WindPoly: Polygonal Mesh Reconstruction via Winding Numbers

Xin He, Chenlei Lv, Pengdi Huang, and Hui Huang(✉)

Visual Computing Research Center, Shenzhen University, Shenzhen, China
hhzhiyan@gmail.com

Abstract. Polygonal mesh reconstruction of a raw point cloud is a valuable topic in the field of computer graphics and 3D vision. Especially to 3D architectural models, polygonal mesh provides concise expressions for fundamental geometric structures while effectively reducing data volume. However, there are some limitations of traditional reconstruction methods: normal vector dependency, noisy points and defective parts sensitivity, and internal geometric structure lost, which reduce the practicality in real scene. In this paper, we propose a robust and efficient polygonal mesh reconstruction method to address the issues in architectural point cloud reconstruction task. It is an iterative adaptation process to detect planar shapes from scattered points. The initial structural polygonal mesh can be established in the constructed convex polyhedral space without assistance of normal vectors. Then, we develop an efficient polygon-based winding number strategy to orient polygonal mesh with global consistency. The significant advantage of our method is to provide a structural reconstruction for architectural point clouds and avoid point-based normal vector analysis. It effectively improves the robustness to noisy points and defective parts. More geometric details can be preserved in the reconstructed polygonal mesh. Experimental results show that our method can reconstruct concise, oriented and faithfully polygonal mesh that are better than results of state-of-the-art methods.

Keywords: Polygonal mesh · Polygon-based winding number strategy · Structural reconstruction

1 Introduction

As an efficient representation of a three-dimensional object, a polygonal mesh has significant advantages in the shape reconstruction task. It carries structured face information with geometric consistency while significantly reducing data volume. Especially for architectural point cloud representation, a polygonal mesh provides accurate and concise geometric structures that are useful for visualization and feature analysis in urban scenes.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/978-3-031-72970-6_17.

A. Leonardis et al. (Eds.): ECCV 2024, LNCS 15105, pp. 294–311, 2025.
https://doi.org/10.1007/978-3-031-72970-6_17

The mainstream reconstruction technical routes include two-step mesh reconstruction and low-poly meshing directly. For the first route, the related methods attempt to reconstruct complex mesh from point cloud to keep more geometric details [3,17,19,20,42,45]. Then, they employ simplification strategies [25,39] to achieve concise meshes for data compression. However, some important geometric features are broken by the simplification to a certain extent. In addition, the performance of the reconstruction is inevitably reduced for incomplete point clouds. For the second route, the methods [1,24,27,35] decompose the 3D space for the incomplete point cloud and directly establish a polygonal mesh to provide a concise structural representation. But they are sensitive to outliers of raw point cloud and lose some accurate geometric features.

In this paper, we propose a robust polygonal mesh reconstruction method to implement low-poly meshing for architectural point clouds. It includes three core components. Firstly, we detect polygonal planes from the raw point cloud, which is inspired by assembling-based surface reconstruction [35]. Secondly, we design an adaptive spatial partitioning scheme that iteratively segments the internal space represented by achieved planes. It is used to control the scope of planar intersection. A set of convex polyhedrons is constructed to fill the internal space. With the polygon-based winding numbers optimization, the polygonal mesh is constructed from the convex polyhedrons, which captures concise representation and more geometric details from noisy and unorganized point clouds. The contributions can be summarized as:

- We present a polygonal plane detection method that clusters points into regular planes without normal vector analysis. It is robust to outliers and noisy points while providing a structural geometric representation for the incomplete and unorganized point cloud.
- We design an adaptive spatial partition to establish a set of convex polyhedrons, which are used to represent the internal space of the architectural point cloud. A large number of plane-based intersection calculations are avoided which improve efficiency and enhance the accuracy of internal geometric details.
- We propose a polygon-based winding numbers optimization to achieve the final polygonal mesh. The optimization strategy fully inherits the advantages of the winding number for surface orientation. At the same time, the strategy considers the consistency constraint with the original input point cloud. It further enhances the quality of the reconstructed polygonal mesh.

2 Related Work

Related methods for polygonal mesh reconstruction can be concluded into three parts: mesh reconstruction, mesh simplification, and low-polygon meshing.

2.1 Mesh Reconstruction

Basically, the mesh reconstruction can be divided into two categories: explicit surface reconstruction and implicit one. The explicit surface reconstruction

is to establish continuous representations for discrete forms by establishing explicit local neighborhood. Representative solutions include ball pivoting [2], scale space [10], Delaunay triangulation [8], and voxel-based reconstruction [41]. Following the development of deep learning, many researchers attempt to learn the prior knowledge from modeling experiences to guide 3D reconstruction [7,15,16,18,43,48]. Compared to the explicit surface reconstruction, the implicit one solves an implicit function, which defines the surface as the zero-level set of the function. Some representative methods include marching cubes [29,36], Poisson surface reconstruction [17,19–21], and radial basis function [5,34,46]. They are more flexible and can handle topologically complex shapes.

2.2 Mesh Simplification

To preserve more geometric features, reconstructed meshes usually carry more vertices and faces which increase the computational cost of mesh-based rendering and analyzing. Therefore, mesh simplification schemes are proposed to compress the 3D model. Such schemes include geometric-based approximation [4,9,25], Delaunay-based remeshing [38,49], structural simplification [14,39], hierarchy strategy [26], and intrinsic analysis [28,32]. These schemes can approximate a polygonal mesh or reconnect a simplified mesh from the original complex one while keeping some important geometric features. However, these methods require high-precision mesh input, which makes the processing flow cumbersome and unstable.

2.3 Low-Poly Meshing

To address the limitation of mesh simplification, low-poly meshing is proposed to directly establish concise polygonal meshes based on raw inputs. Most representative methods utilize the idea of structural reconstruction to implement the meshing process, including building blocks [27,33], structure-preserving [24], surface elements [22] and plane hypothesis [1,11,12,35]. These methods extract basic representation elements from raw point clouds to construct polygonal meshes while approximating objects from incomplete geometric structures. However, there are limited by normal dependency and lower computational efficiency.

3 Methodology

Overview. The proposed WindPoly can be regarded as a low-poly remeshing scheme. It can be concluded as three parts: polygonal plane detection, adaptive spatial partitioning for convex polyhedron generation, and polygon-based winding numbers optimization. The polygonal plane detection is used to detect primitive planes from the raw point cloud in order to achieve initial structure information. Then, the convex polyhedron generation implements internal structure fitting by using adaptive spatial partitioning. This results in a set of coarse polyhedral elements without correct directions. To achieve the final polygonal

mesh, the polygon-based winding numbers optimization is implemented to orient polyhedral elements with global consistency. Some ambiguous geometric structures are improved during the orientation and consequently obtain more precise internal geometric details. The pipeline is shown in Fig. 1. In the following parts, we explain the implementation details.

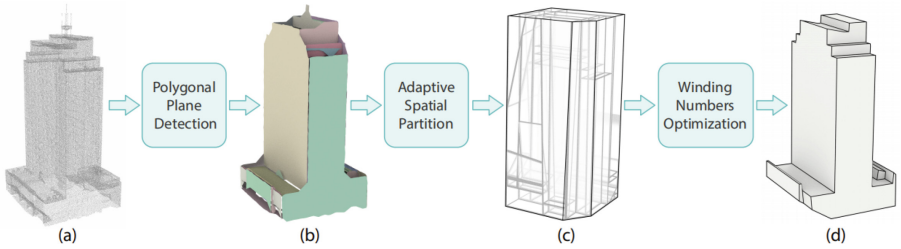

Fig. 1. Pipeline of WindPoly. (a) raw point cloud, (b) detected polygonal planes, (c) adaptive spatial partition, (d) output by polygon-based winding numbers optimization.

3.1 Polygonal Plane Detection

The scanned raw point cloud inevitably carries noisy points and outliers with random distributions. Therefore, we employ a pre-processing step to increase the robustness for raw point clouds. Firstly, we simplify the raw point cloud by Poisson resampling. It reduces the scale of the point cloud and optimizes its densities to be uniform. Then, we remove the outliers based on an automatic neighbor analysis program that is similar to the solution in [31]. Let p_i represent a point in a point cloud P, $N(p_i)$ is the neighbor set of p_i which is achieved by KNN. The judgement of outlier set $\{p_o\}$ can be formulated as

$$\{p_o\} = \{p_i | p_i \notin N(p_j), p_j \in N(p_i), p_i, p_j \in P\}, \tag{1}$$

where p_j is the neighbor point of p_i, and p_i is judged as an outlier when it does not belong to neighbor sets of its neighbors. The formulation is established based on the manifold compactness.

Based on the pre-processed point cloud, we implement polygonal plane detection for raw point clouds. The usual approach is to estimate the normal vector of each points and cluster them to fit related plane [37]. It is a normal vector dependency scheme with limited robustness. To implement a scheme with normal vector independence, we present a practical solution based on existing methods. We firstly extract candidate planes by fitting planar primitives (FPP) [50]. It extracts basic planes based on primitive configuration achieved by

$$U(x) = \omega_f U_f(x) + \omega_s U_s(x) + \omega_c U_c(x), \tag{2}$$

Algorithm 1: Polygonal Plane Detection

Input : Pre-processed point cloud P
Output: Polygonal plane set $\{F\}$

```
Initialization: {F}_c ← FPP(P)
foreach plane F_i ∈ {F}_c do
    foreach plane F_j ∈ {F}_c, j ≠ i do
        if PRR(F_i, F_j) then
            Combine F_i into F_j
            Update {F}_c
        end
    end
end
{F} ← {F}_c
```

where U_f, U_s, and U_c represent fidelity, simplicity and completeness energies with related weights ω_f, ω_s, and ω_c, which describe the clustering property in local neighborhoods. Then, we check the planes and combine them according to the plane refinement regulation (PRR) mentioned in [35]. It can be represented as

$$\begin{aligned} acos\langle n_i, n_j\rangle &< \theta_{the}, \\ N_t &= min(|F_i|, |F_j|)/5, \end{aligned} \tag{3}$$

where n_i and n_j are normal vectors of planes F_i and F_j, $|F_i|$ and $|F_j|$ are points belonging to the two planes, θ_{the} (10° by default) and N_t are control parameters. Once the common point number between two planes is larger than N_t and simultaneously satisfies the first condition in Eq. (3), the plane-based combination is triggered, the two related planes are merged. The required planes are achieved until all candidate planes are checked and combined. The implementation of the method can be concluded in Algorithm 1.

3.2 Adaptive Spatial Partition

Based on detected polygonal planes, we propose an adaptive spatial partition to generate convex polyhedrons for internal structure perception. It inherits the idea of iteratively generating polyhedrons in the convex polyhedral space while using a concise strategy to fit the internal structure without normal vector assistance. The method includes two basic components: convex polyhedral space construction and iterative convex polyhedron searching.

The convex polyhedral space is the convex hull of the object, which is constructed by the external planes selected from the primitive elements. The external plane is detected based on the regulation that is all other primitive elements should be located in the same side of the plane. An instance is shown in Fig. 2. It should be noticed that some outliers and inaccurate planes take influences for external plane judgement. To improve the robustness, we implement an additional check for the intersection of two planes. Let plane F_i to be

a cross plane of F_0, we compute the α shape to achieve the boundary point set $\{p\}_b^0$ of F_0. The α shape represents a point-based region that describes the enveloping shape of associated points. We check points in $\{p\}_b^0$ and delete ones that satisfy $dist(p_j^0, F_i) < \sigma$, where $dist$ is the distance between p_j^0 and F_i. If all the remaining boundary points of $\{p\}_b^0$ are located on one side of F_i, then the entire plane F_0 is judged to be on the side of F_i. The additional check improves the practicality of external plane detection.

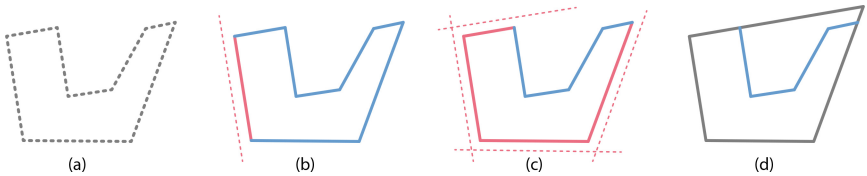

Fig. 2. External plane detection in 2D vision. Gray dotted lines (a) represent the raw point cloud. Red lines represent the external planes (b), blue lines are the other planes located in same side of the external planes (c). Finally, the convex polyhedral space represented by external planes is achieved (d). (Color figure online)

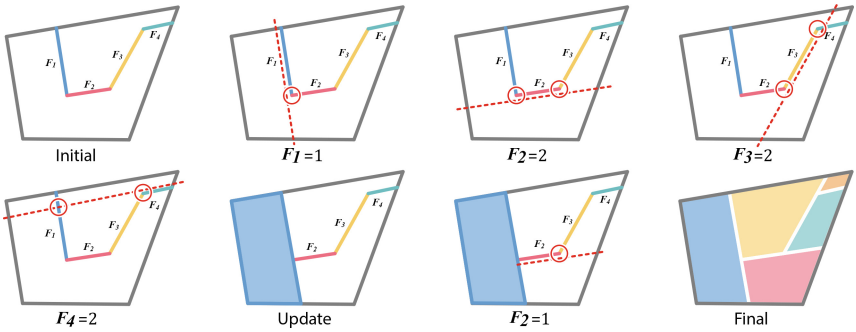

Fig. 3. An instance of adaptive spatial partition in 2D vision. In the initial step, intersection numbers of related planes $F_1 \sim F_4$ are 1, 2, 2, 2. After partition according to the regulation, the convex polyhedral space is divided into a set of convex polyhedrons.

In the convex polyhedral space, we establish convex polyhedrons to perceive the internal structure. PolyFit [35] has developed a solution that implements exhaustive partition to generate convex polyhedrons. However, it requires redundant intersection calculations for all planes with poor robustness. To solve the problem, we employ the adaptive spatial partition to segment the convex polyhedral space. Each internal plane is expanded and intersects with its spatial

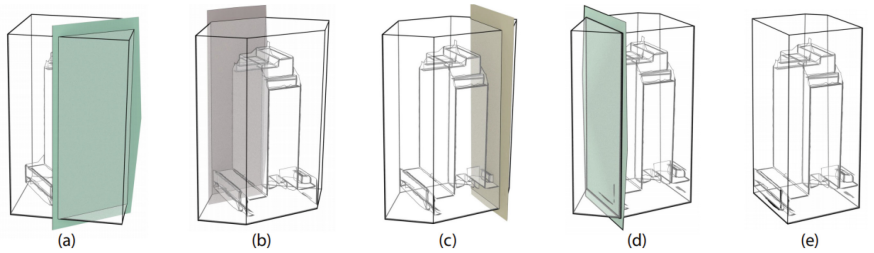

Fig. 4. An instance of adaptive spatial partition in 3D vision. According to the separating plane searching (a)-(d), the 3D structure can be achieved with internal geometric details (e).

Algorithm 2: Adaptive Spatial Partition

Input : Internal plane set $\{F\}_{int}$
Output: Convex polyhedron set $\{CP\}_c$
Initialization
$\{num\}_{int} \leftarrow$ intersection numbers of $\{F\}_{int}$
`AdaptiveFun`($\{F\}_{int}$, $\{num\}_{int}$):
Search the F_i from $\{F\}_{int}$ with minimum value from $\{num\}_{int}$
Cut convex polyhedral space into two sub-spaces
if *sub-space*$_{1,2}$ *has planes* **then**
  $\{F\}_{s1,2} \leftarrow \{F\}_{int}$
  `AdaptiveFun`($\{F\}_{s1,2}$, $\{num\}_{int}$)
end
else if then
  $\{CP\}_c \leftarrow$ sub-space$_{1,2}$
end
return;

sub-region. Benefited from the adaptive spatial partition, the efficiency of convex polyhedron generation can be significantly improved. The adaptive spatial partition can be concluded as an iterative scheme.

As an initial step, the internal planes are collected into the set $\{F\}_{int}$. Intersection numbers $\{num\}_{int}$ between each internal plane and other internal ones are computed and stored at the same time. Then, we select the separating plane F_i with minimum intersection number of $\{num\}_{int}$. According to the F_i, the original convex polyhedral space represented by external planes are divided into two parts. We iteratively search for separating planes in the corresponding part and continuously segment the subspaces until all planes are checked. Each convex polyhedral subspace corresponds to a convex polyhedron. Finally, a set of convex polyhedrons can be constructed. The intersection calculations between internal planes are controlled in the related subspace. It avoids redundant calculations while improving accuracy. The adaptive spatial partition is concluded in Algorithm 2.

For the separating plane searching, if there are multiple planes with same minimum value, we select the one with the largest area. In Fig. 3, we show an instance for the adaptive spatial partition. The partition process tends to search the subspaces from the outside to the inside in the convex polyhedral space (Fig. 4). After that, a coarse structural representation based on a set of polyhedrons is achieved which has detected the internal regions.

3.3 Polygon-Based Winding Numbers

The convex polyhedrons take redundant faces for polygonal mesh representation, which should be removed for the final polygonal mesh representation. Based on the generated convex polyhedrons, we introduce the third part that is to employ winding numbers optimization to determine the orientation of the candidate faces and decide whether to remove the related polyhedrons. As a mature orientation strategy, winding numbers are widely used in recent works [13,44], which optimize a scalar field to define the inside and outside of a closed surface. Generally, such optimization is processed on redundant points which requires a huge computation cost. In our framework, we propose a polygon-based winding number strategy that orients faces directly. Fewer points are used for the computation that significantly improve the efficiency. It can be represented as

$$w(q) = \sum_{i=1}^{N} a_i \frac{(p_i - q) \cdot n_i}{4\pi \left\| p_i - q \right\|^3}, \tag{4}$$

where q is a polyhedron centroid, p_i is the center of face F_i of related polyhedron, a_i is the area of F_i, and n_i is the face-based normal vector. Then, we complete the redefinition of parameters for winding numbers. The value of $w(q)$ can be computed which represents the inside or outside direction. The numerical distribution of winding numbers corresponds to the global consistency of normal vectors, which is suitable for orientation.

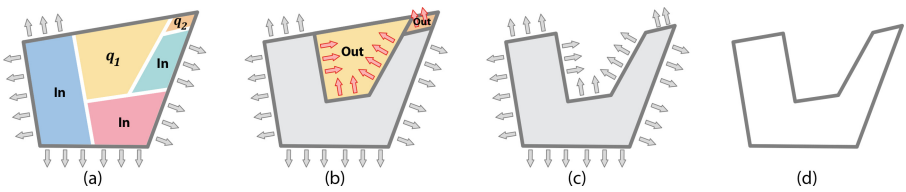

Fig. 5. An instance of polygon-based winding numbers optimization. According to the binary labels of related centroids, the polygonal mesh can be extracted from the convex polyhedron set.

Let $\{CP\}_c$ represent the convex polyhedron set, we collect candidate faces $\{F\}_{cp}$ from $\{CP\}_c$, which meet the condition $A(F_i)_\alpha / A(F_i) > T_r$, ($A(F_i)_\alpha$ is the α shape area of F_i, $A(F_i)$ is the area of F_i, T_r is the control threshold). Then we

search the outside faces $\{F\}_{out}$ from candidate faces based on the convex hull of the model, and assign related normal vectors $\{N\}_{out}$ to the outside of the convex hull. A group of faces with normal vectors for winding number computing has been obtained as initialization ($p_i \in \{F\}_{out}, n_i \in \{N\}_{out}$ for Eq. (4)). Next, we define a judgement energy to determine whether a polyhedron should be removed, which can be represented as

$$E_{dir} = W(\{q\}) + V(\{q\}), \tag{5}$$

where q represents the polyhedron centroid mentioned in Eq. (4). The purpose of Eq. (5) is to achieve a set of labels of $\{q\}$ by minimizing the direction-based energy E_{dir}. Once the label of the polyhedron centroid is provided, the judgement of the related convex polyhedron can be processed. The optimization consists of two terms, $W(\{q\})$ and $V(\{q\})$. The first term $W(\{q\})$ employs the winding numbers optimization to compute the face-based direction energy. It provides a geometrically consistent optimization method for determining the optimal orientation combination of polyhedrons, which can be represented as

$$W(\{q\}) = \sum_{q_k \in \{q\}} \widetilde{w}(q_k), \tag{6}$$

$$\widetilde{w}(q_k) = \begin{cases} 1 - w(q_k), q_k \in \{q\}_{out} \\ w(q_k), q_k \in \{q\}_{in} \end{cases}, \tag{7}$$

where q_k is a polyhedron centroid, $q_k \in \{q\}$, $\widetilde{w}(q_k)$ is used to normalize winding number $w(q_k)$ for optimization, $\{q\}_{in}$ and $\{q\}_{out}$ are subsets of $\{q\}$ with related binary labels, which determine whether the polyhedron should be retained. Once the label of q_k is decided, the first term of direction energy can be achieved. The second term is used to check fitting degree between faces and original point cloud, which assists the determination of direction judgement. It can be computed by

$$V(\{q\}) = \sum_{F_l \in \{q_i, q_j\}} (1 - A_\alpha(F_l)/A(F_l)), \tag{8}$$

where F_l is a common face between the adjacent convex polyhedrons q_i and q_j, $A_\alpha(F_l)$ means the area of α shape of points related to the F_l, $A(F_l)$ is the area of F_l, and the value of $A_\alpha(F_l)/A(F_l)$ can represent the coverage rate. This energy term represents the consistency between original point cloud and boundary faces of convex polyhedrons directly. The optimization is implemented by a max-flow algorithm that has been used in KSR [1]. The corresponding binary labels of $\{q\}$ can be obtained when the direction energy is minimized. An instance is shown in Fig. 5. In each iteration of the optimization strategy, the orientation of the candidate faces located at the boundary can be determined. Such faces are used to calculate E_{dir} that labels the remainder polyhedrons in the next iteration. More details are described in the supplementary material.

Benefiting from the accurate determined directions for polyhedron set, a more accurate internal structure (Fig. 6) can be checked and processed by the polygon-based winding numbers optimization. Related centroids involved in the optimization correspond to the number of polyhedrons, which is much less than the

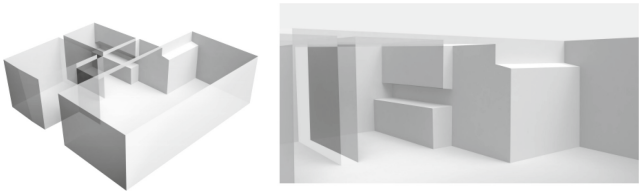

Fig. 6. Visualization of internal structure reconstructed by WindPoly.

original points. Compared to the traditional winding numbers optimization, our method significantly improves efficiency.

4 Experiment

We evaluate the performance of WindPoly in the polygonal mesh reconstruction task. The experimental machine is equipped with Intel i9-13900K, 128GB RAM, RTX4090, with Windows 10 as the operation system and Visual Studio 2019 as the development platform. The experiments include the following parts: 1. we introduce the test dataset and explain some selected metrics to prove quantitative analysis for the reconstruction; 2. we compare different reconstruction methods to report and visualize the advantages of WindPoly; 3. we discuss some potential limitations of WindPoly in practice.

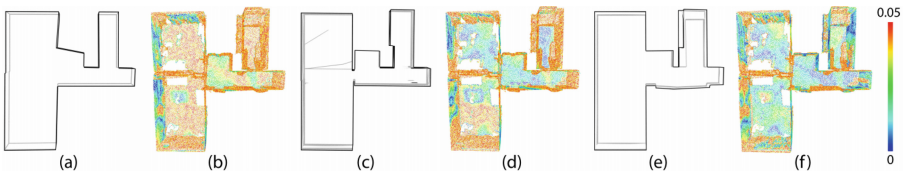

Fig. 7. Reconstruction results and color maps of mapping distances by different methods. (a, b) PolyFit, (c, d) KSR, (e, f) WindPoly.

4.1 Dataset and Metrics

The test point clouds are collected from ABC dataset [23], PolyFit dataset [35], UrbanBIS [47], and BuildingNet [40], which reflect the reconstructed geometric details of different levels. The ABC dataset includes small-scale workpiece models with regular point distributions and clear geometric details. The PolyFit and BuildingNet datasets contain architectural point clouds. The UrbanBIS is a large scale city scene dataset with real scanned architectural point clouds. Based on the data scale mentioned in [14], we select 300 models (100 models from ABC

dataset and 200 models from other architectural datasets) with representative holes or smooth surfaces to evaluate the performance of the reconstruction for structured representation of geometric information.

The quantitative metrics should be established from two perspectives: geometric consistency and data compression efficiency. The geometric consistency can be represented by Hausdorff distance and mean distance, which characterizes the matching degree from points to the polygonal plane. For data compression efficiency, we directly report the point and face numbers of the reconstructed polygonal mesh. It should be noticed that the quality of the data compression should related to the geometric consistency. If the reconstruction result just contains one point, the better performance of data compression means nothing. Therefore, we multiple the Hausdorff distance and simplification rate to be a multiple estimation for fair evaluation.

4.2 Comparisons

Based on the collected dataset and related metrics, we evaluate the performance of different reconstruction methods, including PolyFit [35], KSR [1], IPSR [17], LowPoly [14], and R-LowPoly [6]. Such methods cover the current mainstream solutions. However, the LowPoly and R-LowPoly can not reconstruct mesh from a raw point cloud directly. We employ a voxel-based Delaunay triangulation method [30] to achieve an initial mesh at first. The IPSR achieves a high-quality mesh without data compression. To provide a fair comparison, we employ the simplification from VCG library to concise the mesh.

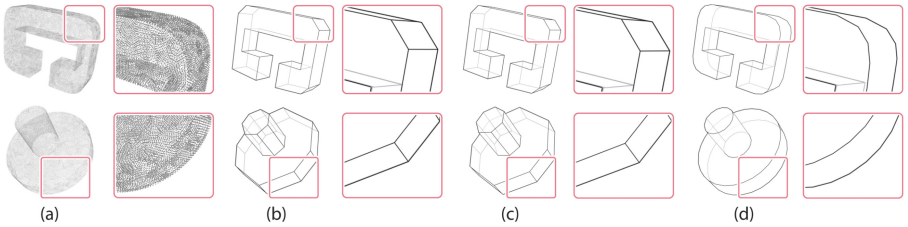

Fig. 8. External geometric details representation by different methods. (a) CAD model, (b) PolyFit, (c) KSR, (d) WindPoly.

To show the performance of internal geometric structure reconstruction, we compute color maps for reconstructed meshes using different low-poly meshing methods. Mapping distances from points to related planes are used to generate colors. The upper layer of the point cloud is peeled off to display the internal color differences. Such color maps are shown in Fig. 7. Comparing to other low-poly meshing schemes, the fitting errors of WindPoly in the internal geometric structure are lower. As mentioned before, WindPoly achieves a balance between planes with different scales. It keeps more accurate geometric features in

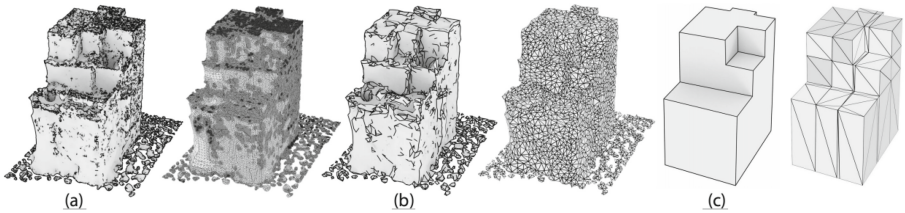

Fig. 9. Visualization of reconstructed meshes and related triangulations by IPSR and WindPoly. (a) IPSR, point = 199.1k, face = 398.1k, (b) IPSR with simplification, point = 5.0k, face = 9.9k, (C) WindPoly, point = 336, face = 168. WindPoly achieves better structural planes and sharp features.

regions with smooth curvature transitions. In Fig. 8, we compare some results by classical low-poly meshing methods and WindPoly. It is clear that our method achieves more accurate results with better geometric consistency. Related qualitative results are reported in Tables 1 and 2.

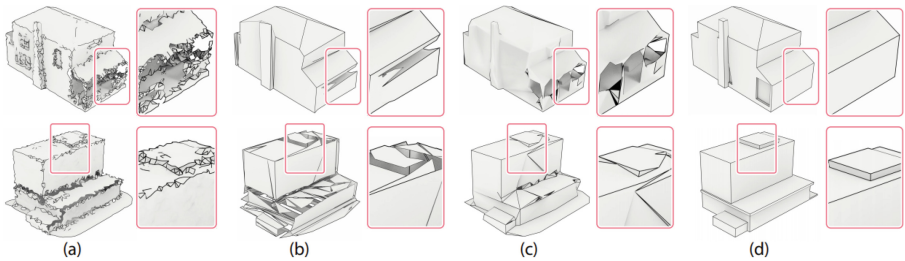

Fig. 10. Visualization of reconstructed meshes by different methods. (a) Initial mesh by [30], (b) LowPoly, (C) R-LowPoly, (d) WindPoly.

To further compare the difference between the two-step reconstruction (mesh reconstruction with simplification) and WindPoly, we exhibit an independent comparison between IPSR and WindPoly. The rendering results of reconstructed meshes are shown in Fig. 9. Benefited from the accurate implicit surface estimation, IPSR achieves more precise geometric details. However, the reconstructed mesh by IPSR take more points and faces even it has been simplified. Some sharp features are smoothed. In contrast, the performance of WindPoly for data compression is significantly improved while keeping sharp features. More results are shown in Fig. 11. As structural methods, LowPoly and R-LowPoly can establish polygonal models with concise structures. However, both methods rely on the quality of the initial mesh. For some broken regions as shown in Fig. 10, such methods cannot automatically repair them and result in noticeable topological errors. In Tables 1 and 2, more qualitative results of structural reconstruction methods are reported. WindPoly achieves better results without initial meshes.

Table 1. Quantitative analysis of different structural reconstruction methods in CAD models from ABC dataset. The related metrics include Hausdorff distance Dis^H, mean distance Dis^M, average point number $\text{p}^{Avg.}$, average face number $\text{F}^{Avg.}$, simplification rate $\text{R}^{Avg.}$, and multiple estimation $\text{RH}^{Avg.} = \text{Dis}^H \times \text{R}^{Avg.}$.

	$\text{Dis}^H \downarrow$	$\text{Dis}^M \downarrow$	$\text{p}^{Avg.} \downarrow$	$\text{F}^{Avg.} \downarrow$	$\text{R}^{Avg.} \downarrow$	$\text{RH}^{Avg.} \downarrow$
PolyFit [35]	0.118	0.014	277	299	0.017	0.0011
KSR [1]	0.111	0.015	**61**	123	0.005	0.0005
LowPoly [14]	0.281	0.018	62	**114**	**0.005**	0.0011
R-LowPoly [6]	0.267	0.014	246	368	0.009	0.0019
WindPoly	**0.061**	**0.002**	273	447	0.024	**0.0010**

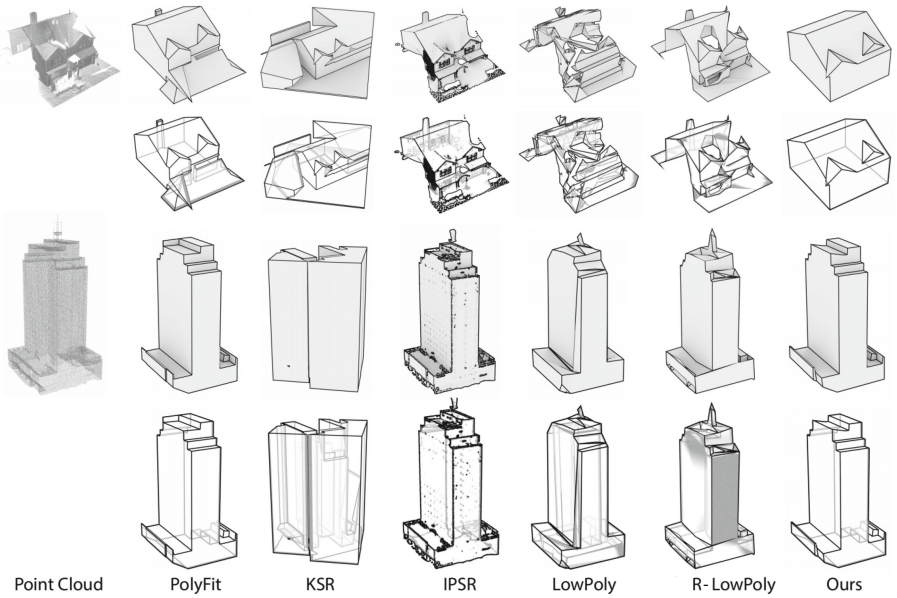

Fig. 11. Comparisons of different polygonal mesh reconstruction methods.

Table 2. Quantitative analysis of different structural reconstruction methods in building models from architectural dataset.

	$\text{Dis}^H \downarrow$	$\text{Dis}^M \downarrow$	$\text{p}^{Avg.} \downarrow$	$\text{F}^{Avg.} \downarrow$	$\text{R}^{Avg.} \downarrow$	$\text{RH}^{Avg.} \downarrow$
PolyFit [35]	5.1652	0.7808	392	433	0.00607	0.0496
KSR [1]	6.7792	0.6845	**157**	312	**0.00362**	0.0398
LowPoly [14]	6.6763	**0.4863**	378	**188**	0.00471	0.0313
R-LowPoly [6]	6.6121	1.8565	476	1064	0.01181	0.0283
WindPoly	**4.5276**	0.5159	549	274	0.00560	**0.0229**

For efficiency analysis, we show time cost reports of different methods in Table 3. KSR is faster but it depends on normal vectors and may lose some geometric structures like instances in Fig. 11. For complex architectural point clouds, IPSR takes more computational cost and outputs meshes with redundant points and faces. LowPoly and R-LowPoly require initial mesh reconstruction. In contrast, WindPoly achieves more concise representation and better geometric structure. It improves the performance for polyhedron searching by the spatial partition strategy which avoids low-contributing planes in convex polyhedron generation. The polygon-based winding numbers optimization further improves the computational efficiency. In summary, WindPoly provides a balanced and effective solution for polygonal mesh reconstruction, as shown in Fig. 12.

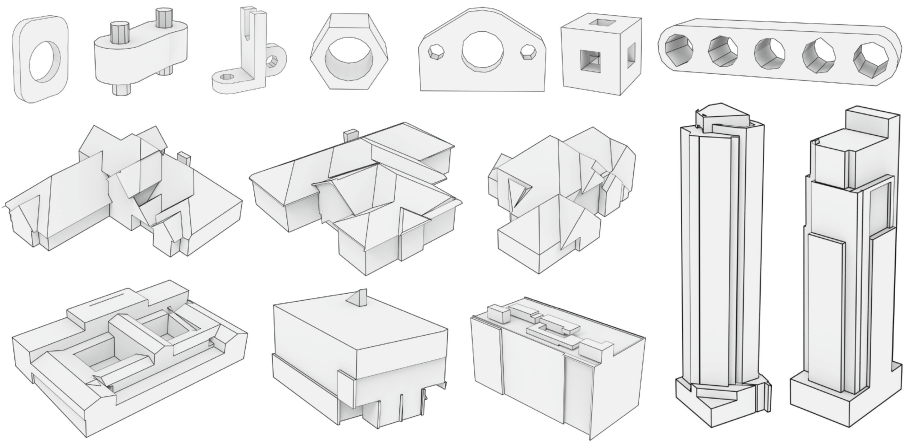

Fig. 12. More practical instances of polygonal meshes by WindPoly.

Limitations. WindPoly is effective for point clouds with significantly structural information. However, once such information of the point cloud is not the main content, the performance of WindPoly is affected to some extent. For some smaller structures in large-scale point clouds, there is a certain probability of producing incorrect results, especially when the point density of the structure is lower. Such smaller structures may be merged into incorrect planes. Some instances are shown in supplementary materials.

Table 3. Time cost reports of different methods in related stages. Initial pre-processing of PolyFit and WindPoly is polyhedron searching; Initial mesh reconstruction of LowPoly and R-LowPoly is voxel-based Delaunay triangulation method [30].

Method	PolyFit		KSR		IPSR		LowPoly		R-LowPoly		WindPoly	
	Initial	Total	Initial	Total	Initial	Total	Initial	Total	Initial	Total	Initial	Total
ABC Dataset	5 s	10 s	–	6 s	–	16 s	17 s	58 s	17 s	78 s	31 s	133 s
PolyFit	6 s	15 s	—	9 s	–	345 s	11 s	103 s	11 s	120 s	19 s	99 s
BuildingNet	56 s	803 s	–	7 s	–	239 s	10 s	80 s	10 s	78 s	20 s	76 s
UrbanBIS	330 s	1707 s	–	11 s	–	1,347 s	9 s	196 s	9 s	98 s	25 s	204 s

5 Conclusion

We propose a polygonal mesh reconstruction method WindPoly to reconstruct a concise 3D representation with accurate structural information for raw point clouds. Three core parts work together to achieve this goal. The polygonal plane detection implements the initial processing to extract primitive planes from the point cloud. Based on the planes, the adaptive spatial partition checks the convex polyhedral space and generate a set of polyhedrons without orientation. Some internal geometric structures are preserved. Finally, the polygon-based winding numbers optimization orients the faces of the polyhedrons and outputs the reconstructed polygonal mesh. The WindPoly can establish structural information from raw point clouds without point-based normal vector assistance. It orients the polygonal mesh with an efficient way while capturing the accurate internal geometric structures. Experiments show that WindPoly achieves a better balance between normal vector independence, geometric consistency and data compression. It can handle CAD models and architectural point clouds, and output high quality polygonal meshes.

Acknowledgements. This work was supported in parts by NSFC (U21B2023, U2001206, 62161146005), Guangdong Basic and Applied Basic Research Foundation (2023B1515120026, 2023A1515110292), DEGP Innovation Team (2022KCXTD025), Shenzhen Science and Technology Program (KQTD20210811090044003, RCJC202007 1411443 5012, JCYJ20210324120213036), Development Funds from Shenzhen University and Guangdong Laboratory of Artificial Intelligence and Digital Economy (SZ).

References

1. Bauchet, J.P., Lafarge, F.: Kinetic shape reconstruction. ACM Trans. Graph. **39**(5), 156:1–156:14 (2020)
2. Bernardini, F., Mittleman, J., Rushmeier, H., Silva, C., Taubin, G.: The ball-pivoting algorithm for surface reconstruction. IEEE Trans. Vis. Comput. Graph. **5**(4), 349–359 (1999)
3. Bolitho, M., Kazhdan, M., Burns, R., Hoppe, H.: Parallel Poisson surface reconstruction. In: International Symposium on Visual Computing, pp. 678–689 (2009)
4. Calderon, S., Boubekeur, T.: Bounding proxies for shape approximation. ACM Trans. Graph. (Proc. SIGGRAPH) **36**(4), 57:1–57:13 (2017)

5. Carr, J.C., et al.: Reconstruction and representation of 3D objects with radial basis functions. In: Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques, pp. 67–76 (2001)
6. Chen, Z., Pan, Z., Wu, K., Vouga, E., Gao, X.: Robust low-poly meshing for general 3D models. ACM Trans. Graph. (Proc. SIGGRAPH) **42**(4), 119:1–119:20 (2023)
7. Chen, Z., Tagliasacchi, A., Zhang, H.: BSP-net: generating compact meshes via binary space partitioning. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 45–54 (2020)
8. Cohen-Steiner, D., Da, F.: A greedy delaunay-based surface reconstruction algorithm. Vis. Comput. **20**, 4–16 (2004)
9. Cohen-Steiner, D., Alliez, P., Desbrun, M.: Variational shape approximation. In: Proceedings of the SIGGRAPH, pp. 905–914 (2004)
10. Digne, J., Morel, J.M., Souzani, C.M., Lartigue, C.: Scale space meshing of raw data point sets. Comput. Graph. Forum **30**(6), 1630–1642 (2011)
11. Fang, H., Lafarge, F.: Connect-and-slice: an hybrid approach for reconstructing 3D objects. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 13490–13498 (2020)
12. Fang, H., Lafarge, F., Desbrun, M.: Planar shape detection at structural scales. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 2965–2973 (2018)
13. Feng, N., Gillespie, M., Keenan, C.: Winding numbers on discrete surfaces. ACM Trans. Graph. (Proc. SIGGRAPH) **42**(4), 36:1–36:17 (2019)
14. Gao, X., Wu, K., Pan, Z.: Low-poly mesh generation for building models. In: Proceedings of the SIGGRAPH, pp. 3:1–3:9 (2022)
15. Hanocka, R., Hertz, A., Fish, N., Giryes, R., Fleishman, S., Cohen-Or, D.: MeshCNN: a network with an edge. ACM Trans. Graph. (Proc. SIGGRAPH) **38**(4), 90:1–90:12 (2019)
16. Hanocka, R., Metzer, G., Giryes, R., Cohen-Or, D.: Point2mesh: a self-prior for deformable meshes. arXiv preprint arXiv:2005.11084 (2020)
17. Hou, F., Wang, C., Wang, W., Qin, H., Qian, C., He, Y.: Iterative Poisson surface reconstruction (IPSR) for unoriented points. ACM Trans. Graph. (Proc. SIGGRAPH) **41**(4), 128:1–128:13 (2022)
18. Huang, J., Chen, H.X., Hu, S.M.: A neural galerkin solver for accurate surface reconstruction. ACM Trans. Graph. (Proc. SIGGRAPH Asia) **41**(6), 229:1–229:16 (2022)
19. Kazhdan, M., Bolitho, M., Hoppe, H.: Poisson surface reconstruction. In: Proceedings of the Eurographics Symposium on Geometry Processing, pp. 61–70 (2006)
20. Kazhdan, M., Hoppe, H.: Screened Poisson surface reconstruction. ACM Trans. Graph. **32**(3), 29:1–29:13 (2013)
21. Kazhdan, M., Chuang, M., Rusinkiewicz, S., Hoppe, H.: Poisson surface reconstruction with envelope constraints. Comput. Graph. Forum **39**(5), 173–182 (2020)
22. Kelly, T., Femiani, J., Wonka, P., Mitra, N.J.: BigSUR: large-scale structured urban reconstruction. ACM Trans. Graph. (Proc. SIGGRAPH Asia) **36**(6), 204:1–204:16 (2017)
23. Koch, S., et al.: ABC: a big cad model dataset for geometric deep learning. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 9601–9611 (2019)
24. Lafarge, F., Alliez, P.: Surface reconstruction through point set structuring. Comput. Graph. Forum **32**(2), 225–234 (2013)
25. Lescoat, T., Liu, H.T.D., Thiery, J.M., Jacobson, A., Boubekeur, T., Ovsjanikov, M.: Spectral mesh simplification. Comput. Graph Forum **39**(2), 315–324 (2020)

26. Li, M., Nan, L.: Feature-preserving 3D mesh simplification for urban buildings. ISPRS J. Photogrammetry Remote Sens. **173**, 135–150 (2021)
27. Li, M., Wonka, P., Nan, L.: Manhattan-world urban reconstruction from point clouds. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9908, pp. 54–69. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46493-0_4
28. Liu, H.T.D., Gillespie, M., Chislett, B., Sharp, N., Jacobson, A., Crane, K.: Surface simplification using intrinsic error metrics. ACM Trans. Graph. (Proc. SIGGRAPH) **42**(4), 118:1–118:17 (2023)
29. Lorensen, W.E., Cline, H.E.: Marching cubes: a high resolution 3D surface construction algorithm. In: Seminal Graphics: Pioneering Efforts that Shaped the Field, pp. 347–353 (1998)
30. Lv, C., Lin, W., Zhao, B.: Voxel structure-based mesh reconstruction from a 3D point cloud. IEEE Trans. Multimed. **24**, 1815–1829 (2021)
31. Lv, C., Lin, W., Zhao, B.: Intrinsic and isotropic resampling for 3D point clouds. IEEE Trans. Pattern Anal. Mach. Intell. **45**(3), 3274–3291 (2022)
32. Lv, C., Lin, W., Zheng, J.: Adaptively isotropic remeshing based on curvature smoothed field. IEEE Trans. Vis. Comput. Graph. 1–15 (2022)
33. Mehra, R., Zhou, Q., Long, J., Sheffer, A., Gooch, A., Mitra, N.J.: Abstraction of man-made shapes. ACM Trans. Graph. **28**(5), 1–10 (2009)
34. Morse, B.S., Yoo, T.S., Rheingans, P., Chen, D.T., Subramanian, K.R.: Interpolating implicit surfaces from scattered surface data using compactly supported radial basis functions. In: Proceedings of the SIGGRAPH (2005)
35. Nan, L., Wonka, P.: Polyfit: polygonal surface reconstruction from point clouds. In: International Conference on Computer Vision, pp. 2353–2361 (2017)
36. Nielson, G.M.: On marching cubes. IEEE Trans. Vis. Comput. Graph. **9**(3), 283–297 (2003)
37. Rabbani, T., Van Den Heuvel, F., Vosselmann, G.: Segmentation of point clouds using smoothness constraint. ISPRS J. Photogrammetry Remote Sens. **36**(5), 248–253 (2006)
38. Rakotosaona, M.J., Aigerman, N., Mitra, N.J., Ovsjanikov, M., Guerrero, P.: Differentiable surface triangulation. ACM Trans. Graph. (Proc. SIGGRAPH Asia) **40**(6), 267:1–267:13 (2021)
39. Salinas, D., Lafarge, F., Alliez, P.: Structure-aware mesh decimation. Comput. Graph. Forum **34**(6), 211–227 (2015)
40. Selvaraju, P., et al.: BuildingNet: learning to label 3D buildings. In: International Conference on Computer Vision, pp. 10377–10387 (2021)
41. Wang, J., Oliveira, M.M., Kaufman, A.E.: Reconstructing manifold and non-manifold surfaces from point clouds. In: Proceedings of the IEEE International Conference on Visualization, pp. 415–422 (2005)
42. Wang, P., Wang, Z., Xin, S., Gao, X., Wang, W., Tu, C.: Restricted delaunay triangulation for explicit surface reconstruction. ACM Trans. Graph. (Proc. SIGGRAPH Asia) **41**(5), 180:1–180:20 (2022)
43. Williams, F., Schneider, T., Silva, C., Zorin, D., Bruna, J., Panozzo, D.: Deep geometric prior for surface reconstruction. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 10130–10139 (2019)
44. Xu, R., et al.: Globally consistent normal orientation for point clouds by regularizing the winding-number field. ACM Trans. Graph. (Proc. SIGGRAPH) **42**(4), 111:1–111:15 (2023)
45. Xu, R., et al.: RFEPS: reconstructing feature-line equipped polygonal surface. ACM Trans. Graph. (Proc. SIGGRAPH Asia) **41**(6), 228:1–228:15 (2022)

46. Xu, Y., Nan, L., Zhou, L., Wang, J., Wang, C.C.: HRBF-fusion: accurate 3D reconstruction from RGB-D data using on-the-fly implicits. ACM Trans. Graph. **41**(3), 35:1–35:19 (2022)
47. Yang, G., Xue, F., Zhang, Q., Xie, K., Fu, C.W., Huang, H.: UrbanBIS: a large-scale benchmark for fine-grained urban building instance segmentation. In: Proceedings of the SIGGRAPH, pp. 16:1–16:11 (2023)
48. Yang, K., Chen, X.: Unsupervised learning for cuboid shape abstraction via joint segmentation from point clouds. ACM Trans. Graph. (Proc. SIGGRAPH) **40**(4), 152:1–152:11 (2021)
49. Yi, R., Liu, Y.J., He, Y.: Delaunay mesh simplification with differential evolution. ACM Trans. Graph. (Proc. SIGGRAPH Asia) **37**(6), 263:1–263:12 (2018)
50. Yu, M., Lafarge, F.: Finding good configurations of planar primitives in unorganized point clouds. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 6367–6376 (2022)