Journal of Electronic Research and Application, 2025, Volume 9, Issue 6
http://ojs.bbwpublisher.com/index.php/JERA
ISSN Online: 2208-3510
ISSN Print: 2208-3502

Research on Reverse Modeling of Parametric CAD Models from Multi-View RGB-D Point Clouds

Yangzhi Zhang*

School of Artificial Intelligence, Zhejiang Dongfang Polytechnic, Wenzhou 325000, Zhejiang, China

**Author to whom correspondence should be addressed.*

Abstract: Existing reverse-engineering methods struggle to directly generate editable, parametric CAD models from scanned data. To address this limitation, this paper proposes a reverse-modeling approach that reconstructs parametric CAD models from multi-view RGB-D point clouds. Multi-frame point-cloud registration and fusion are first employed to obtain a complete 3-D point cloud of the target object. A region-growing algorithm that jointly exploits color and geometric information segments the cloud, while RANSAC robustly detects and fits basic geometric primitives. These primitives serve as nodes in a graph whose edge features are inferred by a graph neural network to capture spatial constraints. From the detected primitives and their constraints, a high-accuracy, fully editable parametric CAD model is finally exported. Experiments show an average parameter error of 0.3 mm for key dimensions and an overall geometric reconstruction accuracy of 0.35 mm. The work offers an effective technical route toward automated, intelligent 3-D reverse modeling.

Keywords: CAD model; RGB-D point cloud; Reverse modeling; Geometric information; Region-growing algorithm

Online publication: December 16, 2025

1. Introduction

In smart manufacturing, industrial inspection, digital twins, and cultural-heritage digitization, rapidly and accurately acquiring a 3-D digital representation of a physical object—and further converting it into an editable, reusable computer-aided design (CAD) model—has become a core technology [1]. This physical-to-digital conversion process is known as reverse engineering or reverse modeling. Traditional reverse modeling relies heavily on expensive contact or laser scanning equipment and involves tedious, time-consuming pipelines that cannot meet modern industrial demands for efficiency and flexibility. Therefore, exploring a high-efficiency, low-cost, and automatic 3-D reverse-modeling method possesses significant theoretical value and broad application prospects [2].

This study first investigates accurate multi-view point-cloud registration and fusion to obtain high-quality 3-D data. It then focuses on a segmentation algorithm that combines color and geometric cues, and on geometric-feature recognition and parameter extraction based on graph neural networks or attention mechanisms, aiming

to intelligently identify basic geometric primitives and their mutual constraints. Finally, we examine how to translate these primitives and constraints into standard CAD-kernel-recognizable parametric modeling instruction sequences, enabling automatic or semi-automatic reconstruction of parametric CAD models.

2. Multi-view RGB-D data pre-processing and high-accuracy point-cloud reconstruction

Raw RGB-D data suffer from depth noise, motion blur, and pose misalignments among views. The goal of this stage is to produce a complete, accurate, and globally consistent 3-D point-cloud model.

2.1. Point-cloud denoising and filtering

For each captured RGB-D frame, the data are first converted into a 3-D point cloud. Let a single-frame point cloud be, where every point comprises 3-D coordinates and RGB color. We adopt a statistical outlier-removal filter to denoise the cloud. The algorithm computes, for every point, the mean distance to its k nearest neighbors, assumes the resulting distribution is Gaussian, and removes any point whose mean distance lies outside the range $\mu \pm \lambda\sigma$. The mean distance for a point p_i is calculated as

$$d_i = \frac{1}{K}\sum_{J=1}^{K}\left\|p_i - p_{ij}\right\| \tag{1}$$

Here, p_{ij} is the j nearest neighbor of point p_i, and d_i represents the average distance. Then, compute the mean d_i and standard deviation μ of all σ values, and remove any points that satisfy $d_i>\mu+\alpha\cdot\sigma$ or $d_i>\mu-\alpha\cdot\sigma$. α is the standard deviation multiplier, typically set to 1.0.

2.2. Multi-view point cloud registration

To align point clouds $\{P_1,P_2,...,P_M\}$ acquired from different viewpoints into a unified global coordinate system, we employ the classical Iterative Closest Point (ICP) algorithm and its variants. The registration task aims to determine the optimal rigid transformation B between two consecutive frames, the point cloud P_s (source) and point cloud P_t $T=[R|t]$, where R denotes the rotation matrix and t the translation vector, such that the transformed source cloud minimizes its deviation from the target cloud.

This objective is encoded by the registration loss function defined in Equation (2):

$$E(R,t) = \sum_{i=1}^{N} w_i \left\|\left(R\cdot p_s^i + t\right) - p_t^{c(i)}\right\|^2 \tag{2}$$

Here, p_s^i is the i point in the source cloud and $p_t^{c(i)}$ is its nearest-neighbor correspondence in the target cloud; w_i is a weighting factor. Minimizing this error function via singular-value decomposition (SVD) yields the optimal rotation R and translation. First, an initial transformation is estimated by coarse registration based on FPFH features, after which ICP refinement is applied to align the point clouds from all viewpoints incrementally [3].

2.3. Point-cloud fusion and reconstruction

After registration, direct concatenation of the clouds produces heavy overlap and non-uniform density. We therefore apply a voxel-grid filter to the merged cloud, downsampling it into a uniformly distributed model. Next, the Poisson surface-reconstruction algorithm converts the filtered cloud into a mesh suitable for visualization and

downstream computation. The method solves the Poisson equation whose general form is given in Equation (3):

$$\nabla^2 x = \nabla \cdot \vec{v} \tag{3}$$

Here, X is the indicator function (equal to 1 inside the object and 0 outside), ∇^2 denotes the Laplacian operator, and $\vec{v}$ is the gradient field approximated from the point-cloud normals. Solving this equation yields a smooth implicit surface X; an isosurface extraction then produces the final triangle mesh.

3. Point-cloud segmentation and geometric primitive recognition

This stage is the core of the proposed method: identifying the basic geometric entities—planes, cylinders, cones, etc., and their parameters from the reconstructed point cloud.

3.1. Region-growing-based geometric segmentation

Because CAD models are dominated by regular geometric features, we adopt an improved region-growing algorithm that uses both normal vectors and RGB color as growing criteria, allowing adjacent but distinct features to be separated more accurately. For every point, its normal is first computed [4]. The seed point with minimum curvature is selected, and a region is grown while the angle between a neighbor's normal and the region's average normal is below the threshold θ_{thresh} and the color difference is below the threshold C_{thresh}.

3.2. Geometric primitive parameter fitting

For each segmented cluster Ck, RANSAC is employed to robustly fit the underlying primitive, automatically discarding any remaining outliers.

To fit a plane, the general equation is $Ax + By + Cz + D = 0$. RANSAC randomly draws three points to instantiate a candidate model M(A B C D), counts the inliers whose distance to M is smaller than threshold d_{thresh}, and retains the model with the maximum number of inliers after a fixed number of iterations.

Point-to-plane distance:

$$d_{thresh}\ (p_i, M) = \frac{\left|A_{xi} + B_{yi} + C_{zi} + D\right|}{\sqrt{A^2 + B^2 + C^2}} \tag{4}$$

For other primitives such as cylinders, the model involves more parameters—axis origin, direction vector, radius—yet the rationale is identical: by deciding which primitive description best explains a cluster, both the type and its parameters are obtained.

3.3. Inferring inter-primitive constraints

Identifying isolated geometric primitives is insufficient for reconstructing a parametric model; it is essential to recover the spatial constraint relationships among them, such as parallel, perpendicular, concentric, coplanar, and equal-radius constraints. This paper proposes a constraint-relation reasoning module based on a graph neural network. All identified geometric primitives are $\{G_l, G_2, ..., G_L\}$ treated as nodes in a graph. The feature vector of every node encodes its type (e.g., plane, cylinder), its parameters—such as the plane normal, cylinder radius, and axis direction—and its spatial location. A graph neural network refines each node's representation by iteratively aggregating information from its neighborhood through a message-passing mechanism. Finally, an edge classifier operating on the refined node features predicts whether a specific constraint relationship exists between any pair of

nodes (primitives).

Equation (5) compactly formulates one layer of this GNN message-passing step:

$$h_v^{(l+1)} = \sigma\left(W^{(l)} \cdot \left(h_v^{(l)} \| \, AGGREGATE^{(l)}\left(\left\{h_v^{(l)}, \forall u \in N(v)\right\}\right)\right)\right) \tag{5}$$

Among them, is the feature vector of node v at the l-th layer, N(v) is the set of neighboring nodes of v, AGGREGATE is the aggregation function (sum, average), || represents vector concatenation, $W^{(l)}$ is the trainable weight matrix, and σ is the nonlinear activation function. Through multi-layer propagation, nodes can perceive the global geometric structure, thereby accurately inferring constraint relationships.

3.4. Automated generation of parametric CAD models

In this phase, the identified primitives and their constraint relationships will be transformed into a sequence of modeling commands executable by the CAD kernel [5].

First, determine a base feature (e.g., the largest plane) as the sketch reference plane based on the spatial positions of the primitives. Then, generate a modeling sequence following the order of "reference positioning–sketch drawing (projecting identified primitive boundaries)–feature operations (extrusion, rotation, scanning)–constraint application (e.g., fixed, vertical)." All dimensional parameters of the primitives, such as radius, length, angle, and constraint relationships, are replaced by parametric variables.

Due to errors in point cloud measurement and model fitting, the directly generated model may not fully satisfy the constraints. Therefore, we formulate an overall optimization problem. Let all geometric parameters to be optimized (dimensions, positions) be represented as vector X. The optimization objective is to minimize the overall distance between the reconstructed CAD model surface and the original point cloud, while satisfying all inferred geometric constraints.

The objective function is defined as follows:

$$Minx = \left(\sum_{i=1}^{N} d\left(p_{i,S}(X)\right)^2 + \lambda \sum_{J=1}^{C} \psi\left(constra\,\mathrm{int}_j(X)\right)\right) \tag{6}$$

Among them, $\sum_{i=1}^{N} d\left(p_{i,S}S(X)\right.$ represents the distance from point p_i in the original point cloud to the surface S of the CAD model defined by parameter X, ψ(*constra* int(X)) is the penalty term for constraint violation, and ë is the weight coefficient. By solving this optimization problem, a set of optimal CAD model parameters that strictly satisfy geometric constraints can be obtained, resulting in a precise and editable parametric CAD model.

4. Experimental results and analysis

4.1. Experimental environment and dataset

4.1.1. Experimental platform

The experiment was conducted on a workstation equipped with an Intel Core i7-12700K processor, 32GB of memory, and an NVIDIA GeForce RTX 3080 graphics card. The point cloud data acquisition device was an Intel RealSense D435i depth camera. The operating system is Ubuntu 20.04 LTS. The algorithm development framework adopts Python 3.8 and PyTorch 1.11.0, point cloud processing uses the Open3D library, and the parameterized CAD model generation part relies on the Open CASCADE geometry kernel. M1 is a

traditional commercial software process based on grid reconstruction and manual feature extraction, which uses CloudCompare for point cloud processing and imports it into SolidWorks for manual modeling. M2 is an improved version of Point2Cylinder, an end-to-end cloud reconstruction network based on deep learning, aimed at reconstructing basic geometric voxels.

4.1.2. Experimental dataset

Due to the scarcity of publicly available datasets containing raw RGB-D sequences and corresponding parameterized CAD models, this study independently constructed a test dataset. The dataset contains 20 typical mechanical part models with different geometric complexities, which are divided into three levels of complexity based on the number of geometric elements contained in the model and the complexity of the constraint relationships:

(1) Simple objects (S series) are mainly composed of a small number of planes with simple constraint relationships, such as cubes and L-shaped blocks.

(2) Medium complex objects (M series), including plane, cylinder, cone, and other basic elements, with common constraints such as parallel, perpendicular, concentric, perforated supports, and stepped axes.

(3) Complex objects (C-series) include a mixture of free-form surfaces and regular features, or objects with complex Boolean relationships, such as vortex paths and complex shells.

For each object, use RealSense D435i to collect RGB-D images from 8 uniformly distributed angles, and use reverse modeling to reconstruct the complete point cloud as the algorithm input.

4.2. Experimental results and analysis

4.2.1. Results of geometric primitive recognition and parameter extraction

First, the accuracy of geometric primitive recognition was evaluated. **Table 1** presents the statistics of geometric primitive recognition and parameter extraction accuracy. For simple (S series) and moderately complex (M series) objects, the geometric primitive recognition rate of the proposed method reached 100%, and the average parameter error PE was controlled within 0.3 mm, demonstrating high recognition accuracy and robustness. This indicates that the region growing segmentation algorithm combined with color and normal vectors and the RANSAC fitting strategy adopted in this paper are very effective for regular geometric objects. For complex objects C series, due to the presence of free-form surfaces that cannot be described by simple primitives, the recognition rate decreased, reaching a minimum of 86.7%, and the parameter error also increased significantly. This is expected, as the proposed method is mainly targeted at parametric models, and free-form surfaces require different representation methods. Overall, the method performs well on objects dominated by regular shapes.

Table 1. Statistics of geometric primitive recognition and parameter extraction accuracy

Object number	Actual quantity	Recognized quantity	Recognition rate (%)	Average step error PE (mm)
S1	6	6	100.0	0.12
S2	8	8	100.0	0.15
S3	7	7	100.0	0.11
S4	9	9	100.0	0.13
S5	6	6	100.0	0.14
S6	7(6P,1C)	7	100.0	0.18
M1	9(7P,2)	9	100.0	0.21
M2	10(P,4C)	10	100.0	024

Table 1 (Continued)

Object number	Actual quantity	Recognized quantity	Recognition rate (%)	Average step error PE (mm)
M3	11(8R,3C)	11	100.0	0.26
M4	12(9P,3C)	12	100.0	0.23
M5	10(7P,2C,15F)	10	100.0	028
M6	13(10P,3C)	13	100.0	0.25
C1	15(10P,4C,1F)	13	86.7	0.52
C2	16(12P,4C)	16	100.0	029
C3	14(9P,4C,1F)	12	86.7	0.48
C4	18(14P,4C)	18	100.0	0.30
C5	16(10R,5C,1F)	14	87.5	0.55
C6	17(1P,5C,1F)	15	88.2	0.50

4.2.2. Comparative analysis of overall reconstruction accuracy and efficiency

Figure 1 compares the overall reconstruction accuracy and efficiency, comparing the performance of the proposed method, traditional manual methods, and deep learning in terms of overall geometric accuracy (GA) and reconstruction time (RT). The average GA of the proposed method is 0.35 mm, significantly better than the 0.61 mm of the deep learning method. This is because deep learning networks tend to produce smoothing effects on complex shape details, resulting in a loss of accuracy. Although the proposed method is slightly lower than the accuracy of experienced engineers manually operating M1, which is 0.28 mm, the gap is within an acceptable range of 0.07 mm, fully meeting the needs of engineering applications.

The advantage of the method proposed in this paper is extremely evident in efficiency. The average reconstruction time is only 158 seconds, while the traditional manual method requires an average of 2,170 seconds, representing a more than 10-fold improvement in efficiency. Compared with deep learning, although the method proposed in this paper is slightly slower, it achieves a significant improvement in accuracy. This indicates that the method proposed in this paper achieves a good balance between accuracy and efficiency, enabling high-quality automated modeling.

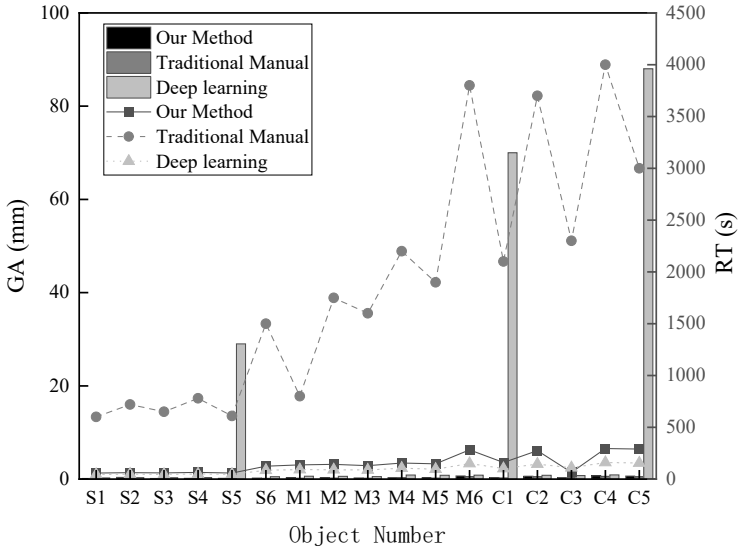

Figure 1. Comparison of overall reconstruction accuracy and efficiency

4.2.3. Analysis of the validity of constraint relation recognition

The core of the parametric model lies in constraints. The results of geometric constraint relationship recognition are shown in **Table 2**, with recall rate and precision rate as evaluation indicators. The constraint reasoning module based on GNN in this paper performs well overall. The average recall rate of constraint recognition reaches 94.6%, and the average precision rate is as high as 97.1%. This means that most of the true constraint relationships have been successfully identified, and most of the identified constraint relationships are correct. A few errors mainly occur on complex objects. When there are slight errors in the fitting of primitives themselves, GNN may produce ambiguity in the judgment of some critical states. However, overall, this module can reliably reconstruct the topological structure of the model.

Table 2. Recognition results of geometric constraints

Series	Sample number	Recall rate (%)	Precision (%)
S Series (simple)	S1-S6	96.8	98.5
M Series (moderately complex)	M1-M6	95.2	97.3
C Series (complex)	C1-C6	92.1	95.6
Overall average	-	94.6	97.1

The method proposed in this paper can efficiently and automatically reconstruct high-precision parameterized CAD models from multi-view RGB-D point clouds. On regular objects, its accuracy is close to manual modeling, while its efficiency is improved by an order of magnitude. Although there are limitations in processing free-form surfaces, this positioning is in line with the main application scenarios of parameterized modeling. This method provides a practical and automated solution for reverse engineering.

5. Conclusion

This paper focuses on the automatic reverse reconstruction from multi-view RGB-D point clouds to parametric CAD models, and proposes a complete technical solution. The main research conclusions are as follows:

(1) This article combines multi-view point cloud reconstruction, geometric primitive recognition, constraint relationship reasoning, and parametric model generation to achieve automatic conversion from low-level sensor data to high-level CAD models, significantly reducing the dependence of traditional methods on human intervention.

(2) The improved region growing segmentation algorithm and RANSAC fitting strategy adopted in this paper perform well for objects composed of regular geometric features. Experiments show that the method achieves a 100% recognition rate for simple and moderately complex objects, with an average parameter error controlled within 0.3 mm, laying a solid foundation for high-precision parametric reconstruction. Compared with traditional manual modeling, the proposed method achieves an average geometric error of 0.35 mm in reconstruction accuracy and improves modeling efficiency by an order of magnitude, providing a practical solution to the efficiency bottleneck problem in reverse engineering.

In summary, the method in this paper fully verifies the feasibility of achieving efficient and high-precision parametric model reverse modeling based on consumer-grade RGB-D sensors, which has positive significance for promoting the popularization and application of reverse engineering technology in small and medium-sized enterprises.

Disclosure statement

The author declares no conflict of interest.

References

[1] Vido M, de Oliveira Neto GC, Lourenço SR, et al., 2024, Computer-Aided Design and Additive Manufacturing for Automotive Prototypes: A Review. Applied Sciences, 14(16): 7155.

[2] Thamaraimanalan T, Ramalingam S, 2024, Hybrid Artificial Neural Network-Based Grasshopper Optimization Algorithm for Anomaly Detection in Wireless Body Area Networks. IETE Journal of Research, 70(4): 3738–3752.

[3] Wang Q, Jing W, Chi K, et al., 2024, Cross-Difference Semantic Consistency Network for Semantic Change Detection. IEEE Transactions on Geoscience and Remote Sensing, 62: 1–12.

[4] Fan L, Yu M, Hu W, et al., 2024, Reducing Charge-Recombination Losses in Photovoltaic Cells by Spontaneous Reconstruction of n/p Homojunction in a Monolithic Perovskite Film Using Black Phosphorus Nanosheets. Chemical Engineering Journal, 479: 147861.

[5] Zhang C, Polette A, Pinquié R, et al., 2025, Reinforcement Learning-Based Parametric CAD Models Reconstruction from 2D Orthographic Drawings. Computer-Aided Design, 2025: 103925.