke widths at each skeleton point according to the radius of the constructed pressure-sensing circles. The pressure information of the strokes could be automatically mapped to the radius of the pressure-sensing circle according to a given function, and the adaptive -widths were generated, which enables transfer of the art style of the Chinese font and maintains the imitation accuracy of the calligraphy. Although the previously mentioned methods can achieve accurate mapping from pressure-sensitive information to specific effects, the established virtual brush or stroke models were relatively complex. Furthermore, since the calculation complexity of some abovementioned approaches was too high to achieve satisfactory visual effects on human eyes, graphics processing units (GPUs) with high performance should be used, which made it difficult to meet the high-efficiency requirements of data processing in the Web environment. Thus, we present a polyline method to calculate the stroke widths at each skeleton point and simulate the real strokes with adaptive -widths in LRPDP, which enables transfer of the art style of the text font and maintains the imitation accuracy of the calligraphy. Fig. 6 shows the generation process of the pressure-sensitive trajectory using the collected original trajectory, which performs lower computational complexity than the approaches presented in the literatures [33]–[38]. So that, pressure-sensitive trajectory with adaptive -widths can be rapidly generated even with low performance GPUs in the Web environment.

III. ANGLE-CHORD HEIGHT UNITED TRAJECTORY SIMPLIFICATION ALGORITHM (ACUTS)

A. OVERVIEW OF THIS ALGORITHM

To ensure the simplification of the trajectory data and the high degree of lightweight, we proposed the ACUTS algorithm, shown in Fig. 1.

First, the original hand-drawn trajectory T_o that was acquired by the tablet should be resampled in real time based on the chord length threshold, i.e., iteratively determine whether the distance between adjacent points is greater than the given chord length threshold l_ε. Thus, the presimplified trajectory T_{ps} can be acquired. Second, T_{ps} should be simplified based on a sliding window in which the pressure-sensitivity values, the angle deviation threshold α and chord height deviation threshold d are comprehensively considered, to acquire the simplification trajectory T_s. Note that by judging whether the variance in the pressure values of the several

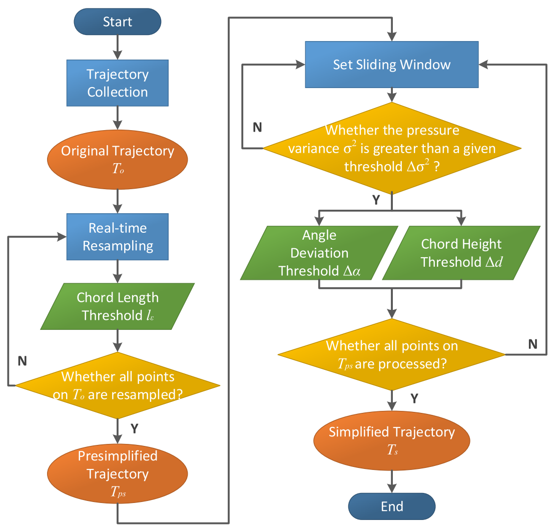
Start
Trajectory Collection
Original Trajectory T_o
Real-time Resampling
Chord Length Threshold l_ε
N
Whether all points on T_o are resampled?
Y
Presimplified Trajectory T_{ps}
Set Sliding Window
N
Whether the pressure variance σ^2 is greater than a given threshold $\Delta\sigma^2$?
Y
Angle Deviation Threshold $\Delta\alpha$
Chord Height Threshold Δd
Whether all points on T_{ps} are processed?
N
Y
Simplified Trajectory T_s
End

FIGURE 1. Overview of the ACUTS algorithm.

points in a sliding window is greater than a given threshold, we judge whether the removal operation can be performed in this window. If the variance is greater than a given threshold, the pressure value in the window changes greatly, which indicates that these points are greatly affected by pressures and need to be retained. Conversely, when the variance is less than a given threshold, we can decide whether to remove some redundant points based on the chord height parameter and angle parameter.

B. THEORETICAL METHOD OF THE RESAMPLING PROCESS

The original hand-drawn trajectory collected by the tablet is a series of discrete points with serial numbers, timestamps, coordinate values, pressure values and directions [39], which can be formulated as

$$T_o = \{S_i\,(x_i, y_i, t_i, p_i), i = 1, 2, 3, \cdots, n\} \quad (1)$$

where x_i and y_i represent the coordinate values of the i-th frame point in the tablet CS, and the unit is 10^{-2} mm. If the size of the effective acquisition area is $U \times V$, $0 \le x_i \le U$ and $0 \le y_i \le V$. t_i represents the timestamp and p_i indicates the pressure level captured by the pressure pen at this point. If the total pressure level of the tablet is D, $1 \le p_i \le D$, $p_i \in \mathbb{N}^*$. Note that although the pressure orientation of a point collected by the tablet changes as the angle between the stroke and the tablet changes in the process of collecting trajectory data, the actual value p_i collected by the tablet is the component of the actual pressure p_i in the direction perpendicular to the tablet plane. Let the angle between the stroke and the tablet be τ at p_i; it can be assumed that $p_i = P_i \sin\tau$.

T_o has the characteristics of a large amount of data, inconsistent data density, and inhomogeneous distribution of vertices because of user habits and outside interference [40]. Thus, preprocessing should be executed before the

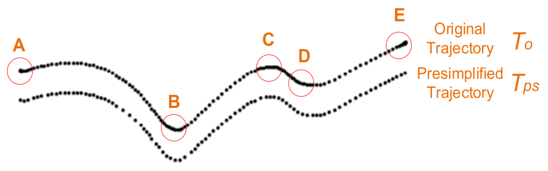

FIGURE 2. Resampling effect of the original trajectory. The resampling effect is better in areas of large curvature, such as A, B, C, D and E.

simplification of T_o to make it relatively uniform and improve the efficiency of data transmission and processing from the client to the cloud server [41].

Thus, a real-time resampling method that is based on the chord length threshold l_ε, which is performed during the data acquisition process to dilute the dense areas of the original hand-drawn trajectory and eliminate the noise points, is proposed in this paper. We define the chord length threshold l_ε as the maximum distance allowed for two adjacent points in the hand-drawn trajectory data. Note that the specific value of the chord length threshold l_ε needs to be defined by the hand-drawn 3D modeling user, but it is necessary to consider the handwriting accuracy of the tablet and the tablet width. Let the handwriting accuracy be d_{acc} and the tablet width be w. It can be assumed that $d_{acc} \le l_\varepsilon \le w$.

After this resampling process, the presimplified trajectory data are formulated as follows:

$$T_{ps} = \left\{S_j\left(x_j, y_j, t_j, p_j\right), j = 1, 2, 3, \cdots, m, m \le n\right\} \quad (2)$$

The resampling method shows that $\overline{S_{j+1}S_j} \ge l_\varepsilon$ in T_{ps}; that is, the chord lengths of two arbitrary adjacent points are greater than l_ε. The implementation effect of this method is shown in Fig. 2. The original hand-drawn trajectory T_o has obvious vibration and stagnation at the start point A and end point E. In places where the curvature changes are relatively large (such as B, C, and D), the original hand-drawn trajectory is dense due to the slow speed of hand- drawing. After resampling, the points near A, B, C, D, and E are significantly simplified, and the smoothness and continuity of the trajectory are preserved.

C. THEORETICAL METHOD OF THE SIMPLIFICATION PROCESS

Although the density of the resampled presimplified trajectory data is relatively uniform and the trajectory retains most of the features, there are still many redundant points. Therefore, the further simplification process is essential because it can ensure a lower data volume of the reconstructed 3D stroke mesh with the premise of ensuring no distortion to fluently realize 3D rendering in the Web browser.

We applied a sliding window mechanism for our simplification algorithm, in which the window size W_S is the number of processing points at one time. The sliding window moves forward in sequence according to a certain scale, which not only ensures that the algorithm processes only a limited number of the trajectory points in a loop to improve the execution efficiency but also avoids data congestion caused by concurrent requests from multiple clients. Thus, T_{ps} can be efficiently simplified in the Web environment.

Before removing redundant points, we first need to determine whether the points in the current sliding window are greatly affected by their pressure values, that is, whether the points need to be simplified based on the pressure values. We believe that if the pressure value of these points in a sliding window changes greatly, for the painter, these points are relatively important and show semantics to some extent in the entire trajectory, which cannot be easily removed. Conversely, if the pressure values of these points change slightly, for the painter, these points are not relatively important and do not show semantics, which can be removed according to the curvature change. We use the variance of the pressure values σ^2 to measure the degree of pressure change, which is formulated as

$$\sigma^2 = \frac{\sum_{k=1}^{W_S} (p_k - \mu)^2}{W_S} \tag{3}$$

where p_k represents the pressure value of the k-th point, μ represents the arithmetic mean of the pressure values in this sliding window, and W_S is the window size. The pressure variance threshold $\Delta\sigma^2$ should also be set artificially to measure whether the pressure value changes greatly. If $\sigma^2 > \Delta\sigma^2$, the pressure value of these points in a sliding window changes greatly, which cannot be easily removed. Conversely, if $\sigma^2 \leq \Delta\sigma^2$, the pressure value of these points in a sliding window changes slightly, which can be removed according to the curvature change.

After the previously mentioned pressure-based filter, we start the simplification process. According to the characteristics of the hand-drawn trajectory, the points with larger curvature are denser, while the points with smaller curvature are sparser [42]. The angle and chord height of the trajectory are two factors that have a great influence on the curvature change [42].

The chord height method determines whether to remove the middle point by presetting the chord height threshold d and then comparing d and d, where d represents the chord height of the middle point. As shown in Fig 4 (a), the chord height of S_2 is d_2, which represents the height of triangle $S_1S_2S_3$ from S_2.

The angle deviation method determines whether to remove the point by presetting the angle threshold α and then comparing α and α, where α represents the angle deviation of the chord and edge instead of the chord height d.

Note that regardless of a chord height deviation or angle deviation, they are both parameters that measure the local curvature of the hand-drawn trajectory. If we use only one deviation to decide whether to remove the points, local cumulative errors may occur. Thus, points in the smooth area that cause trajectory distortion are easily removed [43,44], as shown in Fig. 3 (a) and Fig. 3 (b).

Therefore, we take into account both angle and chord height to confine the simplified processing objects within the sliding window. The values of d and α [42] can be formulated

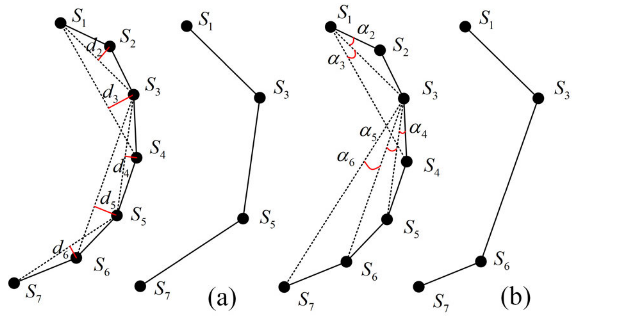

FIGURE 3. Two types of trajectory simplification methods: (a) represents the chord height method and (b) represents the angle deviation method.

as

$$\begin{cases} \Delta d = 0.006\,\overline{S_1S_m} \\ 10^\circ \leq \Delta\alpha \leq 20^\circ \end{cases} \tag{4}$$

where $\overline{S_1S_m}$ represents the chord length of the presimplified trajectory T_{ps} between the first point and the last point.

D. DETAILED STEPS OF THE ACUTS ALGORITHM

The detailed steps of this algorithm are explained as follows:

Step 1: Acquire an arbitrary point in the process of drawing by the tablet. For example, in Fig. 4 (a), point S_i of the original trajectory T_o is acquired by the tablet.

Step 2: Calculate the chord length value of the two neighboring points. For example, when point S_{i+1} is acquired, calculate the chord length l_{ii+1} of $\overline{S_iS_{i+1}}$ with S_i as the starting point.

Step 3: Determine the relationship between the actual calculated chord length value and a given chord length threshold. If the actual chord length value is greater than the threshold, the current point needs to be retained; otherwise, the current point should be removed until all the trajectory points are traversed. For example, given the chord length threshold l_ε, as shown in Fig. 4 (b), if $l_{ii+1} \geq l_\varepsilon$, S_{i+1} is retained as the new start point, the process should return to ***Step 2*** to calculate the chord length $\overline{S_{i+1}S_{i+2}}$. If $l_{ii+1} < l_\varepsilon$, S_{i+1} is removed, the process should return to ***Step 2***. The chord length $\overline{S_iS_{i+2}}$ is calculated until it is greater than the chord length threshold l_ε.

Step 4: After the resampling process from ***Step 1*** to ***Step 3***, the original trajectory can be resampled to the presimplified trajectory. For example, in Fig. 4 (c), the original hand-drawn trajectory T_o is acquired and resampled to obtain the presimplified trajectory T_{ps}.

Step 5: The data simplification process should be started with the initialization of four parameters: sliding window size, pressure variance threshold, chord height threshold and angle deviation threshold. For example, in Fig. 4 (d), we initialize the sliding window size $W_S = 3$, pressure variance threshold $\Delta\sigma^2$, chord height threshold d and angle deviation threshold α.

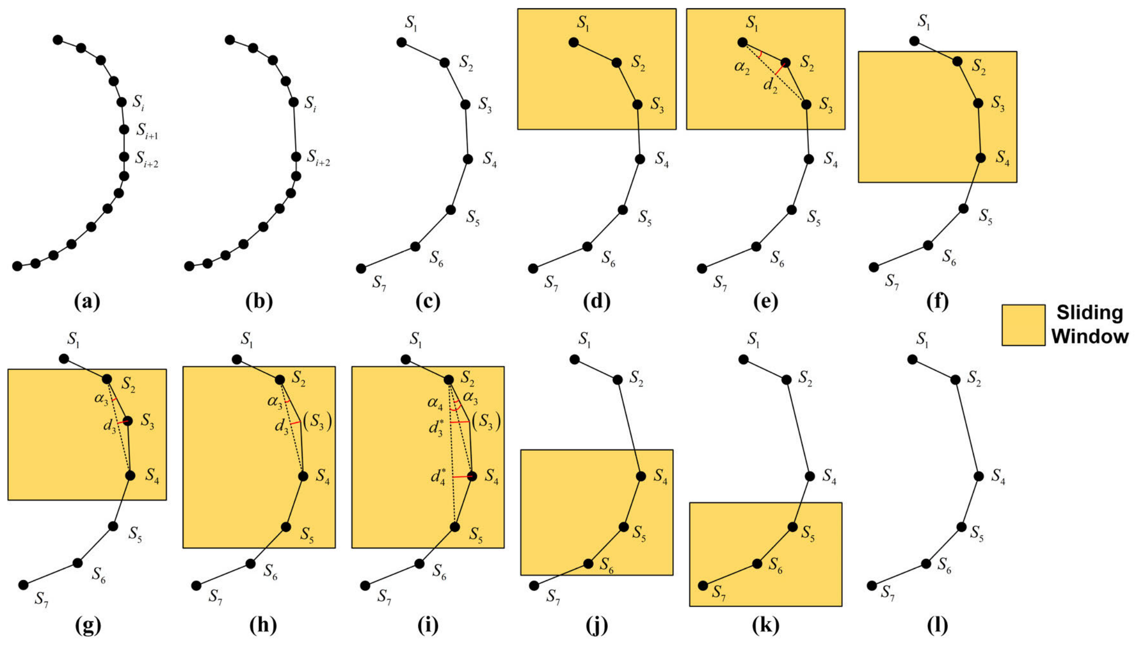

FIGURE 4. Detailed steps of the ACUTS algorithm.

Step 6: Calculate the pressure variance to determine whether to start the simplification process. If the pressure variance σ^2 is less than or equal to the given pressure variance threshold $\Delta\sigma^2$ in a window, it is possible that one or some redundant points in this window would be removed, and the process should proceed to ***Step 7***. Conversely, if the pressure variance σ^2 is more than the given pressure variance threshold $\Delta\sigma^2$ in a window, all the points in this window need to be retained, and this window should be moved forward to the next new position, followed by recalculation in ***Step 6***. For example, in Fig. 4 (d), we start with the window containing S_1, S_2 and S_3. If the pressure variance of S_1, S_2 and S_3 is less than the given threshold, it is possible that S_2 would be removed, and the process should proceed to the next step.

Step 7: Calculate the angle value and chord height value of the current middle point with its two neighbor points. If the calculated angle value is greater than the given angle threshold or the calculated chord height value is greater than the given chord height threshold, the current middle point needs to be retained, and the sliding window should be moved one point forward and recalculated in ***Step 6***. Conversely, if both the calculated angle value and chord height value are less than or equal to the given thresholds, the current middle point needs to be removed but the local shape feature should be retained. The sliding window moves one point forward, but the start point should be retained to further determine whether one or some redundant points in this new window need to be removed according to ***Step 6*** and this step. For example, in Fig. 4 (e), S_1 and S_3 are connected, and the angle α_2 between $\overline{S_1S_2}$ and $\overline{S_1S_3}$ and the chord height d_2 from S_2 to $\overline{S_1S_3}$ should be calculated. If $\alpha_2 > \alpha$ or $d_2 > d$, the local curvature of the trajectory varies greatly from S_1 to S_3, i.e., S_2 should be retained, and the window should be moved one point forward to the new location that contains S_2, S_3 and S_4, which is shown in Fig. 4 (f).

Step 8: The sliding window should be moved one point forward. Then, the process returns to ***Step 6*** to determine whether the simplification process in this new window should be executed. For example, in Fig. 4 (f), the new window contains S_2, S_3 and S_4. If the pressure variance of S_2, S_3 and S_4 is less than the given threshold, it is possible that S_3 would be removed, and the process should proceed to the next step.

Step 9: Return to ***Step 7*** and calculate the angle value and chord height value of the current new middle point with its two neighboring points. For example, in Fig. 4 (g), S_2 and S_4 are connected, and the angle α_3 between $\overline{S_2S_3}$ and $\overline{S_2S_4}$ and the chord height d_3 from S_3 to $\overline{S_2S_4}$ should be calculated. If $\alpha_3 \leq \alpha$ and $d_3 \leq d$, the redundant point S_3 should be removed, but the local shape feature of the trajectory should be retained, which is shown in Fig. 4 (h). The window moves one point forward but retains S_2, which contains S_2, S_4 and S_5. If the pressure variance of S_2, S_4 and S_5 is still less than the given threshold $\Delta\sigma^2$, the cumulative angle deviation α_4 between $\overline{S_2S_3}$ and $\overline{S_2S_5}$ and the chord height d_4 should also be calculated to determine whether S_4 should be removed. d_4 represents the larger value of d_3^* or d_4^*, which is shown in Fig. 4 (i). If $\alpha_4 > \alpha$ or $d_4 > d$, the local curvature of the trajectory varies greatly along S_2, S_4 and S_5; that is, S_4 should be retained as the new start point.

Step 10: Return to ***Step 8***, move the sliding window one point forward and decide whether the simplification process in this new window should be executed according to ***Step 6***.

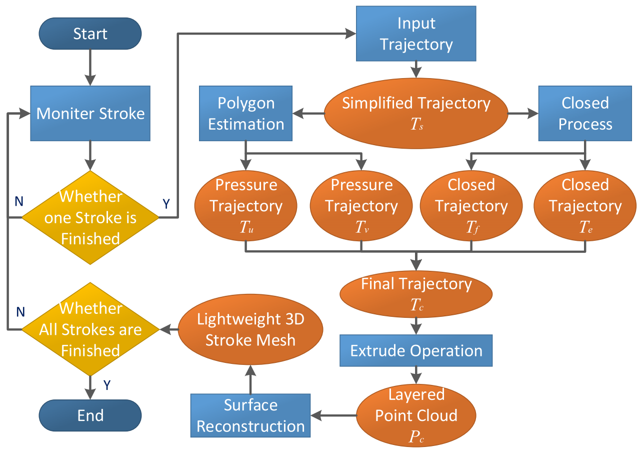
Start
Moniter Stroke
Whether one Stroke is Finished
N
Y
Input Trajectory
Simplified Trajectory T_s
Polygon Estimation
Closed Process
Pressure Trajectory T_u
Pressure Trajectory T_v
Closed Trajectory T_f
Closed Trajectory T_e
Final Trajectory T_c
Extrude Operation
Layered Point Cloud P_c
Surface Reconstruction
Lightweight 3D Stroke Mesh
Whether All Strokes are Finished
N
Y
End

FIGURE 5. Overview of the LRPDP.

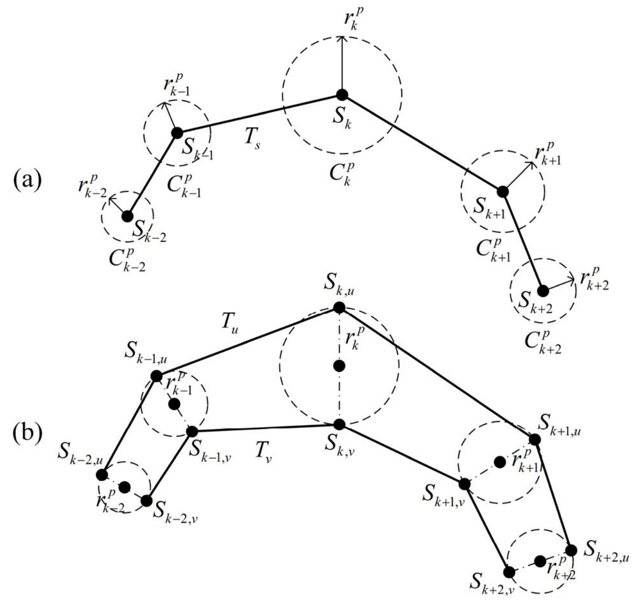

FIGURE 6. Generation of the pressure-sensitive circle and pressure-sensitive trajectory: (a) pressure-sensitive circle; (b) pressure-sensitive trajectory.

For example, in Fig. 4 (j), the new window contains S_4, S_5 and S_6. If the pressure variance of S_4, S_5 and S_6 is more than the given threshold, S_4, S_5 and S_6 need to be retained, and the window should be moved forward to the new location that contains S_5, S_6 and S_7. Similarly, in Fig. 4 (k), if the pressure variance of S_5, S_6 and S_7 is more than the given threshold, S_5, S_6 and S_7 need to be retained as well.

Contrasting two types of results in Fig. 3 (b) and Fig. 4 (l), the ACUTS algorithm can avoid the trajectory distortion caused by the accumulation of errors without significantly increasing the number of reserved points in T_s

IV. LIGHTWEIGHT RECONSTRUCTION ALGORITHM BASED ON PRESSURE -SENSITIVE OF DISCRETE POINTS (LRPDP)

The simplification trajectory T_s can be considered discrete points and cannot be directly reconstructed into a 3D stroke mesh. Thus, a layered 3D point cloud should be generated based on T_s [45]. Therefore, we proposed the LRDPD in this paper; the process is shown in Fig. 5. Initialize the stroke hover time t_h and monitor the status of the strokes. When the pen does not touch the tablet for a time greater than t_h, one stroke has been completed, so the algorithm should be performed immediately. Otherwise, the stroke is monitored. First, the simplified trajectory T_s should be imported and performed by polyline estimation and closed processing. Thus, the pressure-sensitive trajectories T_u and T_v and the closed trajectories T_f and T_e can be acquired. The final trajectory T_c can also be generated. T_c is extruded to generate a layered cloud P_c, and a reconstruction algorithm is performed to generate a lightweight 3D hand-drawn mesh. If the hand drawing is completed, the algorithm is finished; otherwise, the stroke is further monitored.

A. GENERATION OF THE PRESSURE-SENSITIVE TRAJECTORY

The design intention of the hand-drawn designer can be accurately reflected by pressure-sensitive information; thus, it should be contained in the layered point cloud. The simplified trajectory can be formulated as follows:

$$T_s = \{S_k(x_k, y_k, t_k, p_k), k = 1, 2, 3, \cdots, N\} \tag{5}$$

Pressure-sensitive information p_k is contained in T_S, and the direct visualization of p_k is the line width in the trajectory. Thus, p_k should be converted to a pressure-sensitive circle with S_k as the center and r_k^p as the equivalent pressure-sensitive radius to realize the mapping from the pressure-sensing information to the line width. The equivalent pressure-sensitive radius can be formulated as

$$r_k^p = E\chi p_k \tag{6}$$

where E represents the conversion constant, and the value is related to the projection transformation of the PC screen. χ represents the pressure adjustment ratio, $\chi \in [0, 1]$. The pressure-sensitive circle is shown in Fig. 6 (a). Drawing the pressure-sensitive circle C_k^p of the arbitrary point S_k in T_s, two pressure-sensitive points, $S_{k,u}$ and $S_{k,v}$, can be defined on C_k^p along the normal of the velocity of S_k. All the pressure-sensitive points on T_s should be defined such that the pressure-sensitive trajectories of T_u and T_v can be generated, as shown in Fig. 6 (b). T_u and T_v are formulated as follows:

$$\begin{cases} T_u = \{S_{k,u}(x_{k,u}, y_{k,u}), k = 1, 2, 3, \cdots, N\} \\ T_v = \{S_{k,v}(x_{k,v}, y_{k,v}), k = 1, 2, 3, \cdots, N\} \end{cases} \tag{7}$$

If the coordinate value of an arbitrary point in T_u or T_v can be calculated by the difference algorithm in the cubic β-spline curve or cubic NURBS curve, the result is more accurate, but the computational complexity is higher [46]. Thus, a polyline

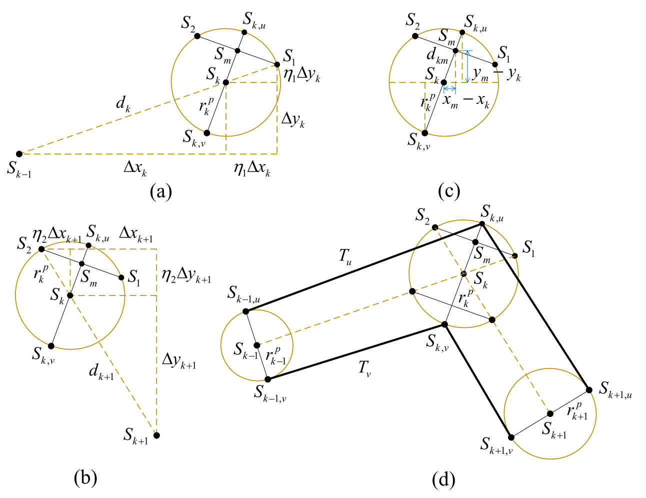

FIGURE 7. Specific steps of pressure-sensitive trajectory estimation using the polyline method.

method is executed to estimate the pressure-sensitive trajectory, which is shown in Fig. 7. Let the coordinates of three consecutive points in T_s be S_{k-1} (x_{k-1}, y_{k-1}), S_k (x_k, y_k), and S_{k+1} (x_{k+1}, y_{k+1}). The distances between two adjacent points are formulated as follows:

$$\begin{cases} d_k = \sqrt{(x_k - x_{k-1})^2 + (y_k - y_{k-1})^2} \\ \quad = \sqrt{\Delta x_k^2 + \Delta y_k^2} \\ d_{k+1} = \sqrt{(x_{k+1} - x_k)^2 + (y_{k+1} - y_k)^2} \\ \quad = \sqrt{\Delta x_{k+1}{}^2 + \Delta y_{k+1}{}^2} \end{cases} \tag{8}$$

Let S_1 (x_{S1}, y_{S1}), S_2 (x_{S2}, y_{S2}) be the points on the pressure-sensitive circle C_k^p. These points can be generated along the vectors $\mathbf{S}_{k-1}\mathbf{S}_k$ and $\mathbf{S}_k\mathbf{S}_{k+1}$ from S_k, as shown in Fig. 7 (a) and Fig. 7 (b), respectively. S_1 and S_2 are formulated as

$$\begin{cases} x_{S1} = x_k + \eta_1 \Delta x_k \\ y_{S1} = y_k + \eta_1 \Delta y_k \end{cases} \quad \begin{cases} x_{S2} = x_k + \eta_2 \Delta x_k \\ y_{S2} = y_k + \eta_2 \Delta y_k \end{cases} \tag{9}$$

where $\eta_1 = r_k^p / d_k$ and $\eta_2 = r_k^p / d_{k+1}$. The midpoint S_m (x_m, y_m) of the line segment S_1S_2 can be calculated, as shown in Fig. 7 (c), which is formulated as follows:

$$\begin{cases} x_m = x_k + 0.5\,(\eta_1 \Delta x_k + \eta_2 \Delta x_{k+1}) \\ y_m = y_k + 0.5\,(\eta_1 \Delta y_k + \eta_2 \Delta y_{k+1}) \end{cases} \tag{10}$$

Let the distance d_{km} from S_k to S_m be

$$d_{km} = \frac{1}{2}\sqrt{(\eta_1 \Delta x_k + \eta_2 \Delta x_{k+1})^2 + (\eta_1 \Delta y_k + \eta_2 \Delta y_{k+1})^2} \tag{11}$$

Thus, $S_{k,u}$ $(x_{k,u}, y_{k,u})$ and $S_{k,v}$ $(x_{k,v}, y_{k,v})$ can be formulated as

$$\begin{cases} x_{k,u} = x_k + \eta_3\,(x_m - x_k) \\ y_{k,u} = y_k + \eta_3\,(y_m - y_k) \end{cases} \quad \begin{cases} x_{k,v} = x_k + \eta_3\,(x_k - x_m) \\ y_{k,v} = y_k + \eta_3\,(y_k - y_m) \end{cases} \tag{12}$$

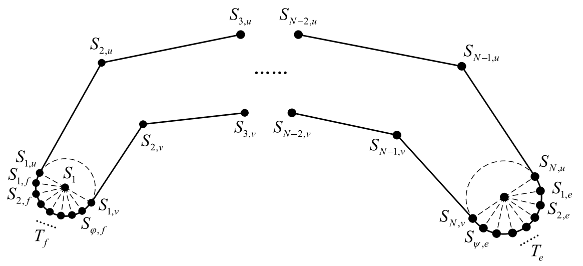

FIGURE 8. Closed treatment of trajectory sketch.

where η_3 is formulated as follows:

$$\begin{aligned} \eta_3 &= \frac{r_k^p}{d_{km}} \\ &= \frac{2r_k^p}{\sqrt{(\eta_1 \Delta x_k + \eta_2 \Delta x_{k+1})^2 + (\eta_1 \Delta y_k + \eta_2 \Delta y_{k+1})^2}} \end{aligned} \tag{13}$$

From Equation (10), $S_{k,u}$ $(x_{k,u}, y_{k,u})$ and $S_{k,v}$ $(x_{k,v}, y_{k,v})$ can also be formulated as

$$\begin{cases} x_{k,u} = x_k + \eta_4 r_k^p \\ y_{k,u} = y_k + \eta_5 r_k^p \end{cases} \quad \begin{cases} x_{k,v} = x_k - \eta_4 r_k^p \\ y_{k,v} = y_k - \eta_5 r_k^p \end{cases} \tag{14}$$

where η_4 and η_5 are formulated as follows:

$$\begin{cases} \eta_4 = \dfrac{\eta_1 \Delta x_k + \eta_2 \Delta x_{k+1}}{\sqrt{(\eta_1 \Delta x_k + \eta_2 \Delta x_{k+1})^2 + (\eta_1 \Delta y_k + \eta_2 \Delta y_{k+1})^2}} \\ \quad = \dfrac{\eta_1 \Delta x_k + \eta_2 \Delta x_{k+1}}{2d_{km}} \\ \eta_5 = \dfrac{\eta_1 \Delta y_k + \eta_2 \Delta y_{k+1}}{\sqrt{(\eta_1 \Delta x_k + \eta_2 \Delta x_{k+1})^2 + (\eta_1 \Delta y_k + \eta_2 \Delta y_{k+1})^2}} \\ \quad = \dfrac{\eta_1 \Delta y_k + \eta_2 \Delta y_{k+1}}{2d_{km}} \end{cases} \tag{15}$$

The coordinates of each point on T_u and T_v can be calculated according to the previously mentioned process, which is shown in Fig. 7 (d).

B. GENERATION OF THE LAYERED POINT CLOUD AND MESH RECONSTRUCTION

A single stroke nonclosed sketch can be constructed from the two pressure-sensitive trajectories T_u and T_v, which are not connected. Thus, a closed treatment should be performed. The pressure-sensitive circles C_1^p and C_N^p, which correspond to the start point S_1 and the end point S_N of the simplified trajectory T_s, are retained. The portion outside the pressure-sensitive trajectory should be retained. Thus, the trajectories T_f and T_e are generated, as shown in Fig. 8.
where φ and ψ represent the point numbers in T_f and T_e, respectively. The empirical formulas $\varphi = Ir_1^p$ and $\psi = Ir_N^p$ are given, where I represents the conversion constant, the value of which is related to the projection transformation

of the PC screen. All these points in T_f and T_e are evenly distributed around C_1^p and C_N^p, respectively, and the coordinates can be calculated by their parametric equations. T_f and T_e are formulated as follows:

$$\begin{cases} T_f = \left\{S_{\varepsilon,f}\left(x_{\varepsilon,f}, y_{\varepsilon,f}\right), \varepsilon = 1, 2, 3, \cdots, \varphi\right\} \\ T_e = \left\{S_{\tau,e}\left(x_{\tau,e}, y_{\tau,e}\right), \tau = 1, 2, 3, \cdots, \psi\right\} \end{cases} \tag{16}$$

Thus, the eventual trajectory $T_c = \{T_f, T_u, T_v, T_e\}$ can be generated. The layered point cloud can also be generated by copying T_c along the extruding direction with step d_s. Let T_c be on the datum $z = 0$; then, the extruding operation can be formulated as

$$z = z + (-1)^n \cdot \xi \cdot d_s, \quad \sigma \in N, \ n = 1, 2 \tag{17}$$

where d_s represents the extruding step, ξ represents the number of translation copy layers and the maximum value is $\xi_m = \left[D/d_S\right]$, where D represents the total extruding length. When $n = 1$, the extruding direction is negative along the Z axis. When $n = 2$, the extruding direction is positive along the Z axis. The layered 3D point cloud P_c that is obtained after extrusion can be formulated as

$$P_c = \left\{P_c\left(T_f\right), P_c\left(T_u\right), P_c\left(T_v\right), P_c\left(T_e\right)\right\} \tag{18}$$

where $P_c(T_f)$, $P_c(T_u)$, $P_c(T_v)$ and $P_c(T_e)$ represent the layered point clouds after extruding T_f, T_u, T_v and T_e, respectively, which are formulated as follows:

$$P_c\left(T_f\right) = \begin{Bmatrix} S_{1,f}\left(x_{1,f}, y_{1,f}, 0\right) & \cdots & S_{\varphi,f}\left(x_{\varphi,f}, y_{\varphi,f}, 0\right) \\ S_{1,f}\left(x_{1,f}, y_{1,f}, d_s\right) & \cdots & S_{\varphi,f}\left(x_{\varphi,f}, y_{\varphi,f}, d_s\right) \\ \vdots & \ddots & \vdots \\ S_{1,f}\left(x_{1,f}, y_{1,f}, \sigma_m d_s\right) & \cdots & S_{\varphi,f}\left(x_{\varphi,f}, y_{\varphi,f}, \sigma_m d_s\right) \end{Bmatrix} \tag{19}$$

$$P_c\left(T_u\right) = \begin{Bmatrix} S_{1,u}\left(x_{1,u}, y_{1,u}, 0\right) & \cdots & S_{N,u}\left(x_{N,u}, y_{N,u}, 0\right) \\ S_{1,u}\left(x_{1,u}, y_{1,u}, d_s\right) & \cdots & S_{N,u}\left(x_{N,u}, y_{N,u}, d_s\right) \\ \vdots & \ddots & \vdots \\ S_{1,u}\left(x_{1,u}, y_{1,u}, \sigma_m d_s\right) & \cdots & S_{N,u}\left(x_{N,u}, y_{N,u}, \sigma_m d_s\right) \end{Bmatrix} \tag{20}$$

$$P_c\left(T_v\right) = \begin{Bmatrix} S_{1,v}\left(x_{1,v}, y_{1,v}, 0\right) & \cdots & S_{N,v}\left(x_{N,v}, y_{N,v}, 0\right) \\ S_{1,v}\left(x_{1,v}, y_{1,v}, d_s\right) & \cdots & S_{N,v}\left(x_{N,v}, y_{N,v}, d_s\right) \\ \vdots & \ddots & \vdots \\ S_{1,v}\left(x_{1,v}, y_{1,v}, \sigma_m d_s\right) & \cdots & S_{N,v}\left(x_{N,v}, y_{N,v}, \sigma_m d_s\right) \end{Bmatrix} \tag{21}$$

$$P_c\left(T_e\right) = \begin{Bmatrix} S_{1,e}\left(x_{1,e}, y_{1,e}, 0\right) & \cdots & S_{\psi,e}\left(x_{\psi,e}, y_{\psi,e}, 0\right) \\ S_{1,e}\left(x_{1,e}, y_{1,e}, d_s\right) & \cdots & S_{\psi,e}\left(x_{\psi,e}, y_{\psi,e}, d_s\right) \\ \vdots & \ddots & \vdots \\ S_{1,e}\left(x_{1,e}, y_{1,e}, \sigma_m d_s\right) & \cdots & S_{\psi,e}\left(x_{\psi,e}, y_{\psi,e}, \sigma_m d_s\right) \end{Bmatrix} \tag{22}$$

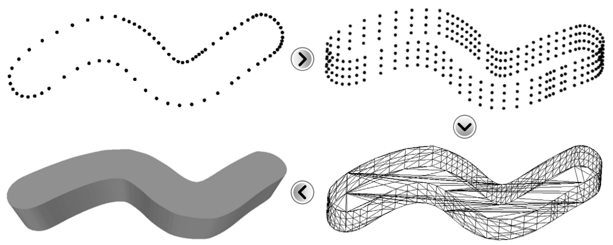

FIGURE 9. Process of hand-drawn trajectory mesh reconstruction based on the screened Poisson surface reconstruction algorithm.

A screened Poisson surface reconstruction algorithm was performed on the layered 3D point cloud [47]. Thus, the hand-drawn lightweight 3D stroke mesh can be generated; the process is shown in Fig. 9.

V. EXPERIMENT AND ANALYSIS

A. SYSTEM FRAMEWORK AND SERVICE CONFIGURATION

To verify the feasibility of the method proposed in this paper, an experimental platform was developed and used as the testing environment, which can be treated as a simple data-driven serverless Web application. In this serverless environment, the cloud provider dynamically manages the allocation of resources whereas the developers purely focus on their applications. The overall framework of this experimental platform is shown in Fig. 10. The sketch drawing step is executed on the client, which is responsible for collecting the original hand-drawn trajectory, real-time resampling, and uploading the resampled trajectory to the specific data simplification proxy via socket communication [48]. The simplification is based on the sliding window mechanism, and both the angle and chord height are considered. The final simplification trajectory is downloaded to the Web browser via HTTP communication, and pressure-sensitive mesh reconstruction is performed to generate the lightweight 3D stroke mesh.

The serverless Web application mainly address the Web distributed scenarios. It is essential to offer a consistent user experience across different connection types because of the cross-platform characteristic in the Web environment. Also, it is critical to simplify the development process of the data-driven serverless Web application [49]. Thus, we use the method named UMLPMSC (UML Profile for Multi-Cloud Service Configuration) proposed in reference [50] to model the service configuration design requirements at high abstraction level, which is shown in Fig. 11. This system performs four major functions: RunSlidingWindow, RunJudgment, GetAngleDeviation and GetChordHeight. Stereotype handler is mapped to each function that is called to be execution.

To evaluate the effectiveness of our approach, two students (student A and student B) from Software Engineering department with appropriate programming background were selected. Before assigning the development task, both students have undergone a training phase. During the training period, student A learned the concepts of UMLPMSC while

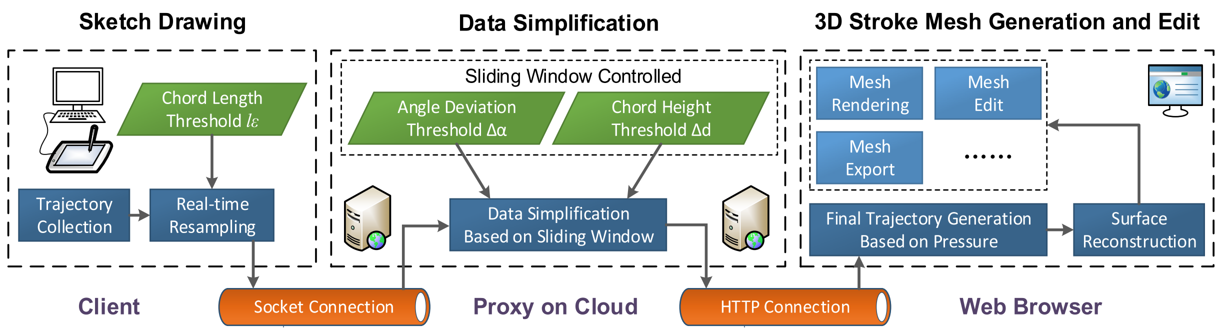
Sketch Drawing
Data Simplification
3D Stroke Mesh Generation and Edit
Chord Length Threshold lε
Trajectory Collection
Real-time Resampling
Sliding Window Controlled
Angle Deviation Threshold Δα
Chord Height Threshold Δd
Data Simplification Based on Sliding Window
Mesh Rendering
Mesh Edit
Mesh Export
......
Final Trajectory Generation Based on Pressure
Surface Reconstruction
Client
Socket Connection
Proxy on Cloud
HTTP Connection
Web Browser

FIGURE 10. Overall framework of the experimental platform.

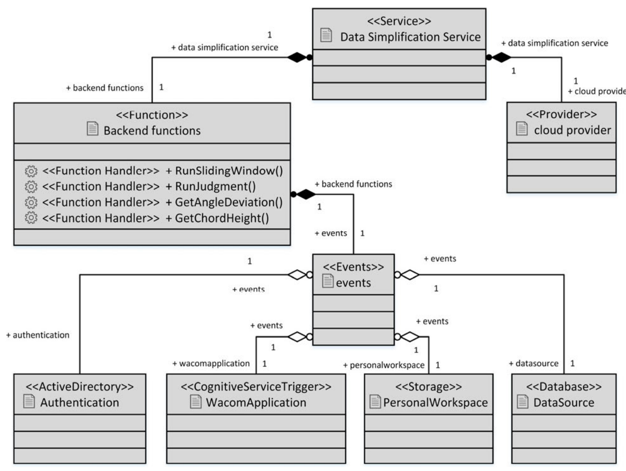
<<Service>> Data Simplification Service
+ data simplification service
+ backend functions
+ cloud provider
<<Function>> Backend functions
<<Function Handler>> + RunSlidingWindow()
<<Function Handler>> + RunJudgment()
<<Function Handler>> + GetAngleDeviation()
<<Function Handler>> + GetChordHeight()
<<Provider>> cloud provider
<<Events>> events
+ events
+ authentication
+ wacomapplication
+ personalworkspace
+ datasource
<<ActiveDirectory>> Authentication
<<CognitiveServiceTrigger>> WacomApplication
<<Storage>> PersonalWorkspace
<<Database>> DataSource

FIGURE 11. System framework service configuration on cloud.

student B did not learn. During the development process, student A created UML models for the assigned case study and applied the UMLPMSC concepts. On the other hand, student B developed the same case study through the conventional programming approach at lower level of implementation.

By summing up all, student A has taken around 6 days for the development of a complete case study (including the training time), while student B has taken around 10 days for the development of same case study. It further strengthens our claim that the proposed approach simplifies the development of Web application in a serverless cloud environment.

B. ESTABLISHMENT OF EXPERIMENTAL PLATFORM

To simulate the high concurrency of the Web environment and ensure a reasonable sense of interactive experience for the testers, this experimental platform has certain requirements for the computer hardware environment, software environment and tablet configuration:

1) HARDWARE ENVIRONMENT

There are certain hardware requirements for the cloud server and client in this experimental platform, which are shown in Table 1.

TABLE 1. Hardware environment of the cloud server and the client.

Hardware	PC	Cloud Server
CPU	Intel(R) i5-7200U 2.5 GHz	Intel Xeon-E5 2620V4 2.1 GHz
GPU	NVIDIA GeForce 940MX 1GB	NVIDIA GeForce GT720M 1GB
RAM	DDR4 2100 8GB	DDR4 RECC 32GB
Hard Disk	SSD 128GB	SAS 600G 15000RPM
Monitor	14 inch, 1920×1080	14 inch, 1920×1080

TABLE 2. Tablet configuration.

Indexes	Read Speed	Read Resolution	Pressure Level	Handwriting Accuracy
Value	150 points/sec	2540 LPI	1024	0.25 mm

2) SOFTWARE ENVIRONMENT

For the front-end software environment, the generation and rendering of the 3D stroke mesh in the Web environment is realized based on WebGL. For the back-end software environment, Microsoft Visual Studio 2013 and JetBrains WebStorm 2016 are employed as the Integrated Development Environment (IDE). For the database, the Microsoft SQL Server 2008 R2 is selected to store user information and 3D stroke mesh data. For data communication, the socket.io module in Node.js is selected to realize data transmission in the Web environment based on WebSocket.

3) TABLET CONFIGURATION

The hand-drawn trajectory data are acquired by the Wacom Bamboo CTL-671/K0-F tablet; the configuration is shown in Table 2.

4) EXPERIMENTAL PARAMETERS

We use the following parameters, which are shown in Table 3, to initialize the experiment. The values of all these parameters are determined according to experience after several operations.

C. TRAJECTORY SIMPLIFICATION EXPERIMENTS

The hand-drawn trajectories in this experiment constitute a set of classic contour curves proposed by Teh and Chin *et al.* [51], Prasad [52], which are employed to verify the

TABLE 3. Experimental parameters.

Indexes	λ	φ	$\Delta\alpha$	Δd	E	χ	W_S	$\Delta\sigma^2$
Value	1.0	0.5	15°	0.74	1	0.5	3	1500

TABLE 4. Definition of the stroke speed.

Range of the average speed	$\bar{v} \leq 250$	$250 < \bar{v} \leq 350$	$350 < \bar{v} \leq 450$	$\bar{v} > 450$
Strokes speed	200 px/sec	300 px/sec	400 px/sec	500 px/sec

simplification of the digital curve and the effect of the feature point extraction algorithm: Figure-8; Multiple semicircles; Chromosome-shaped and Leaf-shaped.

For the hand-drawn experiment, how to control for the variable "stroke speed" has an important role during the experimental process. The instantaneous speed of the stroke depends on many factors, including personal habits and changes in the curvature of the graph. When drawing the same trajectory, the instantaneous speed of each sampling point is different. Therefore, with the premise of ensuring that the experimenter draws the pattern at a uniform speed, we applied the average speed of a complete drawing of a trajectory $\bar{v}$ as the basis for the normalized strokes speed, which is formulated as

$$\bar{v} = \frac{N}{T} \tag{23}$$

where N represents the number of points from a whole hand-drawn curve, and T represents the time required to draw a curve. According to the calculated average speed $\bar{v}$, we divide the stroke speed into 200 px/sec, 300 px/sec, 400 px/sec and 500 px/sec; the corresponding relationship is shown in Table 4.

If we were to collect, compare and analyze the hand-drawn trajectory data collected by the experimenter each time, such as the best or worst cases, the sparseness of the sampled data points would vary greatly due to the influence of the experimenter's different stroke speeds and would objectively produce deviations in the performance comparison of the different simplification algorithms. To ensure the reliability of the experimental data, each of the following sets of data was obtained by averaging 10 experiments.

We have compared the proposed method with the latest five trajectory compression methods. Two of these methods are the batch compression method which contains the BQS [15], and the improved DP (IDP) algorithm [16]. Another three methods are the online compression method, which contains the online DPTS (ODPTS) [19], the improved sliding window trajectory compression (ISWTC) algorithm [20] and a parallel online trajectory compression approach (PSQUISH-E) [21]. Four different stroke speeds from 200 px/sec to 500 px/sec were applied to draw the four classic contour curves in these experiments. The simplified trajectories of the four hand-drawn classic contour curves using different trajectory compression methods are shown in Fig. 12. Two aspects are shown in Fig. 12. The final collected trajectory shows a simplified effect, which proves that the algorithm that we proposed achieves the simplification of the collected original trajectory. However, with the increase in the stroke speed, the effect of simplification becomes increasingly obvious, which proves that our proposed algorithm shows adaptability to the trajectory drawn by the different stroke speeds.

To quantitatively evaluate the simplification algorithm, we employed only hand-drawn trajectory data with a stroke speed of 300 px/sec as the analysis object. We believe that the stroke speed of 300 px/sec is closest to the normal speed of hand- drawing. The experimental metrics shown in Table 5 were used to measure the simplification effect, which includes the compression ratio (*CR*), local maximum error ($E_{\max}$), total length error (E_l), mean stochastic error distance (*SED*) [53], [54], and whole processing time t. Note that the metric t includes the whole process from trajectory data acquisition to trajectory data upload, followed by the trajectory data download and visualization in the Web browser. We show the results of the data simplification experiments for four hand-drawn trajectories using the 6 different methods shown in Table 6.

As shown in Table 6, by calculating the CR of the four types of trajectories using different simplification methods, the data reduction effects of these 6 methods can be analyzed. For the same trajectory, the number of trajectory vertices after simplification at different stroke speeds is obviously reduced. However, our method performs best in terms of data reduction because the *CRs* of these 4 types of classic contour curves are 85.1%, 92.4%, 95.7% and 95.2%, which are obviously higher than those of the other algorithms.

As shown in Table 6, by calculating the mean SED, local maximum error ($E_{\max}$) and total length error (E_l), the similarity between the reduced trajectory and the original trajectory can be analyzed. For the metric of *SED*, our method yields a value of 0.073 on average, which is obviously lower than those of the other five algorithms (0.087, 0.094, 0.118, 0.116 and 0.095).

As shown in Table 6, by counting the whole processing time t, we can implicitly understand the time complexity of these six different simplification algorithms. The average t of our algorithm is 11.0 s, which is lower than those of the other five algorithms (14.9 s, 14.4 s, 13.4 s, 12.5 s and 11.7 s). It can be seen from these experimental results that our algorithm performs less processing time than other five algorithms, which shows lower computational complexity in the Web environment.

In addition, with different hand-drawn speeds, the average local maximum error of the four hand-drawn patterns is 1.055, and the average total length error is 4.38%, which are all lower than the testing results of 1.5 and 5.0%, respectively, from the reference [52]. The average $E_{\max}$ of our method is also lower than those of the other five algorithms

TABLE 5. Experimental metrics of trajectory simplification.

Indexes	Formula	Interpretation
CR	$CR=\left(1-C/N\right)\times 100\%$	Ratio of the trajectory data compressed during reduction, where N represents the number of vertices in the original trajectory and C represents the number of vertices in the simplified trajectory.
$E_{\max}$	$E_{\max}=\max\limits_{1\le i\le n} e_i$	e_i represents the vertical Euclidean distance between a vertex in the original hand-drawn trajectory and the corresponding segment of the simplified trajectory.
E_l	$E_l={L_n}/{L_c}\times 100\%$	L_n and L_c represent the total length of the original hand-drawn trajectory and simplified trajectory data, respectively.
SED	$SED=\int_{-\infty}^{+\infty}\left\|T_n\left(e\right)-T_c\left(e\right)\right\|de$	$T_n\,(e)$ represents the cumulative distribution function of the forecast error e between the original hand-drawn contour curve and the standard contour curve. $T_c\,(e)$ represents the cumulative distribution function of the forecast error e between the simplified trajectory and the standard contour curve.
t	None	Whole processing time of different simplification algorithm

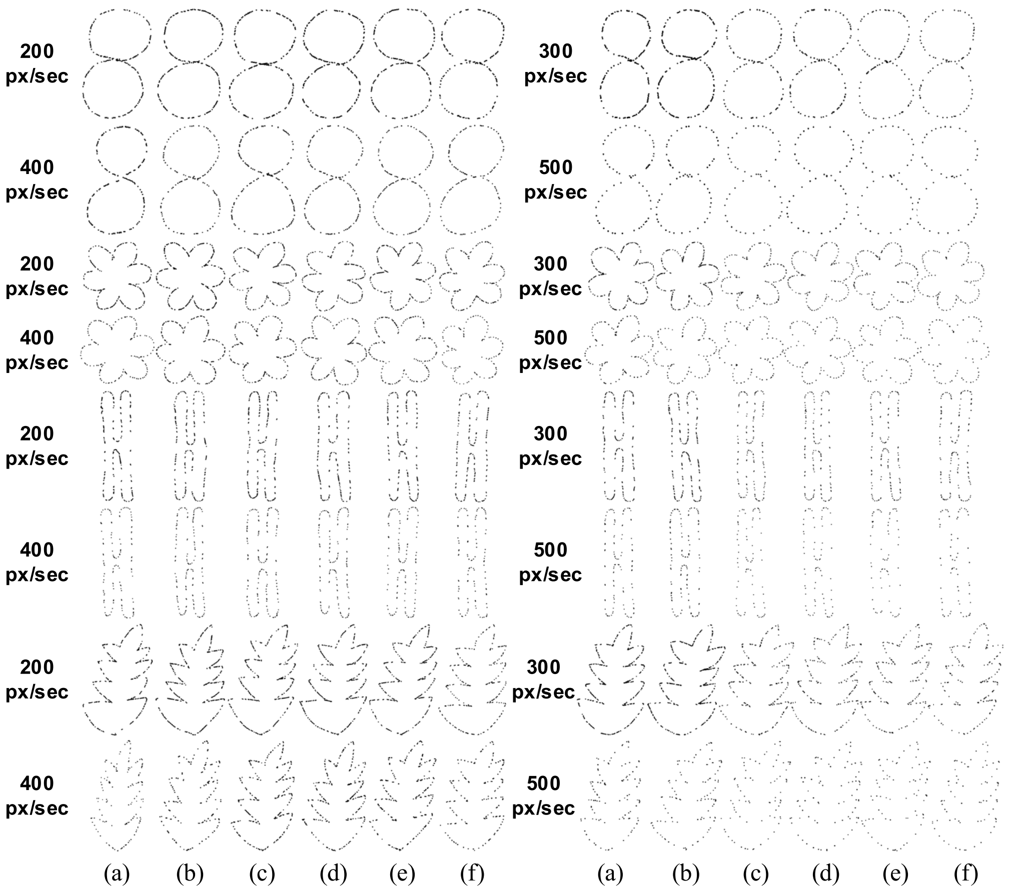

FIGURE 12. Simplified trajectories of the four hand-drawn classic contour curves using different trajectory compression methods and different stroke speeds: (a) BQS; (b) IDP; (c) ISWTC; (d) PSQUISH-E; (e) ODPTS; and (f) our method.

(1.33, 1.33, 1.70, 1.54 and 1.23), and the average E_l of our method is lower than those of the other five algorithms (5.17%, 5.04%, 6.40%, 6.28% and 5.30%). Therefore, the ACUTS proposed in this paper can ensure not only the light weight of hand-drawn trajectory data to meet the requirements of high-efficiency data transmission

TABLE 6. Results of the data reduction experiments for the four hand-drawn trajectories using six different methods.

Graphic Type	Method	N	C	CR	SED	$E_{\max}$/px	E_l/%	t/sec
Figure-8	BQS	632	107	83.1%	0.07	0.96	1.84%	9.9
	IDP	616	112	81.8%	0.068	1.06	1.47%	9.4
	ISWTC	679	146	78.5%	0.074	1.32	1.98%	8.9
	PSQUISH-E	641	132	79.4%	0.072	1.21	1.92%	7.7
	ODPTS	644	117	81.8%	0.068	0.96	1.90%	7.1
	Ours	692	103	85.1%	0.062	0.84	1.54%	6.3
Multiple semicircles	BQS	1593	132	91.7%	0.09	1.32	6.54%	16.5
	IDP	1508	143	90.5%	0.093	1.24	5.79%	16.1
	ISWTC	1061	137	87.1%	0.101	1.65	9.12%	15.4
	PSQUISH-E	1198	189	84.2%	0.942	1.57	8.95%	13.8
	ODPTS	1222	147	88.0%	0.086	1.28	6.89%	13.1
	Ours	1376	104	92.4%	0.081	1.05	4.71%	12.3
Chromosome-shaped	BQS	2336	136	94.2%	0.074	1.32	3.79%	13.8
	IDP	2228	148	93.4%	0.105	1.17	3.95%	13.2
	ISWTC	1976	193	90.2%	0.141	1.67	4.62%	11.5
	PSQUISH-E	2336	163	93.0%	0.140	1.49	4.59%	10.9
	ODPTS	2373	121	94.9%	0.084	1.18	4.01%	10.3
	Ours	2498	107	95.7%	0.061	0.95	3.32%	9.8
Leaf-shaped	BQS	2536	157	93.8%	0.112	1.73	8.52%	19.4
	IDP	2579	169	93.4%	0.109	1.83	8.93%	18.9
	ISWTC	2680	228	91.5%	0.157	2.16	9.89%	17.8
	PSQUISH-E	2154	221	89.7%	0.152	1.89	9.64%	17.4
	ODPTS	2322	189	91.9%	0.140	1.59	8.41%	16.3
	Ours	2616	126	95.2%	0.090	1.38	7.95%	15.6

and low computational complexity of data processing in the Web environment but also the high accuracy of the simplified trajectory data.

D. LIGHTWEIGHT 3D RECONSTRUCTION EXPERIMENTS

The width of the trajectory 3D stroke mesh can be changed using the pressure-sensitive information with the lightweight 3D reconstruction method proposed in this paper. The pressure versus time curves of the four types of trajectories are shown in Fig. 13. The abscissa represents the hand-drawing time, and the ordinate represents the pressure level instead of the number of vertices collected.

As shown in Fig. 13, with an increase in the hand-drawn speed, the pressure declines, which is in accordance with the normal hand-drawn result. The pressure could be more stable by drawing the simple trajectories, such as Figure-8 and Chromosome-shaped trajectories, while the pressure could be changed drastically and periodically by drawing the complex trajectories, such as Multiple semicircle and Leaf-shaped.

To quantitatively evaluate the proposed lightweight 3D reconstruction algorithm, we also applied the hand-drawn trajectory data as the analysis object. We tested various parameters for the reconstructed 3D stroke meshes at different stroke speeds from 200 px/sec to 500 px/sec and performed a quantitative analysis with 300 px/sec as a typical case for two main reasons. First, the stroke speed of 300 px/sec is closest to the actual hand-drawing speed, and after many experiments, it has been proven that this stroke speed can ensure the high fidelity of the reconstructed 3D stroke mesh. The 3D stroke mesh can be reconstructed in a relatively fast time. Second, although different stroke speeds will lead to different amounts of data of the 3D stroke meshes, their lightweight degrees are similar. The data that we tested confirmed this view.

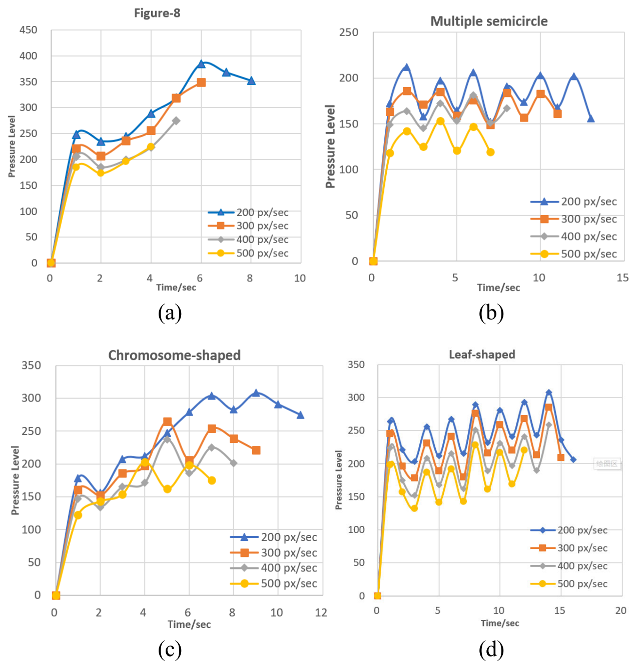

FIGURE 13. Pressure versus time curves of the four types of trajectories: (a) Figure-8; (b) Multiple semicircles; (c) Chromosome-shaped; (d) Leaf-shaped.

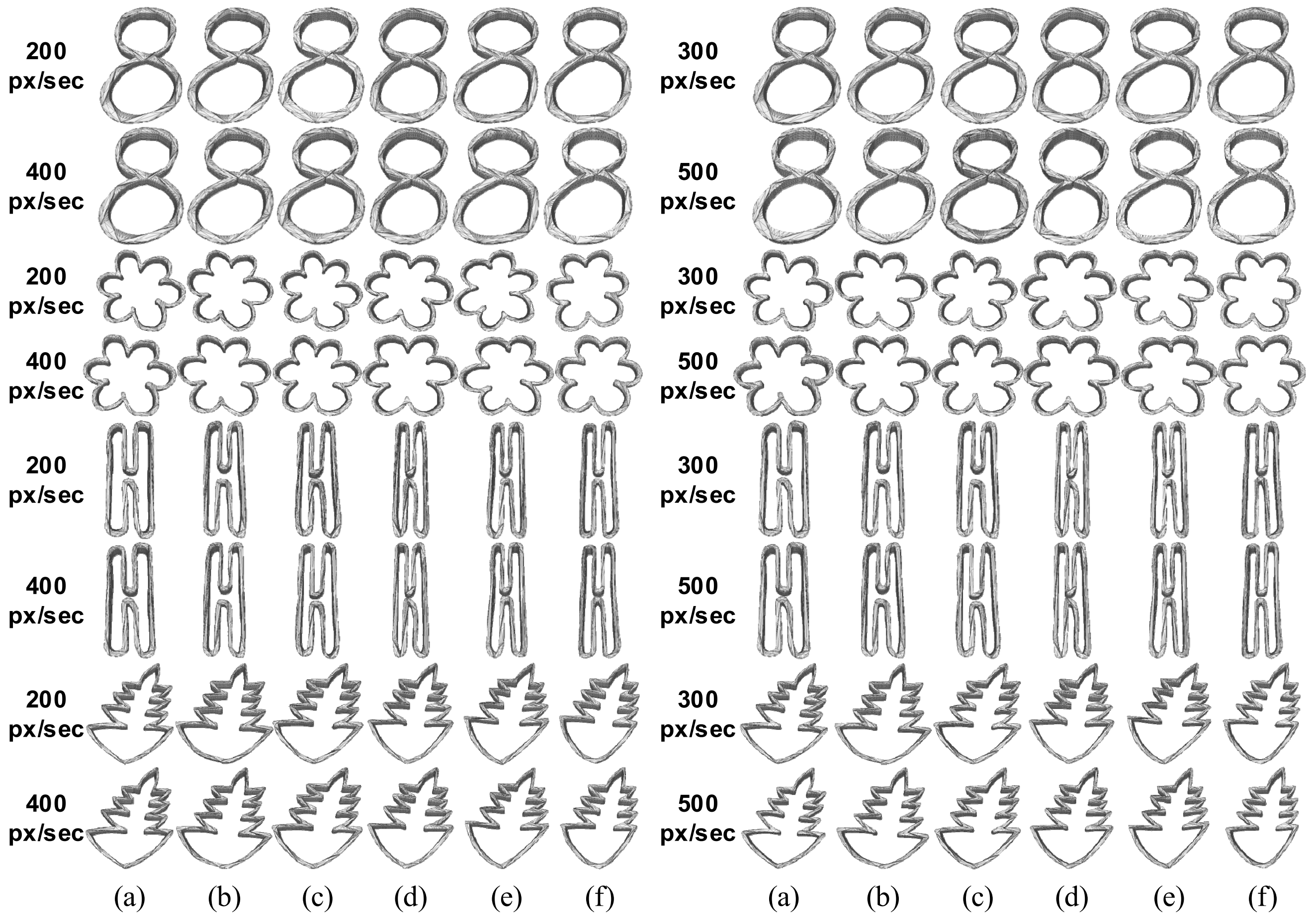

FIGURE 14. Reconstructed 3D stroke meshes of the four hand-drawn classic contour curves using different trajectory simplification algorithms and different stroke speeds: (a) BQS; (b) IDP; (c) ISWTC; (d) PSQUISH-E; (e) ODPTS; and (f) our method.

We utilized the original stroke mesh that was reconstructed from the original hand-drawn trajectory data as the reference mesh to evaluate the lightweight degree and quality of the reconstructed mesh. We need only to specify the specific width of the trajectory 3D stroke mesh; the equivalent pressure-sensitive radius is automatically set to a specified constant, and the width of the original stroke mesh will be constant. We hope to show that the 3D stroke mesh with pressure-sensitive information generated by our proposed method is more realistic and closer to the effect of real hand painting based on the comparative experimental results.

The experimental metrics shown in Table 7 were used to measure the reduction effect of the reconstructed 3D stroke mesh, which includes the reduction percentages of the number of vertices ($\eta_{N'}$), number of triangular facets (η_F), data volume of the 3D stroke mesh (η_S), and 3D reconstruction time (η_T).

Considering that we did not study the 3D reconstruction algorithm, only the screened Poisson surface reconstruction algorithm [47] was employed for 3D reconstruction. We performed 3D reconstruction experiments on the hand-drawn trajectory data generated by the five different trajectory simplification algorithms [15], [16], [19]–[21] mentioned in the previous section. In addition, we treat the reconstructed 3D stroke mesh from BQS [15] as the original mesh. These reconstructed meshes can be treated as contrasting data to demonstrate the effectiveness of our method. The experimental results are shown in Table 8 and Fig. 14. The visualization effects of the 3D reconstruction are shown in Fig. 15.

As shown in Table 8 and Fig 14 (a), (b), (c), and (d), our method performs best in terms of the lightweight processing of 3D stroke meshes because the $\eta_{N'}$ values of these 4 types of classic contour curves are 69.1%, 69.3%, 65.3% and 64.5%, respectively; the η_F values of these 4 types of classic contour curves are 64.6%, 67.5%, 56.8% and 60.4%, respectively; the η_S values of these 4 types of classic contour curves are 70.3%, 68.5%, 62.9% and 57.7%, respectively; and the η_T values of these 4 types of classic contour curves are 64.9%, 67.0%, 60.6% and 63.2%, respectively. These four metrics achieve the maximum values among the six methods except BQS [15], which indicates that our 3D reconstruction algorithm is capable of reconstructing lightweight 3D stroke meshes with the least amount of data.

TABLE 7. Experimental metrics of lightweight 3D reconstruction.

Indexes	Formula	Interpretation
$\eta_{N'}$	$\eta_{N'} = \left(1 - {N'}/{N}\right) \times 100\%$	Reduction percentage of the number of vertices, where *N* represents the number of original mesh vertices by BQS [15], and *N'* represents the number of mesh vertices using the specific trajectory simplification algorithms.
η_F	$\eta_F = \left(1 - {F'}/{F}\right) \times 100\%$	Reduction percentages of the number of triangular facets, where *F* represents the number of original mesh facets by BQS [15], and *F'* represents the number of mesh facets using the specific trajectory simplification algorithms.
η_S	$\eta_S = \left(1 - {S'}/{S}\right) \times 100\%$	Reduction percentages of the data volume, where *S* represents the data volume of original mesh by BQS [15], and *S'* represents the data volume of mesh using the specific trajectory simplification algorithms.
η_T	$\eta_T = \left(1 - {T'}/{T}\right) \times 100\%$	Reduction percentages of the 3D reconstruction time, where *T* represents the reconstruction time of original mesh by BQS [15], and *T'* represents the reconstruction time of mesh using the specific trajectory simplification algorithms.

TABLE 8. Experimental results of lightweight 3D reconstruction.

Graphic Type	Method	*N'*	$\eta_{N'}$/%	*F*	η_F/%	*S*/MB	η_S/%	*T*/s	η_T/%
Figure-8	BQS	5789	0.0%	10090	0.0%	3.10	0.0%	2.683	0.0%
	IDP	2581	55.4%	5189	48.6%	1.27	59.0%	1.931	28.0%
	ISWTC	2326	59.8%	4678	53.6%	1.15	62.9%	1.189	55.7%
	PSQUISH-E	2564	55.7%	5185	48.6%	1.24	60.0%	1.960	26.9%
	ODPTS	2478	57.2%	4916	51.3%	1.19	61.6%	1.572	41.4%
	Ours	2333	59.7%	4670	53.7%	1.17	62.3%	1.056	60.6%
Multiple semicircles	BQS	7349	0.0%	13897	0.0%	3.59	0.0%	4.618	0.0%
	IDP	3421	53.4%	6831	50.8%	1.68	53.2%	2.793	39.5%
	ISWTC	3315	54.9%	6625	52.3%	1.62	54.9%	2.514	45.6%
	PSQUISH-E	3509	52.3%	7017	49.5%	1.69	52.9%	3.254	29.5%
	ODPTS	3461	52.9%	6928	50.1%	1.71	52.4%	3.163	31.5%
	Ours	2388	67.5%	4776	65.6%	1.19	66.9%	1.905	58.7%
Chromosome-shaped	BQS	6854	0.0%	11025	0.0%	3.21	0.0%	4.269	0.0%
	IDP	2513	63.3%	5017	54.5%	1.24	61.4%	1.898	55.5%
	ISWTC	2606	62.0%	5249	52.4%	1.29	59.8%	2.256	47.2%
	PSQUISH-E	2896	57.7%	5817	47.2%	1.36	57.6%	2.897	32.1%
	ODPTS	2609	61.9%	5240	52.5%	1.30	59.5%	2.165	49.3%
	Ours	2507	63.4%	5014	54.5%	1.25	61.1%	1.893	55.7%
Leaf-shaped	BQS	7785	0.0%	13949	0.0%	3.26	0.0%	6.216	0.0%
	IDP	2905	62.7%	5828	58.2%	1.43	56.1%	3.421	45.0%
	ISWTC	2879	63.0%	5771	58.6%	1.42	56.4%	2.916	53.1%
	PSQUISH-E	3231	58.5%	6459	53.7%	1.58	51.5%	4.758	23.5%
	ODPTS	3050	60.8%	6109	56.2%	1.54	52.8%	4.256	31.5%
	Ours	2944	62.2%	5888	57.8%	1.47	54.9%	3.976	36.0%

In addition, we show the trajectory strokes with the pressure-sensitive information and its 3D reconstruction effects, as well as the trajectory strokes without the pressure-sensitive information and its 3D reconstruction effects, which are shown in Fig 15. The trajectory with an inconsistent width can be generated from the pressure-sensitive information of different sampling points using the ACUTS algorithm and LRPDP proposed in this paper. Compared with the 3D stroke meshes with the same width, the 3D stroke meshes generated from the pressure-sensitive trajectory can not only simulate the real handwriting but also reduce the amount of data to some extent. The speed and quality of the online 3D rendering can be effectively improved in the Web environment by the method proposed in this paper.

VI. LIMITATION

The limitation of this study is the insufficient utilization of the pressure-sensitive information. For the hand-drawn pressure-sensitive trajectories, the pressure-sensitive information is so important that it can directly reflect the design intention of the painter. The full use of pressure-sensitive information can not only further reduce the amount

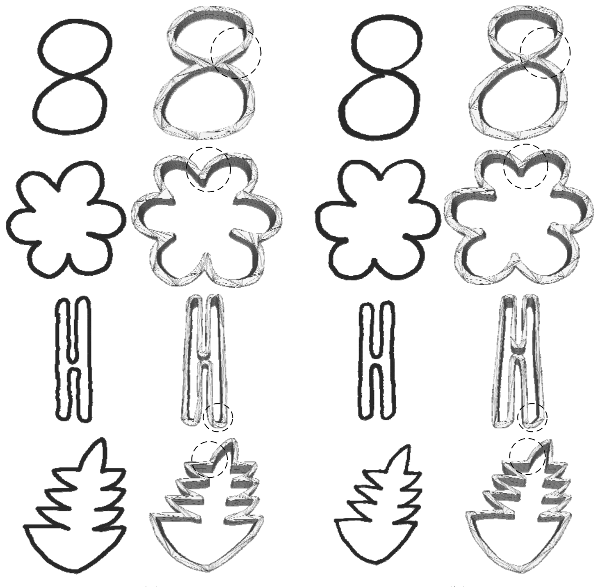

FIGURE 15. Visualization effects of the adaptive-width mapped from the stroke pressure with black dotted circle areas, which make the 3D stroke mesh more realistic: (a) original trajectory strokes and their corresponding 3D reconstructed meshes; (b) lightweight pressure-sensitive trajectory strokes and their corresponding 3D reconstructed meshes.

of 3D stroke mesh data to meet the smooth 3D rendering using WebGL but also render the reconstructed 3D stroke mesh similar to the real hand-drawn pattern. However, the pressure-sensitivity information was not fully utilized in this study, which is mainly reflected in the following two aspects.

(1) Pressure-sensitive information was not utilized enough in the simplification process of trajectory points. In this paper, the pressure values were utilized only to determine whether several points need to be removed in a sliding window. We calculated the variance of the pressure values to determine whether the points in the current sliding window are greatly affected by their pressure values. Note that we did not fully utilize the pressure-sensitivity information, which did not have a significant impact on the final experimental results. This issue will be a key area of our future work.

(2) Pressure-sensitive information should be considered to study a new metric, which can reveal the accuracy of the reconstructed 3D stroke mesh. In the experimental section, the 3D stroke mesh was reconstructed based on the collected pressure-sensitive trajectory data. However, we performed only a quantitative analysis of the lightweight degree of the 3D stroke mesh and did not perform a quantitative analysis of the fidelity and quality of the 3D stroke mesh based on relevant metrics that contain pressure values. Note that this approach will not substantially affect the contribution of this article. This issue will be another key area of our future work.

VII. CONCLUSION

This study is aimed to develop a lightweight processing method for hand-drawn pressure-sensitive trajectories that are oriented toward Web-based 3D modeling, in which three implications are summarized as follows: First, an angle-chord height united trajectory simplification algorithm (ACUTS) is proposed. In the resampling process, the original data are resampled in real time based on the chord length threshold. In the data simplification process, the resampled discrete points are simplified by comprehensively considering the angle deviation and chord height deviation with the sliding window mechanism, which can make the trajectory data achieve a high degree of accuracy and light weight. Second, the LRPDP is proposed. The pressure-sensitive information obtained by the tablet is mapped into adaptive-width hierarchical point cloud data to ensure that the generated hand-drawn trajectory 3D stroke mesh performs with light weight with inconsistent widths. This algorithm can reduce the amount of data while ensuring the visualization effect and satisfying the requirements of the lightweight data in the Web environment. A Web-based CAD software that supports digital tablet access, which provides a new way to realize hand-drawn 3D modeling in the Web environment, was developed.

In future work, the size of the sliding window should be dynamically adjusted according to the density of the trajectory vertices in the ACUTS, which is suitable for real-time drawing. The mesh reconstruction algorithm should be improved to adapt to the point cloud data of the special topology generated by the single-column trajectory data in the LRPDP. In addition, the pressure-sensitive information should be considered to study a new metric, which can measure the accuracy of the reconstructed 3D stroke mesh in consideration of the pressure-sensitive information. Some stochastic methods should be utilized for improving the performance of our algorithm and the quality of the collected data in future work, such as the Monte Carlo method.

REFERENCES

[1] K. Lupinetti, J.-P. Pernot, M. Monti, and F. Giannini, "Content-based CAD assembly model retrieval: Survey and future challenges," *Comput.-Aided Des.*, vol. 113, pp. 62–81, Aug. 2019, doi: 10.1016/j.cad.2019.03.005.

[2] Y. Wu, F. He, D. Zhang, and X. Li, "Service-oriented feature-based data exchange for cloud-based design and manufacturing," *IEEE Trans. Services Comput.*, vol. 11, no. 2, pp. 341–353, Mar. 2018, doi: 10.1109/TSC.2015.2501981.

[3] G. Andreadis, G. Fourtounis, and K.-D. Bouzakis, "Collaborative design in the era of cloud computing," *Adv. Eng. Softw.*, vol. 81, pp. 66–72, 2015, doi: 10.1016/j.advengsoft.2014. 11.002.

[4] L. Avila and M. Bailey, "A computer graphics back-to-school special," *IEEE Comput. Graph. Appl.*, vol. 36, no. 5, pp. 95–96, Sep. 2016, doi: 10.1109/MCG.2016.104.

[5] J. Qi, G. Jiang, G. Li, Y. Sun, and B. Tao, "Intelligent human-computer interaction based on surface EMG gesture recognition," *IEEE Access*, vol. 7, pp. 61378–61387, 2019, doi: 10.1109/access.2019.2914728.

[6] Q. Qi, Q. Huo, J. Wang, H. Sun, Y. Cao, and J. Liao, "Personalized sketch-based image retrieval by convolutional neural network and deep transfer learning," *IEEE Access*, vol. 7, pp. 16537–16549, 2019, doi: 10.1109/access.2019.2894351.

[7] Y. Li, H. Lei, S. Lin, and G. Luo, "A new sketch-based 3D model retrieval method by using composite features," *Multimedia Tools Appl.*, vol. 77, no. 2, pp. 2921–2944, Jan. 2018, doi: 10.1007/s11042-017-4446-y.

[8] A. Maroosi and H. K. Bizaki, "Multiple frequencies determination of sinusoidal real waveform by multiple sensors with low sampling rate," *IEEE Sensors J.*, vol. 17, no. 24, pp. 8404–8411, Dec. 2017, doi: 10.1109/jsen.2017.2765280.

[9] H.-Y. Cheng, C.-C. Yu, V. Gau, and C.-L. Lin, ‘‘Video-based signature verification and pen-grasping posture analysis for user-dependent identification authentication,’’ *IET Comput. Vis.*, vol. 6, no. 5, pp. 388–396, Sep. 2012, doi: 10.1049/iet-cvi.2010.0136.

[10] G. Orbay and L. Burak Kara, ‘‘Pencil-like sketch rendering of 3D scenes using trajectory planning and dynamic tracking,’’ *J. Vis. Lang. Comput.*, vol. 25, no. 4, pp. 481–493, Aug. 2014, doi: 10.1016/j.jvlc.2014.02.002.

[11] J. Jiang and X. Wang, ‘‘Review on trajectory data compression,’’ *J. East China Normal Univ., Natural Sci.*, vol. 5, pp. 61–76, 2015.

[12] D. H. Douglas and T. K. Peucker, ‘‘Algorithms for the reduction of the number of points required to represent a digitized line or its caricature,’’ *Cartographica, Int. J. Geographic Inf. Geovisualization*, vol. 10, no. 2, pp. 112–122, Dec. 1973.

[13] C. Long, R. C.-W. Wong, and H. V. Jagadish, ‘‘Direction-preserving trajectory simplification,’’ *Proc. VLDB Endowment*, vol. 6, no. 10, pp. 949–960, Aug. 2013, doi: 10.14778/2536206.2536221.

[14] H. Abbasi, M. Olyaee, and H. R. Ghafari, ‘‘Rectifying reverse polygonization of digital curves for dominant point detection,’’ *Int. J. Comput. Sci. Issues*, vol. 10, no. 3, p. 154, 2013.

[15] J. Liu, K. Zhao, P. Sommer, S. Shang, B. Kusy, and R. Jurdak, ‘‘Bounded quadrant system: Error-bounded trajectory compression on the go,’’ in *Proc. IEEE 31st Int. Conf. Data Eng.*, Apr. 2015, pp. 987–998, doi: 10.1109/ICDE.2015.7113350.

[16] L. Zhao and G. Shi, ‘‘A method for simplifying ship trajectory based on improved Douglas–Peucker algorithm,’’ *Ocean Eng.*, vol. 166, pp. 37–46, Oct. 2018, doi: 10.1016/j.oceaneng.2018.08.005.

[17] J. Muckell, J.-H. Hwang, V. Patil, C. T. Lawson, F. Ping, and S. S. Ravi, ‘‘SQUISH: An online approach for GPS trajectory compression,’’ in *Proc. 2nd Int. Conf. Comput. Geospatial Res. Appl. COM.Geo*, 2011, pp. 1–8.

[18] J. Muckell, P. W. Olsen, J.-H. Hwang, C. T. Lawson, and S. S. Ravi, ‘‘Compression of trajectory data: A comprehensive evaluation and new approach,’’ *GeoInformatica*, vol. 18, no. 3, pp. 435–460, Jul. 2014, doi: 10.1007/s10707-013-0184-0.

[19] Z. Deng, W. Han, L. Wang, R. Ranjan, A. Y. Zomaya, and W. Jie, ‘‘An efficient online direction-preserving compression approach for trajectory streaming data,’’ *Future Gener. Comput. Syst.*, vol. 68, pp. 150–162, Mar. 2017, doi: 10.1016/j.future.2016.09.019.

[20] M. Gao, G. Y. Shi, and W. F. Li, ‘‘Online compression algorithm of AIS trajectory data based on improved sliding window,’’ *J. Traffic Transp. Eng.*, vol. 18, no. 3, pp. 218–227, 2018.

[21] W. Han, Z. Deng, J. Chu, J. Zhu, P. Gao, and T. Shah, ‘‘A parallel online trajectory compression approach for supporting big data workflow,’’ *Computing*, vol. 100, no. 1, pp. 3–20, Jan. 2018, doi: 10.1007/s00607-017-0563-8.

[22] J. Luo, B. Wu, F. Shen, and Z. Wu, ‘‘Design and implementation of the handwriting device supporting pen-interaction,’’ *Yi Qi Yi Biao Xue Bao/Chin. J. Sci. Instrum.*, vol. 33, no. 9, pp. 2114–2124, 2012.

[23] D. Wang, Y. Zhang, C. Yao, J. Wu, H. Jiao, and M. Liu, ‘‘Toward force-based signature verification: A pen-type sensor and preliminary validation,’’ *IEEE Trans. Instrum. Meas.*, vol. 59, no. 4, pp. 752–762, Apr. 2010, doi: 10.1109/TIM.2009.2037871.

[24] H. Jiao, D. Wang, Y. Zhang, and L. Fang, ‘‘Signature verification using handwriting friction force,’’ *Zidonghua Xuebao/Acta Automatica Sinica*, vol. 37, no. 7, pp. 883–890, 2011.

[25] C. Guo, Z. X. Hou, G. Q. Yang, and S. Z. Zheng, ‘‘The simulation of the brush stroke based on force feedback technology,’’ *Math. Problems Eng.*, vol. 2015, Jan. 2015, Art. no. 164821, doi: 10.1155/2015/164821.

[26] K. Tamilarasi and S. Nithya Kalyani, ‘‘Design and implementation of deep learning strategy based smart signature verification system,’’ *Microprocessors Microsyst.*, vol. 77, Sep. 2020, Art. no. 103119, doi: 10.1016/j.micpro.2020.103119.

[27] J. Heckeroth, E. Kupferschmid, T. Dziedzic, N. Kalantzis, B. Geistová Čakovská, C. Fernandes, M. J. Branco, K. Axelsson Spjuth, A. Kerkhoff, P. Vaccarone, J. Zimmer, and P. Schmidt, ‘‘Features of digitally captured signatures vs. Pen and paper signatures: Similar or completely different?’’ *Forensic Sci. Int.*, vol. 318, Jan. 2021, Art. no. 110587, doi: 10.1016/j.forsciint.2020.110587.

[28] Y. Gong, C. Xuan, J. Lou, and Q. Li, ‘‘3D automotive sketching system based on pressure-sensitivity of digital pen,’’ *Nongye Jixie Xuebao/Trans. Chin. Soc. Agricult. Machinery*, vol. 46, no. 8, pp. 314–318 and 288, 2015, doi: 10.6041/j.issn.1000-1298.2015.08.043.

[29] F. Guo and C. Zhang, ‘‘Haptic data compression by curve fitting,’’ *Jisuanji Fuzhu Sheji Yu Tuxingxue Xuebao/J. Comput.-Aided Des. Comput. Graph.*, vol. 27, no. 3, pp. 508–513, 2015.

[30] M. Meng, Y. Ge, Z. Wu, P. Fang, and F. Shen, ‘‘Accuracy of a force sensitive digital tablet in determining the position of force application,’’ in *Proc. IEEE Int. Conf. Inf. Acquisition*, Jun. 2005, p. 5.

[31] J. Li, Q. Wu, Y. Ma, and H. Ding, ‘‘Dynamic signature verification method based on timing feature fusion,’’ *Jisuanji Yingyong Yanjiu/Appl. Res. Comput.*, vol. 37, no. 7, pp. 2032–2036, 2020, doi: 10.19734/j.issn.1001-3695.2019.01.0011.

[32] L. Donati, S. Cesano, and A. Prati, ‘‘A complete hand-drawn sketch vectorization framework,’’ *Multimedia Tools Appl.*, vol. 78, no. 14, pp. 19083–19113, Jul. 2019, doi: 10.1007/s11042-019-7311-3.

[33] C. Xuan, Q. Li, Y. Gong, and J. Lou, ‘‘Depth conversion and expression of 3D free-hand stroke based on pressure-sensitive digital pen,’’ *Nongye Jixie Xuebao/Trans. Chin. Soc. Agricult. Machinery*, vol. 47, no. 5, pp. 366–371, 2016, doi: 10.6041/j.issn.1000-1298.2016.05.050.

[34] D. Wang, Y. Zhang, and C. Yao, ‘‘Stroke-based modeling and haptic skill display for chinese calligraphy simulation system,’’ *Virtual Reality*, vol. 9, nos. 2–3, pp. 118–132, Jan. 2006, doi: 10.1007/s10055-005-0012-4.

[35] C. Xing and Z. Wu, ‘‘Handwriting representation of Chinese brush calligraphy based on three-dimension force information,’’ *Jisuanji Xitong Yingyong/Comput. Syst. Appl.*, vol. 18, no. 10, pp. 144–147, 2009.

[36] M. Otsuki, K. Sugihara, A. Toda, F. Shibata, and A. Kimura, ‘‘A brush device with visual and haptic feedback for virtual painting of 3D virtual objects,’’ *Virtual Reality*, vol. 22, no. 2, pp. 167–181, Jun. 2018, doi: 10.1007/s10055-017-0317-0.

[37] N. Xie, T. Zhao, Y. Yang, and H. T. Shen, ‘‘Web-based SBLR method of multimedia tools for computer-aided drawing,’’ *Multimedia Tools Appl.*, vol. 78, no. 1, pp. 799–816, Jan. 2019, doi: 10.1007/s11042-018-5949-x.

[38] D.-T. Liang, D. Liang, S.-M. Xing, P. Li, and X.-C. Wu, ‘‘A robot calligraphy writing method based on style transferring algorithm and similarity evaluation,’’ *Intell. Service Robot.*, vol. 13, no. 1, pp. 137–146, Jan. 2020, doi: 10.1007/s11370-019-00298-3.

[39] Y. H. Han, W. W. Sun, and B. H. Zheng, ‘‘COMPRESS: A comprehensive framework of trajectory compression in road networks,’’ *ACM Trans. Database Syst.*, vol. 42, no. 2, p. 49, Jun. 2017, doi: 10.1145/3015457.

[40] C. Lv, F. Chen, Y. Xu, J. Song, and P. Lv, ‘‘A trajectory compression algorithm based on non-uniform quantization,’’ in *Proc. 12th Int. Conf. Fuzzy Syst. Knowl. Discovery (FSKD)*, Aug. 2015, pp. 2469–2474.

[41] J. M. Calvin and M. K. Nakayama, ‘‘Resampled regenerative estimators,’’ *ACM Trans. Model. Comput. Simul.*, vol. 25, no. 4, p. 25, Nov. 2015, doi: 10.1145/2699718.

[42] K. Renny Simba, N. Uchiyama, and S. Sano, ‘‘Real-time smooth trajectory generation for nonholonomic mobile robots using Bézier curves,’’ *Robot. Comput.-Integr. Manuf.*, vol. 41, pp. 31–42, Oct. 2016, doi: 10.1016/j.rcim.2016.02.002.

[43] L. J. Van Bogaert, ‘‘The relation between height, foot length, pelvic adequacy and mode of delivery,’’ *Eur. J. Obstetrics, Gynecology, Reproductive Biol.*, vol. 82, no. 2, pp. 195–199, 1999, doi: 10.1016/s0301-2115(98)00232-2.

[44] K. H. Lee, H. Woo, and T. Suk, ‘‘Data reduction methods for reverse engineering,’’ *Int. J. Adv. Manuf. Technol.*, vol. 17, no. 10, pp. 735–743, May 2001, doi: 10.1007/s001700170119.

[45] Y. Qiu, X. Zhou, and X. Qian, ‘‘Direct slicing of cloud data with guaranteed topology for rapid prototyping,’’ *Int. J. Adv. Manuf. Technol.*, vol. 53, nos. 1–4, pp. 255–265, Mar. 2011, doi: 10.1007/s00170-010-2829-6.

[46] J. Zhang, J. Zheng, and Q. Gao, ‘‘Numerical solution of the degasperis–procesi equation by the cubic B-spline quasi-interpolation method,’’ *Appl. Math. Comput.*, vol. 324, pp. 218–227, May 2018, doi: 10.1016/j.amc.2017.11.058.

[47] M. Kazhdan and H. Hoppe, ‘‘Screened Poisson surface reconstruction,’’ *ACM Trans. Graph.*, vol. 32, no. 3, pp. 1–13, Jun. 2013, doi: 10.1145/2487228.2487237.

[48] H. D. Mustafa, S. N. Merchant, U. B. Desai, and B. M. Baveja, ‘‘Green symbiotic cloud communications: Virtualized transport layer and cognitive decision function,’’ *IEEE Access*, vol. 5, pp. 13409–13421, 2017, doi: 10.1109/access.2016.2644727.

[49] F. Samea, F. Azam, M. Rashid, M. W. Anwar, W. H. Butt, and A. W. Muzaffar, ‘‘A model-driven framework for data-driven applications in serverless cloud computing,’’ *PLoS ONE*, vol. 15, no. 8, Aug. 2020, Art. no. e0237317, doi: 10.1371/journal.pone.0237317.

[50] F. Samea, F. Azam, M. W. Anwar, M. Khan, and M. Rashid, ‘‘A UML profile for multi-cloud service configuration (UMLPMSC) in event-driven serverless applications,’’ in *Proc. 8th Int. Conf. Softw. Comput. Appl.*, Feb. 2019, pp. 431–435, doi: 10.1145/3316615.3316636.

[51] C.-H. Teh and R. T. Chin, "On the detection of dominant points on digital curves," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 11, no. 8, pp. 859–872, Aug. 1989, doi: 10.1109/34.31447.

[52] D. K. Prasad, "PRO: A novel approach to precision and reliability optimization based dominant point detection," *J. Optim.*, vol. 2013, Sep. 2013, Art no. 345287, doi: 10.1155/2013/345287.

[53] F. X. Diebold and M. Shin, "Assessing point forecast accuracy by stochastic error distance," *Econ. Rev.*, vol. 36, nos. 6–9, pp. 588–598, Oct. 2017, doi: 10.1080/07474938.2017.1307247.

[54] O. M. Ardakani, N. Ebrahimi, and E. S. Soofi, "Ranking forecasts by stochastic error distance, information and reliability measures," *Int. Stat. Rev.*, vol. 86, no. 3, pp. 442–468, Dec. 2018, doi: 10.1111/insr.12250.

FEIYU ZHAO was born in Wuhan, Hubei, China, in 1992. He received the M.S. degree in mechanical engineering from Wuhan Polytechnic University, Wuhan, China, in 2016, and the Ph.D. degree in mechanical engineering from the Wuhan University of Technology, Wuhan, in 2019.

He is currently a Lecturer with the College of Computer Science, South-Central University for Nationalities. His research interests include cloud-based CAD, smart manufacturing, and virtual simulation.

WEI TIAN was born in Hubei, China, in 1979. He received the B.S. degree from the Hubei University of Technology, Wuhan, China, in 2002, the M.S. degree from the Wuhan University of Technology, Wuhan, in 2006, and the Ph.D. degree from Wuhan University, Wuhan, in 2014.

He currently works with South-Central University for Nationalities. He is engaged in the study of plasma processing.

ZONGXIAO ZHU was born in Wuhan, Hubei, China, in 1978. He received the B.S. and M.S. degrees in electrical engineering from Xi'an Jiaotong University, Xi'an, China, in 2000 and 2003, respectively, and the Ph.D. degree in control science and engineering from the Huazhong University of Science and Technology, Wuhan, in 2014.

He is currently an Associate Professor with the College of Computer Science, South-Central University for Nationalities. His research interests include computer vision, virtual simulation, and artificial intelligence.

SHENG LEI was born in Huanggang, Hubei, China, in 1988. He received the M.S. degree in mechanical design and theory from the Wuhan University of Technology, Wuhan, China, in 2013, and the Ph.D. degree in mechanical design and theory from the Huazhong University of Science and Technology, Wuhan, in 2016.

He is currently a Lecturer with the College of Computer Science, South-Central University for Nationalities. His research interests include finite element analysis, dynamic analysis of machine tools, and dynamic monitoring of engineering structures.

DELONG KONG was born in Linxia, Gansu, China, in 1988. He received the M.S. degree in traffic engineering from Lanzhou Jiaotong University, Lanzhou, China, in 2014.

He is currently a Lecturer with the College of Computer Science, South-Central University for Nationalities. His research interests include intelligent transportation, train control systems, and signal processing.