www.cad-journal.net
Computer-Aided Design AND APPLICATIONS

# Low-poly Mesh Generation from Low-quality Point Clouds Based on Projections

Shinichi Sano[1], Hiroaki Date[2] and Satoshi Kanai[3]

[1]Hokkaido University, sano.shinichi.b0@elms.hokudai.ac.jp
[2]Hokkaido University, hdate@ist.hokudai.ac.jp
[3]Hokkaido University, kanai@ssi.ist.hokudai.ac.jp

Corresponding author: Shinichi Sano, sano.shinichi.b0@elms.hokudai.ac.jp

**Abstract.** Recently, it has become possible to acquire point clouds of objects and environments using cameras and LiDAR on small mobile devices, such as iPads and iPhones. However, the acquired point clouds are of low quality (large lack of points, varying point densities, and high-level noise), and generating practical 3D models from point clouds faces challenges. In this study, we propose a simple method for generating beautified low-poly mesh models from low-quality point clouds using 2D projection, image processing, and geometric contour-based operations. The proposed approach first determines the optimal projection directions by analyzing surface normals and creates point cloud projection images, which represent the object's silhouettes. Subsequently, image-based filtering for noise removal and hole filling was applied to the 2D silhouette images, and symmetries were detected. Next, regularized 2D contour polylines of the silhouettes were extracted, and primitives were generated by sweeping the polylines. Finally, a 3D mesh was reconstructed by computing the intersection of the primitives. The experimental results demonstrate that the proposed method can reconstruct low-poly meshes with symmetries and regularities, such as parallel and orthogonal meshes, from low-quality point clouds with a large lack of points.

**Keywords:** Point cloud, 3D scanning, Low-poly mesh, projection
**DOI:** https://doi.org/10.14733/cadaps.2026.803-813

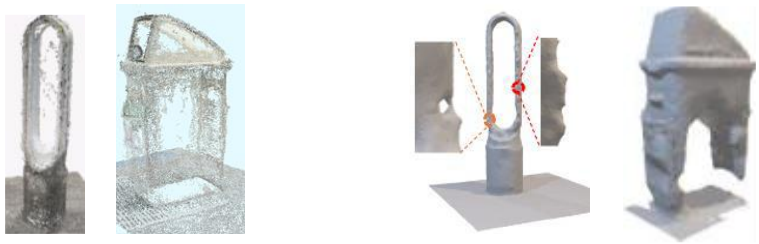
## 1 INTRODUCTION

Over the past two decades, the use of point clouds of large-scale environments and structures acquired by terrestrial, mobile, airborne, and UAV laser scanning systems has become common. Acquired precise and dense point clouds are used in several fields, such as plant engineering, product development, civil engineering, architecture, and land surveying. In contrast, currently, point clouds can also be acquired using small mobile devices such as iPads and iPhones. With the growing availability of mobile devices equipped with LiDAR and photogrammetry applications, non-experts can acquire 3D data of their surroundings. In such devices, point clouds are often generated using

small, mounted LiDAR devices or photogrammetry. However, the quality of point clouds varies depending on the system and the method used for point cloud acquisition. As shown in Fig. 1(a), low-quality point clouds with a large lack of points, variations in point densities, and high-level noise can often be acquired. The quality of the point clouds obtained from laser scanning and photogrammetry significantly depends on the surface properties of the target objects. Consequently, low-quality meshes with undesired bumps or large holes are obtained, as shown in Fig. 1(b). Room for improvement exists in the rapid generation of useful meshes using the existing methods [26].

In this paper, a simple and fast algorithm for beautified low-poly mesh generation from low-quality point clouds is presented. The proposed method is based on a low-poly mesh generation method that uses high-density meshes of buildings [9]. In the proposed method, the input low-quality point clouds are first projected onto the 2D planes. Then, noise removal, hole filling, and beautifications are performed on the projected plane using image and polygon processing techniques. The resulting contours of the objects were swept along the projection directions, and the final mesh was obtained as an intersection of the swept volumes. The proposed method is applicable to point clouds independent of the acquisition sources, and no additional data is needed, such as images. Because output meshes are regularized and lightweight, they can be used for digital archives of surrounding objects and construction of complex scenes for AR/VR and entertainment.

Related works are introduced in Section 2. The proposed method is described in Section 3, and the results of the proposed method for low-quality point clouds are presented and compared in Section 4.



(a) (b)

Fig. 1: Point clouds acquired by a mobile device and generated 3D meshes: (a) Low-quality point clouds and (b) Meshes generated by existing methods [23].

## 2 RELATED WORKS

Several methods exist for generating meshes from point clouds [26]. These methods can be classified into two main categories. The first is a surface-based method that generates mesh surfaces directly for a given point cloud. The Delaunay-based method [1] defines surface triangles according to Delaunay criteria. The ball pivoting algorithm [3] creates triangles for three points that contact a sphere pivoting on the surface of the point clouds. These methods connect the points in a given point cloud to construct a surface. However, they are not suitable for low-quality point clouds, because they tend to preserve noise and are not robust to point clouds of different densities. In particular, it is difficult to fill the holes in the given data appropriately. As the other method, subdivision surface fitting [12,18] can be used for reconstructing surfaces from point clouds. Although these methods create smooth surfaces approximating input point clouds, initial meshes are required, and there are challenges for creating appropriate initial meshes from low-quality point clouds. Our method has a potential to support the generation of initial meshes for subdivision surfaces.

The second method is a volume-based method. This method generates a mesh by dividing space into volumes and classifying regions as either interior or exterior. One popular volume-based method is the Poisson surface reconstruction (PSR) [13], and some extensions exist, such as screened PSR

(SPSR) [14], PSR with envelope constraints [15], and stochastic methods [25]. These methods reconstruct the surface by finding a scalar indicator function whose gradient field fits the field defined by the normals of the given point clouds over the entire space and then extract the isosurfaces [17] of the indicator function. SPSR [14] is an improved version of PSR [13] that introduces a screening term into the conventional PSR, enabling mesh generation that faithfully reflects the shape of the input points while maintaining tolerance. PSR with an envelope [15] adopts an envelope that represents the boundaries of free spaces as Dirichlet constraints to improve the correctness of the resulting meshes. The stochastic method [25] extends the PSR method with a Gaussian distribution to improve the robustness to noisy inputs. These methods can generate smooth surfaces from point clouds; however, producing correct and beautified low-poly meshes from low-quality point clouds is difficult owing to the difficulty of accurate normal calculation and the influences of large holes and noises. Recently, deep learning-based methods have been developed. DeepSDF [21] is a method that represents the shape of an object as a continuous signed distance function (SDF) using neural networks. For an arbitrary point, SDF returns the signed distance to the shape surface, indicating whether the point lies inside or outside. Although this approach can generate smooth and noise-resistant meshes, it is computationally expensive and requires that the SDF be trained on data-appropriate functions. Although other neural network-based methods have also been proposed, the correctness of the surfaces depends on the training data [26].

As the novel methods for the 3D model and scene representation, recently, neural radiance field (NeRF) [2] and 3D Gaussian Splatting [4] have garnered attention. 3D Gaussian Splatting represents a scene as a collection of 3D Gaussians that are projected and composited onto a screen space to generate free-viewpoint images with high speed and quality. Although both methods are powerful for image synthesis, they require additional processing to output a mesh instead of a point cloud and are computationally expensive. In addition, these technologies are combined with image or video capturing and are not applicable only to point clouds. The proposed method is applicable to point clouds generated by these technologies.

The proposed method is based on the projection-based low-poly mesh generation method for buildings proposed by Gao et al. [9]. This method generates a simplified mesh from a dense mesh using multiple contour sweep geometry intersection operations and mesh simplification. First, silhouettes of a building were created for each projection direction and 3D primitives were generated by sweeping the silhouette. Subsequently, a product set of the primitives was created to produce a 3D mesh model without concave features. Finally, the mesh was simplified by restoring the minor concave features. Their method achieved higher shape fidelity and simplicity than existing approaches. In this study, we applied this strategy to create a low-poly mesh from low-quality point clouds because of its applicability to image and geometry processing in the 2D domain.

In the proposed method, regularization for parallelism and orthogonality is considered in the resulting model. GlobFit [16], proposed by Li et al., is a mesh reconstruction method that considers regularization by incorporating geometric constraints. This method constructs a consistent model by considering global relationships (e.g., parallelism and orthogonality) when fitting geometric primitives, such as planes and cylinders, from point cloud data. They first detect geometric primitives such as planes, cylinders, and circles using RANSAC on an input point cloud. They then analyze the global relationships among these primitives and determine the optimal placement of the primitives through nonlinear optimization, treating these relationships as variables. This method achieved consistent reconstruction results for structured scenes. However, this depended heavily on the initial fitting. Therefore, it is not well suited for low-quality point clouds with high noise levels and large missing regions, which are the focus of this study. Áron Monszpart et al. proposed RAPter [20], a method for reconstructing man-made scenes such as buildings as regular planar arrangements. Their method extracts initial planar candidates from a point cloud and selects the most consistent arrangement by evaluating both the goodness of fit of the data and the geometric regularities (e.g., parallelism, orthogonality, and equal spacing). This method has the advantage of being able to reconstruct planar structures of artificial scenes from noisy point clouds in a regularized manner. However, this is time consuming because mixed-integer problems must be solved. In the proposed method, regularization is performed on the projected planes; therefore, simpler methods can be

applied. In the proposed method, a part of the regularization method proposed by Takahashi et al. [27] was used for the regularization of the 2D contours. As the low-poly mesh generation, mesh simplification methods for dense meshes can be used [5,6,10]. However, it is difficult to preserve regularities in the input meshes while simplification process. In our method, regularities are recognized and imposed to 2D simple polygons, and the final shapes are created by intersection of swept volume of the 2D simple polygons. Therefore, low-poly meshes with regularities can be created efficiently.

## 3 PROPOSED METHOD FOR GENERATING LOW-POLY MESHES FROM LOW-QUALITY POINT CLOUDS.

### 3.1 Method Overview

The input for this method is a low-quality point cloud of the target object acquired by small mobile devices, as shown in Fig. 1(a). We assumed that the local floor near the objects was also scanned, and a point cloud of the local floor was acquired. The output of the method is a low-poly 3D mesh with regularities such as symmetry, parallel, and orthogonality. In this study, the projection-based simplified mesh generation method for buildings proposed by Gao et al. [9] was extended to generate a low-poly mesh from a low-quality point cloud. In Gao's method, a low-poly mesh is created by simplifying the mesh of the intersection of the swept volumes of the silhouettes obtained by projecting a given high-density mesh in multiple directions. This method has the advantage of allowing the application of 2D images and geometric processing. In the proposed method, noise removal, hole filling, and regularization were performed using 2D processing. As the final 3D mesh was represented by the intersection of a set of swept volumes of 2D projections (silhouettes) of a point cloud, the target objects of our study were limited to those whose major surfaces appeared in the silhouettes, such as office supplies, furniture, and home appliances.

Fig. 2 shows an overview of the proposed method. First, the projection direction that adequately captured the geometry of the major surfaces of the object was determined by evaluating the normals of the point cloud (A1). The input point cloud was then projected onto planes perpendicular to the projection directions to create a set of silhouette images. Noise removal and hole completion were performed using image processing techniques considering symmetries, and a set of silhouette contour polylines was extracted (A2). Next, the polylines were simplified and regularized (A3). Finally, a set of primitives was generated by sweeping the contour polylines along the projection directions, and the simplified intersection of the primitives was extracted as a low-poly mesh (A4). The features of the proposed method include the fast and robust generation of a low-poly mesh with regularities using simple 2D images and contour processing from low-quality point clouds.

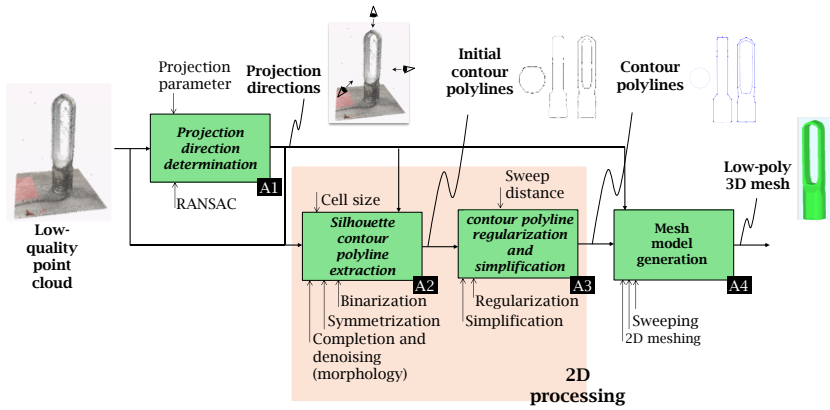


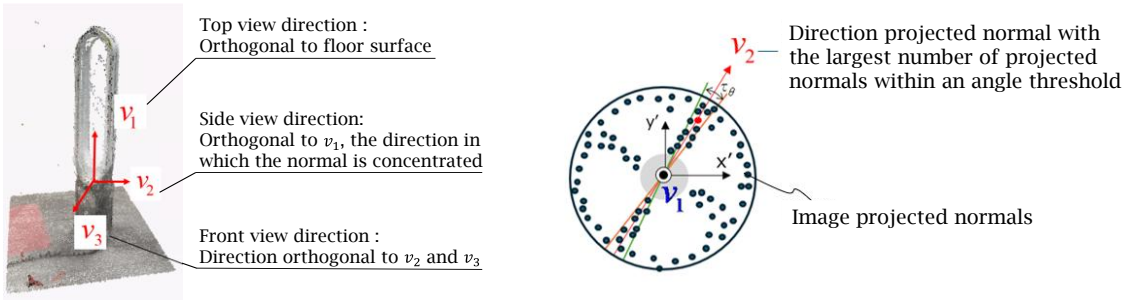

Fig. 2: Overview of the proposed method.

### 3.2 Projection Direction Determination

In this study, it is assumed that the geometry of the major surfaces of the object can be captured by the silhouettes from orthogonal three directions of the top, front, and side views of the input object, as shown in Fig. 3(a). The projection direction to obtain the top view is denoted by $\mathbf{v}_1$, and it is extracted as the normal of the floor detected by plane fitting using RANSAC [8,24]. The other projection directions should be determined such that the projections along the directions can catch the silhouettes of the major surfaces of the objects. Because major surfaces may include a relatively large number of points, the projection direction that can catch the geometry of the major surfaces on the silhouette can be considered as a direction perpendicular to the direction of a large number of normals. In the proposed method, because the second and third directions, $\mathbf{v}_2$ and $\mathbf{v}_3$ are orthogonal to $\mathbf{v}_1$, they are determined using projected point normals of the object on a plane $f$ perpendicular to $\mathbf{v}_1$. The normals that are nearly parallel to $\mathbf{v}_1$ should not be considered in the determination of $\mathbf{v}_2$ and $\mathbf{v}_3$. First, normals $\{\mathbf{n}_i\}$ that satisfy $|\mathbf{n}_i \cdot \mathbf{v}_1| < \tau_n$ are extracted and projected onto plane $f$. Second direction $\mathbf{v}_2$ is determined as the direction in which the projected normals $\{\hat{\mathbf{n}}_i\}$ are concentrated. This direction can be obtained by Equation (1), as shown in Fig. 3(b).

$$\mathbf{v}_2 = \arg\max_{\hat{\mathbf{n}}_i \in \hat{N}} |S_i|, \; S_i = \left\{ \hat{\mathbf{n}}_j \mid angle(\hat{\mathbf{n}}_i, \hat{\mathbf{n}}_j) < \tau_\theta, \hat{\mathbf{n}}_j \in \hat{N} \right\} \tag{1}$$

where, $\hat{N}$ is a set of projected normals, $angle(\boldsymbol{a}, \boldsymbol{b})$ is the angle between $\boldsymbol{a}$ and $\boldsymbol{b}$, and $\tau_\theta$ is a threshold. The third direction $\mathbf{v}_3$ that can catch the silhouette of major surfaces is calculated by the outer product of $\mathbf{v}_1$ and $\mathbf{v}_2$.




(a) (b)

Fig. 3: Projection directions: (a) Projection directions and (b) Side view direction.

### 3.3 Silhouette Contour Polyline Extraction

In this method, outliers in the point cloud are first removed using a point cloud filtering technique, followed by silhouette shaping of the image generated by projecting the point cloud. The shaping processes in this step are hole filling and symmetry recognition. Small holes can be filled by morphological processing; however, missing parts in large regions are difficult to complete. Therefore, a region-based hole filling was applied. Symmetry recognition is performed by a simple symmetry metric that uses inverted images. Using 2D projections makes each process more robust and efficient than 3D processing.

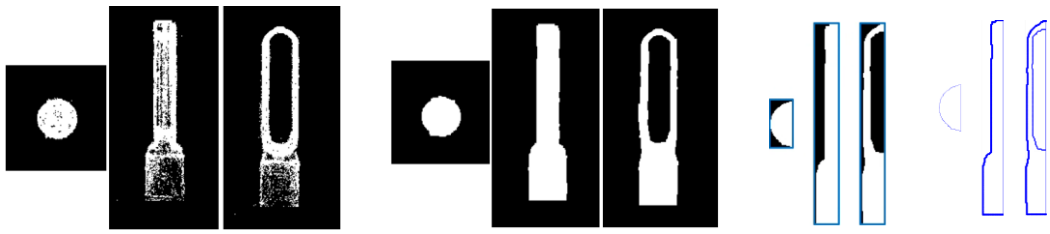
First, the SOR filter for noise removal [22] is applied to the input point cloud, the resulting point cloud is projected onto a plane perpendicular to the projection direction $v_d$ (where $d \in \{1,2,3\}$), and silhouette images $I_s^d$ are created. Here, each $I_s^d$ is a binary image, where white cells represent the foreground (object), including the projected points, and black cells represent the background, as shown in Fig. 4(a). Next, a closing operation based on the morphology is applied to $I_s^d$ to remove

small holes and interpolate the missing points. The holes are then detected as connected background pixels, and if the size of the holes is smaller than the given threshold $\tau_f$, the holes are filled. Finally, an opening operation based on the morphology is applied to remove noise, and the modified silhouette images $I_m^d$ are obtained, as shown in Fig. 4(b).

Subsequently, the symmetry of each $I_m^d$ is determined. First, each image size is adjusted to the AABB of the foreground of the modified images, and the degree of symmetry $r_{sym}$ is evaluated using Equation (2).

$$r_{sym} = \frac{|R_{org} \cap R_{mir}|}{|R_{org}|} \tag{2}$$

where $R_{org}$ is a pixel set of foregrounds of the image and $R_{mir}$ is that of the mirrored image. $r_{sym}$ is the proportion of common foreground pixels in the original and mirrored images. If $r_{sym} > \tau_s$ ($\tau_s$ : threshold), the silhouette is considered symmetrical, and the half image of the intersection between the original and mirrored images is extracted as $I_h^d$ as shown in Fig. 4(c). Finally, the contour polylines (point sequences) of $I_h^d$ for symmetric silhouettes and $I_m^d$ for the other silhouettes were extracted, as shown in Fig. 4(d).



(a) (b) (c) (d)

Fig. 4: Projected image and contour polyline: (a) Projected image, (b) Silhouette image, (c) Half image, and (d) Contour point sequences.

### 3.4 Contour Generation

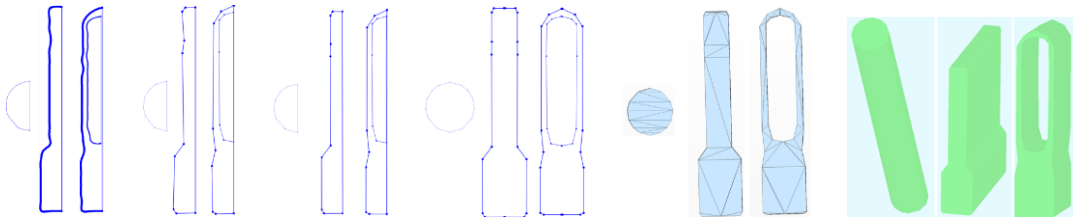
Extracted contours include undesired small bumps due to the rasterization of project points and scanning noises. Polyline smoothing is applied to remove these bumps on the contours, as shown in Fig. 5(a). In the proposed method, an extension of the $\lambda$ -$\mu$ algorithm [28] is used. Small bumps on the contour can be classified into two types: bumps from noise and rasterization and bumps from small shapes. Small shapes must be preserved, and the others should be removed. This can be performed by evaluating the reliability of the cell. Cells including a large number of projected points can be identified as a part of a small shape. In contrast, cells including the small number of projected points can be recognized as noise or the effect of rasterization. Therefore, in the proposed method, the strength of the contour smoothing for small bump removal is adaptively controlled to maintain the small shapes of the input object and to remove noise and rasterization effects.

In the $\lambda$ -$\mu$ algorithm, the smoothing strength can be adaptively controlled using the passband frequency $k$. Based on the above assumptions, the strength of smoothing, $k$, for a vertex $i$ on a contour polyline was calculated using Equation (3).

$$k(n_i) = \begin{cases} \beta \exp(n_i) + \alpha & n_i < \tau_n \\ 0.1 & otherwise \end{cases} \tag{3}$$

where $n_i$ denotes the number of projected points in the image cell corresponding to the vertex $i$ of the contour polyline. In the proposed method, to satisfy the conditions $k(1) \cong 0.05$ and $k(5) \cong 0.1$, parameters $\beta$ and $\alpha$ are set by $\beta = 0.00034$ and $\alpha = 0.04907$. $\tau_n$ is a threshold for the number of points in cell, and $\tau_n = 5$ is used in the experiment. In the smoothing iteration, weights for Laplacian, $\lambda$ and $\mu$, are determined as $\lambda = (k - \sqrt{k^2 - 6k + 10})/(3k - 5)$ and $\mu = (k + \sqrt{k^2 - 6k + 10})/(3k - 5)$, respectively. In the method, a weak smoothing operation is applied to vertices that have large $n_i$ by adjusting the coefficients in the iteration of the smoothing operation.

After polyline smoothing, simplified contour polylines were extracted using the Douglas–Peucker algorithm [7], as shown in Fig. 5(b). Subsequently, regularization was performed for the simplified contour polylines. In regularization, circular arcs are first reconstructed by vertex position modification, followed by parallelization and orthogonalization of the straight-line segments. The circular arcs of the polylines were detected by circle fitting using RANSAC [8,24]. The vertex positions of the inliers in circle fitting were moved on the detected circles along the circle radius.



(a) (b) (c) (d) (e) (f)

Fig. 5: Flow of primitive generation: (a) Smoothed polylines, (b) Simplified contour polylines, (c) Regularized polylines, (d) Entire contour polylines, (e) 2D triangular meshes, and (f) Primitives.

After circular arc modification, regularization was performed in parallel and orthogonally. The regularization was based on a simplified version of the method proposed by Takahashi et al. [27]. In this method, the parallel and orthogonal relationships are represented by the graphs shown in Fig. 6. In the graph, the line segments for parallelization are grouped on a node, and the orthogonal relationship is represented by arcs. Each node in the graph represents a line segment or set of line segments, where segments grouped in the same node are parallel to each other. Each edge in the graph represents an orthogonal relationship between connected nodes. The graph is constructed by a simple grouping of directional vectors of straight-line segments. The resulting graph consists of a set of nodes and two connected nodes, as shown in Fig. 6.

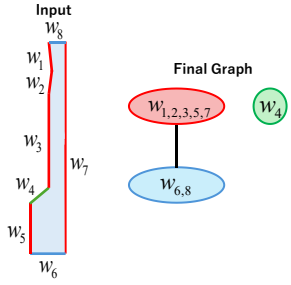



Fig. 6: Examples of graph representations of parallel and orthogonal relationships.

The first step of parallelization and orthogonalization is to find the best directional vectors of the line segments of the contours for each connected component $C$ in the graph. For parallelization, the first

problem is to find a directional vector $\mathbf{d}_a$ for modifying the line segment directions in the contours to be parallel in a node. This is performed simply by straight line fitting for the translated endpoints of the line segments. First, by applying a translation to the endpoints of each line segment, the center point of each segment becomes the origin. The directional vector $\mathbf{d}_a$ for a node can be determined via straight line fitting using the least-squares method for a set of transformed endpoints.

If $C$ contains other nodes, the other directional vector $\mathbf{d}_b$, which is orthogonal to $\mathbf{d}_a$, is determined using a similar method under the constraints of $\mathbf{d}_a \cdot \mathbf{d}_b = 0$. $\mathbf{d}_b$ can be performed using a process similar to that used for the first node. The constraint can be considered as applying 90-degree rotation to translated endpoints in the first node. Therefore, the orthogonal directional vectors $\mathbf{d}_a$ and $\mathbf{d}_b$ can be determined using the following formulas:

$$\mathbf{d}_a = \arg\min_{\mathbf{d}} \left\{ \sum_{i \in P_a} ((\mathbf{R}\mathbf{d}) \cdot \hat{\mathbf{p}}_i)^2 + \sum_{j \in P_b} (\mathbf{d} \cdot \hat{\mathbf{p}}_j)^2 \right\} \tag{4}$$

$$\mathbf{d}_b = \mathbf{R}\mathbf{d}_a \tag{5}$$

where $P_a$ and $P_b$ are the sets of vertices in each graph node, $\hat{\mathbf{p}}_i$ denotes the position of vertex $i$ after translation, and $\mathbf{R}$ is the 90-degree rotation matrix.

Next, the vertex positions of each line segment were modified according to the determined directional vectors. This process can be classified into the following three cases: The positions of the vertices modified in the circular arc reconstruction were assumed to be fixed.

1. If both endpoints are free to move, they are modified to a straight line in the specified direction, passing through the center of gravity.
2. If one of the endpoints is fixed, the position of the other vertex is modified such that the direction of the line segment becomes the derived direction.
3. If both endpoints are fixed, no modification of positions is performed.

If symmetrization is performed, the contour polylines are duplicated and mirrored to generate an entire contour polyline, as shown in Fig. 5(d).

### 3.5 Low-poly Mesh Generation

Primitives were created to generate the final meshes from the closed contour polylines in each projection direction. First, 2D triangular meshes of the silhouettes were created by applying 2D constrained Delaunay triangulation [11] to the resulting contours, as shown in Fig. 5(e). Primitives were then generated by sweeping each 2D mesh along the corresponding projection direction. Consequently, using the proposed method, three primitives were obtained, as shown in Fig. 5(f). Subsequently, a 3D triangular mesh was obtained by calculating the intersection of the three generated primitives. Finally, a mesh simplification method was applied to the resulting triangular meshes. A quadric-error-based mesh simplification method [10] was used in our implementation.

## 4 RESULTS AND EVALUATIONS

The iPhone 15 Pro and SCANIVERSE [23], which is a popular scanning application software, were used to obtain point clouds, and the proposed method was applied to the point clouds of some objects in the university's laboratory. The leftmost column in Fig. 7 shows the photos and point clouds for the target objects: a fan, trash can, and air cleaner, whereas the second column shows the generated 3D meshes with the numbers of vertices and triangles of the meshes.

In the point clouds acquired by the mobile phone and software, large-scale noise and a large lack of points at the sides of the objects can be observed, and different point densities are also observed. Although the quality of the input point clouds is quite low, the results showed that a large lacks of

points are appropriately filled, and low-poly meshes with regularities are generated using the proposed method. In addition, it was confirmed that the projections from the three directions derived from the point normals captured the geometries of the major surfaces of the objects.

The meshes obtained using the popular implicit surface reconstruction method, screened PSR [14], and 3D reconstruction software [23] are shown in the third and fourth columns of the figure. SPSR was applied by using default parameters on the implemented software [19]. The ball pivoting algorithm [3] implemented in free software [19] does not work for low-quality point clouds because of the large difference in the point densities. The resulting meshes contained several incorrect surfaces, inappropriate holes, and small, incorrect bumps on the surfaces. By contrast, low-poly meshes without such problems can be obtained using the proposed method. The computation time for mesh generation using the proposed method was less than 3 seconds on a desktop PC (CPU: Core i7-14700F).

The proposed method requires appropriate parameter settings and the results depend on the specified parameters. In particular, the thresholds for region-based hole filling, $\tau_f$, and symmetry detection, $\tau_s$, should be adjusted depending on the target object. In the current implementation, these parameters are determined interactively. In our experiments, a few tries and errors were sufficient to obtain the desired results. In addition, the image pixel sizes should be determined according to the noise level and point density of the given point clouds. In the experiments, the pixel size was set to 5 mm for all the examples. This parameter may vary according to the scanning device and software used. Automatic parameter determination should be included in future works.

The experimental results for the other objects are shown in Fig. 8. However, because of the nature of this method, some concave portions of the object cannot be recovered appropriately, and low-poly meshes that capture the major surfaces of the objects can be obtained. Improving the representation of the details of the resulting meshes by increasing the projection directions and using depth information from each view will also be included in future work.

## 5 CONCLUSIONS

In this paper, a simple method for generating low-poly meshes from low-quality point clouds was proposed. This method was based on the 2D projection of points, image and contour processing, and intersection extraction of primitives generated by contours. Symmetrization, noise removal, and hole filling were achieved in image processing, and regularization and simplification were realized in contour processing. Through experiments, we confirmed that the proposed method enabled the efficient generation of low-poly meshes from low-quality point clouds of simple small objects. Future work will include the improvement of detail reproducibility by increasing the projection direction, using depth information from each view, and automatic parameter settings.


Fig. 7: Resulting low-poly meshes and a comparison with existing methods.

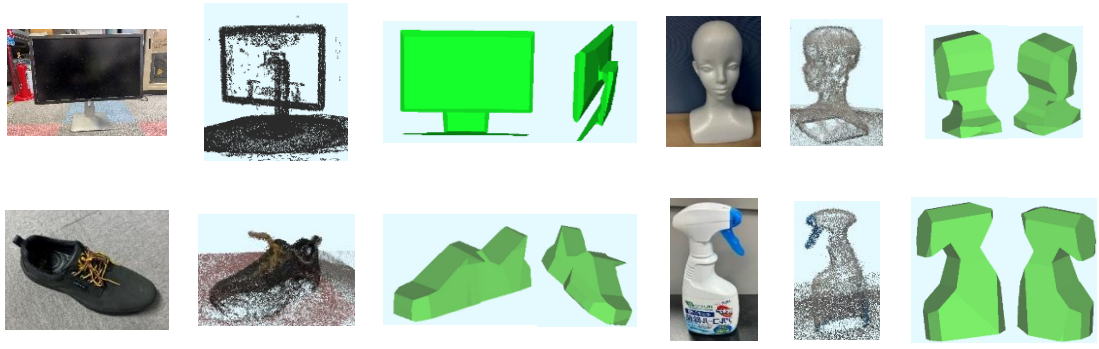


Fig. 8: Resulting low-poly meshes of other objects.

*Shinichi Sano,* https://orcid.org/0009-0003-1614-8536
*Hiroaki Date*, https://orcid.org/0000-0002-6189-2044
*Satoshi Kanai*, https://orcid.org/0000-0003-3570-1782

## REFERENCES

[1] Amenta, N.: The crust algorithm for 3D surface reconstruction, Proceedings of the fifteenth annual symposium on Computational geometry,1999,423-424. https://doi.org/10.1145/304893.30500
[2] Ben, M.; Pratul, P.S.; Mattew, T.; Jonathan, T.B.; Ravi, R.; Ren, N.: NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis, Communications of the ACM, 65(1), 2021, 99-106. https://doi.org/10.1145/3503250
[3] Bernardini, F.; Mittleman, J.; Rushmeier, H.; Silva, C.; Taubin, G.: The ball-pivoting algorithm for surface reconstruction, IEEE Transactions on Visualization and Computer Graphics, 5(4), 2002, 349-359, https://doi.org/10.1109/2945.817351
[4] Bernhard, K.; Georgios, K.; Thomas, L.; George, D.: 3D Gaussian Splatting for Real-Time Radiance Field Rendering, ACM Transactions on Graphics, 42(4), 2023, Article No.:139. https://doi.org/10.1145/3592433
[5] Botsch, M.; Kibbelt, L.; Pauly, M.; Aliez, P.; Levy, B.: Polygon Mesh Processing, AK Peters/CRC Press, 2010.
[6] Cohen-Steiner, D.; Alliez, P.; Desbrun, M.: Variational shape approximation, SIGGRAPH '04: ACM SIGGRAPH 2004 Papers, 2004, 905-914.
[7] Douglas, D. H.; Peucker, T. K.: Algorithms for the reduction of the number of points required to represent a digitized line or its caricature, Cartographica: The International Journal for Geographic Information and Geovisualization, 10(2), 1973, 112-122.https://doi.org/10.3138/FM57-6770-U75U-7727
[8] Fischler, M. A.; Bolles, R. C: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography, Communications of the ACM, 24(6), 1981, 381-395. https://doi.org/10.1145/358669.358692
[9] Gao, X.; Wu, K.; Pan, Z.: Low-poly Mesh Generation for Building Models, Proc. ACM SIGGRAPH 2022, 2022, Article No.: 3.

[10] Garland, M.; Heckbert, P.S.: Surface simplification using quadric error metrics, SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques, 1997, 209-216. https://doi.org/10.1145/258734.258849
[11] Jonathan, R.: Triangle: Engineering a 2D quality mesh generator and Delaunay triangulator, Lecture Notes in Computer Science, 1148, 2005, 203-222.
[12] Kanai, T.: MeshToSS: Converting Subdivision Surfaces from Dense Meshes, VMV '01: Proceedings of the Vision Modeling and Visualization Conference 2001, 2001, 325-332.
[13] Kazhdan, M.; Hoppe, H.; Bolitho, M.: Poisson surface reconstruction, Proceedings of the fourth Eurographics symposium on Geometry processing, 2006, 61-70.
[14] Kazhdan, M.; Hoppe, H.: Screened poisson surface reconstruction, ACM Transactions on Graphics, 32(3), 2013, Article No.: 29. https://doi.org/10.1145/2487228.2487237
[15] Kazhdan, M.; Chuang, M.; Rusinkiewicz, S.; Hoppe, H.: Poisson Surface Reconstruction with Envelope Constraints, COMPUTER GRAPHICS forum, 39(5), 2020, 173-182.
[16] Li, Y.; Chrysathou, Y.; Sharf, A.; Daniel, C.; Niloy, J.:GlobFit: Consistently Fitting Primithives by Discovering Global Relations, ACM Transactions on Graphics, 34(4), 2011, Article No.: 52. https://doi.org/10.1145/2010324.1964947
[17] Lorensen, W. E.; Cline, H. E.; Marching cubes: A high resolution 3D surface construction algorithm, ACM SIGGRAPH Computer Graphics, 21(4), 1987, 163-169. https://doi.org/10.1145/37402.37422
[18] Ma, X.; Keates, S.; Jiang, Y.; Kosinka, J.: Subdivision surface fitting to a dense mesh using ridges and umbilics, Computer Aided Geometric Design, 32(C), 2015, 5-21.
[19] MeshLab, https://www.meshlab.net/, Visual Computing Lab.
[20] Monszpart, A.; Mellado, N.; Brostow, G. J.; Mitra, N.J.: RAPter: Rebuilding man-made scenes with regular arrangements of planes, ACM Transactions on Graphics, 34(4), 2015, Article No.: 103.https://doi.org/10.1145/2766995
[21] Park, J.; Florence, P.; Straub, J.; Newcombe, R.; Lovegrove, S.: DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation, IEEE Conference on Computer Vision and Pattern Recognition, 2019.
[22] Rusu, R. B.; Cousins, S.: Point Cloud Library, IEEE International Conference on Robotics and Automation, 2011, 1-4. https://doi.org/10.1109/ICRA.2011.5980567
[23] Scaniverse, https://scaniverse.com/, Niantec.
[24] Schnabel, R.; Wahl, R.; Klein, R.: Efficient RANSAC for Point-Cloud Shape Detection, Computer Graphics Forum, 26(2), 2007, 214-226. https://doi.org/10.1111/j.1467-8659.2007.01016.x
[25] Sellán, S.; Jacobson, A.: Stochastic Poisson Surface Reconstruction, ACM Transactions on Graphics, 41(6), 2022, Article No.: 227. https://doi.org/10.1145/3550454.3555441
[26] Sulzer, R.; Marlet, R.; Vallet, B.; Landrieu, L.: A Survey and Benchmark of Automatic Surface Reconstruction From Point Clouds, IEEE Transactions on Pattern Analysis and Machine Intelligence, 47(3), 2025, 2000-2019. https://doi.org/10.1109/TPAMI.2024.3510932
[27] Takahashi, H.; Date, H.; Kanai, S.: Automatic Indoor Environment Modeling from Laser-scanned Point Clouds using Graph-based Regular Arrangement Recognition Proc. ICCBEI 2019, 2019, 368-375.
[28] Taubin, G.: A signal processing approach to fair surface design, Proc. SIGGRAPH 1995, 1995, 351-358. https://doi.org/10.1145/218380.218473