Hindawi
Mathematical Problems in Engineering
Volume 2018, Article ID 4673849, 16 pages
https://doi.org/10.1155/2018/4673849

Research Article

A Lightweight Surface Reconstruction Method for Online 3D Scanning Point Cloud Data Oriented toward 3D Printing

Buyun Sheng,[1,2] **Feiyu Zhao**,[1] **Xiyan Yin**,[1] **Chenglei Zhang**,[1] **Hui Wang**,[1] **and Peide Huang**[1]

[1] *School of Mechanical and Electronic Engineering, Wuhan University of Technology, Wuhan, Hubei 430070, China*
[2] *Hubei Digital Manufacturing Key Laboratory, Wuhan University of Technology, Wuhan, Hubei 430070, China*

Correspondence should be addressed to Buyun Sheng; shengby@whut.edu.cn

Received 16 November 2017; Revised 23 February 2018; Accepted 4 March 2018; Published 16 May 2018

Academic Editor: Tudor Barbu

The existing surface reconstruction algorithms currently reconstruct large amounts of mesh data. Consequently, many of these algorithms cannot meet the efficiency requirements of real-time data transmission in a web environment. This paper proposes a lightweight surface reconstruction method for online 3D scanned point cloud data oriented toward 3D printing. The proposed online lightweight surface reconstruction algorithm is composed of a point cloud update algorithm (PCU), a rapid iterative closest point algorithm (RICP), and an improved Poisson surface reconstruction algorithm (IPSR). The generated lightweight point cloud data are pretreated using an updating and rapid registration method. The Poisson surface reconstruction is also accomplished by a pretreatment to recompute the point cloud normal vectors; this approach is based on a least squares method, and the postprocessing of the PDE patch generation was based on biharmonic-like fourth-order PDEs, which effectively reduces the amount of reconstructed mesh data and improves the efficiency of the algorithm. This method was verified using an online personalized customization system that was developed with WebGL and oriented toward 3D printing. The experimental results indicate that this method can generate a lightweight 3D scanning mesh rapidly and efficiently in a web environment.

1. Introduction

3D printing is a technology that manufacture solid parts through accumulating material layer by layer [1], and it can efficiently support the recent accomplishments in 3D digital-to-solid modeling [2]. 3D printing can produce customized objects [3, 4] that can be realized by any merchant to personalize products for a consumer, meet quality expectations, and add personal refinements [5]. With the development of the Internet, 3D printing cloud services platform can provide integrated 3D printing services [6] to serve customers at all levels: from cloud-based 3D scanning of a physical model to personalized and/or customized products [7]. However, cloud-based point cloud data surface reconstruction from 3D scanned point clouds requires not only a low-complexity reconstruction algorithm to meet the high concurrency requirements of cloud service platforms but also the reconstruction of a lightweight mesh to meet the requirements of efficient and real-time data transmission in web environments [8]. Most of the existing surface reconstruction algorithms use an accurate mesh as a reconstruction target without considering a reduction in the complexity of reconstruction algorithms or the lightweight nature of the 3D model data. In addition, a PC connected to a 3D scanner cannot receive the generated point cloud data directly from the cloud server due to Internet security requirements. Thus, real-time point cloud data transmission from the 3D scanner to the cloud service platform cannot be realized along with the process of 3D scanning [9].

To reduce the complexity of the algorithm and generate a lightweight 3D model, in this paper, an online lightweight surface reconstruction algorithm is proposed, which is composed of a point cloud update algorithm (PCU), a rapid iterative closest point algorithm (RICP), and an improved Poisson surface reconstruction algorithm (IPSR). The PCU is used to obtain the latest point cloud data generated by the 3D

scanner in real time. It can remove part of the noise from the point cloud data and makes it lightweight using a filtration approach based on the center of gravity. The 3D scanner used in this research is called Ciclop, which independently generates two point cloud data sets through its two infrared laser modules. Rapid and accurate registration between the two sets of point cloud data is achieved using the RICP. The resulting modified point cloud set restores the outline of the scanned object more accurately. During the process of surface reconstruction by the IPSR, the normal vectors of the point cloud data are pretreated via a recomputation method based on the least squares method. To repair the mesh holes that can easily be generated by the Poisson surface reconstruction algorithm, an iterative postprocessing algorithm for PDE patch generation based on biharmonic-like fourth-order PDEs is executed successively in IPSR, which also reduces the amount of reconstructed mesh data.

To realize real-time point cloud data transmission, a dynamic visualization framework for point cloud data, based on WebSocket, is also proposed in this paper. The point cloud data generated by Ciclop in real-time can be encapsulated as a JSON file in the local server and dynamically displayed in the browser using WebSocket and WebGL. In addition, the fluency and high rendering effect of dynamic visualization of point cloud data is ensured by the Web Worker mechanism in high concurrency environments.

In sum, the novel contributions in this paper are as follows: (1) an online lightweight surface reconstruction algorithm, in which the lightweight operations are conducted at every step, from point cloud data acquisition to preprocessing and to surface reconstruction. This reconstruction ensures that a reduced data volume of the 3D model meets the requirement for web-based data transmission; (2) a dynamic visualization framework for point cloud data based on WebSocket, which achieves online dynamic visualization of point cloud data in high concurrency environments; (3) an online personalized customization system oriented toward 3D printing, which dynamically visualizes the point cloud data through the 3D scanning process and the efficient and rapid reconstruction of the lightweight mesh in the web environment.

The remainder of this paper is organized as follows. Section 2 reviews prior relevant research concerning surface reconstruction. Section 3 elaborates the online lightweight surface reconstruction algorithm, which includes the PCU, the RICP, and the IPSR. Section 4 presents the experiments, which demonstrate the system framework and experimental platform, point cloud update and the results of dynamic visualization tests, point cloud registration tests, and online surface reconstruction tests, and Section 5 concludes the paper.

2. Related Works

2.1. Surface Reconstruction. In the field of surface reconstruction, Zhang et al. [10] proposed a new approach to simultaneously denoise and parameterize unorganized point cloud data, in which the key ingredient was an "as-rigid-as-possible" meshless parameterization that maps a point cloud with disk topology to a 3D plane. Denoising and reconstruction of the point cloud are executed in the same process. Huang et al. [11] proposed a 3D reconstruction system that performs fast 3D modeling using a Kinect sensor and can automatically detect the face region and track head pose using an ICP algorithm. A volumetric integration method was used to fuse the resulting data. Finally, a marching cubes algorithm was used to reconstruct the face model used for display, which effectively improved the face model accuracy. Centin et al. [12] proposed a new Poisson reconstruction algorithm based on an interpolation method and exploited it to efficiently guide a restricted Delaunay framework protecting the input mesh and the boundary curves. The boundary of a repaired hole can be seamlessly integrated with the patch boundary. Garrett et al. [13] proposed an accurate object-tracking method based on an ICP algorithm. Reduced-density friendly point clouds can be reconstructed to create an accurate mesh through a Poisson surface reconstruction algorithm via object tracking. Peyrot et al. [14] proposed a framework to design semiregular meshes directly from stereoscopic images in which feature-preserving samples of the stereoscopic images were extracted to obtain a base mesh. Then, using iterative procedures, a semiregular mesh of the original surface was generated from the base mesh. Liu et al. [15] proposed a method for representing 3D outdoor scenes via 3D laser point clouds that used a fast optimal bearing angle (FOBA) approach to project the 3D laser point clouds to 2D images, which greatly reduced the computational cost of scene segmentation with little loss of accuracy, thus improving the efficiency of the reconstruction. Li et al. [16] proposed a Prominent Cross-Section algorithm embedded with a curvature constraint that can automatically identify the boundary of a damaged area, thereby eliminating any defective point clouds during the reconstruction process. They also proposed an improved iterative ICP algorithm to automatically identify and eliminate any unreliable corresponding pairs. Roth et al. [17] proposed a method for reconstructing the 3D surface model of an individual's face along with albedo information. A 3D Morphable Model was fitted to form a personalized template, and a novel photometric stereo formulation was also developed. This formulation of an accurate 3D face model can be generated from low-quality photo collections and with fewer images based on the albedo information. Boltcheva and Lévy [18] proposed a method for reconstructing a 3D surface triangulation from an input point set based on a restricted Voronoi diagram. The properties of the restricted Voronoi cells were utilized to make an embarrassingly parallel implementation that could process 100 million vertices within a few minutes to achieve rapid mesh reconstruction.

2.2. Network Communications. In the field of network communication, Marion and Jomier [19] proposed a web-based system that focused on collaborative interaction. The system is composed of two innovative technologies: WebGL and WebSocket. The architecture of the proposed system was presented, and the operating process of the system was further elaborated in the field of telemedicine. Zimmer and Kerren [20] suggested a client/server-based visualization

system for collaboratively exploring graphs in which the web application was rendered by WebGL, and the real-time visualization transmission was achieved by WebSocket. Mwalongo et al. [21] proposed an approach to visualize dynamic molecular data using WebGL. The approach exploits HTML5 technologies such as WebSocket and Web Workers and used efficient data encoding techniques to minimize the data transferred to the client. This approach allows scientists to perform analyses from dynamic visualizations of the molecular structure data via a browser. Renambot et al. [22] presented SAGE2, a software framework based on WebSocket that enables local and remote collaboration. 3D-rendering and cross-platform visualization and interaction were supported in this framework, which can be utilized in education, academic conferences, and so on.

2.3. Existing Problems in Current Research. Although any of the current surface reconstruction algorithms [11–14] can reconstruct accurate meshes, the data volume of 3D meshes is larger; thus, the resulting average reconstruction time is also longer. As an example of this increased data volume, for Stanford Bunny point cloud data (1.89 MB), the reconstruction mesh was at least 20 MB and the average reconstruction time was at least 15 s. Although the quality of the point cloud and mesh improved while the reconstruction time decreased due to denoising and registration pretreatments, the surface reconstruction algorithms [10, 16] were not optimized, resulting in a insufficiently lightweight mesh. Consequently, the reconstruction was unable to meet the requirements for network transmission. While the methods of real-time data communication technologies, based on WebSocket [19–22], were achieved only if the effects of the relevant data were stored in a database, the data could be visualized dynamically in the browser. Thus, the real-time data communication between device and cloud server was not achieved. Overall, dynamic visualization of point cloud data in a browser, processed by 3D scanning technology, is difficult to achieve.

3. Online Lightweight Surface Reconstruction Algorithm

The method proposed in this paper is achieved by using an open-source 3D scanner called Ciclop. The structure of Ciclop is shown in Figure 1. The two point cloud sets, denoted by $P_A = \{p_{A_0}, p_{A_1}, \ldots, p_{A_n}\}$ and $P_B = \{p_{B_1}, p_{B_2}, \ldots, p_{B_n}\}$, are generated from infrared laser modules A and B, respectively.

To ensure the efficiency and speed of surface reconstruction and the reconstruction of a lightweight mesh, an online lightweight surface reconstruction algorithm is proposed in this paper. The algorithm includes the PCU, the RICP, and the IPSR. These three subalgorithms are executed in turn. A flow-chart of the online lightweight surface reconstruction algorithm is shown in Figure 2.

The process in the PCU can be described as two point cloud sets, P_{A_i} and P_{B_i}, which are generated by the infrared laser module A and B, respectively, from a set of depth images $I = \{I_1, I_2, \ldots, I_i, \ldots, I_n\}$ which are acquired continuously

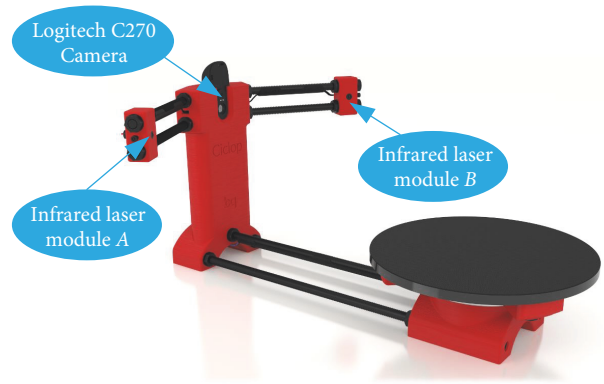


Figure 1: Open source Ciclop 3D scanner.

from Ciclop's Logitech C270 camera. The point cloud data is updated during the process of 3D scanning. The determination of whether to terminate the process is executed after each iteration. The holonomic point cloud sets P_A and P_B should be generated over several iterations.

The RICP performs the registration between P_A and P_B. First, P_A is selected as the benchmark, and a preregistration of P_A and P_B is executed based on the spatial relationship between the infrared laser modules A and B. Thus, P_B is rotated to the position of $[P_B]$. Then, the ICP algorithm transforms point cloud set $[P_B]$, rotating it to a new position, $\overline{P_B}$. Finally, the solution of modified point cloud set P^* is conducted using the spatial relationship between P_A and $\overline{P_B}$.

The IPSR first executes a pretreatment step that reconstructs the normal vectors of the modified point cloud set P^* based on the least squares method. Subsequently, the Poisson surface reconstruction and a postprocessing step that generates PDE patches based on a biharmonic-like fourth-order PDEs are conducted successively. The lightweight mesh M is generated through the above procedures.

3.1. Point Cloud Update Algorithm. A point cloud update algorithm (PCU) is proposed in this paper. The updating process involves the following process: whenever the Ciclop turntable is rotated by 1.8 degrees, a new depth image I_i is generated by the Logitech C270, adding the new feature points p_i and p_j to the point cloud sets P_{A_i} and P_{B_i}, respectively. P_A and P_B will be refined after several iterations.

To meet the requirements of efficient and real-time data transmission based on B/S, the new feature points generated from I_i should be filtered before they are added into the existing point cloud sets. As shown in Figure 3, first, new feature points are selected. Then, new triangular structures are formed based on these new feature points and the base of a triangular structure. These new triangular structures are considered the base triangular structures that need further refinement.

The filtration approach is conducted based on a center of gravity approach that can be described as follows.

Step 1. Three feature points p_A, p_B, p_C that are about to be reconstructed as mesh triangle are obtained from the depth

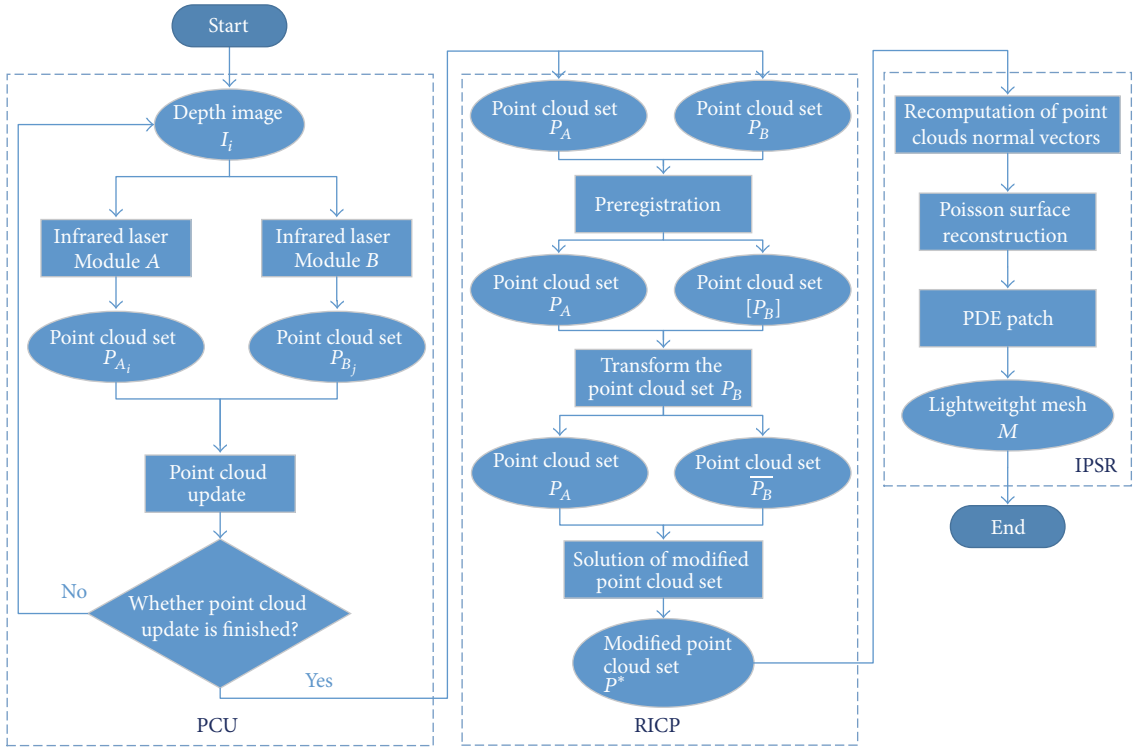


Figure 2: Online lightweight surface reconstruction algorithm.

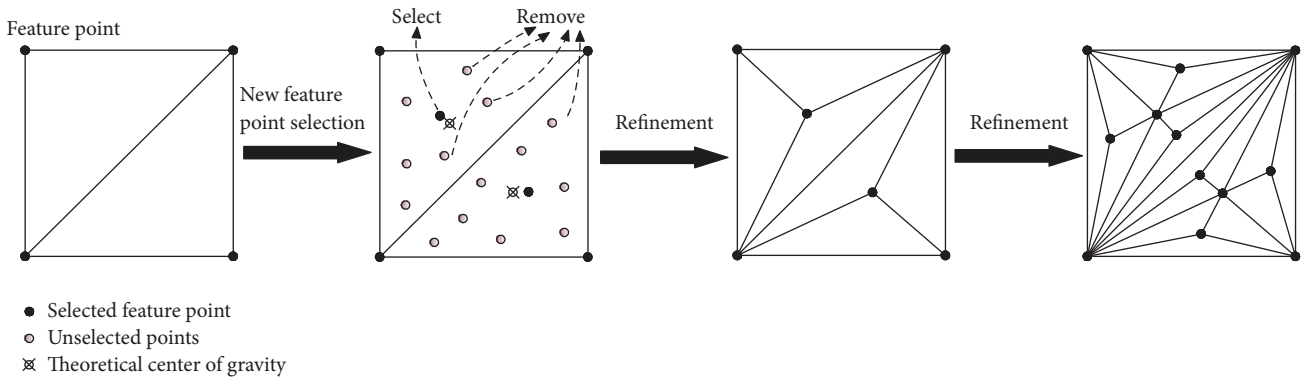


Figure 3: Filtration of new feature points.

image I_i, and their 2D coordinates $p_A(x_A, y_A)$, $p_B(x_B, y_B)$, and $p_C(x_C, y_C)$ are extracted. Their center of gravity is computed by $p_G = (x_G, y_G) = ((x_A + x_B + x_C)/3, (y_A + y_B + y_C)/3)$.

Step 2. The 2D points obtained from the depth image I_{i+1} are imported into the existing point cloud set. Any points located inside the triangle (ignoring the points on the boundary) are selected and collected as the point set $\{p_1, p_2, \ldots, p_n\}$.

Step 3. The variable $d_{\min}$ is defined as the minimum distance. A bubble sort algorithm [23] is utilized to compute the point (p_i) nearest to p_G, which is determined using the distance formula $d_{\min} = \sqrt{(x_G - x_i)^2 + (y_G - y_i)^2}$.

Step 4. The new feature point p_i is selected, and three sub-triangles are generated from this new feature point with the other three existing feature points, after which the filtration process is complete.

Compared with the point cloud update approach that adds all the points into the point cloud set, our approach effectively achieves a denoised and lightweight point cloud to some extent.

3.2. Rapid Iterative Closest Point Algorithm. Fundamentally, the RICP can be regarded as a solution of a least squares problem. The method was proposed by Besl and McKay [24], who determined that this approach can accomplish

registration between two point cloud sets P_A and P_B [25]. Let $P_A = \{p_{A_0}, p_{A_1}, \ldots, p_{A_n}\}$ be the reference point cloud set and $P_B = \{p_{B_1}, p_{B_2}, \ldots, p_{B_n}\}$ be the sample point cloud set. A translation vector $\mathbf{t}$ and a rotation matrix R which transform P_B to P_A can be computed by ICP.

To achieve rapid registration between different point cloud sets and reduce the number of iterations, a preregistration process is conducted between P_A and P_B, i.e., the rotation matrix R' and translation vector $\mathbf{t}'$ forming a rigid transformation from the position $O_B(x_B, y_B, z_B)$ to the position $O_A(x_A, y_A, z_A)$ of the respective infrared laser modules, B and A, are computed in a world-coordinate system. Then, P_B is preassociated with R' and $\mathbf{t}'$, which greatly reduces the time to find the nearest points between two point cloud sets and number of ICP iterations.

(1) Preregistration. Translation vector $\mathbf{t}'$ is computed from $O_A(x_A, y_A, z_A)$, $O_B(x_B, y_B, z_B)$ and formulated as follows:

$$\mathbf{t}' = O_A - O_B, \tag{1}$$

where O_A and O_B should be expressed as pure quaternions (i.e., $O'_A(x_A, y_A, z_A, 0)$ and $O'_B(x_B, y_B, z_B, 0)$). Let the origin of the world-coordinate system be O; then, the rotation unit quaternion from $\mathbf{OO'_B}$ to $\mathbf{OO'_A}$ can be computed as $q(q_x, q_y, q_z, q_w)$. According to [26],

$$\mathbf{OO'_A} = q \cdot \mathbf{OO'_B} \cdot q^{-1}, \tag{2}$$

where q^{-1} is the conjugate quaternion of q and expressed as $q^{-1}(-q_x, -q_y, -q_z, q_w)$. Thus, q can be solved from (2), and q can be transformed to the rotation matrix R', which is formulated as follows:

$$R' = \begin{bmatrix} q_w^2 + q_x^2 - q_y^2 - q_z^2 & 2(q_x q_y - q_w q_z) & 2(q_x q_z + q_w q_y) \\ 2(q_x q_y + q_w q_z) & q_w^2 - q_x^2 + q_y^2 - q_z^2 & 2(q_y q_z - q_w q_x) \\ 2(q_x q_z - q_w q_y) & 2(q_y q_z + q_w q_x) & q_w^2 - q_x^2 - q_y^2 + q_z^2 \end{bmatrix}. \tag{3}$$

Thus, the rigid transformation of P_B based on R' and $\mathbf{t}'$ is implemented to transform P_B to the point cloud set $[P_B]$, which is formulated as follows:

$$[P_B] = R' P_B + \mathbf{t}'. \tag{4}$$

(2) The Transformation of the Point Cloud Set. The ICP relies on identifying the nearest neighbors between both point sets. To find the matches between P_A and $[P_B]$, a k-nearest neighbors search [27] is used that adopts Euclidean distance as the distance measure:

$$\operatorname{dist}([P_B], P_A) = \sqrt{\left([p_{B_1}] - p_{A_1}\right)^2 + \left([p_{B_2}] - p_{A_2}\right)^2 + \cdots + \left([p_{B_n}] - p_{A_n}\right)^2}. \tag{5}$$

Only one neighbor is returned when P_A and $[P_B]$ are in exact spatial correspondence. When too many points are close (relative to an algorithmic threshold), the match is rejected in the current frame and not considered further. Therefore, a *kd-tree* data structure is used for all points, and an *approximate nearest neighbor search* [28] is implemented to accelerate the search.

All the associated points should be weighted. The weight value is based on the comparability of normal vectors and formulated as follows:

$$w_{ij} = \mathbf{n}_i \cdot \mathbf{n}_j. \tag{6}$$

The normal vectors of two matching points must be aligned in a similar direction after the two point clouds are approximately aligned. The output of this step is a point cloud set consisting of the N corresponding points in P_A and $[P_B]$, along with the weight values w_{ij} for each match $0 \le w_{ij} \le 1$.

To transform the point sets, the translation vector $\mathbf{t}$ and rotation matrix R are computed from $[P_B]$ to P_A. The translation vector $\mathbf{t}$ can be computed as the difference between the corresponding centroids of P_A and $[P_B]$, which can be formulated as follows:

$$g_{P_A} = \frac{1}{N_{P_A}} \sum_{i=1}^{N_{P_A}} p_{A_i}$$
$$g_{[P_B]} = \frac{1}{N_{[P_B]}} \sum_{j=1}^{N_{[P_B]}} \left[p_{B_j}\right]. \tag{7}$$

The translation in step k is given as a rotation from the previous step, $\Delta\mathbf{t} = g_{[P_B]} - R \cdot g_{P_A}$. Assume that total number of iterations is K; then, the final translation vector is

$$\mathbf{t} = \mathbf{t}_{K-1} + \Delta\mathbf{t}. \tag{8}$$

To calculate the rotation matrix R, all the points must be first moved to their centroids and formulated as

$$p'_{A_i} = p_{A_i} + g_{P_A}$$
$$\left[p'_{B_i}\right] = \left[p_{B_i}\right] + g_{[P_B]}. \tag{9}$$

According to [29], the incremental rotation matrix ΔR is computed by

$$\Delta R = V U^T, \tag{10}$$

where V and U are orthogonal matrices that result from a singular value decomposition where

$$W = U \sum V^T = \sum_{i=1}^{N_{P_A}} \left[p'_{B_i}\right] \left(p'_{A_i}\right)^T. \tag{11}$$

Here, ΔR is the orientation increment between the matching points in $[p_{B_j}]$ and p_{A_i}, which stem from $[P_B]$ and P_A, respectively. The final rotation matrix R is formulated as follows:

$$R = R_{K-1} + \Delta R. \tag{12}$$

ICP aligns the two point clouds by solving a least squares problem [30] with an error function $\sigma(R, \mathbf{t})$ formulated as

$$\sigma(R, \mathbf{t}) = \frac{1}{N_{P_A}} \sum_{i=1}^{N_{P_A}} w_i \left\| \left[p_{B_i}\right] - R \cdot p_{A_i} - \mathbf{t} \right\|^2, \quad (13)$$

where N_{P_A} represents the number of points in P_A. Points that do not meet the error metric (such as a set Euclidean distance between p_{A_i} and $[p_{B_j}]$ for outlier removal) have a weight of 0 so that they do not contribute to the error function. The termination criterion $[\sigma]$ is checked after all the iteration steps have been processed. The iteration continues until the termination criterion $\Delta\sigma = \sigma_k - \sigma_{k-1} < [\sigma]$ is reached in step k, meaning that the error delta between two subsequent frames is below the termination criterion $[\sigma]$. The resulting translation vector $\mathbf{t}$ and the rotation matrix R are generated in step k.

(3) Solution of the Modified Point Cloud Set. After transforming the point cloud set from $[P_B]$ to the position registered with P_A, a new point cloud set $\overline{P_B}$ is generated. The modified point cloud set P^*, for arbitrary $p_i^* \in P^*$, p_i^* is computed as shown in the following:

$$p_i^* = \frac{\left(p_{A_i} + \overline{p_{B_j}}\right)}{2}. \quad (14)$$

3.3. Improved Poisson Surface Reconstruction Algorithm. Surface reconstruction can be accomplished by a Poisson surface reconstruction algorithm. Typical PDE Poisson equations are constructed and solved iteratively to extract the isosurface and realize surface reconstruction. However, there are several disadvantages to this algorithm: it is strongly dependent on normal vectors and tends to form mesh holes. Therefore, the IPSR is proposed in this paper. The point cloud pretreatment, including recomputing the normal vectors based on the least squares method and postprocessing a generated PDE patch based on biharmonic-like fourth-order PDEs, is conducted successively to generate an accurate, connected, and lightweight mesh.

3.3.1. Recomputation of Point Clouds Normal Vectors. The recomputation of the normal vectors of a point cloud proposed in this paper is based on the least squares method. Then, the computational complexity of the surface reconstruction algorithm is reduced based on the accuracy of the normal vectors [31].

An arbitrary coordinate point p_i^*, in a point cloud set P^*, and all the K neighbor points of p_i^* can be regarded as being on approximately the same plane, L, which is formulated as follows:

$$Ax + By + Cz + D = 0. \quad (15)$$

The residual sum of squares between all the K-neighbor points and L is

$$d = \sum_{i=1}^{n} \left(Ax_i^* + By_i^* + C - z\right)^2. \quad (16)$$

The best fit can be regarded as the minimum d; therefore, we take the partial derivative of d with A, B, and C. Then, the equations are

$$\begin{aligned} \frac{\partial d}{\partial A} &= 2\left(A\sum_{i=1}^{n} {x_i}^2 + B\sum_{i=1}^{n} x_i y_i + C\sum_{i=1}^{n} x_i - \sum_{i=1}^{n} x_i z_i\right) \\ \frac{\partial d}{\partial B} &= 2\left(A\sum_{i=1}^{n} x_i y_i + B\sum_{i=1}^{n} {y_i}^2 + C\sum_{i=1}^{n} y_i - \sum_{i=1}^{n} y_i z_i\right) \\ \frac{\partial d}{\partial C} &= 2\left(A\sum_{i=1}^{n} x_i + B\sum_{i=1}^{n} y_i + nC - \sum_{i=1}^{n} z_i\right). \end{aligned} \quad (17)$$

Let $\partial d/\partial A = 0$, $\partial d/\partial B = 0$, and $\partial d/\partial C = 0$. Finally, the result can be formulated as follows:

$$\begin{bmatrix} A \\ B \\ C \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{n} {x_i^*}^2 & \sum_{i=1}^{n} x_i^* y_i^* & \sum_{i=1}^{n} x_i^* \\ \sum_{i=1}^{n} x_i^* y_i^* & \sum_{i=1}^{n} {y_i^*}^2 & \sum_{i=1}^{n} y_i^* \\ \sum_{i=1}^{n} x_i^* & \sum_{i=1}^{n} y_i^* & n \end{bmatrix}^{-1} \begin{bmatrix} \sum_{i=1}^{n} x_i^* z_i^* \\ \sum_{i=1}^{n} y_i^* z_i^* \\ \sum_{i=1}^{n} z_i^* \end{bmatrix}. \quad (18)$$

The coefficients A, B, and C can be solved via (18) to generate the normal vector n_i of p_i^*. Moreover, the vector orientations are processed for consistency; that is, an arbitrary normal vector n_j of point p_j^*, which is near point p_i^*, is selected. Transvection of n_i and n_j is conducted. If $n_i \cdot n_j > 0$, the orientation of n_i is kept; otherwise, n_i is adjusted to the opposite orientation: $-n_i$.

3.3.2. Poisson Surface Reconstruction. In a Poisson surface reconstruction, the solution of the surface to be reconstructed is regarded as the solution of a 3D indicator function χ. The reconstructed surface is then obtained by extracting the appropriate isosurface [32]. This process is depicted in Figure 4.

To solve the Poisson equation, the 3D space should be discretized. Therefore, an adaptive *octree* is built. The closer the border is, the larger the density of the *octree* is [32]. Based on the *octree*, the base function F is defined as

$$F(x, y, z) \equiv \left(B(x)\, B(y)\, B(z)\right)^{*n} \quad \text{with } B(t) = \begin{cases} 1 & |t| < 0.5 \\ 0 & \text{otherwise,} \end{cases} \quad (19)$$

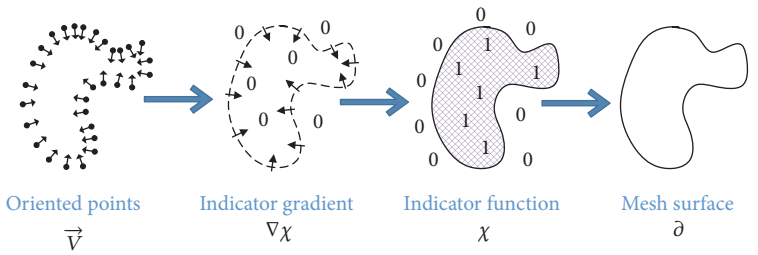


Figure 4: The process of Poisson surface reconstruction.

where ${}^{*}n$ represents the nth convolution. Every point sample becomes a leaf node o in a depth D *octree*; therefore, the unit-integral "node function" centered around node o is formulated as

$$F_o(q) = F\left(\frac{q - o \cdot c}{o \cdot w}\right)\frac{1}{(o \cdot w)^3}, \tag{20}$$

where $o \cdot c$ represents the center of o and $o \cdot w$ represents the width of o.

The vector field $\vec{V}(q)$ is determined from the normal vector $\vec{n_i}$ of p_i^* in a point cloud set P^*. The gradient field of the indicator function χ can be expressed as $\nabla\chi$. The purpose of the Poisson surface reconstruction is to solve a function χ whose gradient best approximates a vector field $\vec{V}(q)$ defined by P^*. This can be formulated as follows:

$$\min_{\chi} \quad \left\|\nabla\chi - \vec{V}\right\|. \tag{21}$$

The problem shown in (21) can be solved by the Poisson equation $\Delta\chi = \nabla \bullet \vec{V}$. A vector field $\vec{V}$ can be approximately formulated as

$$\vec{V}(q) = \sum_{p^* \in P^*} \sum_{o \in Ngbr_D(p*)} \alpha_{o,s} F_o(q)\,\vec{n}, \tag{22}$$

where $Ngbr_D(s)$ are the eight depth-D nodes closest to p^* and $\{\alpha_{o,s}\}$ represents the trilinear interpolation weights. To solve the problem that $\nabla \bullet \vec{V}$ and $\Delta\chi$ are not in the function space, the equation can be simplified by solving for the χ function by minimizing

$$\begin{aligned}&\sum_o \left\|\left\langle \Delta\chi - \nabla\cdot\vec{V}, F_o\right\rangle\right\|^2 \\ &= \sum_o \left\|\langle \Delta\chi, F_o\rangle - \left\langle \nabla\cdot\vec{V}, F_o\right\rangle\right\|^2.\end{aligned} \tag{23}$$

Given a vector v, whose oth coordinate is $\vec{v_o} = \langle \nabla \cdot \vec{V}, F_o\rangle$, the goal is to solve for the function χ. The vector can be obtained by projecting the Laplacian of χ onto each F_o, ensuring that it is as close to v as possible. To express (23) in matrix form, let $\chi = \sum_o x_o F_o$; then, the problem is transformed into one of solving for the vector x_o.

$$\begin{aligned}&\sum_o \left\|\langle \Delta\chi, F_o\rangle - \left\langle \nabla\cdot\vec{V}, F_o\right\rangle\right\|^2 \\ &= \sum_{o'} \left\|\sum_o x_o \langle \Delta F_o, F_{o'}\rangle - \left\langle \nabla\cdot\vec{V}, F_{o'}\right\rangle\right\|^2.\end{aligned} \tag{24}$$

Let the matrix L be contributed as the dot product of the Laplacian with each of the F_o values; for all o, o'; the (o, o')th entry of L is set to

$$\begin{aligned}L_{o,o'} \equiv &\left\langle \frac{\partial^2 F_o}{\partial x^2}, F_{o'}\right\rangle + \left\langle \frac{\partial^2 F_o}{\partial y^2}, F_{o'}\right\rangle \\ &+ \left\langle \frac{\partial^2 F_o}{\partial z^2}, F_{o'}\right\rangle.\end{aligned} \tag{25}$$

Thus, the solution of the indicator function χ can be regarded as finding

$$\min_{x} \quad \|Lx - v\|^2. \tag{26}$$

The isosurface from the indicator function χ can be extracted to *octree* representations through the method used in previous adaptations of the marching cubes [33].

3.3.3. Generation of PDE Patches. The lightweight methods are implemented in the abovementioned algorithms (i.e., PCU in Section 3.1, RICP in Section 3.2, and the recomputation of point clouds normal vectors in Section 3.3.1). Because of the highly lightweight point cloud data, it is easy to generate mesh holes near areas with complex curvature variations, which could influence the visualization of the reconstructed 3D model. Thus, in this paper, an alternative approach to PDE patch generation based on biharmonic-like fourth-order PDEs is proposed. The algorithm is utilized to repair mesh holes on the reconstructed surface according to Section 3.3.2. Each hole is represented by patches with their own uv coordinate system. The preservation of irregular and sharp details on the surface will be found by matching the respective patches to the surface at various sizes and orientations. According to [34], a biharmonic-like fourth-order PDE is formulated as follows:

$$\left(\frac{\partial^2}{\partial u^2} + a^2\frac{\partial^2}{\partial v^2}\right)^2 S(u, v) = 0, \tag{27}$$

where $S(u, v)$ is regarded as the hole $H(u, v)$ to be repaired, and $0 \le u \le 1$, $0 \le v \le 2\pi$. Additionally, the PDE patch $P(u, v)$ can be formulated by (27). The analytic solution of (27) can be computed by of separating the variables and formulating the problem as

$$H(u, v) = A_0(u) + \sum_{n=1}^{N} \left[A_n(u) \cos(nv) + B_n(u) \sin(nv)\right], \tag{28}$$

where $A_0(u)$, $A_n(u)$, $B_n(u)$ are, respectively, formulated as follows:

$$\begin{aligned} A_0(u) &= \alpha_{00} + \alpha_{01}u + \alpha_{02}u^2 + \alpha_{03}u^3 \\ A_n(u) &= \alpha_{n1}e^{anu} + \alpha_{n2}ue^{anu} + \alpha_{n3}e^{-anu} + \alpha_{n4}ue^{-anu} \\ B_n(u) &= \beta_{n1}e^{anu} + \beta_{n2}ue^{anu} + \beta_{n3}e^{-anu} + \beta_{n4}ue^{-anu}. \end{aligned} \tag{29}$$

The PDE coefficients $\alpha_{00}, \alpha_{01}, \ldots, \alpha_{n3}, \alpha_{n4}$ and $\beta_{11}, \beta_{12}, \ldots, \beta_{n3}, \beta_{n4}$ are vectors valued by the boundary conditions. $A_0(u)$ is regarded as the "spine" of the surface, and $A_n(u)\cos(nv) + B_n(u)\sin(nv)$ is an n-order "radius" vector. The amplitude of the "radius" term decays as the frequency increases (if $N \to \infty$), so that the solution of the PDE patch is mostly determined by a first-order "radius" vector $A_1(u)\cos(v) + B_1(u)\sin(v)$ and a second-order "radius" vector $A_2(u)\cos(2v) + B_2(u)\sin(2v)$. Therefore, (28) can be approximately reformulated as follows:

$$H(u, v) = A_0(u) + A_1(u)\cos(v) + B_1(u)\sin(v) + A_2(u)\cos(2v) + B_2(u)\sin(2v). \tag{30}$$

The number of vector-valued PDE coefficients is $M_c = 4 \times (2N + 1)$. The higher M_c is, the higher the complexity of the reconstruction is. To generate a lightweight mesh, we let $N = 3$. Then, the boundary conditions that fit (27) are formulated as follows:

$$\begin{aligned} H(0, v) &= C_0(v) \\ H(u_1, v) &= C_1(v) \\ H(u_2, v) &= C_2(v) \\ H(1, v) &= C_3(v), \end{aligned} \tag{31}$$

where the conditions $C_0(v)$, $C_1(v)$, $C_2(v)$, and $C_3(v)$ are isoparm boundary curves on the surface patch at $u = 0, u_1, u_2, 1$, respectively, where $0 < u_1, u_2 < 1$. To precisely approach the mesh representation of complex geometric shapes, this method needs to use sufficient PDE boundary curves, which must be extracted from the vertices of the original polygon mesh representation. Figure 5 illustrates the layout of the various PDE boundary curves for an individual PDE patch, where $C_0(v)$ degenerates into a single point.

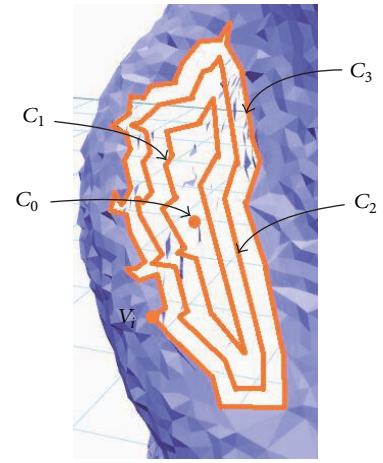


Figure 5: Layout of PDE boundary curves.

4. Experiment and Analysis

4.1. System Framework and Establishment of Experimental Platform. To verify the feasibility of the proposed lightweight surface reconstruction method proposed in this paper, an online personalized customization system oriented toward 3D printing was developed and used as the experimental environment. The system framework is shown in Figure 6 and can be divided into 4 units: Central Control Unit; Online Personalized Customization Unit; 3D Scanning Unit; and 3D Printing Unit. The Central Control Unit runs on a PC and a cloud server, and the collaborative operation of the other 3 units is controlled by the Central Control Unit, which is the core of the whole system. The Online Personalized Customization Unit runs in Google Chrome directly. The dynamic visualization of the point cloud generated by the 3D Scanning Unit can be rendered; that is, the point cloud generated in real-time can be displayed to users. Moreover, the point clouds are reconstructed as meshes. Modifications of color, texture, dimension, slicing of the STL model, and generation of G codes are also supported in the Online Personalized Customization Unit. G codes can be imported by the 3D Printing Unit for the 3D printing tasks.

(1) Hardware Framework of 3D Scanning Unit. The 3D Scanning Unit is composed of the open source 3D scanner Ciclop, whose hardware framework is shown in Figure 7. The Ciclop is composed of an Arduino UNO Motherboard (control core), a Logitech C270 (camera), two infrared laser modules, and other mechanical structures. Its turn table is driven by a NEMA 17 stepper motor which rotates the object undergoing scanning at 1.8°/s around its central axis. Simultaneously, two sets of 5 mW infrared laser modules, set at an angle of 135°, are coordinated with the Logitech C270 to generate depth data at different angles. Thus, the point cloud data of the scanned object can be generated as well [35].

(2) Dynamic Visualization of Point Cloud Data. To achieve the dynamic visualization, the point cloud data from the 3D Scanning Unit to the Online Personalized Customization

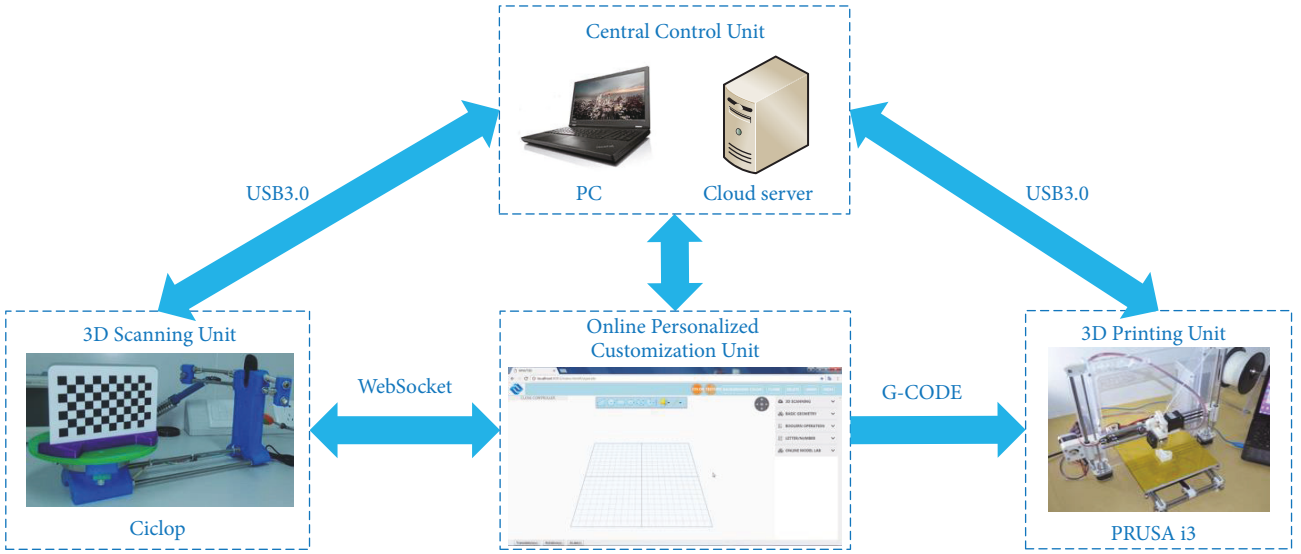


Figure 6: Framework of the online personalized customization system oriented toward 3D printing.

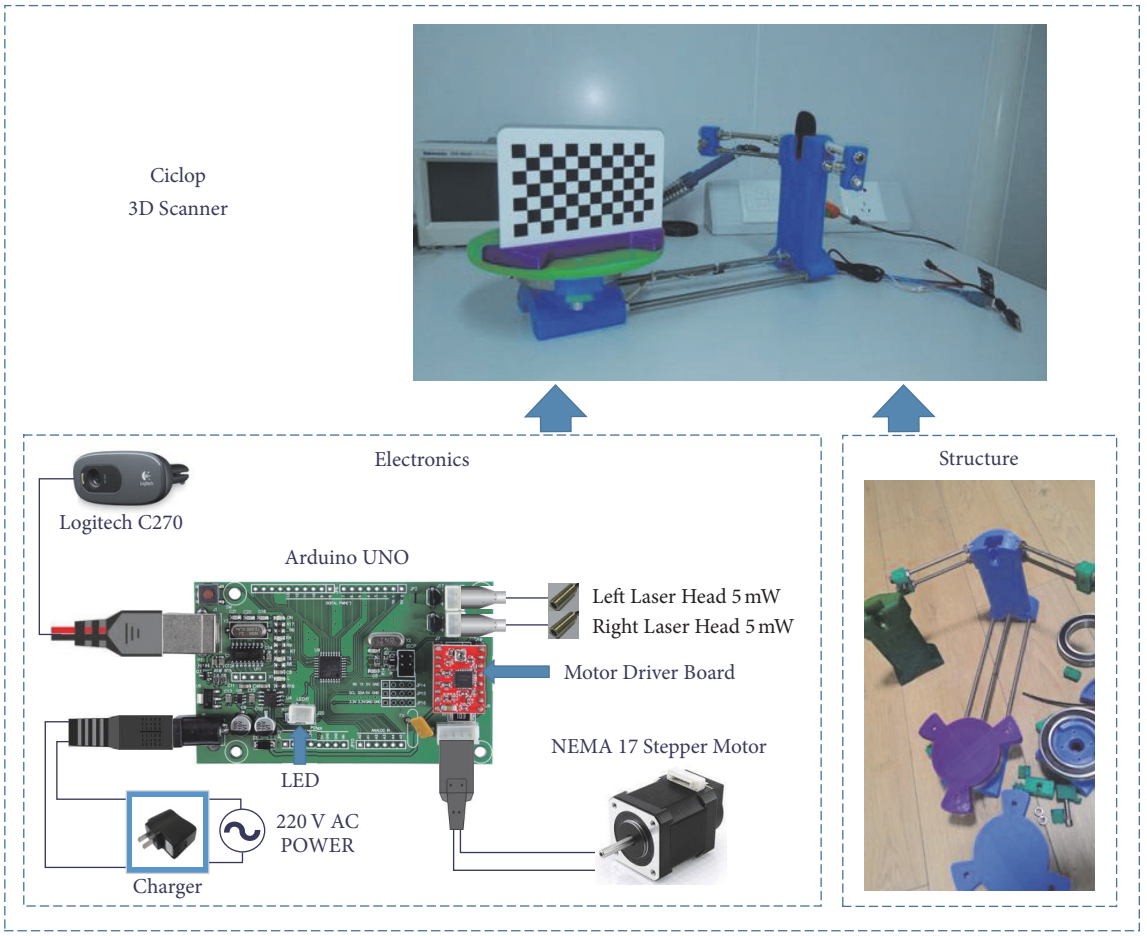


Figure 7: Hardware framework of 3D Scanning Unit.

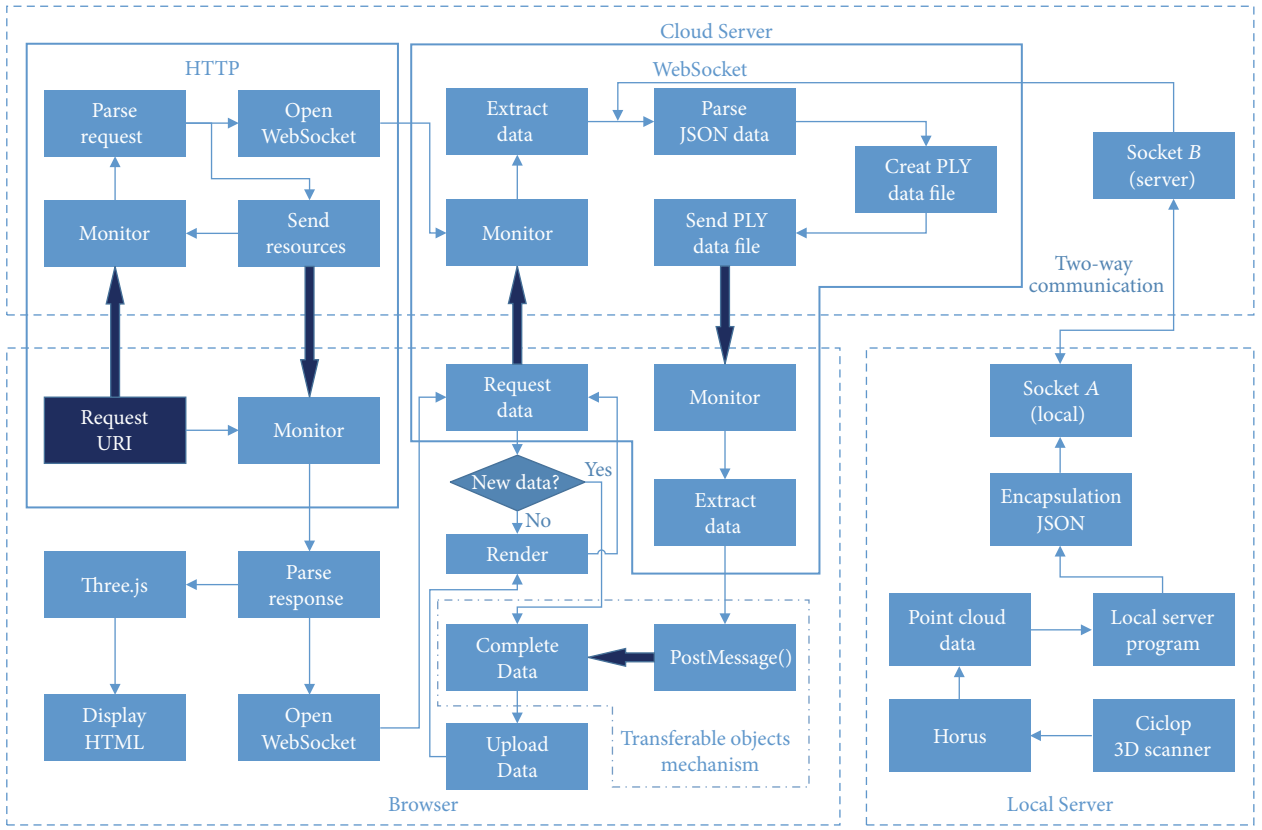


Figure 8: Dynamic point cloud visualization framework.

Unit needs to be transmitted in real time. A proposed dynamic visualization framework for point clouds based on WebSocket is shown in Figure 8. The thin arrows represent the control flow within the server or browser, while the wide arrows represent the data flow between memory spaces. WebSocket is a real-time communication protocol for HTML5 that can achieve real-time data transmission between a browser and a cloud server [36]. Thus, the point cloud data that are generated by Ciclop in real time can be dynamically displayed in the browser by WebSocket and WebGL. The framework is divided into three parts: cloud server, local server, and browser.

The interaction is initiated by the browser, which sends a request URI to the cloud server. The cloud server parses the request and sends the appropriate JavaScript code back to the browser, which dynamically visualizes and renders the point cloud. Meanwhile, a request is also sent to Ciclop. The point clouds generated by Ciclop are extracted by its underlying computer software, Horus, and encapsulated as JSON files by the local server. A socket connection can be established between a local server and cloud server to upload JSON files. Subsequently, a WebSocket connection will also be established through JavaScript code running in the Browser to fetch and parse the JSON files. After the request URI is sent from the browser to the cloud server, a monitor is established to determine whether the latest JSON file to be parsed in the browser through WebSocket connection exists. The point cloud data will be dynamically updated by WebGL when the JSON file exists.

JavaScript is generally single-threaded; consequently, the performance requirements of the computer hardware are higher. To allow the user-interface to remain responsive, the browser fetches and renders the JSON files. The fetching operation runs in a separate Web Worker tread, while the rendering runs in the main JavaScript thread. All long-running code paths need to run outside the main thread [37]. Moreover, to avoid browser congestion by too many messages, the next time step of a JSON file is always requested explicitly by the browser. That is, only when the browser acquires a new rendering request for the current point cloud data to GPU does it send the cloud server a new request for new JSON files. Thus, this framework ensures that each frame will be rendered at least once.

(3) Establishment of the Experimental Platform. The hardware configuration of the Central Control Unit is shown in Table 1.

The environmental calibration parameters of the 3D scanning operation to meet the requirements of the experimental environment are configured in Horus, the computer software underlying Ciclop, as shown in Table 2.

4.2. Point Cloud Update and Dynamic Visualization Experiments. Using the dynamic point cloud visualization

Table 1: Hardware configuration of Central Control Unit.

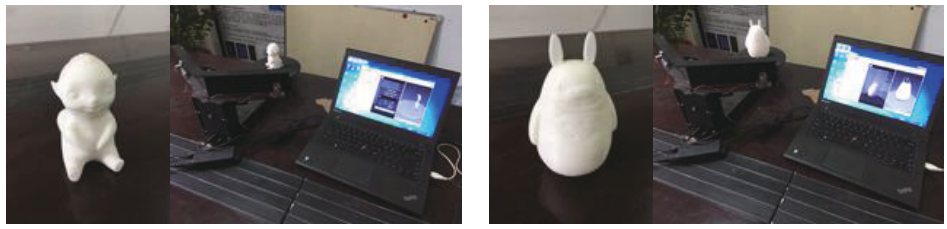
Hardware	PC	Cloud server
CPU	Intel Core i7 7700K 4.2 GHz	Intel Skylake Xeon Platinum 8163 2.5 GHz
GPU	ASUS RX550 4 GB	ASUS NVIDIA GeForce GTX1080 8 GB
RAM	8 GB	10 GB
Hard Disk	SSD 128 GB	SSD 40 GB
Bandwidth	10 Mbps	50 Mbps

Table 2: Environmental calibration parameters of experiments.

Indexes	Brightness	Contrast	Saturation	Exposure	Step degrees	Photograph interval
Value	150	34	32	10	0.45°	0.5 s

Table 3: Generation time and amount of data on two sets of point clouds.

Model type	Dynamic visualization with PCU		Dynamic visualization without PCU	
	Load time/s	Size/MB	Load time/s	Size/MB
Huba	109	1.60	94	5.19
Totoro	126	1.89	112	6.54



(a) (b)

Figure 9: Experimental environments of 3D scanning.

framework and the PCU as proposed in Section 3.1, experiments assessing the online 3D scanning were performed on the "Huba" and "Totoro" models to verify the effect of point cloud dynamic visualization and the lightweight effect of the PCU. The experimental environment is shown in Figure 9, and the dynamic visualization processes in Google Chrome are shown in Figure 10. To optimize the visualization effects, the background color of Google Chrome was set to black.

The generation time and amount of data of the two sets of point clouds (representing the two models "Huba" and "Totoro") were recorded. For comparison purposes, the two circumstances were implemented both with and without the PCU. The results are shown in Table 3.

As listed in Table 3, with the PCU, the size of the point cloud data of the "Huba" model decreases by 69.2% (from 5.19 MB to 1.60 MB), while the size of the point cloud data of the "Totoro" model decreases by 71.1% (from 6.54 MB to 1.89 MB) during the point cloud dynamic visualization process. Although the load time of each model increases by approximately 15% with the PCU, the growth is not substantial compared to the lightweight point cloud data. Experimental analysis shows that lightweight dynamic visualization of point cloud data in a web environment can be accomplished by the PCU in conjunction with the dynamic visualization framework for point clouds based on WebSocket as proposed in this paper.

4.3. Point Cloud Registration Experiments. Experiments on the registration of two sets of point clouds, P_A and P_B, which belong to the "Huba" and "Totoro" models, were conducted by RICP and ICP, respectively, to generate a modified point cloud set P^*. The termination criteria of the MSE threshold was set to $[\sigma] = 0.005$ mm, and the number of iterations and the registration time of the two point cloud sets by RICP and ICP were recorded and are listed in Table 4, and the convergence curves of the registration error with the two sets of point clouds are shown in Figure 11.

The registration effects of the two sets of point clouds calculated by the different algorithms are shown in Figure 12.

With the RICP, the number of iterations decreased from 12 to 7, and the time decreased by 45.8% from 59 s to 32 s for the "Huba" model, while the number of iterations decreased from 15 to 8, and the time decreased by 46.8% from 77 s to 41 s for the "Totoro" model. These results are listed in Table 4. Figure 11 shows that the MSEs of the two models decrease by 0.039 mm and 0.036 mm after one iteration.

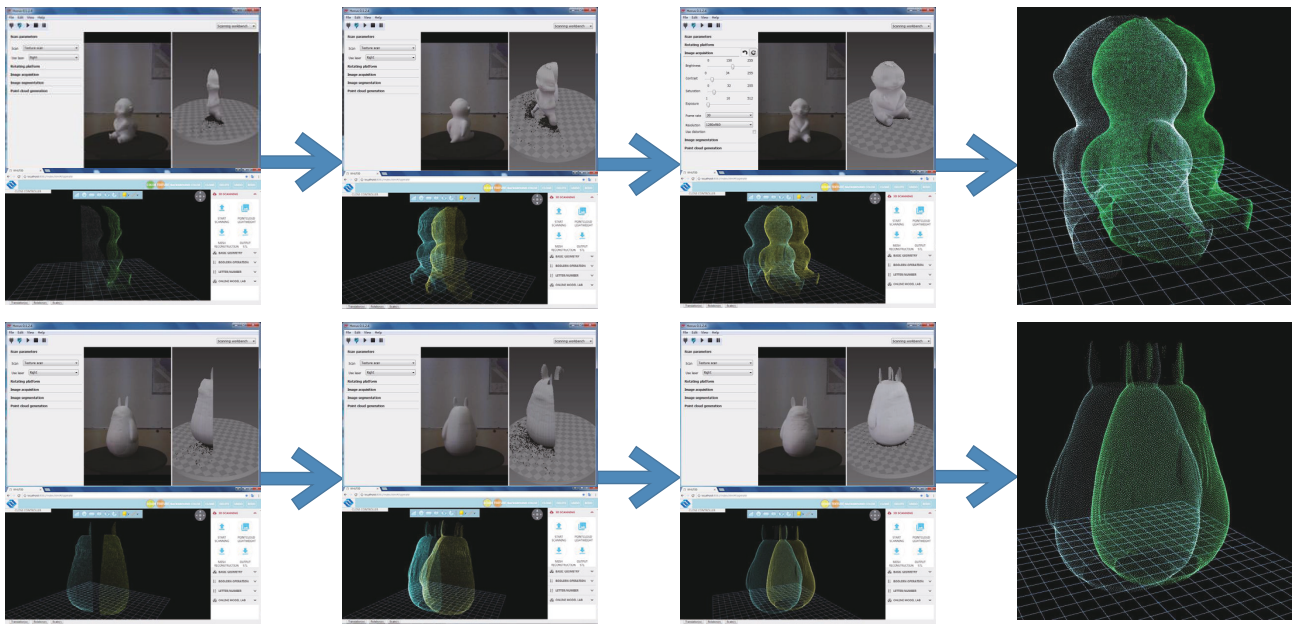


Figure 10: Dynamic visualization processes of point cloud.

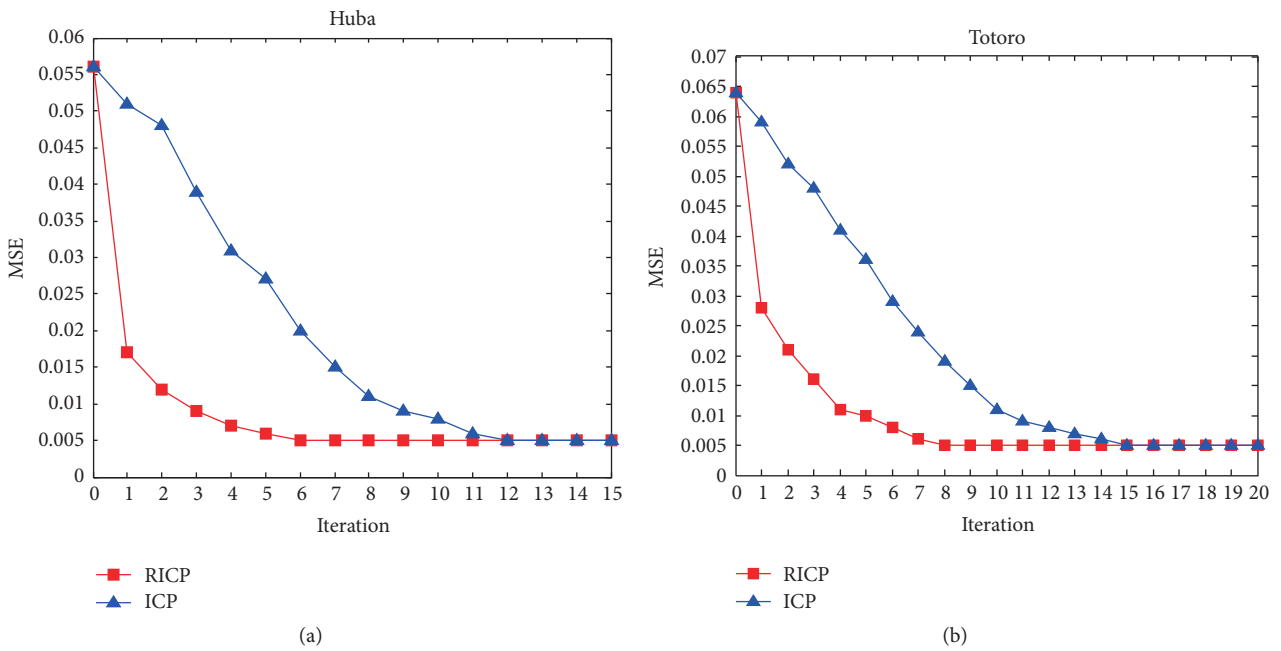


Figure 11: Convergence curves of the registration error.

Table 4: Number of iterations and elapsed time of registration by RICP and ICP.

Registration algorithm	Huba			Totoro		
	$[\sigma]$/mm	Iteration	Time/sec	$[\sigma]$/mm	Iteration	Time/sec
RICP	0.005	7	32	0.005	8	41
ICP		12	59		15	77

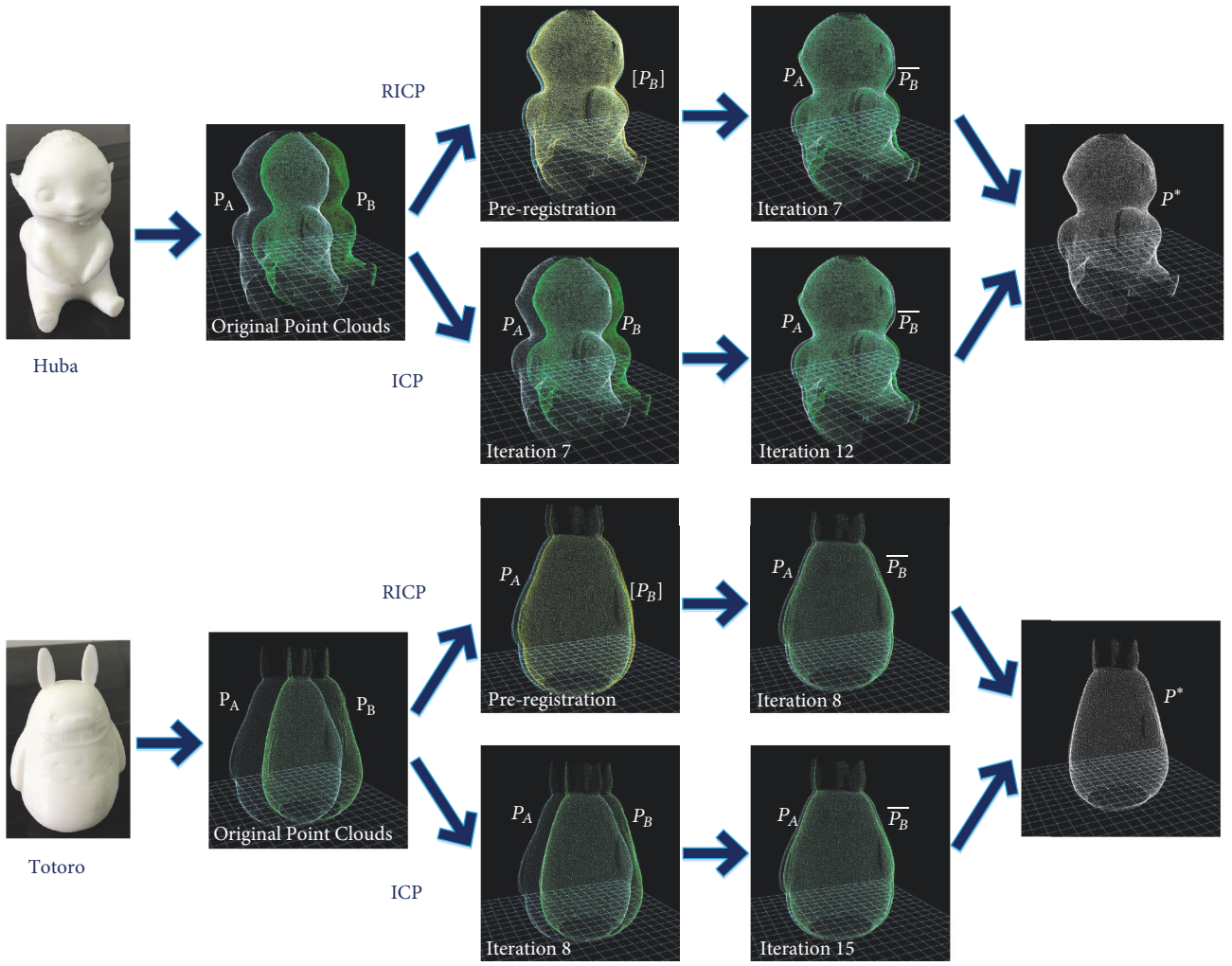


Figure 12: Visualization effect of registration by RICP and ICP.

Table 5: Relevant indicators of surface reconstruction by IPSR and PSR.

Model	Points	Algorithm type	Triangular facets	Size/MB (STL)	Reconstruction time/s	Average FPS	Occupied CPU
Huba	85432	PSR	73735	14.4	3.4	62	8%
		IPSR	61476	10.5	2.9	61	7%
Totoro	129463	PSR	101950	20.1	7.9	76	10%
		IPSR	89759	17.2	6.5	74	8%

The experimental analysis shows the increase in algorithmic efficiency from preregistering the point cloud data in the RICP. The visualization effects of the registrations by the RICP and ICP are shown in Figure 12.

4.4. Online Surface Reconstruction Experiments. Using the IPSR proposed in Section 3.3, modified point cloud data sets of the "Huba" and "Totoro" were reconstructed by both IPSR and Poisson surface reconstruction (PSR) algorithm in the web environment. The online surface reconstructions by both algorithms are shown in Figure 13.

Relevant indicator values were recorded during the process of surface reconstruction to assess IPSR and PSR, including triangular facets, size, reconstruction time, average frames per second (FPS), and occupied CPU percentage [38]. The specific values of these metrics are listed in Table 5, while four histograms of the triangular facets, size, reconstruction time, and average FPS are shown in Figure 14.

As Figure 13 shows, several single-point spikes and holes exist on the polygon mesh reconstructed by PSR. In contrast, the polygon meshes reconstructed by IPSR are accurate, connected, and lightweight. In fact, the surface is relatively

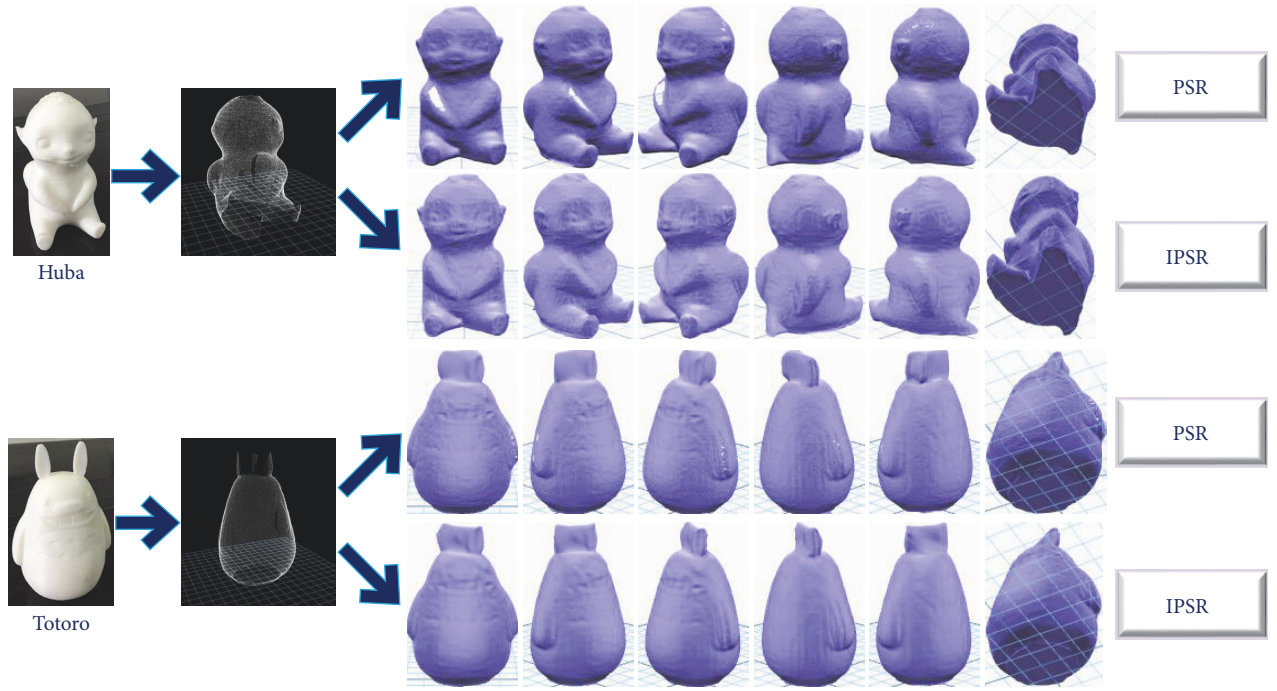


Figure 13: Online surface reconstruction effect by IPSR and PSR.

smooth, without any obvious single-point spikes on the hole boundaries.

In addition to the differences in Table 5 and Figure 14, compared with PSR, the triangular facets of the "Huba" mesh created by IPSR decreased by 16.6%, while the size and reconstruction time decreased by 27.1% and 14.7%, respectively. The triangular facets of the "Totoro" mesh decreased by 12.1%, while the size and reconstruction time decreased by 14.4% and 17.7%, respectively. The lightweight extent of each mesh produced by IPSR is relatively obvious. Although the decrease in average FPS is barely noticeable (1.6% and 2.6% for the "Huba" mesh and "Totoro" mesh, resp.) both are still relatively acceptable with tiny fluctuations around a median value of approximately 70. Users reported that the system ran smoothly without any obvious decrease in rendering speed.

The experimental analysis shows that the reconstruction efficiency, the accuracy, and the absence of obvious spikes in the mesh reconstructed by IPSR are more desirable given the lightweight data requirements of a web environment. Additionally, meshes on a 3D printing cloud server platform can be personalized and customized using this method.

5. Conclusions

In this paper, a lightweight surface reconstruction method for online 3D scanning point cloud data oriented toward 3D printing is proposed. To achieve low algorithmic complexity and generate a lightweight 3D model, an online lightweight surface reconstruction algorithm is proposed, which is composed of a point cloud update algorithm (PCU), a rapid iterative closest point algorithm (RICP), and the improved Poisson surface reconstruction algorithm (IPSR). The PCU is used to denoise the point cloud data to create a lightweight version in real time. The RICP is used to perform rapid and accurate registration between two sets of point cloud data. The IPSR is used to generate the lightweight mesh, and the postprocessing of the PDE patch generation based on biharmonic-like fourth-order PDEs is executed to repair the mesh holes on the reconstructed lightweight mesh, which improves the 3D model visualization. In addition, to achieve real-time point cloud data transmission in conjunction with the 3D scanning process in a web environment, a dynamic visualization framework for point cloud data based on WebSocket is also proposed. This approach achieves comprehensive dynamic visualization of the point cloud data in the browser. The fluency and high rendering effect are ensured through the Web Worker mechanism in the high concurrency environment. Finally, an online personalized customization system oriented toward 3D printing is developed based on the proposed method.

Conflicts of Interest

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

Acknowledgments

This work is supported by the National Key Technology Research and Development Program of China (no. 2016YFB1101700) and Nature Science Foundation of Hubei Province, China (no. 2015CFA115).

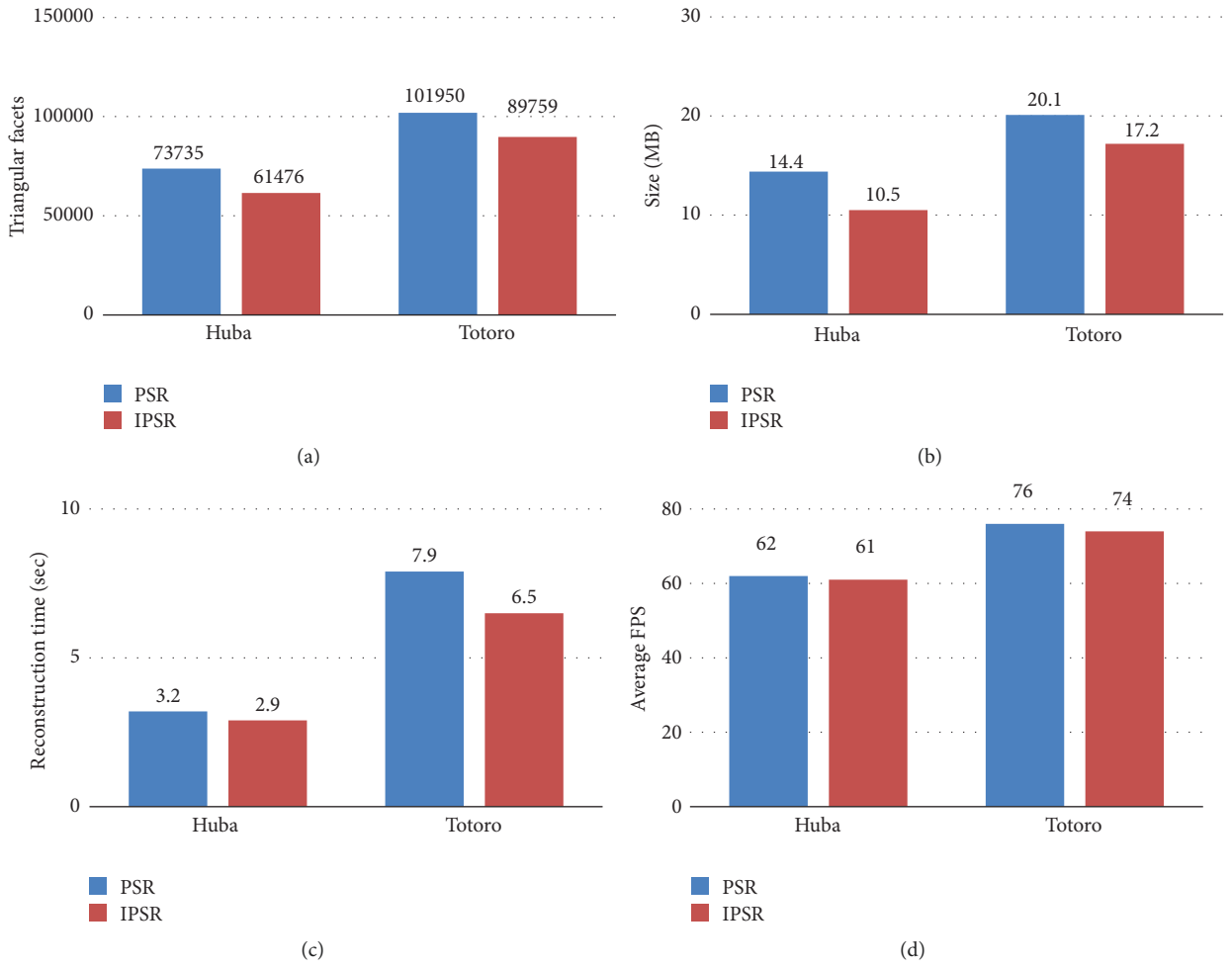


Figure 14: Histograms of relevant indicators.

References

[1] J. Klein, M. Stern, G. Franchin et al., "Additive Manufacturing of Optically Transparent Glass," *3D Printing and Additive Manufacturing*, vol. 2, no. 3, pp. 92–105, 2015.

[2] P. Markillie, "A third industrial revolution," *The economist*, vol. 21, pp. 3–12, 2012.

[3] X. Yao and Y. Lin, "Emerging manufacturing paradigm shifts for the incoming industrial revolution," *The International Journal of Advanced Manufacturing Technology*, vol. 85, no. 5-8, pp. 1665–1676, 2016.

[4] W. Oropallo and L. A. Piegl, "Ten challenges in 3D printing," *Engineering with Computers*, vol. 32, no. 1, pp. 135–148, 2016.

[5] C. Lim, "The Implementation of 3D Printing in Customized Interactive Design for Elderly Welfare Technology," in *HCI International 2014 - Posters' Extended Abstracts*, vol. 435 of *Communications in Computer and Information Science*, pp. 299–303, Springer International Publishing, Cham, 2014.

[6] J. Mai, L. Zhang, F. Tao, and L. Ren, "Customized production based on distributed 3D printing services in cloud manufacturing," *The International Journal of Advanced Manufacturing Technology*, vol. 84, no. 1-4, pp. 71–83, 2016.

[7] J. Xu, L. Ding, and P. E. D. Love, "Digital reproduction of historical building ornamental components: From 3D scanning to 3D printing," *Automation in Construction*, vol. 76, pp. 85–96, 2017.

[8] W. Liu, X. Zhou, X. Zhang, and Q. Niu, "Three-dimensional (3D) CAD model lightweight scheme for large-scale assembly and simulation," *International Journal of Computer Integrated Manufacturing*, vol. 28, no. 5, pp. 520–533, 2015.

[9] M. Wenzel, A. Klinger, and C. Meinel, "Tele-board prototyper - Distributed 3D modeling in a web-based real-time collaboration system," in *Proceedings of the 2016 International Conference on Collaboration Technologies and Systems, CTS 2016*, pp. 446–453, usa, November 2016.

[10] L. Zhang, L. Liu, C. Gotsman, and H. Huang, "Mesh reconstruction by meshless denoising and parameterization," *Computers and Graphics*, vol. 34, no. 3, pp. 198–208, 2010.

[11] X. Huang, X. Chen, T. Tang, and Z. Huang, "Marching cubes algorithm for fast 3D modeling of human face by incremental data fusion," *Mathematical Problems in Engineering*, vol. 2013, Article ID 203609, 2013.

[12] M. Centin, N. Pezzotti, and A. Signoroni, "Poisson-driven seamless completion of triangular meshes," *Computer Aided Geometric Design*, vol. 35/36, pp. 42–55, 2015.

[13] T. Garrett, S. Debernardis, J. Oliver, and R. Radkowski, "Poisson Mesh Reconstruction for Accurate Object Tracking with Low-Fidelity Point Clouds," *Journal of Computing and Information Science in Engineering*, vol. 17, no. 1, Article ID 011003, 2017.

[14] J.-L. Peyrot, F. Payan, and M. Antonini, "From stereoscopic images to semi-regular meshes," *Signal Processing: Image Communication*, vol. 40, pp. 97–110, 2016.

[15] Y. Liu, F. Wang, A. M. Dobaie, G. He, and Y. Zhuang, "Comparison of 2D image models in segmentation performance for 3D laser point clouds," *Neurocomputing*, vol. 251, pp. 136–144, 2017.

[16] L. Li, C. Li, Y. Tang, and Y. Du, "An integrated approach of reverse engineering aided remanufacturing process for worn components," *Robotics and Computer-Integrated Manufacturing*, vol. 48, pp. 39–50, 2017.

[17] J. Roth, Y. Tong, and X. Liu, "Adaptive 3D Face Reconstruction from Unconstrained Photo Collections," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 11, pp. 2127–2141, 2017.

[18] D. Boltcheva and B. Lévy, "Surface reconstruction by computing restricted Voronoi cells in parallel," *Computer-Aided Design*, vol. 90, pp. 123–134, 2017.

[19] C. Marion and J. Jomier, "Real-time collaborative scientific WebGL visualization with WebSocket," in *Proceedings of the 17th International Conference on 3D Web Technology, Web3D 2012*, pp. 47–50, USA, August 2012.

[20] B. Zimmer and A. Kerren, "Harnessing WebGL and WebSockets for a Web-based collaborative graph exploration tool," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics): Preface*, vol. 9114, pp. 583–598, 2015.

[21] F. Mwalongo, M. Krone, M. Becher, G. Reina, and T. Ertl, "GPU-based remote visualization of dynamic molecular data on the web," *Graphical Models*, vol. 88, pp. 57–65, 2016.

[22] L. Renambot, T. Marrinan, J. Aurisano et al., "SAGE2: A collaboration portal for scalable resolution displays," *Future Generation Computer Systems*, vol. 54, pp. 296–305, 2016.

[23] J. Alnihoud and R. Mansi, "An enhancement of major sorting algorithms," *International Arab Journal of Information Technolog*, vol. 7, no. 1, pp. 55–62, 2010.

[24] P. J. Besl and N. D. McKay, "A method for registration of 3-D shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, 1992.

[25] R. Radkowski, "Object Tracking With a Range Camera for Augmented Reality Assembly Assistance," *Journal of Computing and Information Science in Engineering*, vol. 16, no. 1, p. 011004, 2016.

[26] J. Vince, *Quaternions for Computer Graphics*, Springer, London, Uk, 2011.

[27] J. M. Lee, "Fast k-Nearest Neighbor Searching in Static Objects," *Wireless Personal Communications*, vol. 93, no. 1, pp. 147–160, 2017.

[28] M. Greenspan and M. Yurick, "Approximate k-d tree search for efficient ICP," in *Proceedings of the 4th International Conference on 3-D Digital Imaging and Modeling (3DIM '03)*, pp. 442–448, IEEE, October 2003.

[29] K. S. Arun, T. S. Huang, and S. D. Blostein, "Least-squares fitting of two 3D point sets," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 9, no. 5, pp. 698–700, 1987.

[30] J. H. Lee, S. Y. Hwang, and C. H. Ryu, "Localization between curved shell plate and its unfolded shape in different coordinate systems for ship-hull plate forming," *Mathematical Problems in Engineering*, vol. 2011, Article ID 257804, 2011.

[31] X. Wang, J. Cai, Z. Wu, and M. Zhou, "Normal Estimation and Normal Orientation for Point Cloud Model Based on Improved Local Surface Fitting," *Journal of Computer-Aided Design Computer Graphics*, vol. 27, no. 4, pp. 614–620, 2015.

[32] M. Kazhdan, M. Bolitho, and H. Hoppe, "Poisson surface reconstruction," in *Proceedings of the fourth Eurographics symposium on Geometry processing*, pp. 61–70, Cagliari, Sardinia, Italy, 2006.

[33] W. E. Lorensen and H. E. Cline, "Marching cubes: a high resolution 3D surface construction algorithm," *ACM SIGGRAPH Computer Graphics*, vol. 21, no. 4, pp. 163–169, 1987.

[34] Y. Sheng, A. Sourin, G. G. Castro, and H. Ugail, "A PDE method for patchwise approximation of large polygon meshes," *The Visual Computer*, vol. 26, no. 6-8, pp. 975–984, 2010.

[35] L. Sun, X. Suo, Y. Liu, M. Zhang, and L. Han, "3D Modeling of Transformer Substation Based on Mapping and 2D Images," *Mathematical Problems in Engineering*, vol. 2016, Article ID 9320502, 2016.

[36] D. Wang and J. Shao, "The HTML5 graphic and image processing technology," *Agro FOOD Industry Hi Tech*, vol. 28, no. 3, pp. 2956–2959, 2017.

[37] J. Verdú, J. J. Costa, and A. Pajuelo, "Dynamic web worker pool management for highly parallel javascript web applications," *Concurrency Computation*, vol. 28, no. 13, pp. 3525–3539, 2016.

[38] S. Sun, M. Zhang, and Z. Gou, "Smoothing Algorithm for Planar and Surface Mesh Based on Element Geometric Deformation," *Mathematical Problems in Engineering*, vol. 2015, Article ID 435648, 2015.