International Journal of Computer Vision
https://doi.org/10.1007/s11263-024-02059-x

Efficient High-Quality Vectorized Modeling of Large-Scale Scenes

Xiaojun Xiang[1] · Hanqing Jiang[1] · Yihao Yu[1] · Donghui Shen[1] · Jianan Zhen[1] · Hujun Bao[2] · Xiaowei Zhou[2] · Guofeng Zhang[2]

Received: 31 July 2022 / Accepted: 11 March 2024

Abstract

We present a novel high-quality vectorized modeling pipeline for large-scale scenes. With a reconstructed dense point cloud and its corresponding multi-view source images and camera parameters as input, our system can efficiently reconstruct a geometrically complete and detail-preserved vectorized model. Unlike most existing planar shape assembling methods which cannot handle large-scale vectorized modeling well due to the limitation of memory and computation, we can achieve complete high-quality vectorized modeling for complicated large-scale scenes in a time and memory efficient way. Our pipeline first carries out a 3D semantic segmentation on the dense point cloud, by performing 2D semantic labeling on the source images with a semantic segmentation network and fusing the 2D semantic labels into the point cloud. According to the fused dense 3D semantic labels, we then divide the scene into main structure including the grounds, walls and ceilings, and isolated objects that do not belong to the main structure. After the scene division completes, vectorized modeling is performed successively on the main structure and isolated objects to extract their polygonal models respectively instead of vectorizing the whole scene, to improve both time and memory efficiencies. Additionally, the previously vectorized main polygonal structures are used as priors to refine the segmentation and guide the vectorization of the objects to ensure the geometrical completeness and topological consistency of the entire vectorized model. Especially, during the vectorization procedure, a well designed binary space partition tree is designed to better slice the space so that high-quality polygonal mesh with more geometric details can be reconstructed with both time and memory efficiencies. Experiments with quantitative and qualitative evaluations on large-scale scenes demonstrate the accuracy and efficiency of the proposed vectorization pipeline. We also compare our method with state-of-the-art planar shape reconstruction methods to show its effectiveness in reconstructing large-scale vectorized models.

Keywords Vectorized modeling · Main structure · Isolated objects · Binary space partition tree

1 Introduction

Vectorized modeling from 3D point clouds is increasing attractive in recent years, because a vectorized model has considerable advantages in reducing computation and memory overhead due to its light-weight Computer-Aided Design

Communicated by Yasuyuki Matsushita.

Xiaojun Xiang, Hanqing Jiang and Yihao Yu are contributed equally to this work.

✉ Guofeng Zhang
zhangguofeng@zju.edu.cn

Xiaojun Xiang
xiangxiaojun@sensetime.com

Hanqing Jiang
jianghanqing@sensetime.com

Yihao Yu
yuyihao@sensetime.com

Donghui Shen
shendonghui1@sensetime.com

Jianan Zhen
zhenjianan@sensetime.com

Hujun Bao
baohujun@zju.edu.cn

Xiaowei Zhou
xwzhou@zju.edu.cn

[1] SenseTime Research, SenseTime Group Inc., 29F, Tianren Tower, No. 188, Liyi Road, Hangzhou 311215, Zhejiang, China

[2] State Key Lab of CAD&CG, Zhejiang University, ZijinGang Campus, Hangzhou 310058, Zhejiang, China

Published online: 20 May 2024

(CAD) style representation, which is preferred by a variety of applications such as rendering, simulation, navigation and building information modeling (BIM). However, automatically vectorizing a complicated large-scale scene with its geometric details preserved in a time and memory efficient way has long been a challenging problem in computer vision.

The most popular way to reconstruct a 3D polygonal vectorized model consists of two steps: a planar primitive abstraction and a polygonal vectorized model reconstruction. A planar primitive is represented by a set of inlier 3D points and a supporting plane which fits the inliers. All the planar primitives are abstracted from the input 3D point cloud, and are then assembled into a compact polygonal vectorized model, which is the core and the most difficult part of the entire pipeline. A typical solution is to slice the 3D space into an intermediate set of polyhedrons by binary space partition (BSP) tree (Boulch et al., 2014; Chauve et al., 2010; Fang et al., 2021; Murali and Funkhouser, 1997; Mura et al., 2016), plane expansion (Nan and Wonka, 2017; Oesau et al., 2014), kinetic frameworks Bauchet and Lafarge (2020) or soft-connectivity analysis Fang and Lafarge (2020). The vectorized model is then extracted by a min-cut framework or a mixed-integer programming approach. These methods are robust but have difficulty in handling complicated structures well with over thousands of planar primitives due to their extensive time and memory consumption. Some other research works focus on the vectorization of large-scale scenes, including general indoor scenes (Fang et al., 2021; Han et al., 2021a; Ochmann et al., 2019) and urban environments (Bauchet and Lafarge, 2019; Duan and Lafarge, 2016; Han et al., 2021b; Verdie et al., 2015; Zhu et al., 2020, 2018). They segment the scenes into certain semantic parts and vectorize only a limited portion of the scenes including ceilings, grounds, walls and buildings, with other semantic parts ignored. With the help of 3D semantic segmentation, these works can handle large-scale scenes but turn out to be difficult in reconstructing a complete vectorized model of the entire scene with the detailed structures and objects extracted. Other typical approaches for object or scene vectorization like (Arikan et al., 2013; Chen and Chen, 2008; Schindler et al., 2011) snap all the planar primitives according to their spatial connections. However, these works are more suitable for high-quality point cloud such as Lidar data, since their performance depends heavily on the geometrical correctness and completeness of the adjacency relationship of the abstracted primitives.

We present a novel high-quality vectorized modeling system which combines semantic segmentation to solve the above problems. More specifically, we aim at converting an oriented dense point cloud of a large-scale scene to a geometrically complete, detail-preserved and topological consistent polygonal surface mesh in a time and memory efficient way. In other words, a vectorized modeling system should be able to successfully vectorize a piece-wise planar scene containing over tens of thousands of planar primitives within several hours on a computing platform with memory limitation (usually no more than 128GB) for practical application purpose, which is difficult to achieve by most SOTA works. Starting from the point cloud and its multi-view source images and corresponding camera parameters, we first perform 3D semantic segmentation on the point cloud based on a 2D segmentation network combined with 3D semantic fusion. With the 3D semantic labels, we extract main structure including the grounds, walls and ceilings from the dense point cloud, with other parts not belonging to the main structure as isolated objects. After the scene division completes, vectorized modeling is performed successively on the main structure and isolated objects to reconstruct a complete vectorized model. During the vectorization, a well designed BSP tree is proposed to better slice the 3D space, which enables our approach to assemble more planar primitives in a time and memory efficient way for preserving better geometric details. Additionally, the previously vectorized polygonal geometry of the main structure are used as priors to not only constrain the vectorization of the incoming isolated objects to maintain the completeness and topological consistency of the entire vectorized model, but also guide the refinement of 3D semantic segmentation to further improve the vectorized modeling of the whole scene. Figure 1 shows some large-scale indoor and outdoor examples of our vectorization approach, from which we can see that our method is capable of assembling more planar primitives to a high-quality polygonal vectorized model with more geometric details, better geometrical completeness and higher accuracy, compared to other SOTA methods like PolyFit Nan and Wonka (2017), Kinetic Shape Reconstruction (KSR) Bauchet and Lafarge (2020) and VecIM Han et al. (2021a).

Our main contributions can be summarized as:

- We propose an efficient high-quality vectorized modeling pipeline for complicated large-scale scenes, by dividing the whole scene into main structure and isolated objects according to the 3D semantic segmentation of the scenes, for a time and memory efficient vectorization purpose.
- A novel BSP strategy is proposed to generate a lightweight convex polyhedra set, which enables our system to assemble thousands of planar primitives to a detail-preserved vectorized model in only a few minutes.
- We innovatively use the already vectorized main polygonal structures as prior to guide the vectorization process of the isolated parts and further refine the 3D semantic segmentation. In this way, a high-quality vectorized model with complete geometry, consistent topology and correct semantic attributes can be reconstructed.

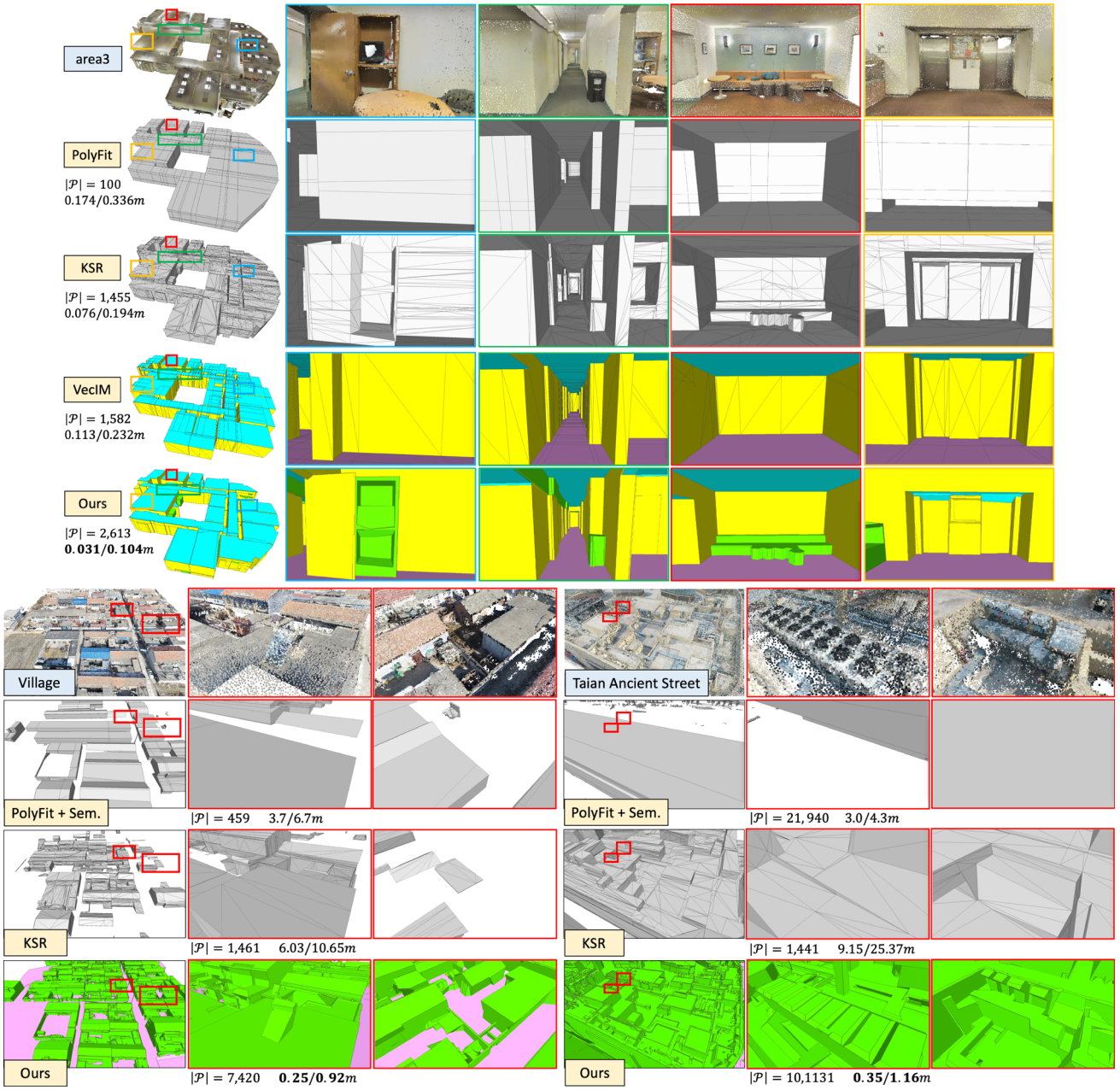

Fig. 1 Examplar point clouds and corresponding 3D vectorized models of an indoor case "area3" from S3DIS Armeni et al. (2017) on top, and two outdoor cases "Village" and "Taian Ancient Street" on left bottom and right bottom, by our vectorized modeling pipeline, with rectangular regions magnified to show the vectorized structure details. For case "area3" we also compare our method with SOTA works PolyFit Nan and Wonka (2017), KSR Bauchet and Lafarge (2020) and VecIM Han et al. (2021a), with the number of participated planar primitives and modeling accuracy by Mean Hausdorff Error (MHE) Cignoni et al. (1998) and Root Mean Squared Error (RMSE) in meters also given for each method. For the two large-scale outdoor cases, only PolyFit and KSR are compared, since VecIM even cannot succeed due to its limitation to indoor scenes. From the detailed comparison results we can see that our vectorization method can well handle large-scale complicated scenes composed of piecewise-planar main structure and isolated objects, by assembling the largest number of primitives to achieve the best geometrical completeness, details and accuracy

This paper is organized as follows. Section 2 briefly presents related work. Section 3 gives an overview of the proposed vectorized modeling system. The semantic scene segmentation module and the scene vectorization module are described in Sects. 4 and 5 respectively. Finally, we evaluate the proposed solution in Sect. 6.

2 Related Work

Vectorized modeling methods reconstruct a CAD style model from 3D point cloud or raw triangle mesh of an object. Existing state-of-the-art (SOTA) works usually start with a planar primitive abstraction procedure, followed by the reconstruction of polygonal vectorized model from the abstracted planar primitives. Planar primitive abstraction is generally achieved by performing region growing algorithm (Marshall et al., 2001; Rabbani et al., 2006) or RANSAC Schnabel et al. (2007) on the input data. Some works like (Boulch et al., 2014; Fang et al., 2021; Nan and Wonka, 2017) additionally merge the coplanar primitives to reduce the over-segmentation regions, or regularize them based on the prior structure knowledge of man-made architecture such as parallelism and orthogonality (Fang et al., 2021; Li et al., 2016b; Monszpart et al., 2015; Verdie et al., 2015). Recently, Yu and Lafarge (2022) propose to refine the planar primitives from unorganized 3D point clouds by optimizing a multi-objective energy function, which shows efficacy for the quality improvement of the reconstructed compact mesh.

After planar primitive abstraction finishes, all the planar primitives are assembled together to form a compact polygonal surface mesh, which is the most difficult step. Existing polygonal vectorized model reconstruction methods can be generally divided into two categories: connectivity-based methods and slicing-based methods. The connectivity-based methods like (Arikan et al., 2013; Chen and Chen, 2008; Schindler et al., 2011) usually analyse the adjacency relationship of the planar primitives and reconstruct the vectorized model based on the boundary information of each primitive, including the corners, edges and facets. For example, Chen and Chen (2008) compute the edges through plane intersections and human interactions, and the corners are extracted from a cluster graph and arranged to constitute the final polygonal surface mesh. Arikan et al. (2013) propose an interactive 3D modeling system, which creates an initial polygonal surface mesh by detecting planar polygon primitives from point clouds and automatically snapping them together according to the local polygon adjacency relations. Then, the users are allowed to interactively modify the polygons, which are interleaved with the automatic snapping to further refine the polygonal mesh. However, most of these methods require user interactions and rely heavily on the quality of the input data, which restricts their availability in the real-world applications with noisy and large-scale point clouds captured by consumer-level devices or generated by multi-view stereo (MVS) methods. The slicing-based methods such as (Boulch et al., 2014; Bauchet and Lafarge, 2020; Bouzas et al., 2020; Chauve et al., 2010; Fang and Lafarge, 2020; Fang et al., 2021; Han et al., 2021a; Huang et al., 2022; Li et al., 2016b; Mura et al., 2016; Nan and Wonka, 2017; Oesau et al., 2014; Wang et al., 2017) are more robust to defect-laden data, whose idea is to first partition the whole 3D space, which is commonly the 3D bounding box of the input data, into an intermediate set of polygonal facets sliced by the detected planar primitives. A typical SOTA work of this category is PolyFit Nan and Wonka (2017), which slices the space by expanding all the planar primitives infinitely to produce a set of extremely dense polygonal facets. This exhaustive slicing way is difficult to process a scene containing more than one hundred planar primitives due to its huge memory and time complexity. Recently, some more efficient partition methods are proposed to speed up the slicing process. For example, Bauchet and Lafarge (2020) design a kinetic data structure where all the planar shapes grow at constant speed until collisions occur. Fang and Lafarge (2020) propose a hybrid slicing method by finding out the structural facets first and constrain the slicing range of the other primitives according to their spacial relationship. Han et al. (2021a) convert the 3D space partition problems to multiple 2D segment or cell assembly optimizations, which is mainly designed for indoor scenes. Another kind of more light-weight slicing method is based on BSP tree (Boulch et al., 2014; Chauve et al., 2010; Fang et al., 2021; Murali and Funkhouser, 1997; Mura et al., 2016), which inserts planar primitives sequentially, with each primitive only splitting the polyhedral cells that are traversed. The main challenge of BSP based methods is to choose a proper insertion order for the planar primitives to generate a correct light-weight partition in a time efficient way. Some BSP based piecewise-planar reconstruction works like (Boulch et al., 2014; Chauve et al., 2010; Fang et al., 2021) sort primitives according to their sizes and structures, and Fang et al. (2021) additionally consider the adjacency relationships of the primitives during the slicing procedure to ensure the correctness of the partition. After the slicing process completes, the vectorized model is extracted by labeling the cells with a min-cut formulation (Bauchet and Lafarge, 2020; Fang et al., 2021; Li et al., 2016b; Mura et al., 2016), or selecting a portion of polygonal facets to form the final surface mesh through mixed-integer programming approach (Boulch et al., 2014; Fang and Lafarge, 2020; Nan and Wonka, 2017). As verified in the practical experiments of KSR Bauchet and Lafarge (2020), the min-cut solver is much more stable and faster than the integer programming, especially if we have a huge number of polyhedral cells. Inspired by the slicing-based methods, our vectorization approach also uses BSP tree for

space partition. However, unlike most existing BSP based works that can hardly handle thousands of planar primitives efficiently, we propose a novel BSP strategy to solve it in an efficient way.

Recently, deep learning based methods are also proposed for object vectorization. For example, Zeng et al. (2018) design a sequence of shape grammar rules and reconstruct buildings through a deep neural network. Chen et al. (2020) propose BSP-Net for training a network to reconstruct polygonal surface mesh from point cloud. Yu et al. (2022) introduce CAPRI-Net to learn 3D CAD shapes via constructive solid geometry operations. However, due to the complexity of the vectorization problem, most of these networks are limited to vectorize small objects.

In addition, there are lots of works which focus on vectorizing urban buildings in the form of levels of detail (LODs). Most of these methods build city models with levels of detail ranging from LOD2 to LOD3 (Arefi et al., 2008; Han et al., 2021b; Verdie et al., 2015; Xie et al., 2021; Zhu et al., 2018, 2020), or specially reconstruct 2.5D models (Bauchet and Lafarge, 2019; Duan and Lafarge, 2016; Lafarge and Mallet, 2012; Li et al., 2016; Poullis and You, 2009), while our goal is to provide a complete and detailed 3D vectorized model for more complicated general scenes including urban buildings and indoor scenes.

Most existing vectorization methods are effective for small-scale scenes, but can hardly be directly used for a complicated city-scale scene which contains more than ten thousands of planar primitives, due to their huge memory and time consumption. To reconstruct large-scale vectorized models, some of these methods such as (Fang et al., 2021; Han et al., 2021a; Mura et al., 2016; Oesau et al., 2014) semantically decompose the input 3D point clouds into isolated parts, and only vectorize the main structure such as ceilings, walls, grounds and buildings, with other isolated objects ignored. The 3D semantic segmentation methods can be divided into three categories: geometric feature based methods, machine learning based approaches, and neural network based methods. The geometric feature based methods (Fang et al., 2021; Mura et al., 2016) decompose the scene according to the geometric attributes of planar primitives, including the size, elevation, planarity, horizontality, verticality, and z-direction of the scene. This kind of methods usually depend on strong prior knowledge about the scene and are not suitable for uncommon or complex environments. The machine learning based methods also compute the geometric attributes first, and then label the entire scene through traditional Markov Random Fields (MRF) (Lafarge and Mallet, 2012; Rouhani et al., 2017; Verdie et al., 2015; Zhu et al., 2018) or Conditional Random Fields (CRF) optimization (Hermans et al., 2014; Pham et al., 2019; Wolf et al., 2015; Yang et al., 2017). Recently, many neural network based methods are proposed to solve the decomposition problem. There are already some 2D CNNs based semantic segmentation works such as (Chen et al., 2018; Wang et al., 2020) which perform successfully on natural scenes. Choy et al. (2019) and Graham et al. (2018) extend the feature extraction ability of 2D CNNs to fit 3D data and achieves good performance on indoor 3D scenes. However, 3D CNNs based methods would consume lots of time and memory when it comes to large-scale outdoor scenes. Instead of extracting features from 3D data, Kundu et al. (2020) choose to fuse 2D segmentation results into 3D scenes with camera-world translation, which avoids time and computation consuming process of training and inferencing in 3D data. After the semantic scene decomposition completes, these works only reconstruct the main vectorized structures, and ignore the other isolated objects. Fang et al. (2021) try to complete the structures of the isolated objects by performing vertex filling and Poisson Surface Reconstruction Kazhdan et al. (2006), instead of vectorizing these objects. In comparison, our system can achieve geometrically complete and topological consistent vectorized modeling with geometric details preserved for both main structure and isolated objects of complicated large-scale scenes in a time and memory efficient way.

3 System Overview

We now outline the main steps of the proposed vectorized modeling system, as shown in Fig. 2. Suppose we have a large-scale normal oriented dense point cloud denoted as $\mathcal{D}$ and its corresponding N multi-view source images $\mathcal{I} = \{I_i \mid i = 1, \ldots, N\}$ and camera parameters $\{\mathbf{M}_i = \mathbf{K}_i [\mathbf{R}_i \mathbf{t}_i] \mid i = 1, \ldots, N\}$ as input. The point cloud can be captured by 3D laser scanner or generated by SOTA MVS methods with the multi-view images and camera parameters. Our system first carries out a semantic segmentation module to decompose the scene into main structure including the grounds, walls and ceilings, and isolated objects which do not belong to any part of the main structure. When the semantic decomposition completes, we start to vectorize the main structure first, by abstracting and expanding their planar primitives according to the adjacency relationships, and partitioning the space into an intermediate set of polyhedrons through our well designed BSP tree. The vectorized model of main structure is extracted by labeling the cells as inside or outside via a min-cut optimization. After that, the vectorized model of the main structure is fed back to the semantic segmentation module to refine the semantic labeling accuracy of the entire scene including the main structure and the isolated objects. The updated semantic segmentation leads a second time of vectorization to get the final vectorized model of the main structure with better geometry. After we have a refined semantic segmentation and vectorized model of the

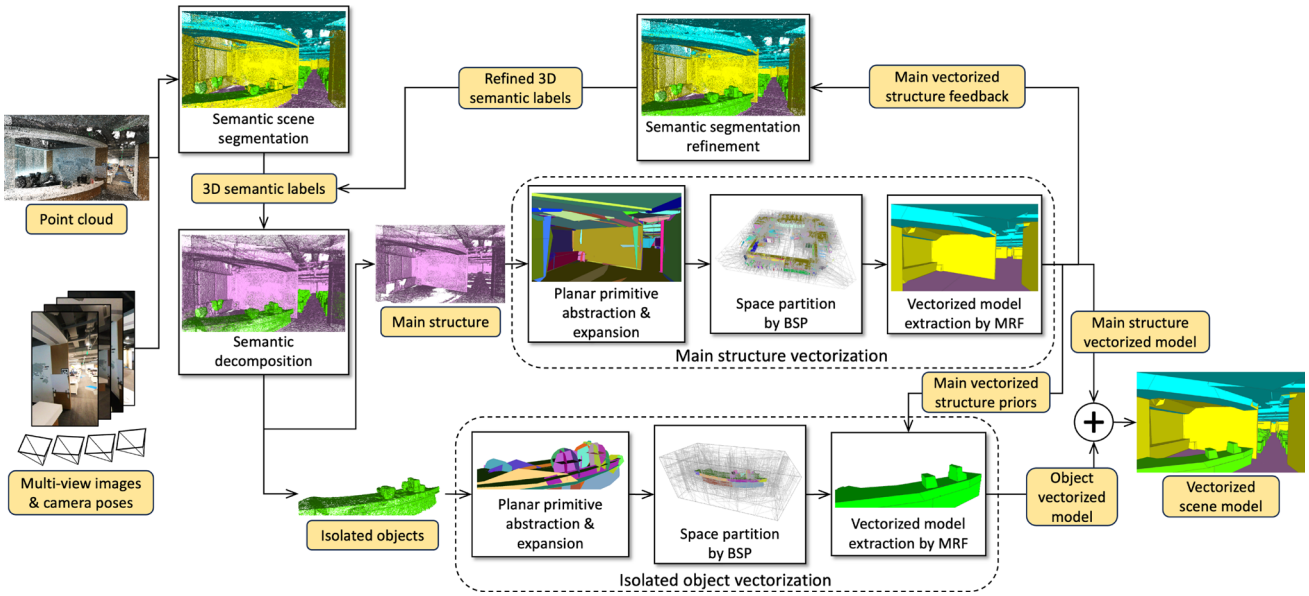

Fig. 2 System framework of our vectorized modeling pipeline, which starts from a semantic segmentation step to divide the scene point cloud into the main structure part and the isolated object parts, followed by vectorization processes for each part to get its polygonal model. Especially, the main vectorized structure provides feedback for semantic segmentation refinement to lead a second time of main vectorized structure extraction, and is also used as prior to guide the vectorization of the isolated objects for topological consistency. The vectorized models of the main structure and the isolated objects make up the final complete polygonal model of the scene

Table 1 The influence of different settings of w_c and w_b on the number of sliced space cells $|\mathcal{C}_\mathbf{m}|$, the number of associated polygonal faces $|\mathcal{F}_\mathbf{m}|$, time (seconds) and memory consumption (GB) of the main structure space partition for case "Office"

w_c/w_b	$\lvert\mathcal{C}_\mathbf{m}\rvert$	$\lvert\mathcal{F}_\mathbf{m}\rvert$	Mem.(GB)	Time(s)
20/10	**213,199**	**728,553**	**0.55**	**30.2**
5/10	246,486	845,693	0.63	33.8
10/10	223,751	764,076	0.57	32.1
50/10	227,028	775,541	0.59	48.3
20/5	220,509	754,391	0.58	39
20/25	247,372	845,391	0.62	32.2
20/50	261,118	893,118	0.66	38.8

The entries with best performance are given in bold

main structure, the vectorization of the isolated objects is guided with the main vectorized structure as priors to ensure their topological consistency. The refined polygonal surface meshes of the main structure and the isolated objects constitute the final vectorized model of the whole scene whose polygonal facets are denoted as $\mathcal{F}$, as shown in Fig. 1. In the following sections, each step will be described in detail.

4 Semantic Scene Segmentation

To semantically decompose the scene into main structure including the grounds, walls and ceilings, and isolated objects of the rest parts, a 3D semantic segmentation of the input point cloud is necessary to classify each 3D point into semantic labels $\mathbf{S}$ including *ground*, *ceiling*, *wall*, *roof*, *building*, *car* and *other object*. We use a 2D-to-3D fusion strategy to ensure generalization of the 3D semantic point cloud segmentation. First, a DeeplabV3+ network Chen et al. (2017) is applied for inferring per-pixel semantic probability features of each source image I_i, where every pixel $\mathbf{x} \in I_i$ has its semantic probability $P_s(\mathbf{x}, s)$ for each semantic label $s \in \mathbf{S}$. We pre-train our 2D semantic segmentation model on ADE20K dataset (Zhou et al., 2017, 2019) for indoor scenes and Semantic Drone Dataset[1] for outdoor scenes, and then fine-tune it with additionally five UAV captured cases. Then, the inferred 2D semantic probability features are mapped to the 3D point cloud, by projecting each 3D point $\mathbf{X} \in \mathcal{D}$ to the multiple source images using the camera parameters $\{\mathbf{M}_i = \mathbf{K}_i\,[\mathbf{R}_i\mathbf{t}_i] \mid i = 1, \ldots, N\}$, where $\mathbf{K}_i$ is the projection matrix and $[\mathbf{R}_i\mathbf{t}_i]$ is the global-to-local pose matrix of I_i including the rotation part $\mathbf{R}_i$ and translation part $\mathbf{t}_i$. The semantic probability features of all the visible projections are fused using a similar fusion approach to Kundu et al. (2020). Here we filter out invisible projections by checking the consistency of the rendered depth buffer and the projection depths of each image view. The semantic probability

[1] http://dronedataset.icg.tugraz.at.

Table 2 The semantic segmentation accuracy measurements of "Village" and "Office" before and after semantic segmentation refinement, with Intersection over Union (IoU) given for each semantic type, and Mean Intersection over Union (MIoU) given for total semantic accuracy metrics

Cases	Accuracy	Wall	Building	Ground	Ceiling	Roof	Car	Other	MIoU
Village	Before Refine	N/A	70.81%	67.28%	N/A	76.63%	61.01%	47.14%	64.57%
	After Refine	N/A	**84.52%**	**91.41%**	N/A	**84.82%**	61.01%	47.14%	**73.78%**
Office	Before Refine	77.54%	N/A	75.95%	84.03%	N/A	N/A	79.03%	79.14%
	After Refine	**91.91%**	N/A	**87.2%**	**94.27%**	N/A	N/A	**92.76%**	**91.54%**

The entries with best performance are given in bold

features of all the visible projections are simply averaged to obtain the fused 3D semantic probability features as in Kundu et al. (2020), where each 3D point $\mathbf{X}$ has its semantic probability $P_s(\mathbf{X}, s)$ for each semantic label $s \in \mathbf{S}$. With the fused semantic probability feature, each 3D point is assigned by the semantic with the maximal probability as its semantic label $S(\mathbf{X}) = \arg\max_s P_s(\mathbf{X}, s)$. The 2D and 3D segmentation results of indoor case "Office" and outdoor case "Village" are shown in Fig. 3, with the 3D semantic segmentation accuracy measurements on the two cases given in the "Before Refine." rows of Table 2, where we use manually annotated semantic ground-truth (GT) to compute Intersection over Union (IoU) for each semantic type and Mean Intersection over Union (MIoU) for total semantic accuracy metrics of each case. From the semantic accuracy evaluation we can see that our 2D-to-3D fusion strategy can ensure a reliable 3D semantic prediction of the scene point cloud with high MIoU for the following scene vectorization process.

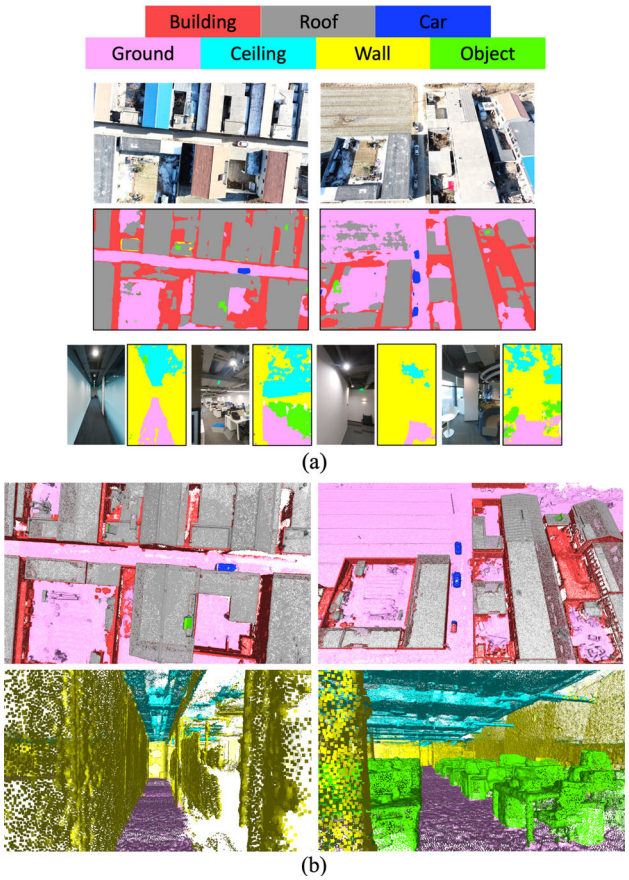

Fig. 3 **a** Examplar 2D semantic segmentations of cases "Village" and "Office". **b** The fused 3D semantic segmentations for point clouds of "Village" and "Office"

5 Large-Scale Scene Vectorization

Since we have the 3D semantic labeling of the scene point cloud, we can divide the scene into main structure and isolated objects, then vectorize the two parts respectively to reduce the computational complexity and memory consumption for large-scale scenes. For simplification of the semantic categories, we rearrange the original semantic labels into four categories by keeping the semantic labels *ground*, *ceiling* and *wall* which belongs to the main structure, and unifying the other semantic labels as *object* to denote isolated objects. Additionally, the original labels *door* and *window* are relabeled as part of *wall* for simplification. In this way, each 3D point $\mathbf{X}$ has its relabeled semantic category denoted as $\hat{S}(\mathbf{X})$. Vectorized modeling is carried out on the main structure and isolated object parts decomposed by the relabeled semantic segmentation. Especially, the vectorized model of the main structure will be used to refine the semantic segmentation accuracy and guide the vectorization of the isolated objects. We will describe the vectorized modeling and semantic refinement steps in detail.

5.1 Planar Primitive Abstraction

Given the semantic relabeled 3D point cloud $\mathcal{D}$, our goal is to abstract a set of planar primitives $\mathcal{P} = \{\mathbf{P}_i \mid i = 1, \ldots, N\}$ from the point cloud and classify them into main structure and isolated objects. Each primitive $\mathbf{P}_i$ consists of a set of inlier 3D points and a supporting 3D plane which fits the inliers. For 3D points of each semantic category, we perform the region growing algorithm implemented in CGAL library Lafarge and Mallet (2012) to detect the planar primitives, considering 3D points with Euclidean distance smaller than

a sphere radius $r_s = 5s_a$ as neighborhood with s_a denoting the average spacing of $\mathcal{D}$. For each primitive, at least 10 inliers are required by region growing with each inlier point lying within maximal tolerances of absolute Euclidean distance $0.1m$ and normal deviation 25° from the plane position and normal of the primitive respectively. Finally, the planar primitives with the semantic categories of *ground*, *ceiling* and *wall* constitute the main structure and the connected primitives with *object* semantic category are grouped as isolated objects, as shown in Fig. 4c.

5.2 Space Partition

After we have the division of the planar primitives into main structure and isolated objects, we start to successively vectorize the main structure part and the isolated object parts. For vectorized modeling of each part, unlike the exhaustive slicing strategies proposed by Oesau et al. (2014), Nan and Wonka (2017), we use a light-weight BSP based space slicing strategy which inserts planar primitives successively with each primitive only slicing the polyhedral cells which are traversed. Most existing BSP tree based methods like (Boulch et al., 2014; Chauve et al., 2010; Mura et al., 2016) cannot ensure correct space slicing boundaries with various improper orders in which the primitives are inserted. An incorrect space slicing is illustrated as 2D simulation in Fig. 5a, b. More importantly, space slicing with more primitives will promote model vectorization with better geometric details and completeness, but at the cost of much more time and memory, as the number of polyhedral cells usually increase rapidly with the insertion of planar primitives. Besides, the time and memory efficiency of space partition depends heavily on a proper insertion order of primitives. To better solve these problems, we propose a novel slicing-based space partition approach by determining the proper boundaries of all the planar primitives based on their neighborhood relationships beforehand to ensure the partition intersection correctness, and applying a novel primitive insertion order to partition the space with a large number of primitives in a time and memory efficient way. To be precise, given a set of planar primitives $\mathcal{P}_\mathbf{s}$ and their 3D bounding box $\mathbf{B_s}$ for the main structure part or one of the isolated object part, our goal is to efficiently slice $\mathbf{B_s}$ into an intermediate set of polyhedrons $\mathcal{C}_\mathbf{s}$, where the final vectorized model can be extracted.

Inspired by Fang and Lafarge (2020) and Fang et al. (2021), an adjacency graph $\mathcal{G}$ is constructed for all the planar primitives $\mathcal{P}$ by finding the adjacent neighborhood of all the planar primitives as shown in Fig. 5d, with each pair of primitives considered adjacently connected if the minimal distance between their inlier points is less than r_s. We initialize the boundary of each primitive as the convex hull of the inlier point projections on its supporting plane as shown in Fig. 4d, and expand the boundary towards its neighboring primitives in the adjacency graph. In this way, the final boundary intersections can be determined in advance before slicing, rather than expanding the boundaries during the slicing procedure as in Fang et al. (2021), so as to avoid incorrect space partition caused by an inproper primitive insertion order. To determine the final boundary $B(\mathbf{P}_i)$ of each primitive $\mathbf{P}_i$, we perform uniform sampling on its initial boundary with the average spacing as s_a and each sample vertex is expanded along the direction away from the centroid of the initial boundary towards its adjacent planar primitives. We traverse all the intersection points where the sample vertices collide with the intersection lines between $\mathbf{P}_i$ and its adjacent primitives, and determine its final boundary as the set of intersection points with the furthest distances from the initial sample vertex positions within a maximal expansion range threshold, whose setting depends on the completeness of the point cloud. Noisy MVS point cloud usually requires a larger threshold to increase the probability of collisions between neighboring primitives, but might produce more unnecessary polyhedral cells. We empirically set it to 2 m, which turns out to be large enough for all our experimental cases. Finally, we update the boundaries of all the planar primitives with the expanded sample vertices, to effectively avoid incorrect space slicing caused by inproper insertion order of planar primitives. The entire primitive pre-expansion process is summarized in Algorithm 1. Figure 5 illustrates that a specific insertion order of primitives in Fig. 5a will result in incorrect space partition and vectorized model without pre-expansion of primitive boundaries as shown in Fig. 5b, c, while Fig. 5d–g demonstrate that our primitive pre-expansion strategy can ensure the correctness of the space slicing by the same insertion order and extract a correct vectorized model. Figure 6 is a real example of the case "Village", where the primitive abstraction strategy with expansion during the insertion process will generate abnormal structures such as the protrusion highlighted in red rectangle of Fig. 6c. As can be seen in Fig. 6f, our primitive pre-expansion strategy avoids the incorrect structures by precomputing the expanded primitive boundaries, with the expanded boundary sample vertices marked as red points in Fig. 6d.

With the pre-computed primitive boundaries, we then insert the planar primitives $\mathcal{P}_\mathbf{s}$ into the bounding box $\mathbf{B_s}$ to generate its space partition which consists of a set of polyhedrons whose edges are consistent with the pre-determined boundaries. An intuitive primitive insertion strategy is to sort planar primitives according to their sizes and structures, and insert them in the sorted order, which is applied in some works like (Boulch et al., 2014; Chauve et al., 2010; Fang et al., 2021). This is a simple but inefficient method, which turns out to have difficulties in handling actual scenes containing thousands of planar primitives. To better solve the problem of time and memory consumption during space

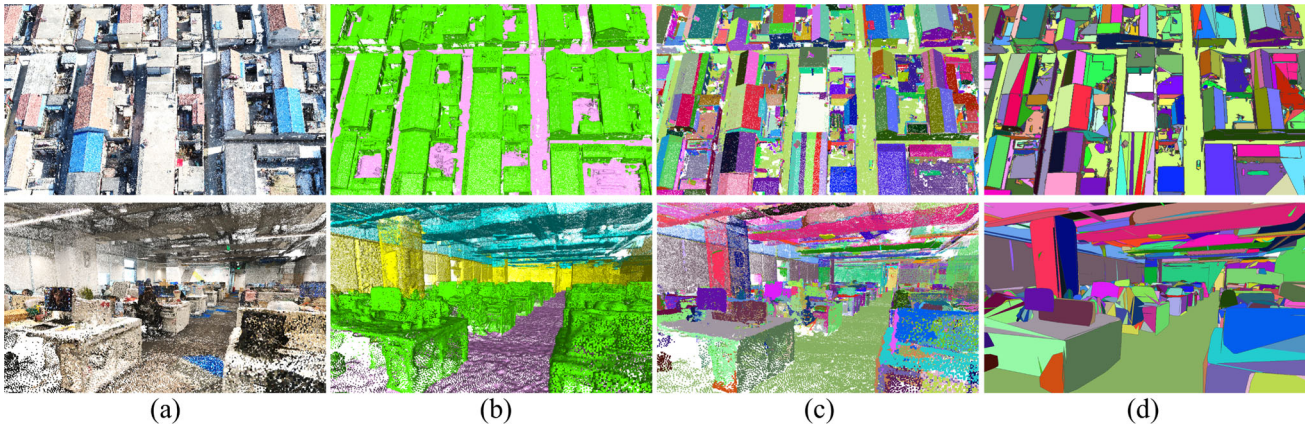

Fig. 4 **a** 3D point clouds of cases "Village" and "Office". **b** Decomposition of (**a**) by semantic categories. **c** 3D planar primitives of (**b**). **d** The supporting planes of (**c**) with convex polygonal boundaries of their inlier projections, denoted by corresponding colors

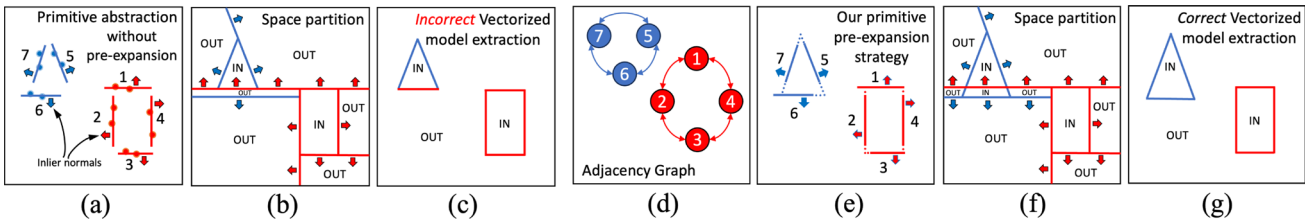

Fig. 5 A 2D simulation of space partition and vectorized model extraction with and without planar primitive pre-expansion. **a** A set of planar primitives with the insertion order denoted by the number, which are abstracted without pre-expansion. **b** Space partition and *in*/*out* labeling by (**a**). **c** Incorrect vectorized surface extraction from (**b**). **d** Adjacency graph constructed from (**a**). **e** Pre-expansion of (**a**) by adjacency graph of (**d**). **f** Correct space partition and *in*/*out* labeling by (**e**). **g** Vectorized model extracted from (**f**) with correct surfaces

Algorithm 1 Planar Primitive Pre-expansion

Require:
$\mathcal{P} = \{\mathbf{P}_i \mid i = 1, \ldots, N\}$: initial planar primitive set;
$\mathcal{G}$: adjacent graph of $\mathcal{P}$;
s_a: average spacing of input point cloud $\mathcal{D}$;
Output: Plane primitive set $\mathcal{P}$ with expanded boundaries;

for $\mathbf{P}_i \in \mathcal{P}$ **do**
  Denote the initial boundary of $\mathbf{P}_i$ as $B(\mathbf{P}_i)$;
  Find the neighboring primitives set of $\mathbf{P}_i$ from $\mathcal{G}$ as $\mathcal{N}(\mathbf{P}_i)$;
  Calculate the intersection lines between $\mathbf{P}_i$ and $\mathcal{N}(\mathbf{P}_i)$ as $L(\mathbf{P}_i)$;
  Perform uniform sampling on $B(\mathbf{P}_i)$ by s_a with the sample vertices denoted as $V_s(\mathbf{P}_i)$;
  for $\mathbf{V} \in V_s(\mathbf{P}_i)$ **do**
    Denote the initial position of $\mathbf{V}$ as $\hat{\mathbf{V}}$;
    while $\mathbf{V}$ is within $2m$ from $\hat{\mathbf{V}}$ **do**
      Expand $\mathbf{V}$ along the direction away from the centroid of $B(\mathbf{P}_i)$
      if $\mathbf{V}$ collides with any line of $L(\mathbf{P}_i)$ **then**
        Update the position of $\mathbf{V}$;
      end if
    end while
  end for
  Update $B(\mathbf{P}_i)$ as $V_s(\mathbf{P}_i)$ with the expanded positions;
end for

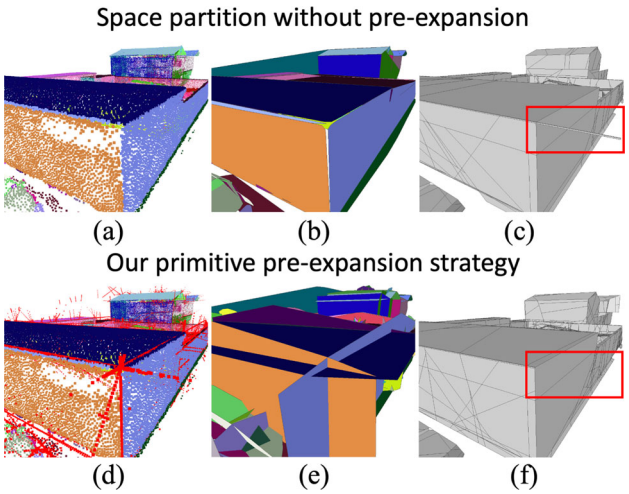

Fig. 6 A real 3D space partition example of the case "Village" with and without primitive pre-expansion. **a** A set of primitives. **b** Space partition by (**a**). **c** Vectorized surface extraction from (**b**) with incorrect structure as highlighted in red rectangle. **d** Pre-expansion of (**a**). **e** Space partition by (**d**). **f** Correct vectorization of (**e**)

division, we innovatively propose a new planar primitive insertion order strategy which considers more about the positional relationships of primitives to solve the space partition problem in a more efficient way.

Considering that the more spaces are partitioned, the more time and memory consumption is required for the space

partition procedure, we tend to design the insertion order of the planar primitives that minimize the final number of polyhedral cells. To achieve this goal, we select an optimal planar primitive respectively for each polyhedral cell to be sliced into two subspaces, by taking into account the positional relationships among the planar primitives. Specifically, for all the polyhedral cells of the BSP tree leaf nodes $\{\mathbf{C}_k \mid k = 1, \ldots, K\}$, we keep a list of associated primitive sets $\{\hat{\mathcal{P}}_k \mid k = 1, \ldots, K\}$, with each associated primitive set $\hat{\mathcal{P}}_k$ containing all the planar primitives which traverse through the space cell $\mathbf{C}_k$. For each planar primitive $\mathbf{P}_i \in \hat{\mathcal{P}}_k$, we calculate its score $W_k(\mathbf{P}_i)$ for slicing $\mathbf{C}_k$ with the formula as follow:

$$W_k(\mathbf{P}_i) = A(\mathbf{P}_i)|\hat{\mathcal{P}}_k| - w_c|\hat{\mathcal{P}}_k^c| - w_b||\hat{\mathcal{P}}_k^l| - |\hat{\mathcal{P}}_k^r||, \qquad (1)$$

where $|\hat{\mathcal{P}}_k|$ is the number of primitives in $\hat{\mathcal{P}}_k$, and $A(\mathbf{P}_i)$ is a normalized area of primitive $\mathbf{P}_i$ divided by the average of the maximal ten primitive areas, which encourages larger primitives to be selected. $|\hat{\mathcal{P}}_k^l|$, $|\hat{\mathcal{P}}_k^r|$ and $|\hat{\mathcal{P}}_k^c|$ count the number of primitives in $\hat{\mathcal{P}}_k$ that lie on the left side, right side, and across both sides of the primitive $\mathbf{P}_i$ respectively accrording to the positional relationships of the planar primitive set $\hat{\mathcal{P}}_k$. w_c and w_b are the weights set to 20 and 10 respectively in our experiments. The variation of the two weights won't sensitively affect the final vectorization result, but will have influence on the number of sliced space cells and the time and memory consumptions. Table 1 gives the statistics of the number of partitioned polyhedral cells denoted by $|\mathcal{C}_\mathbf{m}|$, the number of associated polygonal faces by $|\mathcal{F}_\mathbf{m}|$, time efficiency and memory consumption of different weight settings to verify that our setting of w_c and w_b is the best to achieve the lowest time and memory costs. The cross term $-w_c|\hat{\mathcal{P}}_k^c|$ discourages the selected primitive to have too many intersections with other primitives, so that the planar primitives sliced to the left subspace will not affect the subsequent division of the right subspace and vice versa. The balance term $-w_b||\hat{\mathcal{P}}_k^l| - |\hat{\mathcal{P}}_k^r||$ encourages the remaining planar primitives to be bisected into the two subspaces equally. We select the planar primitive with the maximal score in $\hat{\mathcal{P}}_k$ to divide the convex polyhedral cell space $\mathbf{C}_k$ into two new subspaces $\mathbf{C}_k^l$ and $\mathbf{C}_k^r$. For the rest planar primitives in $\hat{\mathcal{P}}_k$, we check their intersections with the two newly generated subspaces and assign them to the associated primitive sets of $\mathbf{C}_k^l$ and $\mathbf{C}_k^r$. We iteratively perform these operations on all the polyhedral cells including the recursively generated ones, until all the leaf node spaces can no longer be divided. The entire space partition process is summarized in Algorithm 2.

Actually, the positional relationships among the planar primitives which record whether each primitive lies on the left side, right side or across both sides of another primitive, are fixed after the planar primitive expansion finishes. Therefore, we precompute a positional relationship map of all the primitives beforehand in our implementation for speeding up the score calculation during the space partition.

Algorithm 2 BSP Space Partition

Require:
$\mathcal{C}_\mathbf{s} = \{\mathbf{B}_\mathbf{s}\}$: initial polyhedral cells as a single-node BSP tree, which contains only the bounding box of $\mathcal{P}_\mathbf{s}$;
Output:
$\mathcal{C}_\mathbf{s} = \{\mathbf{C}_k \mid k = 1, \ldots, K\}$: the final polyhedral cells partitioned by our BSP slicing;

```
while C_s contains newly generated leaf nodes do
  for each new leaf node C_k ∈ C_s do
    Denote P̂_k as the associated primitive set of C_k;
    if P̂_k ≠ ∅ then
      for P_i ∈ P̂_k do
        Compute the score of P_i as W_k(P_i);
      end for
      Select the primitive with the maximal score as P_m;
      Divide C_k into two subspaces C_k^l and C_k^r with P_m;
      Add C_k^l and C_k^r to BSP tree as new child leaf nodes of C_k;
      Assign the rest primitives in P̂_k to C_k^l and C_k^r as their
  associated primitive sets;
    end if
  end for
  Update C_s with the current leaf nodes of BSP tree;
end while
```

We give a 2D simulated example in Fig. 7 to illustrate that a proper planar primitive insertion order can reduce the number of sliced polyhedral cells while ensuring the correctness of the space partition result. Lines of various lengths in Fig. 7a represent planar primitives of different areas, with the dotted line sections indicating the adjacency relationship between the lines. Figure 7b is the space partition result by the exhaustive insertion strategy of PolyFit Nan and Wonka (2017) which generates a great many polyhedral cells. Figure 7c is the space division result by the insertion order of maximal area without considering the primitive adjacency relationship, which generates fewer polyhedrons but might has some partition errors in the space associated with multiple primitive intersections as highlighted in the rectangle. Figure 7d gives the result of insertion order by maximal area taking into consideration the primitive adjacency relationship by precomputing the primitive boundaries, which generates a correct space partition with fewer polyhedrons than PolyFit. Figure 7e shows the result of our space partition strategy, which demonstrates that our insertion order can reduce the number of sliced polyhedrons for better time and memory efficiency compared to Fig. 7d, while ensuring the correctness of the partition result. Figure 8 is a real example for a local region of the scene "Village", from which we can see that our space partition strategy can perform the best in both minimal number of polyhedral cells and space partition correctness. We further gives the statistical comparison of the numbers of cells together with time and memory consumptions of the space partition step for the city-scale case "SUM

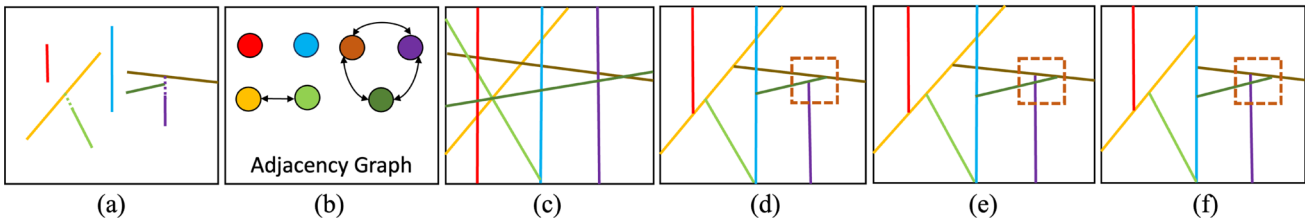

Fig. 7 A 2D simulation of space partition with different primitive insertion strategies. **a** A set of planar primitives. **b** Adjacency graph of (**a**). **c** The space partition result by PolyFit Nan and Wonka (2017). **d** Incorrect space partition by insertion order of maximal area without primitive adjacency relationship. **e** Correct result of insertion order by maximal area considering the adjacency relationship of (**b**). **f** The result of our space partition strategy, which is the best in both minimal number of cells and space partition correctness

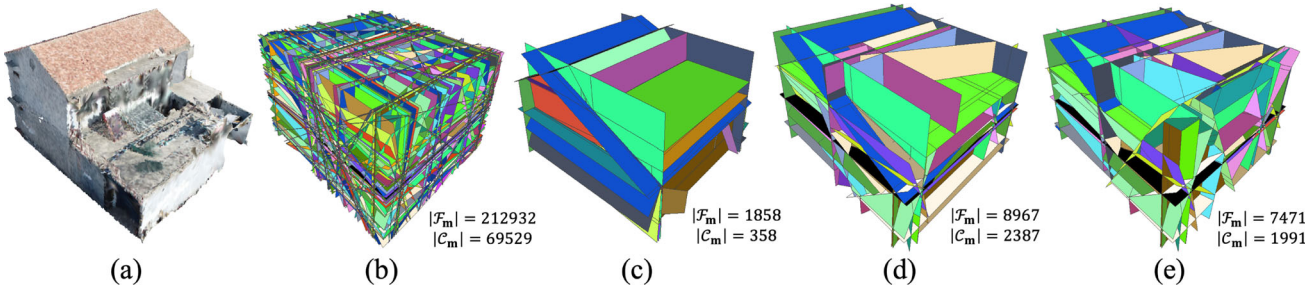

Fig. 8 A real example of space partition with different primitive insertion strategies for a local region of the scene "Village". **a** Point cloud. **b** Space partition result by PolyFit. **c** Space partition by insertion order of maximal area without primitive adjacency relationship. **d** The result of insertion order by maximal area with the adjacency relationship. **e** The result of our space partition strategy, which turns out to be the best in minimal number of cells with correct space partition

Benchmark" on PolyFit, insertion order by maximal area considering the primitive adjacency, and our partition strategy using different numbers of planar primitives in Fig. 9, to demonstrate that our method generates the minimal number of cells with the least consumption of time and memory. Also note that the time and memory consumption and the number of generated cells strongly depends on the number of primitives used for space partition. A larger number of primitives are preferred for better preserving geometric details of the vectorized model. Therefore, for high-quality large-scale vectorization, the time and memory efficiency of space partition step with over thousands of primitives is of great importance.

5.3 Main Vectorized Structure Extraction

After the space partition of the main structure primitives $\mathcal{P}_\mathbf{m}$ completes, we extract a polygonal surface mesh from the set of intermediate polyhedral cells $\mathcal{C}_\mathbf{m} = \{\mathbf{C}_i \mid i = 1, \ldots, M\}$ with all their polygonal facets denoted as $\mathcal{F}_\mathbf{m} = \{\mathbf{F}_i \mid i = 1, \ldots, F\}$. A MRF formulation is applied to label the polyhedral cells as inside or outside, and the final vectorized model is made of the polygonal facets whose two adjacent cells are labeled differently, as illustrated by 2D simulation in Fig. 14f, g.

Our energy function contains a data term E_d and a smooth term E_s defined as follows:

$$
\begin{aligned}
E\left(\mathcal{L}\right) &= E_d\left(\mathcal{L}\right) + \lambda E_s\left(\mathcal{L}\right), \\
E_d\left(\mathcal{L}\right) &= \frac{1}{A} \sum_{\mathbf{C}_i \in \mathcal{C}_\mathbf{m}} \sum_{\mathbf{F}_j \in \partial \mathbf{C}_i} D\left(\mathcal{L}(\mathbf{C}_i), \mathbf{I}(\mathbf{F}_j)\right), \\
E_s\left(\mathcal{L}\right) &= \frac{1}{A} \sum_{(\mathbf{C}_i, \mathbf{C}_j) \in \mathcal{N}(\mathcal{C}_\mathbf{m})} V\left(\mathcal{L}(\mathbf{C}_i), \mathcal{L}(\mathbf{C}_j)\right),
\end{aligned} \tag{2}
$$

where $\mathcal{L} = \{\mathcal{L}(\mathbf{C}_i) \mid \mathbf{C}_i \in \mathcal{C}_\mathbf{m}\}$ with $\mathcal{L}(\mathbf{C}_i) \in \{in, out\}$ denoting the *in*/*out* labeling of polyhedral cells, A is the total area of all the facets of $\mathcal{F}_\mathbf{m}$, $\partial \mathbf{C}_i$ are the set of polygonal facets which compose the boundary of $\mathbf{C}_i$, and $\mathbf{I}(\mathbf{F}_i)$ are all the supporting inlier 3D points inside the polygonal facet $\mathbf{F}_i$. Inspired by KSR Bauchet and Lafarge (2020), the data term E_d measures the coherence between the *in*/*out* label of each polyhedra $\mathbf{C}_i$ and the normal orientations of all the supporting inliers inside its boundary polygonal facets $\partial \mathbf{C}_i$. Specifically, $D\left(\mathcal{L}(\mathbf{C}_i), \mathbf{I}(\mathbf{F}_j)\right)$ is a voting function to evaluate the confidence of the label $\mathcal{L}(\mathbf{C}_i)$, which is defined as follows:

$$
\begin{aligned}
&D\left(\mathcal{L}(\mathbf{C}_i), \mathbf{I}(\mathbf{F}_j)\right) \\
&\quad = \min\left(A(\mathbf{F}_j), \lambda_d \pi s_a^2 \sum_{\mathbf{X} \in \mathbf{I}(\mathbf{F}_j)} d\left(\mathbf{X}, \mathcal{L}(\mathbf{C}_i)\right)\right),
\end{aligned} \tag{3}
$$

where s_a is the average spacing of $\mathcal{D}$, λ_d is a weight set to 0.7 in our experiments, and $d\left(\mathbf{X}, \mathcal{L}(\mathbf{C}_i)\right) \in \{0, 1\}$ is a binary

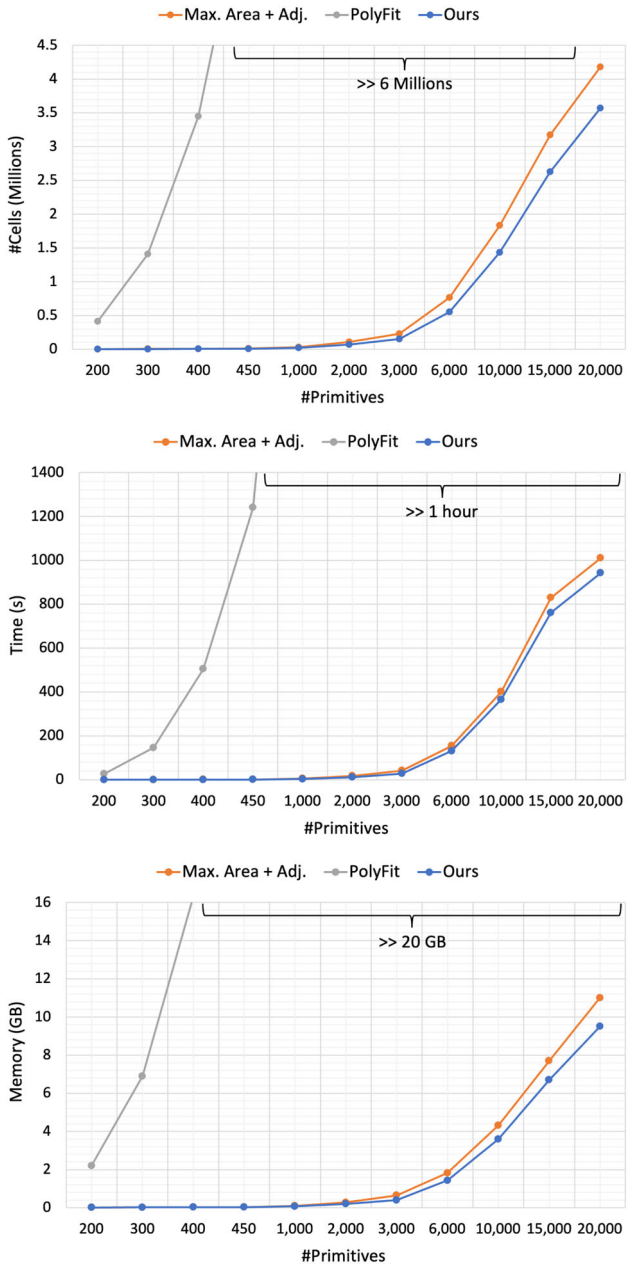

Fig. 9 The statistical comparisons of the numbers of generated cells, and time and memory consumption for case "SUM Benchmark" on different primitive insertion strategies, which verify that our strategy produces the minimal number of cells with the smallest time and memory consumption

function to check the consistency between the normal of $\mathbf{X}$ and the label $\mathcal{L}(\mathbf{C}_i)$. We define $\vec{n}(\mathbf{X})$ as the normal of $\mathbf{X}$ and $\vec{v}(\mathbf{X})$ is the direction from $\mathbf{X}$ to the centroid of polyhedra $\mathbf{C}_i$. If $\vec{n}(\mathbf{X}) \cdot \vec{v}(\mathbf{X}) > 0, d\,(\mathbf{X}, in) = 1$ and $d\,(\mathbf{X}, out) = 0$. Otherwise, $d\,(\mathbf{X}, in) = 0$ and $d\,(\mathbf{X}, out) = 1$. All the inlier points in $\mathbf{I}(\mathbf{F}_i)$ are checked to vote for the label $\mathcal{L}(\mathbf{C}_i)$. The smooth term E_s controls the complexity of the extracted surface mesh by minimize its total area, in which $\mathcal{N}(\mathcal{C}_\mathbf{m})$ represents all the pairs of adjacent cells in $\mathcal{C}_\mathbf{m}$. If $\mathcal{L}(\mathbf{C}_i) \neq \mathcal{L}(\mathbf{C}_j)$, $V\left(\mathcal{L}(\mathbf{C}_i), \mathcal{L}(\mathbf{C}_j)\right) = A(\partial\mathbf{C}_i \cap \partial\mathbf{C}_j)$, where $\partial\mathbf{C}_i \cap \partial\mathbf{C}_j$ is the common boundary facet of $\mathbf{C}_i$ and $\mathbf{C}_j$, and $A(\partial\mathbf{C}_i \cap \partial\mathbf{C}_j)$ is its area. Otherwise, $V\left(\mathcal{L}(\mathbf{C}_i), \mathcal{L}(\mathbf{C}_j)\right) = 0$. $\lambda \in [0, 1]$ is a weight we use to balance the data term and smooth term. As shown in Fig. 10, a small value of λ will improve the completeness of the vectorized model for the missing input point cloud, but produce a less compact model with more facets, while λ with too large value will generate a more compact model but result in the disappearance of some structures. We set $\lambda = 0.2$ for the main structure of outdoor scenes, and $\lambda = 0.5$ for indoor main structure and isolated objects in our experiments to best balance the completeness and compactness of the vectorized model.

Such MRF formulation can be solved by graph-cut algorithm proposed in Boykov et al. (2001), Kolmogorov and Zabin (2004) and Boykov and Kolmogorov (2004) to get an optimal *in*/*out* labeling of the main structure polyhedrons $\mathcal{C}_\mathbf{m}$. All the polygonal facets $\mathcal{F}_\mathbf{m}$ whose two adjacent cells have different labels are assembled to construct the vectorized model of the main structure, as shown in Fig. 11. Then, we iteratively merge two coplanar neighboring facets which have only one common edge to remove the redundant edges in the extracted polygonal surface model. Figure 12a gives a 2D simulated example of facet merging process, from which we can see that the iterative merging stops if the current merging step will produce an inner hole structure, to make sure of a simple polygonal surface representation without hole. This iterative facet merging will produce a more compact vectorized model, as the real case "Office" shown in Fig. 12d. In addition, the inlier 3D points of the planar primitives are attached to the corresponding polygonal facets, which will be used as important priors for the vectorization of the isolated objects, as described in Sect. 5.4.

5.4 Object Vectorized Model Extraction

The vectorized models of the isolated objects are reconstructed by a pipeline similar to that for the main structure described in Sect. 5.3, except for the difference that the vectorized model of the main structure is used as priors for vectorizing isolated objects to ensure topological consistency on the intersections between the objects and the main structure, since our system aims at reconstructing a geometrically complete and topological consistent polygonal surface mesh of the entire scene.

To ensure topological consistency of the intersections between each isolated object and the main structure, our straightforward solution is to slightly enlarge the bounding box of the isolated object by 1 m, and consider those polygonal facets of the main structure vectorized model entirely or partially contained by the expanded bounding box as prior primitives, as illustrated in Fig. 13a, b. These prior primi-

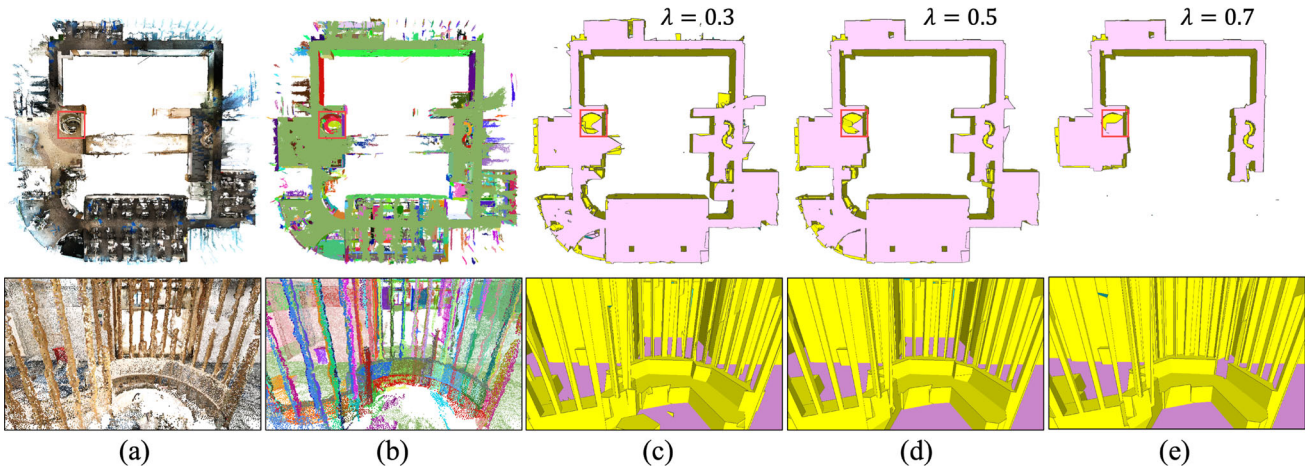

Fig. 10 The influence of different values of smooth weight λ on completeness and compactness of the vectorized model of case "Office". **a** Point cloud. **b** Planar primitives. **c–e** Vectorized models by different values of λ

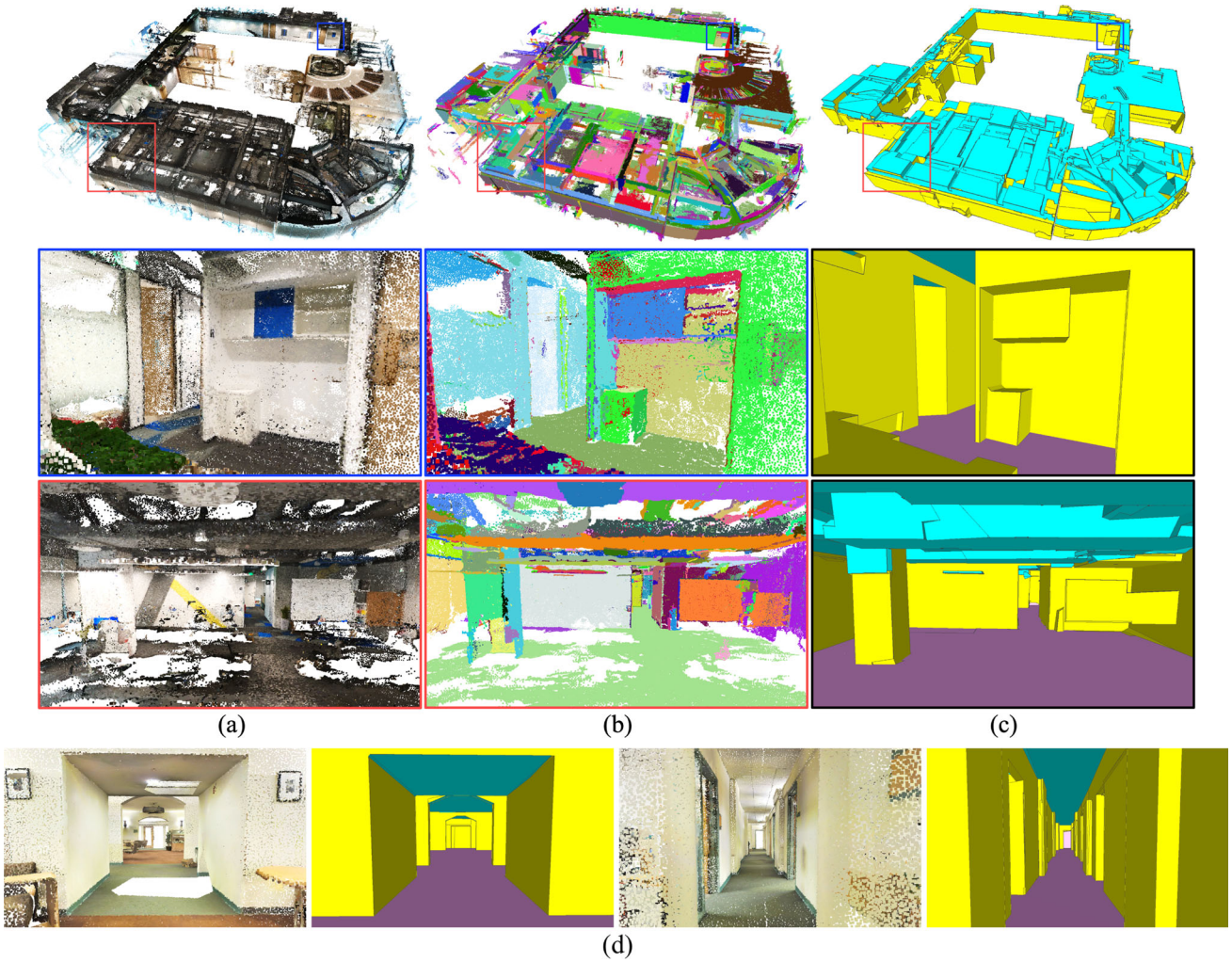

Fig. 11 Main structure vectorization results. **a** Point cloud of case "Office". **b** Main structure planar primitives of (**a**). **c** Main structure vectorized modeling of (**a**). **d** Vectorized main structure of case "area3"

tives are used along with the primitives of the isolated object together for its vectorized modeling, with the main structure inlier points and the inliers of the object treated equally for the *in*/*out* labeling voting of data term as in Eq. 3. However, this strategy will suffer from the inlier lack problem that some polygonal facets of the main structure do not have any supporting inlier, which might cause these prior primitives to disappear during the vectorized modeling of the isolated object. As shown in the 2D simulated example of Fig. 14a, the inlier lack problem is mainly caused by two or more planar primitives which are approximately coplanar but separated by the isolated object. Since only one of them will be extracted as a polygonal facet of the vectorized main structure model in most of these cases like in Fig. 14b, the inlier points of the other approximately coplanar primitives will not appear in the main structure polygonal facets like the primitive on the right side of the object in Fig. 14c, which generates incorrect space cell labeling and leads to topological inconsistency in the vectorized model of the object as shown in Fig. 14c. To solve this problem, we propose to rearrange the inlier points of the main structure, by finding out all the inliers of the original main structure planar primitives which haven't been attached to any of the main structure polygonal facet, and attaching each inlier point to the closest polygonal facet which meets the following three conditions. First, the normal deviation between the polygonal facet and the inlier point does not exceeds an angular threshold whose value depends on the normal smoothness of the point cloud, and is empirically set to 35° for all the scenes reconstructed by MVS. Second, the distance from the inlier point to the facet is smaller than 1 m. Third, the perpendicular projection of the inlier point is inside the boundary of the facet, which is always a convex hull in our cases. After the inlier rearrangement finishes, those primitives which lack inliers will have sufficient inlier voting supports for *in*/*out* labeling. All the main structure polygonal facets which are entirely or partially contained by the isolate object bounding box with their rearranged inliers constitute the prior primitives, and are then combined with the planar primitives of the isolated object for space partition and *in*/*out* labeling to ensure a better topological consistency of the extracted vectorized model, as shown in Fig. 14d. The space cell labeling and polygonal surface extraction for the objects are performed with the same approach as the main structure. Figure 15a–f gives a real example of case "Village" (Fig. 13). Figure 15b contains two approximately coplanar *ground* prior primitives colored in blue and red, where the red colored primitive contains no inlier, which leads to a vectorized object model with inconsistent topology in Fig. 15c. After applying our inlier rearrangement strategy, both prior primitives contains enough inliers which ensures correct *in*/*out* labeling to generate topological consistent vectorized object model, as shown in Fig. 15d.

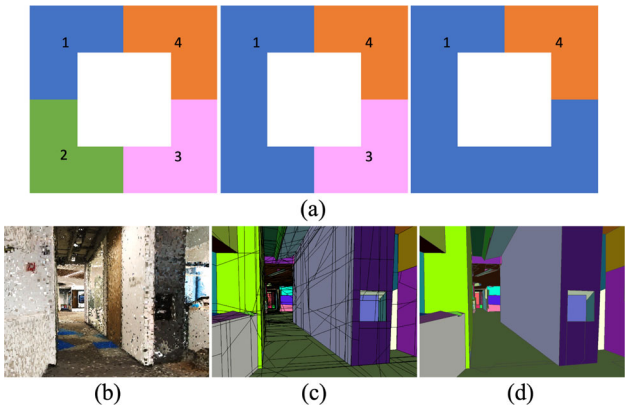

Fig. 12 **a** A 2D simulation of facet merging process. **b** Point cloud of case "Office". **c** Main structure vectorized model of (**b**) before facet merging. **d** Vectorized model of (**b**) after merging, which is a more compact polygonal surface without redundant edges

To avoid topological redundancy between the isolated objects and the main structure surface such as the blue colored polygonal facets shown in Fig. 15e, we further clean up the vectorized model of each object by removing those polygonal facets which exactly overlap with the vectorized model of main structure or contain only supporting inliers from the main structure. The cleaned vectorized object model is shown in Fig. 15f. The vectorized models of all the isolated objects and the main structure compose a complete vectorized model of the entire scene, as the cases "Village", "area3" and "area1" shown in Fig. 15g.

5.5 Semantic Segmentation Refinement

The accuracy of the semantic decomposition is crucial for high-quality vectorized modeling of the main structure. However, there might exist some semantic segmentation errors which will influence the vectorization result, as shown in Fig. 16b, c. To further improve the vectorized modeling result, we propose to feed the vectorized model of the main structure back to the 3D semantic segmentation module to improve the segmentation accuracy.

As can be seen in Fig. 16b, most semantic segmentation errors occur at the boundaries of different semantic labels. We can re-segment 3D points to the semantic labels by taking the vectorized main structure as geometric constraints. To take semantic label *ground* as an example, points segmented as *ground* within a distance threshold d_r to the vectorized ground planes are considered as determined, while points with *ground* label but more than distance d_r far from the vectorized ground structure will be considered as uncertain points. The setting of d_r generally depends on the smoothness of the surface point cloud. In our experiments, we empirically set it to 0.1 m for indoor scenes and 0.2 m for outdoor ones. We conduct the same uncertainty determination for all

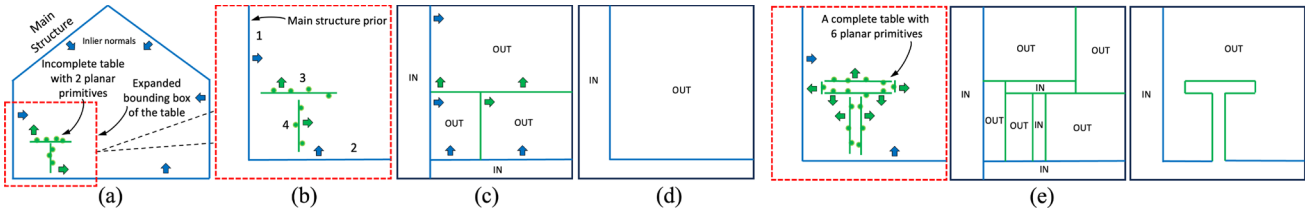

Fig. 13 2D simulation of a failure case of vectorizing a non-watertight object. **a** A vectorized main structure and an incomplete table point cloud to be vectorized, with only two primitives abstracted. **b** The expanded bounding box of the table and the main structure prior. **c** Space partition for the table with inside/outside labeling optimization. **d** The vectorization result of (**b**), in which the table disappears. **e** A completely scanned table with all its planar primitives abstracted, which produces a correct final vectorized model with the table structure extracted

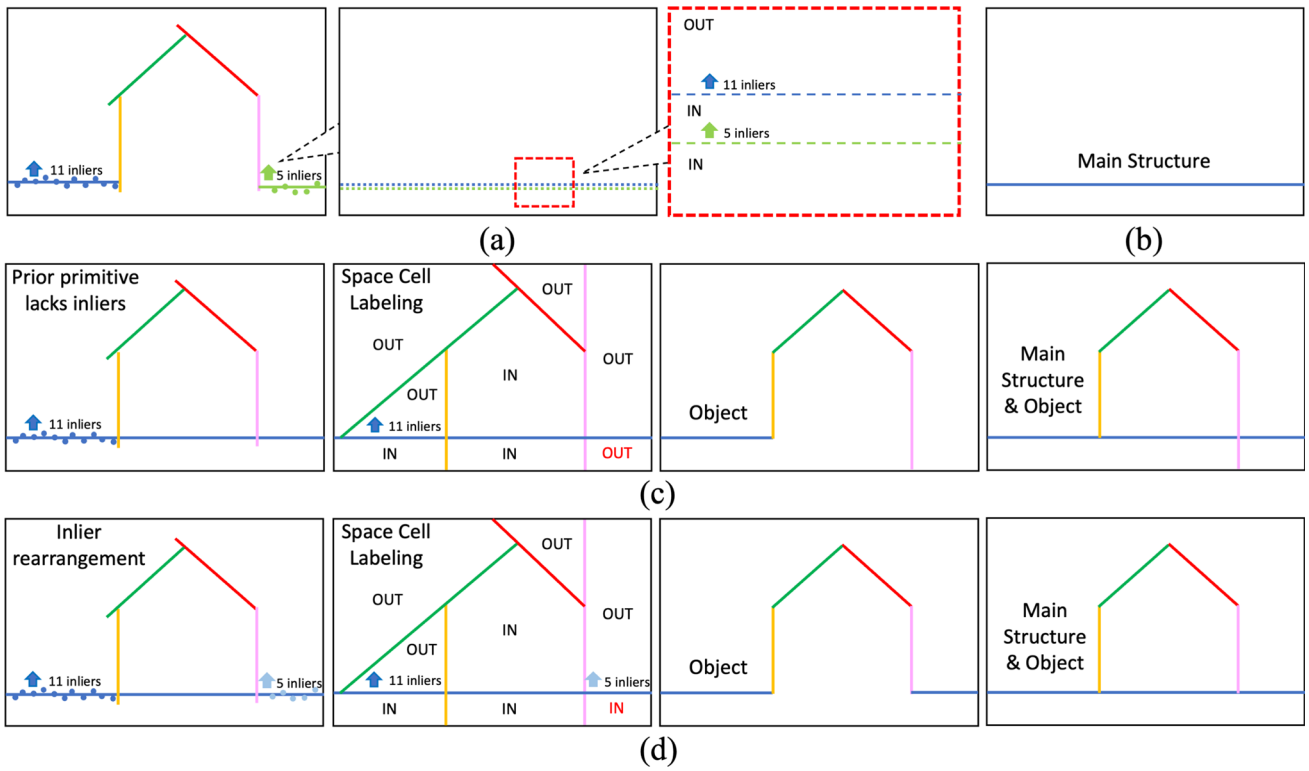

Fig. 14 A 2D simulation of the inlier lack problem. **a** A set of planar primitives with two approximately coplanar *ground* primitives. **b** Vectorized main structure of *ground*. **c** Space partition by the object planar primitives with *ground* structure as prior, which generates incorrect *in*/*out* labeling and inconsistent topology caused by inlier lack problem. **d** Correct space cell labeling by our inlier rearrangement strategy with *ground* structure as prior, which ensures topological consistency of the entire vectorized model

the main structure semantic labels *ground*, *ceiling* and *wall*, and apply a MRF formulation to perform semantic segmentation refinement for the uncertain points. We formulate energy function for the segmentation refinement problem as follows:

$$\begin{aligned} E\left(S\right) &= E_d\left(S\right) + \lambda_s E_s\left(S\right), \\ E_d\left(S\right) &= \sum_{\mathbf{X}\in\mathcal{D}} D_s\left(\mathbf{X}, S(\mathbf{X})\right), \\ E_s\left(S\right) &= \sum_{(\mathbf{X},\mathbf{Y})\in N(\mathcal{D})} V_s\left(S(\mathbf{X}), S(\mathbf{Y})\right), \end{aligned} \tag{4}$$

where $S(\mathbf{X}) \in \mathbf{S}$ is the semantic label of 3D point $\mathbf{X}$ to be predicted, and $N(\mathcal{D})$ denotes the set of neighboring point pairs with a maximal distance threshold d_n between each neighboring pair $(\mathbf{X}, \mathbf{Y})$. We empirically set d_n to 1 m, which turns out to be large enough to semantically relate possible neighboring points for large-scale scenes. The data term $E_d\left(S\right)$ measures the probability of each 3D point $\mathbf{X}$ segmented to $S(\mathbf{X})$, where function $D_s\left(\mathbf{X}, S(\mathbf{X})\right)$ has different probability definitions for determined points and uncertain points, which

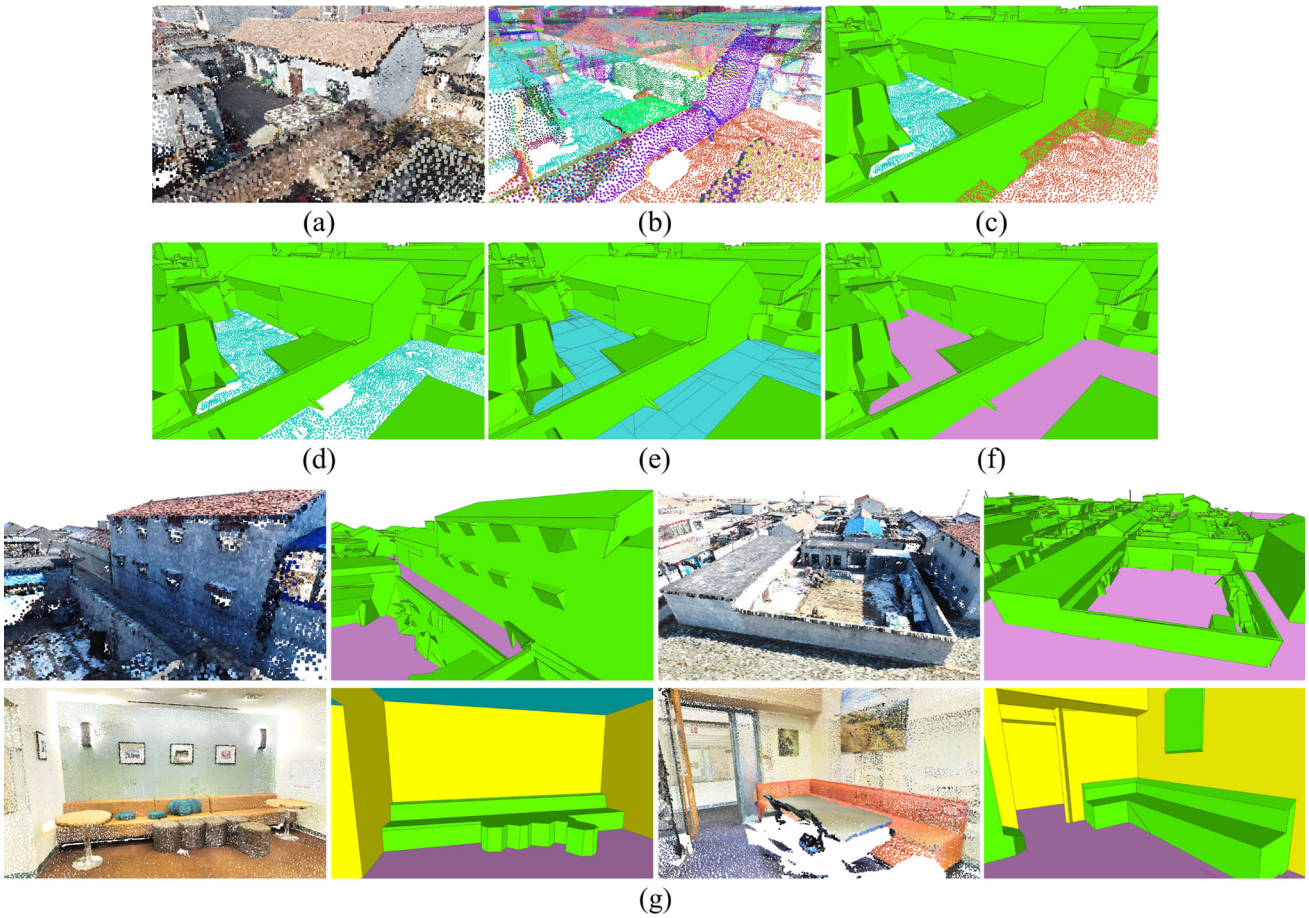

Fig. 15 Real examples of object vectorization. **a** A local region of point cloud for case "Village". **b** Planar primitives of (**a**). **c** Object vectorization of (**a**) with inconsistent topology caused by inlier lack problem. **d** Object vectorization with topological consistency ensured by inlier rearrangement. **e** Topological redundancy between the objects and the *ground* structure, colored in blue. **f** Vectorized object model of (**a**) after cleaning up redundancy. **g** Other local regions of the entire vectorized models of cases "Village" in top row, "area3" in bottom left and "area1" in bottom right, all of which contain both isolated objects and main structure

is defined as follows:

$$D_s\left(\mathbf{X}, S(\mathbf{X})\right) = \begin{cases} -log\left(P_s(\mathbf{X}, S(\mathbf{X}))\right) & \mathbf{X} \text{ is determined} \\ -log(\frac{1}{|\mathbf{S}|}) & \mathbf{X} \text{ is uncertain} \end{cases}, \tag{5}$$

where we continue to use the fused 3D semantic probability $P_s(\mathbf{X}, S(\mathbf{X}))$ for a determined point, while an averaged probability is simply used for an uncertain 3D point instead to eliminate the influence of the inaccurate semantic probability feature. The smooth term $E_s(S)$ encourages the uncertain points to be relabeled semantically consistent with the neighboring determined points. If $S(\mathbf{X}) \neq S(\mathbf{Y})$, $V_s\left(S(\mathbf{X}), S(\mathbf{Y})\right) = -log\left(\arccos\left(\vec{n}(\mathbf{X}) \cdot \vec{n}(\mathbf{Y})\right)/\pi\right)$, where $\vec{n}(\mathbf{X})$ and $\vec{n}(\mathbf{Y})$ are the normals of $\mathbf{X}$ and $\mathbf{Y}$ respectively. Otherwise, $V_s\left(S(\mathbf{X}), S(\mathbf{Y})\right) = 0$. Such smooth term definition forces the segmentation boundaries to coincide with the geometric boundaries with inconsistent normals between neighboring points. λ_s is the smoothness weight, which we set to 0.1 in our experiments.

We solve the energy minimization problem by max-flow/min-cut algorithm proposed in Boykov et al. (2001), Boykov and Kolmogorov (2004) and Kolmogorov and Zabin (2004) to get an optimized semantic labeling S^* for the 3D point cloud $\mathcal{D}$. The refined semantic labels are mapped to the semantic categories to get a refined semantic category labeling $\hat{S}^*$ for $\mathcal{D}$. From the refined semantic segmentation shown in Fig. 16d, we can see that by incorporating the vectorized main structure model as geometric constraints, the segmentation errors at the boundaries between different semantic categories can be significantly reduced. As demonstrated in the two cases "Village" and "Office" of Table 2, the semantic segmentation accuracy in MIoU is significantly improved after the refinement step, compared to the semantic MIoU before refinement. With the refined semantic labeling, the scene is re-decomposed into main structure and isolated

objects. Planar primitive abstraction and space partition are performed on the relabeled point cloud to obtain the finally refined polygonal surface meshes of main structure and isolated objects, as shown in Fig. 16e.

6 Experimental Evaluation

In this section, we evaluate our vectorized modeling pipeline on both high-quality point clouds captured by 3D scanner and more noisy point clouds generated by MVS approach. We experiment on twenty cases composed of ten indoor scenes and ten outdoor ones. Seventeen of them are with MVS point clouds including ten outdoor cases "Village" captured by DJI Phantom 4 Pro, "Century Park1", "Century Park2" and "ZJU CCE" by DJI Phantom 4 RTK, "Taian Ancient Street" by DJI Mavic 2, "Lingang" and "World Expo" by DJI Matrice 300 RTK + Zenmuse P1, "Barn" and "Light House" from the Tanks and Temples Benchmark Knapitsch et al. (2017), and the city-scale "SUM Benchmark" Gao et al. (2021), and seven indoor scenes "Office" captured by Insta360 ONE R, "Tea Room" by Insta360 ONE RS, and "Church", "Auditorium", "Ballroom", "Museum" and "Meeting Room" from the Tanks and Temples, each of which is composed of multi-view digital images, camera poses and dense point cloud reconstructed by DP-MVS Zhou et al. (2021) except for the SUM Benchmark whose meshes are generated by off-the-shelf commercial software ContextCapture[2] as claimed by the benchmark team, and further upsampled by us using Poisson-disk sampling Corsini et al. (2012) with average spacing as 0.4 m to generate a more dense and well-distributed input point cloud. We also experiment on another three indoor cases "area1", "area3" and "area4" from Stanford Large-Scale 3D Indoor Spaces Dataset (S3DIS) Armeni et al. (2017), with each case containing multi-view images, camera poses and point cloud scanned by Matterport Camera equipped with three structured-light sensors and annotated with predefined semantic labels. Note that for each S3DIS case, we skip the semantic segmentation and refinement steps because reliable semantic labeling is already given together with the scanned point cloud as prior. All the experiments are conducted on a computing platform with a 14-Core Intel Xeon E5-2680 CPU and 500 GB memory by Red Hat Linux 4.8.5-4 OS, except for KSR whose Windows executable program runs on a desktop PC with Intel Core i7-11700 CPU and 128 GB memory by Windows 10 OS. Quantitative and qualitative comparisons of our work to the SOTA vectorization methods are reported to show that our vectorized modeling pipeline achieves the best quality in geometric accuracy and details for point clouds of indoor and outdoor scenes. We also report the time and memory consumption of different methods to show the contributions of our work to the outstanding reconstruction scalability and time and memory efficiency of vectorized modeling for large-scale scenes.

Figures 1, 15 and 16 have already demonstrated the vectorized models of cases "Village", "Office" and "area3" in details, together with the accuacy evaluation to show the effectiveness of our vectorized modeling pipeline in better vectorization quality. Figures 17 and 18 provide the vectorized modeling results of cases "Taian Ancient Street" and "SUM Benchmark" respectively, from which we can see that our pipeline can faithfully vectorize high-quality city-scale models while preserving local geometric details as well in a polygonal surface representation, like the magnified local regions shown in Figs. 17e and 18e. For the cases "Church", "area1" and "area4", we qualitatively and quantitatively compare our vectorization approach against other SOTA vectorized modeling works including PolyFit Nan and Wonka (2017), KSR Bauchet and Lafarge (2020) and VecIM Han et al. (2021a) on the generated vectorized models, since other cases even cannot be run successfully by all the SOTA methods on our computing platform due to their huge memory consumption on large-scale point clouds. For PolyFit, apart from directly experimenting it on the three cases, we also combine it with the semantic segmentations of each case, by applying PolyFit on the main structure and each isolated object part separately. As the comparison results of "Church", "area1" and "area4" shown in Figs. 19, 20 and 21 respectively, PolyFit, KSR and VecIM focus on vectorizing the main structure of the scenes, and will lose some geometric details of isolated objects in the final polygonal models if the scenes are large with complicated object structures. This is because for large-scale scenes the number of planar primitives used for vectorization should be large enough to ensure the geometric details, which will causes PolyFit, KSR and VecIM to consume huge execution time. Too many primitives will cost PolyFit extremely long time in its mixed-integer programming based optimization step. Therefore, empirically, there should be no more than 120 primitives for PolyFit to make sure of a successful vectorization, which will inevitably lose some local details. Moreover, VecIM turns out to be more suitable for Manhattan World indoor scenes, and might have difficulties in handling planar surfaces with nondominant directions or isolated objects. KSR handles the collisions of planar primitives dynamically to improve space partition efficiency, but still requires more than one day and huge memory to process a large scene with over two thousands primitives due to its high computational and memory cost. As can be seen in Figs. 19, 20 and 21, PolyFit combined with semantic segmentation can extract polygonal structures of some isolated objects, but the geometric accuracy of the vectorized objects is not guaranteed because only 30 $\sim$ 70 primitives are used for

[2] https://www.bentley.com/en/products/brands/contextcapture

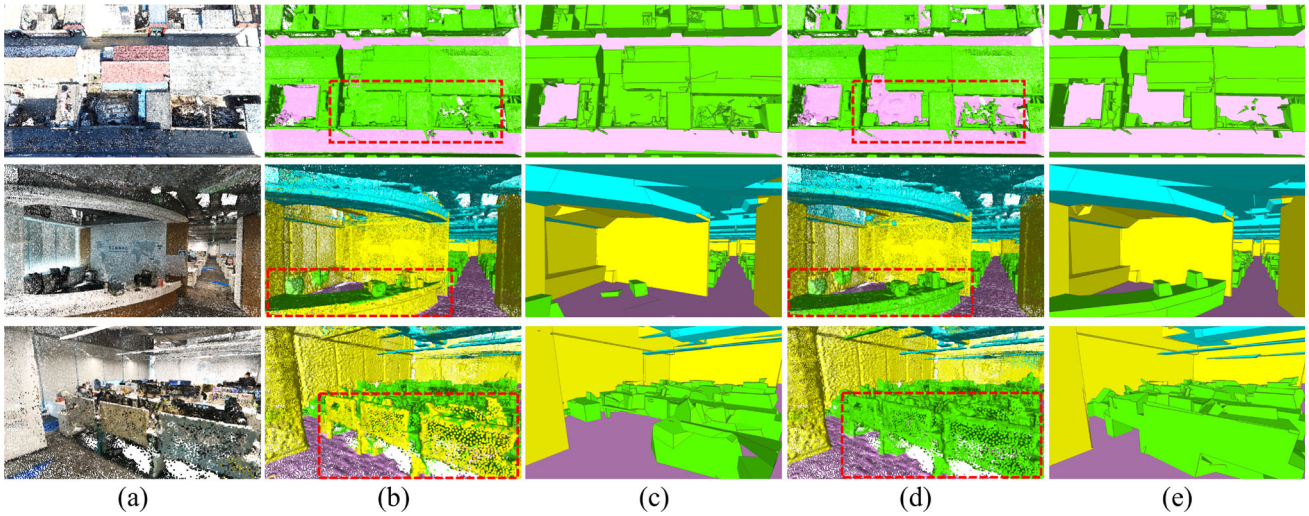

Fig. 16 Semantic segmentation refinement of case "Village" for top row, and "Office" for middle and bottom rows. **a** Point clouds. **b** Semantic segmentations with errors highlighted in red rectangles. **c** Vectorization results by (**b**) with incorrect structures. **d** Refined semantic segmentations. **e** Refined vectorized models by (**d**)

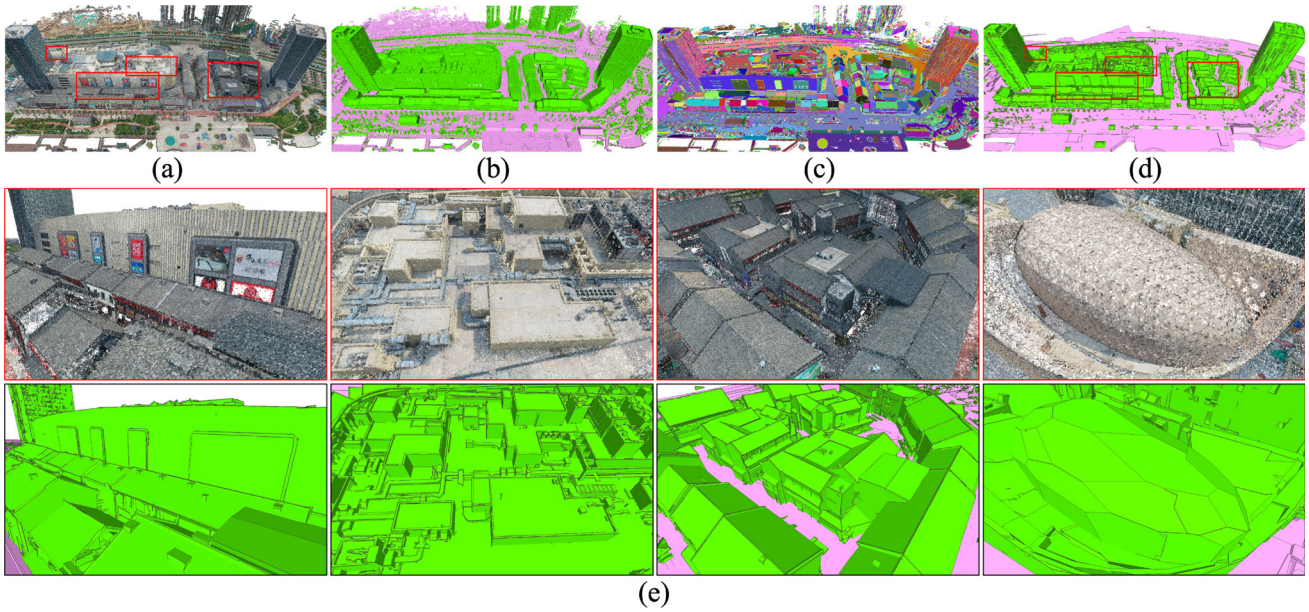

Fig. 17 **a** Point cloud of "Taian Ancient Street". **b** Segmentation by semantic categories. **c** Planar primitives. **d** Vectorized modeling result. **e** Magnified local regions of (**a**) and (**d**) to show the preserved geometric details in the vectorized model

each isolated object part to ensure successful vectorization. In comparison, our vectorization pipeline can successfully handle thousands of primitives and perform better than the other works in the finally generated high-quality vectorized models with more complete polygonal surfaces and better local geometric details of complicated structures. We further provide quantitative evaluation of the vectorization results of the three cases by PolyFit, PolyFit with semantic segmentation, KSR, VecIM, and our method in Fig. 22. For vectorized modeling accuracy evaluation, we use MeshLab[3] to produce Mean Hausdorff Error (MHE) Cignoni et al. (1998) and Root Mean Squared Error (RMSE) between the vectorized models and their input point clouds. From the model accuracy evaluation in Fig. 22 we can see that our pipeline reconstructs the vectorized models with a centimeter-to-decimeter-level accuracy, which turns out to be the best overall in both MHE and RMSE. Beside "Church", "area1", "area3" and "area4",

[3] https://www.meshlab.net.

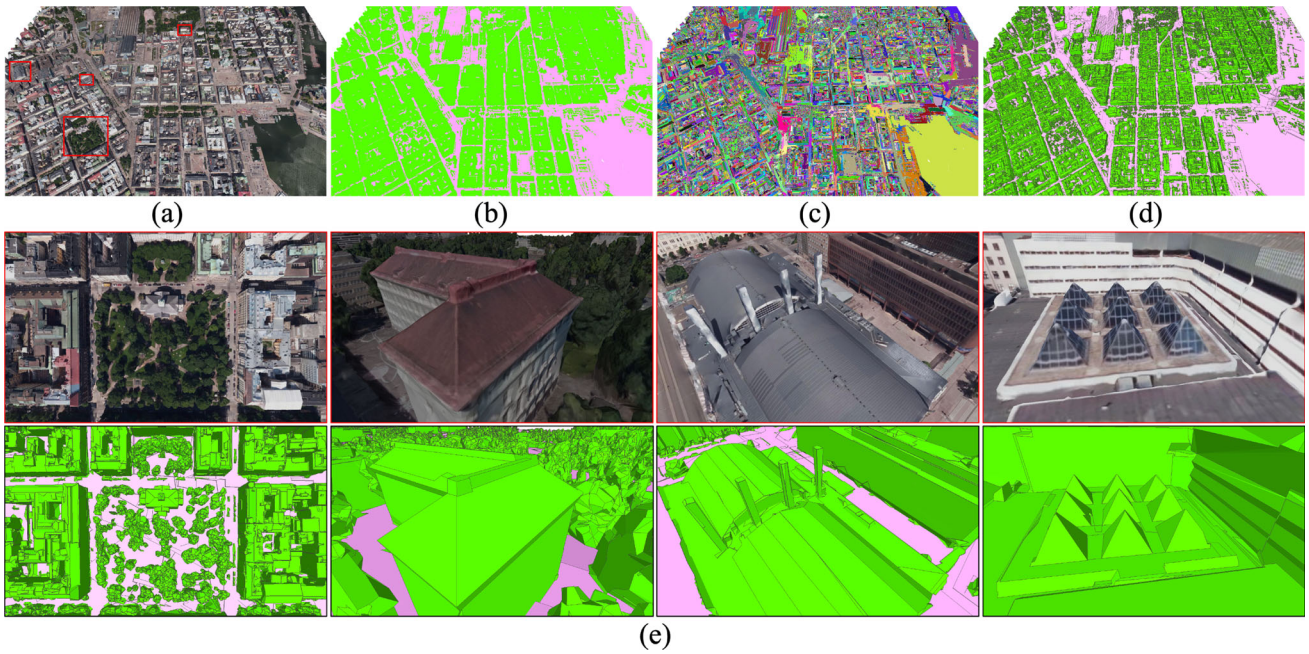

Fig. 18 **a** Point cloud of "SUM Benchmark". **b** Semantic segmentation. **c** Planar primitives. **d** Vectorized model. **e** Magnified regions of **a** and **d** to show the recovered geometric details in polygonal surface representation

we also provide quantitative comparison of our method to PolyFit with semantic segmentation, VecIM and KSR on all the other six indoor cases and ten outdoor cases in Table 3, including the evaluation of the number of used primitives, the peak memory consumption, the total execution time, and the geometric accuracy by MHE and RMSE. Note that VecIM cannot work successfully on outdoor scenes, and even fails on the two indoor cases "Auditorium" and "Museum" due to its piece-wise planar indoor hypothesis. As can be seen in the table, KSR produces lower accuracy for outdoor scenes than indoor ones due to the disappearance of the ground structure and some geometric details in its vectorized models. All the quantitative comparisons demonstrate that our approach can assemble more planar primitives to robustly reconstruct a polygonal model with better geometric accuracy than other methods in a time and memory efficient way, especially for large-scale outdoor cases.

Figures 19, 20 and 21 also gives a comparison of the time and memory consumption of our method to other SOTA works. For each method, we report the number of used primitives denoted as $|\mathcal{P}|$, the number of generated polygonal facets as $|\mathcal{F}|$, the execution time, and the peak memory consumption. It can be seen that our pipeline is the most efficient on time, memory and the number of primitives supported, which runs more than twice faster than PolyFit, PolyFit with semantic segmentation, and VecIM, and costs moderate memory with far more primitives used to generate polygonal facets with higher quality. PolyFit and VecIM cannot support so many primitives as ours to accomplish a successful vectorization due to the time and memory limitation. PolyFit with semantic segmentation is able to support more primitives at the cost of longer execution time. Table 4 gives the time statistics of our pipeline on all the steps, together with the peak memory consumption for cases "Church", "Office", "area4", "Village", "Taian Ancient Street" and "SUM Benchmark". Note that for "area4" and "SUM Benchmark", we skip the semantic segmentation and refinement steps and use the given semantic labeling of the point cloud as prior. From the statistics we can see that the time and memory efficiency of the vectorization process strongly depends on the size of the input point cloud and the number of primitives used for vectorized modeling. However, even the largest scale case "SUM Benchmark" with 30.4 millions of points and 493428 planar primitives can be successfully finished in 3.1 h by our method, for which other SOTA methods definitely will have both time and memory limitations.

7 Discussion

Our system is not designed to vectorize a non-watertight object well by the inside/outside labeling optimization, which is similar to SOTA works (Bauchet and Lafarge, 2020; Fang et al., 2021; Han et al., 2021a; Nan and Wonka, 2017). The non-watertight point cloud is usually caused by insufficient scanning of only its partial faces, such as the incomplete table shown in Fig. 15g. Such incomplete object might disappear or produce inaccurate structure in the final vectorized model.

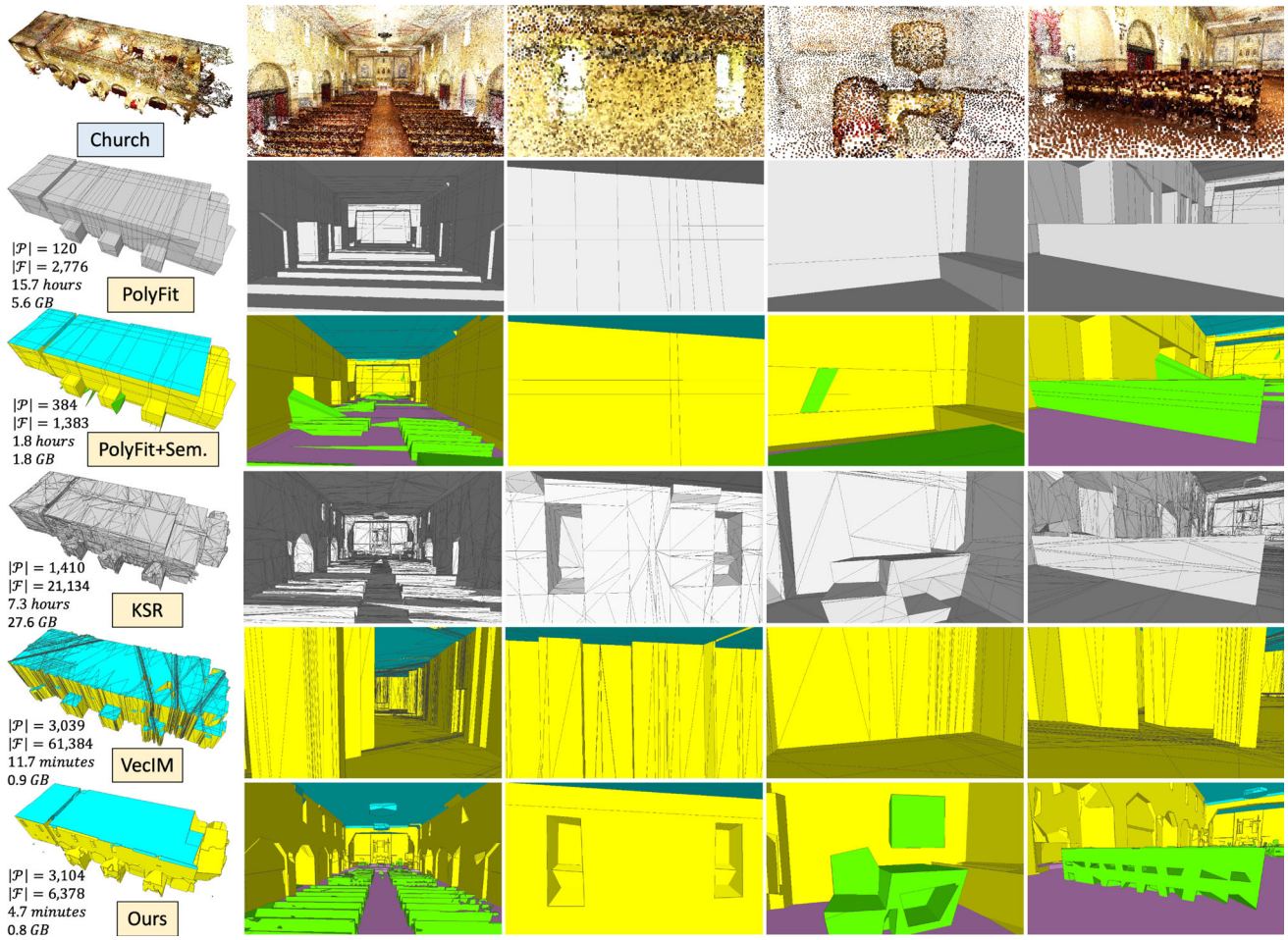

Fig. 19 Comparison of our pipeline with SOTA works PolyFit Nan and Wonka (2017), PolyFit with semantic segmentation, KSR Bauchet and Lafarge (2020) and VecIM Han et al. (2021a) on case "Church", with number of used primitives, time consumption, peak memory cost, and magnified local vectorized models given for each method

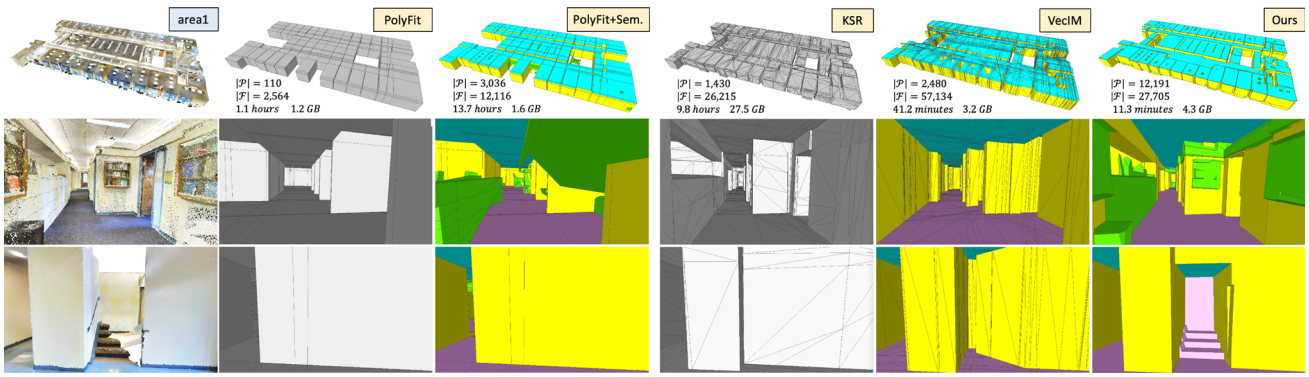

Fig. 20 Comparison of our method with PolyFit, PolyFit with semantic segmentation, KSR and VecIM on case "area1", with time and memory consumption, and magnified local vectorized model details given for each method

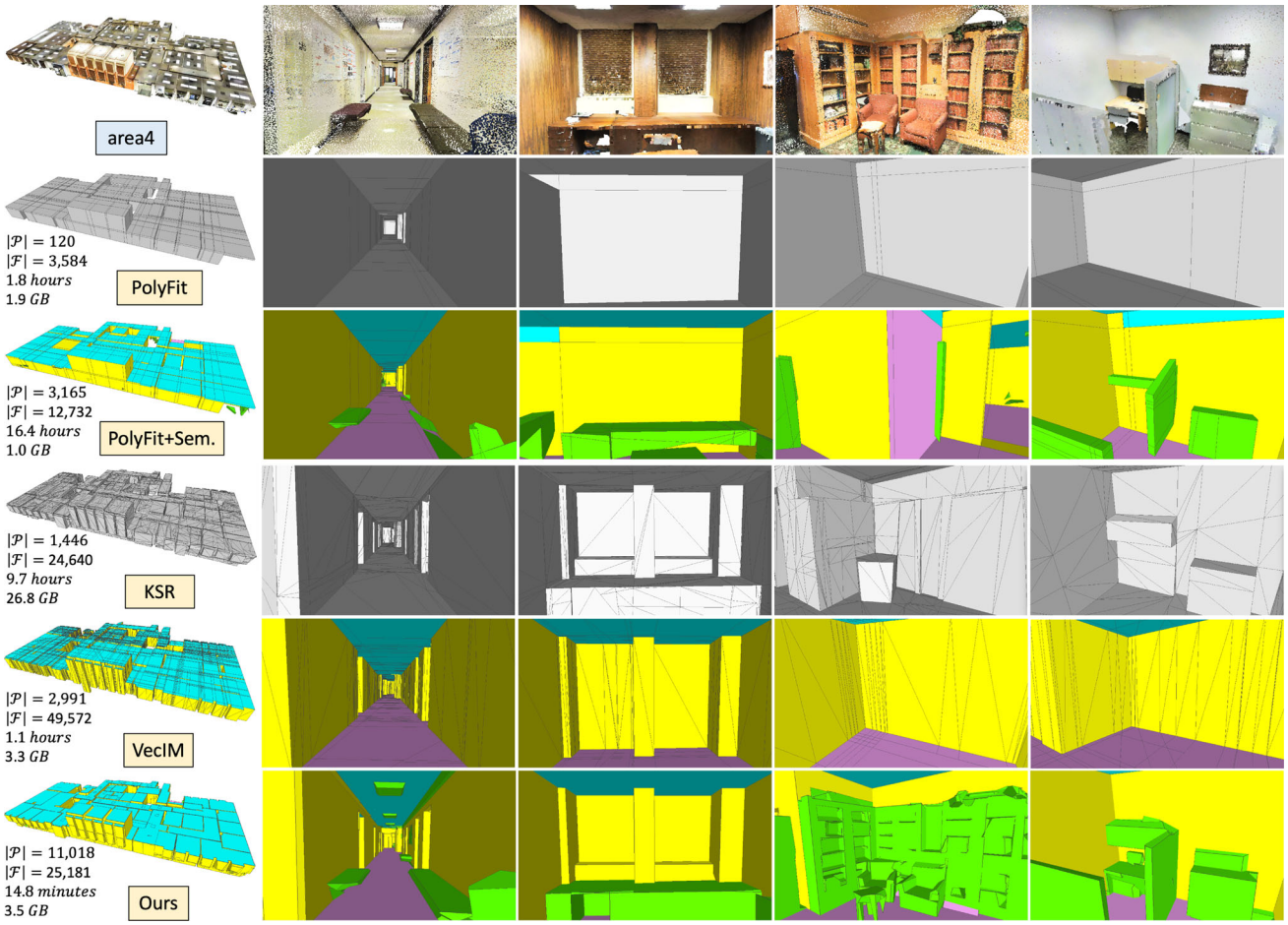

Fig. 21 Comparison of our method with PolyFit, PolyFit with semantic segmentation, KSR and VecIM on case "area4", with time and memory consumption, and magnified local vectorized model details given for each method

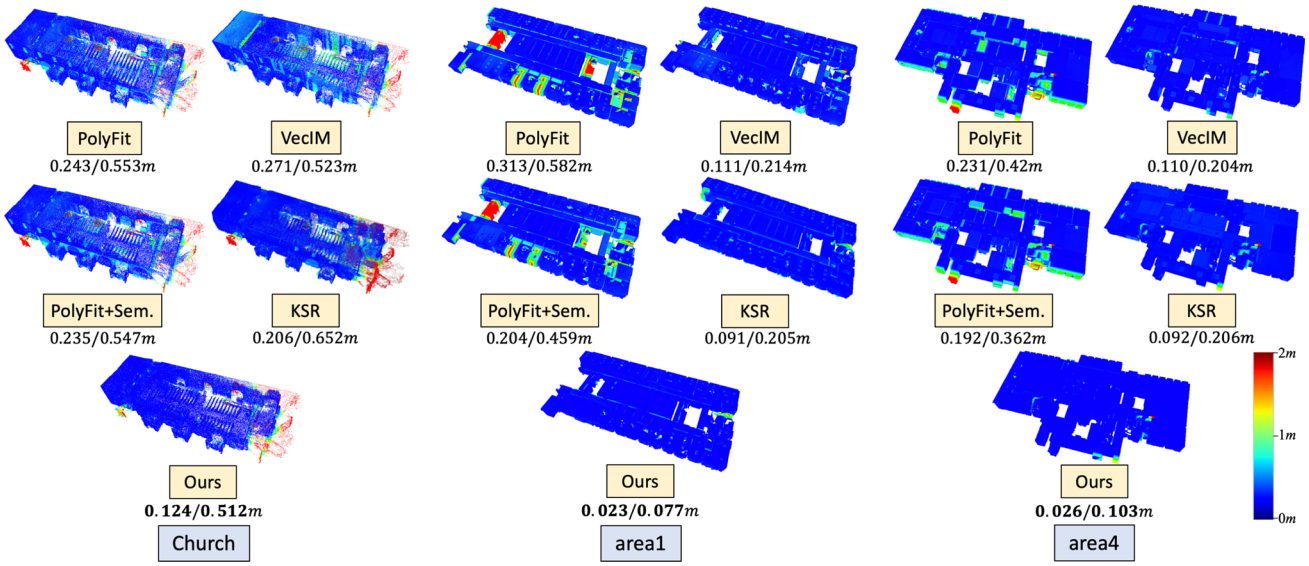

Fig. 22 Accuracy evaluation of the vectorized models of cases "Church", "area1" and "area4" by PolyFit, PolyFit with semantic segmentation, KSR, VecIM and our method, which are given by MHE and RMSE in meters

Table 3 Quantitative comparison of our method to PolyFit Nan and Wonka (2017) with semantic segmentation, VecIM Han et al. (2021a) and KSR Bauchet and Lafarge (2020), on the other six indoor cases and ten outdoor cases, with evaluation of the number of primitives denoted by $|\mathcal{P}|$, the peak memory (GB), the total time (minutes, without semantic segmentation and refinement time), and the geometric accuracy by MHE and RMSE (meters)

Methods	PolyFit + Sem.				VecIM				KSR				Ours			
Cases	$\|\mathcal{P}\|$	Mem	Time	Accuracy	$\|\mathcal{P}\|$	Mem	Time	Accuracy	$\|\mathcal{P}\|$	Mem	Time	Accuracy	$\|\mathcal{P}\|$	Mem	Time	Accuracy
Office (indoor)	90	2.2	320.4	0.41/0.89	662	1.9	129.03	0.4/0.66	1,420	26.3	443.7	0.20/0.55	4,861	6.5	13.9	**0.13/0.5**
Tea room	44	0.83	51.6	0.66/1.49	1,080	1.53	31.24	0.44/1.09	1,421	26.4	468.2	0.27/0.84	2,396	0.6	1.8	**0.05/0.23**
Auditorium	68	2.7	291.6	0.3/0.45	N/A	N/A	N/A	N/A	1,590	34.2	638.8	0.17/0.29	1,578	1.1	1.7	**0.09/0.2**
Ballroom	75	1.9	140.0	0.19/0.3	340	0.58	7.52	0.85/2.10	342	0.7	6.2	0.07/0.16	351	0.1	0.16	**0.07/0.15**
Meeting room	71	1.4	96.3	0.21/0.39	417	0.41	7.36	0.53/0.83	412	1.53	15.6	0.12/0.31	406	0.07	0.17	**0.09/0.2**
Museum	33	0.7	168.7	0.56/0.9	N/A	N/A	N/A	N/A	545	3.88	43.7	0.06/0.11	552	0.28	0.44	**0.06/0.09**
Barn (outdoor)	30	0.9	110.5	0.49/1.0	N/A	N/A	N/A	N/A	294	0.69	7.7	1.03/1.93	292	0.12	0.17	**0.04/0.06**
Light house	49	1.6	82.9	1.0/1.9	N/A	N/A	N/A	N/A	291	0.85	9.3	1.37/2.33	288	0.1	0.15	**0.2/0.62**
ZJU CCE	322	0.4	154.4	9.0/12.0	N/A	N/A	N/A	N/A	1,441	26.6	407.1	7.13/8.8	3,429	0.66	1.14	**0.47/1.37**
Century Park1	1,712	0.8	1199.8	2.9/4.1	N/A	N/A	N/A	N/A	1,401	23	413.4	9.12/14.88	11,029	0.74	3.27	**0.5/1.3**
Century Park2	2,198	0.5	271.1	5.2/7.1	N/A	N/A	N/A	N/A	1,448	21.6	302.2	11.03/17.29	14,490	1.55	5.41	**0.21/0.75**
Lingang	12,113	0.9	7887.4	25.4/41.6	N/A	N/A	N/A	N/A	1,440	27.9	371.3	165.31/226.16	57,421	2.6	18.7	**1.31/3.03**
World Expo	1,223	0.8	2414.3	87.2/104.6	N/A	N/A	N/A	N/A	1,453	31.9	479.5	102.86/141.03	56,446	4.9	18.34	**0.97/3.69**
Village	459	0.5	222.4	3.7/6.7	N/A	N/A	N/A	N/A	1,461	26.5	472.6	6.03/10.65	7,420	1.1	3	**0.25/0.92**
Taian	21,940	1.0	8665.4	3.0/4.3	N/A	N/A	N/A	N/A	1,441	27.8	467.7	9.15/25.37	101,131	9.1	56.7	**0.35/1.16**
SUM	34,201	1.1	5351.6	15.9/25.6	N/A	N/A	N/A	N/A	1,420	25.9	657.3	143.21/204.42	493,428	31.3	187.9	**0.13/0.36**

Table 4 The time statistics on all the steps of our pipeline (minutes), and the peak memory consumption (GB) for cases "Church", "Office", "area4", "Village", "Taian Ancient Street" and "SUM Benchmark"

Cases	Church	Office	area4	Village	Taian	SUM
#Points (millions)	0.9	6.5	10.6	2.5	20.7	30.4
#Main structure primitives	1,058	2,805	1,414	97	1,337	10,702
#Object primitives	2,046	2,056	9,604	7,323	99,794	482,726
#Objects	27	21	152	38	1,636	2,698
Semantic segmentation	1.3	36.7	N/A	2.4	60.9	N/A
Primitive abstraction	0.2	1.1	1.6	0.4	5.4	4.9
Primitive pre-expansion	0.2	1.9	3.9	0.2	2.4	4.6
Main structure vectorization	0.1	3.2	0.9	0.3	3.4	20.2
Segmentation refinement	3.8	13.5	N/A	5	38.3	N/A
Object vectorization	3.7	1.5	8.6	1.2	34.3	158.2
Total time (minutes)	9.8	64.1	14.8	10.4	155.9	187.9
Peak mem. (GB)	0.8	6.5	3.5	1.1	9.1	31.3

A 2D simulated example is shown in Fig. 13 for easy understanding. The incomplete table has only two planar primitives abstracted in Fig. 13a, including a top surface and a front side of its supporting column, with other faces missing, which causes this table to disappear by our inside/outside labeling as can be seen in Figs. 13c, d and 15g, since it is a non-watertight structure when combined with the main structure. However, if the table is completely scanned with all its faces composing a watertight point cloud, its six primitives can be fully abstracted and assembled with the main structure for a correct inside/outside labeling, so as to produce a complete vectorized model as illustrated in Fig. 13e.

Additionally, our vectorization results depends on the accuracy of semantic segmentation. For unknown scenes, severe segmentation errors will influence the final vectorized models, and even cannot be well corrected by the segmentation refinement step. Besides, the semantic segmentation step consumes a large proportion of the total time. How to reduce the influence of semantic segmentation while still keeping the time and memory efficiency of our large-scale vectorized modeling is a problem worth studying in future. Besides, for city-scale scenes far larger than "SUM Benchmark", the time and memory limitations might still become a problem for vectorized modeling. How to further partition these scenes into small blocks for parallel processing as a more powerful time and memory efficient city-scale vectorization strategy remains to be our future work.

8 Conclusion

A novel vectorization pipeline is proposed in this work for high-quality vectorized modeling of large-scale scenes with point clouds as input. The large-scale scenes are decomposed by 3D semantic segmentation into main structure and isolated objects, which are vectorized respectively to a global polygonal surface model for a time and memory efficient purpose. To better preserve geometric details in the generated polygonal model, we propose a novel BSP strategy for a light-weight space partition to efficiently assemble thousands of planar primitives, which is difficult for other SOTA vectorization methods. Moreover, we innovatively use the already vectorized main polygonal structures as prior to guide the vectorization of the isolated objects and further refine the 3D semantic segmentation, so as to reconstruct a high-quality vectorized model with complete geometry, consistent topology and correct semantic attributes.

Acknowledgements The authors wish to thank Liyang Zhou, Zhuang Zhang, Fei Jiao, Yi Su, Xinru Hou, Yue Li, Shaoqing Lu, Bing Yan and Yuzhou Liu for their kind helps in the experiments and the development of the proposed vectorized modeling system. This work was partially supported by NSF of China (No. 61932003).

Data Availability The datasets generated and analysed during the current study are not publicly available due to privacy protection, but are available from the corresponding author on reasonable request.

References

Arefi, H., Engels, J., Hahn, M., & Mayer, H. (2008). Levels of detail in 3D building reconstruction from LiDAR data. In *ISPRS conference*.

Arikan, M., Schwärzler, M., Flöry, S., et al. (2013). O-snap: Optimization-based snapping for modeling architecture. *ACM Transactions on Graphics, 32*(1), 1–15.

Armeni, I., Sax, S., Zamir, A. R., & Savarese, S. (2017). Joint 2D-3D-semantic data for indoor scene understanding. arXiv preprint arXiv:1702.01105

Bauchet, J. P., & Lafarge, F. (2019). City reconstruction from airborne LiDAR: A computational geometry approach. In *3D GeoInfo*.

Bauchet, J. P., & Lafarge, F. (2020). Kinetic shape reconstruction. *ACM Transactions on Graphics, 39*(5), 1–14.

Boulch, A., de La Gorce, M., & Marlet, R. (2014). Piecewise-planar 3D reconstruction with edge and corner regularization. In *Computer graphics forum*, pp. 55–64.

Bouzas, V., Ledoux, H., & Nan, L. (2020). Structure-aware building mesh polygonization. *ISPRS Journal of Photogrammetry and Remote Sensing, 167*, 432–442.

Boykov, Y., & Kolmogorov, V. (2004). An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 26*(9), 1124–1137.

Boykov, Y., Veksler, O., & Zabih, R. (2001). Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 23*(11), 1222–1239.

Chauve, A. L., Labatut, P., & Pons, J. P. (2010). Robust piecewise-planar 3D reconstruction and completion from large-scale unstructured point data. In *IEEE conference on computer vision and pattern recognition*, pp. 1261–1268.

Chen, J., & Chen, B. (2008). Architectural modeling from sparsely scanned range data. *International Journal of Computer Vision, 78*(2), 223–236.

Chen, L. C., Papandreou, G., Schroff, F., & Adam, H. (2017). Rethinking atrous convolution for semantic image segmentation. arXiv preprint arXiv:1706.05587

Chen, L. C., Zhu, Y., Papandreou, G., Schroff, F., & Adam, H. (2018). Encoder–decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 801–818.

Chen, Z., Tagliasacchi, A., & Zhang, H. (2020). BSP-Net: Generating compact meshes via binary space partitioning. In *IEEE/CVF conference on computer vision and pattern recognition*, pp. 45–54.

Choy, C., Gwak, J., & Savarese, S. (2019). 4D spatio-temporal convnets: Minkowski convolutional neural networks. In *IEEE/CVF conference on computer vision and pattern recognition*, pp. 3075–3084.

Cignoni, P., Rocchini, C., & Scopigno, R. (1998). Metro: Measuring error on simplified surfaces. In *Computer graphics forum*, Wiley Online Library, pp. 167–174.

Corsini, M., Cignoni, P., & Scopigno, R. (2012). Efficient and flexible sampling with blue noise properties of triangular meshes. *IEEE Transactions on Visualization and Computer Graphics, 18*(6), 914–924.

Duan, L., & Lafarge, F. (2016). Towards large-scale city reconstruction from satellites. In *European conference on computer vision*, pp. 89–104.

Fang, H., & Lafarge, F. (2020). Connect-and-slice: An hybrid approach for reconstructing 3D objects. In *IEEE/CVF conference on computer vision and pattern recognition*, pp. 13490–13498.

Fang, H., Pan, C., & Huang, H. (2021). Structure-aware indoor scene reconstruction via two levels of abstraction. *ISPRS Journal of Photogrammetry and Remote Sensing, 178*, 155–170.

Gao, W., Nan, L., Boom, B., et al. (2021). SUM: A benchmark dataset of semantic urban meshes. *ISPRS Journal of Photogrammetry and Remote Sensing, 179*, 108–120.

Graham, B., Engelcke, M., & Van Der Maaten, L. (2018). 3D semantic segmentation with submanifold sparse convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 9224–9232.

Han, J., Rong, M., Jiang, H., et al. (2021). Vectorized indoor surface reconstruction from 3D point cloud with multistep 2D optimization. *ISPRS Journal of Photogrammetry and Remote Sensing, 177*, 57–74.

Han, J., Zhu, L., Gao, X., et al. (2021). Urban scene LOD vectorized modeling from photogrammetry meshes. *IEEE Transactions on Image Processing, 30*, 7458–7471.

Hermans, A., Floros, G., & Leibe, B. (2014). Dense 3D semantic mapping of indoor scenes from RGB-D images. In *IEEE international conference on robotics and automation*, pp. 2631–2638.

Huang, J., Stoter, J., Peters, R., et al. (2022). City3D: Large-scale building reconstruction from airborne LiDAR point clouds. *Remote Sensing, 14*(9), 2254.

Kazhdan, M., Bolitho, M., & Hoppe, H. (2006). Poisson surface reconstruction. In *Eurographics symposium on geometry processing*.

Knapitsch, A., Park, J., Zhou, Q. Y., & Koltun, V. (2017). Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics (ToG), 36*(4), 1–13.

Kolmogorov, V., & Zabin, R. (2004). What energy functions can be minimized via graph cuts? *IEEE Transactions on Pattern Analysis and Machine Intelligence, 26*(2), 147–159.

Kundu, A., Yin, X., Fathi, A., Ross, D., Brewington, B., Funkhouser, T., & Pantofaru, C. (2020). Virtual multi-view fusion for 3D semantic segmentation. In *European conference on computer vision*, Springer, pp. 518–535.

Lafarge, F., & Mallet, C. (2012). Creating large-scale city models from 3D-point clouds: A robust approach with hybrid representation. *International Journal of Computer Vision, 99*(1), 69–85.

Li, M., Nan, L., Smith, N., et al. (2016). Reconstructing building mass models from UAV images. *Computers and Graphics, 54*, 84–93.

Li, M., Wonka, P., & Nan, L. (2016b). Manhattan-world urban reconstruction from point clouds. In *European conference on computer vision*, pp. 54–69.

Marshall, D., Lukacs, G., & Martin, R. (2001). Robust segmentation of primitives from range data in the presence of geometric degeneracy. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 23*(3), 304–314.

Monszpart, A., Mellado, N., Brostow, G. J., et al. (2015). RAPter: Rebuilding man-made scenes with regular arrangements of planes. *ACM Transactions on Graphics, 34*(4), 103–1.

Mura, C., Mattausch, O., & Pajarola, R. (2016). Piecewise-planar reconstruction of multi-room interiors with arbitrary wall arrangements. In *Computer graphics forum*, pp. 179–188.

Murali, T., & Funkhouser, T. A. (1997). Consistent solid and boundary representations from arbitrary polygonal data. In *Symposium on interactive 3D graphics*, pp. 155–ff.

Nan, L., & Wonka, P. (2017). PolyFit: Polygonal surface reconstruction from point clouds. In *IEEE international conference on computer vision*, pp. 2353–2361.

Ochmann, S., Vock, R., & Klein, R. (2019). Automatic reconstruction of fully volumetric 3D building models from oriented point clouds. *ISPRS Journal of Photogrammetry and Remote Sensing, 151*, 251–262.

Oesau, S., Lafarge, F., & Alliez, P. (2014). Indoor scene reconstruction using feature sensitive primitive extraction and graph-cut. *ISPRS Journal of Photogrammetry and Remote Sensing, 90*, 68–82.

Pham, Q. H., Hua, B. S., Nguyen, T., & Yeung, S. K. (2019). Real-time progressive 3D semantic segmentation for indoor scenes. In *IEEE winter conference on applications of computer vision*, pp. 1089–1098.

Poullis, C., & You, S. (2009). Automatic reconstruction of cities from remote sensor data. In *IEEE conference on computer vision and pattern recognition*, pp. 2775–2782.

Rabbani, T., Van Den Heuvel, F., & Vosselmann, G. (2006). Segmentation of point clouds using smoothness constraint. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences, 36*(5), 248–253.

Rouhani, M., Lafarge, F., & Alliez, P. (2017). Semantic segmentation of 3D textured meshes for urban scene analysis. *ISPRS Journal of Photogrammetry and Remote Sensing, 123*, 124–139.

Schindler, F., Wörstner, W., & Frahm, J. M. (2011). Classification and reconstruction of surfaces from point clouds of man-made objects. In *IEEE international conference on computer vision workshops*, pp. 257–263.

Schnabel, R., Wahl, R., & Klein, R. (2007). Efficient RANSAC for point-cloud shape detection. In *Computer graphics forum*, pp. 214–226.

Verdie, Y., Lafarge, F., & Alliez, P. (2015). LOD generation for urban scenes. *ACM Transactions on Graphics, 34*(3), 30.

Wang, J., Sun, K., Cheng, T., et al. (2020). Deep high-resolution representation learning for visual recognition. *IEEE Transactions on Pattern Analysis and Machine intelligence, 43*(10), 3349–3364.

Wang, R., Xie, L., & Chen, D. (2017). Modeling indoor spaces using decomposition and reconstruction of structural elements. *Photogrammetric Engineering and Remote Sensing, 83*(12), 827–841.

Wolf, D., Prankl, J., & Vincze, M. (2015). Fast semantic segmentation of 3D point clouds using a dense CRF with learned parameters. In *IEEE international conference on robotics and automation*, pp. 4867–4873.

Xie, L., Hu, H., Zhu, Q., et al. (2021). Combined rule-based and hypothesis-based method for building model reconstruction from photogrammetric point clouds. *Remote Sensing, 13*(6), 1107.

Yang, S., Huang, Y., & Scherer, S. (2017). Semantic 3D occupancy mapping through efficient high order crfs. In *IEEE/RSJ international conference on intelligent robots and systems*, pp. 590–597.

Yu, F., Chen, Z., Li, M., et al. (2022). CAPRI-Net: Learning compact CAD shapes with adaptive primitive assembly. In *IEEE/CVF conference on computer vision and pattern recognition*, pp. 11768–11778.

Yu, M., & Lafarge, F. (2022). Finding good configurations of planar primitives in unorganized point clouds. In *IEEE/CVF conference on computer vision and pattern recognition*, pp. 6367–6376.

Zeng, H., Wu, J., & Furukawa, Y. (2018). Neural procedural reconstruction for residential buildings. In *European conference on computer vision*, pp. 737–753.

Zhou, B., Zhao, H., Puig, X., et al. (2017). Scene parsing through ADE20K dataset. In *IEEE conference on computer vision and pattern recognition*, pp. 633–641.

Zhou, B., Zhao, H., Puig, X., et al. (2019). Semantic understanding of scenes through the ADE20K dataset. *International Journal of Computer Vision, 127*(3), 302–321.

Zhou, L., Zhang, Z., Jiang, H., et al. (2021). DP-MVS: Detail preserving multi-view surface reconstruction of large-scale scenes. *Remote Sensing, 13*(22), 4569.

Zhu, L., Shen, S., Gao, X., & Hu, Z. (2018). Large scale urban scene modeling from MVS meshes. In *European conference on computer vision*, pp. 614–629.

Zhu, L., Shen, S., Gao, X., et al. (2020). Urban scene vectorized modeling based on contour deformation. *ISPRS International Journal of Geo-Information, 9*(3), 162.