Deep Polycuboid Fitting for Compact 3D Representation of Indoor Scenes

Gahye Lee Hyejeong Yoon Jungeon Kim Seungyong Lee

POSTECH

{gahye0509, hjyoon02, jungeonkim, leesy}@postech.ac.kr

Abstract

This paper presents a novel framework for compactly representing a 3D indoor scene using a set of polycuboids through a deep learning-based fitting method. Indoor scenes mainly consist of man-made objects, such as furniture, which often exhibit rectilinear geometry. This property allows indoor scenes to be represented using combinations of polycuboids, providing a compact representation that benefits downstream applications like furniture rearrangement. Our framework takes a noisy point cloud as input and first detects six types of cuboid faces using a transformer network. Then, a graph neural network is used to validate the spatial relationships of the detected faces to form potential polycuboids. Finally, each polycuboid instance is reconstructed by forming a set of boxes based on the aggregated face labels. To train our networks, we introduce a synthetic dataset encompassing a diverse range of cuboid and polycuboid shapes that reflect the characteristics of indoor scenes. Our framework generalizes well to real-world indoor scene datasets, including Replica, ScanNet, and scenes captured with an iPhone. The versatility of our method is demonstrated through practical applications, such as virtual room tours and scene editing.

1. Introduction

Understanding 3D scene geometry is a fundamental challenge in computer vision. While point clouds have been widely utilized for representing 3D scene geometry, they primarily serve as a raw data representation that inherently lacks part-level and object-level structures. To address this limitation, shape abstraction approaches have been developed to model 3D geometries as compositions of simple primitives, such as planes [7,34], cuboids [5,36], and superquadrics [14,16]. These abstract representations provide a structured and interpretable description of scene geometry, which is useful for various applications, such as scene understanding [12, 24], robot navigation [22, 23], scene reconstruction [6, 31], and AR/MR [1, 9].

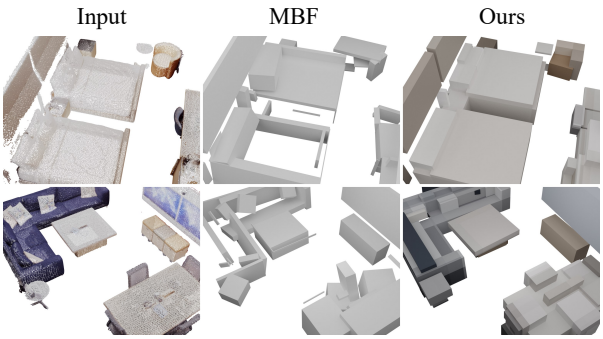

Figure 1. Comparison of shape abstraction results on Replica dataset. Compared to MBF [21], which generates many thin boxes covering only small portions of input points, our method detects high-quality polycuboids that comprehensively abstract the overall shapes of objects.

Indoor scenes consist of interior surfaces, such as walls, floors, and ceilings, as well as man-made objects including furniture and appliances. Given that these components often exhibit rectilinear shapes, cuboids could be a preferred representation for indoor scenes [27, 28, 34]. However, cuboids inherently struggle to represent concave structures, such as L-shaped sofas, and are insufficient to capture the diversity of object shapes in indoor scenes. In contrast, polycuboids, which combine multiple cuboids, can provide a more expressive and accurate representation.

In this paper, we explore the use of polycuboids to compactly represent the underlying geometries of 3D indoor scenes. Specifically, we introduce a data-driven approach that learns to fit a set of polycuboids to a noisy point cloud. To the best of our knowledge, leveraging polycuboids for 3D representation has not been explored previously.

Fitting polycuboid instances to a noisy point cloud is challenging. Constructing reliable priors that generalize across diverse polycuboid configurations and object shapes is inherently ill-posed. Partial or noisy scan data further complicates the robust detection of polycuboids. Moreover,

existing cuboid fitting methods cannot be directly extended to construct polycuboids, as they typically rely on heuristic plane fitting algorithms and tend to overly decompose scenes (Fig. 1). In that case, additional filtering or merging steps are required to refine small or thin cuboids and obtain a structured and meaningful polycuboid representation.

In this paper, we explore the geometric priors of a cuboid, the fundamental unit of a polycuboid, for polycuboid detection, We define six cuboid face types and twelve spatial relationships, encoded using discrete and finite labels. By extending these priors, we adapt cuboid-based relationships to comprehensively represent diverse structures of polycuboids (Fig. 3). Instead of a top-down approach, we adopt a bottom-up approach detecting individual polycuboid faces and then aggregating them into polycuboid instances.

Given a point cloud, we detect individual polycuboid faces using a transformer network. A graph neural network is then used to infer spatial relationships of the detected faces to form potential polycuboid instances. Finally, each polycuboid instance is reconstructed by assembling cuboids based on the label information of aggregated faces. Our method enables robust detection of plausible polycuboids from a noisy or even incomplete point cloud (Fig. 1).

To train our transformer and graph neural networks, we introduce a synthetic polycuboid dataset encompassing a diverse range of cuboid and polycuboid shapes that reflect the characteristics of indoor scenes. Leveraging this synthetic dataset, our framework can faithfully handle the diverse shapes in indoor scenes. We evaluate our method on three types of real scan dataset, ScanNet [2], Replica [26], and scenes captured using an iPhone. We also explore practical applications of our framework, such as virtual room tour and scene editing, combined with an off-the-shelf texture mapping technique.

Our main contributions can be summarized as follows;

- We introduce the polycuboid representation for representing individual components of an indoor scene in a lightweight and controllable manner.
- We present a novel deep learning-based framework for polycuboid fitting of a noisy input point cloud, formulating the problem as cuboid component labeling and their aggregation into polycuboid instances.
- We demonstrate the effectiveness of our framework on real-world indoor scene datasets, and show its practical applications such as virtual room tours and scene editing.

2. Related Work

Cuboid fitting There have been studies that leverage cuboid shapes to represent indoor scenes or man-made objects [5, 7, 15, 21, 24]. Some methods [7, 15, 21] detect orthogonal planar segments to establish a set of cuboids. He et al. [5] incorporates corners for cuboid fitting that are generated by intersecting lines and planes extracted from an image. Tianjia et al. [24] proposed a method that first fits cuboids to separated point clusters and then optimizes the cuboid parameters. However, most previous methods rely on heuristic plane fitting and generate undesirable cuboids like planar boxes, requiring post-processing to filter out invalid cuboids. Our approach fully explores important geometric constraints within a cuboid, such as adjacency relationships between cuboid faces. This enables robust generation of plausible polycuboids from a noisy point cloud without post-processing.

Primitive-based object abstraction Parsing 3D scanned objects into simple primitives, such as planes, cylinders, spheres, superquadrics, and cuboids, has been studied in the context of shape abstraction and reconstruction. PolyFit [16] introduced a framework for reconstructing polygonal surfaces from noisy points by selecting outer faces from a set of hypothesized faces. A few works [11, 25, 33] have learned to fit various primitive types to a point cloud for representing an object. Recently, superquadrics have regained attention due to their ability to achieve low fitting errors [13, 17, 32]. A cuboid has long been emphasized as a geometric primitive due to its significant benefit in representing manufactured objects [27, 28, 34]. Despite their impressive results, these methods assume a clean input geometry and are limited to object-scale data.

Primitive-based scene abstraction A scene typically contains various man-made objects that often exhibit piecewise smooth or planar surfaces. Given this characteristic, several studies have employed simple primitives, such as lines and planes, to approximate the underlying geometry of the target scene. Zhou et al. [35] reconstruct a 3D wireframe from a single RGB image by leveraging 2D geometric cues such as junctions, lines, and vanishing points based on the Manhattan assumption. Cuboids have also been widely adopted as primitives for scene-scale abstraction [7, 15, 21, 24]. Especially, similar to ours, MonteBoxFinder [21] detects cuboids in a scene-scale point cloud. However, previous cuboid fitting methods are constrained to representing only the convex regions, making it challenging to capture concave objects in a scene. In contrast, our polycuboid fitting method effectively represents both convex and concave objects, enhancing the overall representation capability.

3. Deep Polycuboid Fitting

Our framework converts a point cloud of an indoor scene into a set of polycuboids for abstract scene representation in two key steps: polycuboid component detection (Section 3.1) and polycuboid reconstruction (Section 3.2). In

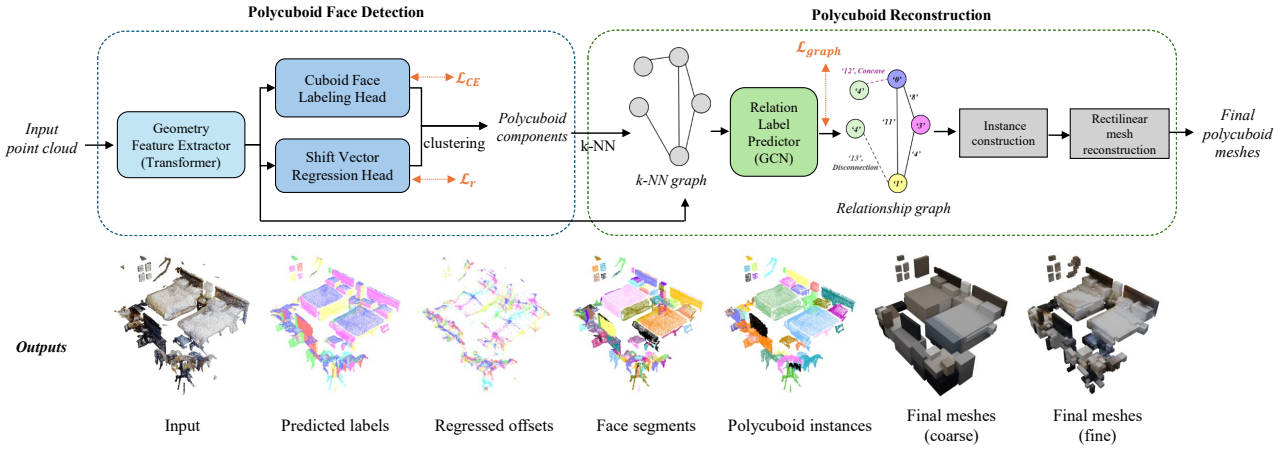

Figure 2. Overall process of our polycuboid fitting framework. Initially, a transformer network estimates point-wise face labels and offsets that are used to detect face segments. Detected face segments are then aggregated into polycuboid instances using spatial relations predicted by a graph convolutional network. Finally, each polycuboid instance is reconstructed as a polycuboid mesh in a coarse or fine detail level.

the polycuboid component detection stage, we first infer a cuboid face label and a shift vector for each 3D point in the input point cloud. Using this information, we cluster points to detect polycuboid face components via a density-based algorithm. In the polycuboid reconstruction stage, we infer fourteen predefined spatial relationships within a polycuboid for the detected faces by using a graph convolutional network (GCN). Based on the inferred spatial relationships, connected faces are aggregated into polycuboid instances. Finally, each polycuboid is reconstructed as a set of cuboids using the label information from the aggregated faces. This process is illustrated in Fig. 2.

3.1. Polycuboid Component Detection

We adopt a bottom-up approach to detect polycuboid face components from an input point cloud. Our method infers a cuboid face label and a center shift vector for each 3D point, and this information is used to cluster points, where each cluster represents a face component.

Point-wise cuboid face labeling For cuboid face labeling, we formulate a multiclass classification problem where each class represents one of the six cuboid faces. As illustrated in Fig. 3, any face of a polycuboid can be assigned one of six cuboid face labels (corresponding to the six directional axes: $\pm x, \pm y, \pm z$). To implement the labeling task, we employ the Stratified Transformer [10] as our backbone network, which is especially designed for scene geometry understanding of large-scale point clouds.

Given an input point cloud, the backbone network extracts point-wise features $\mathbf{F}$. These features are then passed through a two-layer MLP head to predict probability vectors $\mathbf{s} \in \mathbb{R}^6$ for cuboid face labels. The final label for each point is determined by selecting the label with the highest probability. The backbone and head networks are trained using a cross-entropy loss defined as

$$\mathcal{L}_{CE} = \frac{1}{N} \sum CE(\mathbf{s}_i, \mathbf{s}_i^*), \tag{1}$$

where N is the number of points and $\mathbf{s}^*$ is the ground-truth label.

Point-wise shift vector regression Assigning cuboid face labels alone is not sufficient for accurately detecting polycuboid faces. 3D points with the same cuboid face label but belonging to different cuboids should not be grouped into the same face. Rather than relying on a heuristic distance threshold, we employ a neural network to regress shift vectors, enabling robust detection of distinct faces.

Shift vector regression has been widely used for instance-level association in object detection and segmentation [8, 20, 29]. In our case, the shift vector $\mathbf{o} \in \mathbb{R}^3$ represents the displacement of each point to the center of its corresponding polycuboid face. Similar to face labeling, shift vector regression utilizes a shared backbone network and a two-layer MLP head. The network is trained using l_2 regression loss defined as

$$\mathcal{L}_r = \frac{1}{N} \sum \|\mathbf{o}_i - \mathbf{o}_i^*\|_2, \tag{2}$$

where $\mathbf{o}^*$ is the ground-truth shift vector.

Polycuboid face detection Using the estimated face labels and shift vectors, we detect polycuboid faces by grouping spatially adjacent points that share the same labels. For each set of points sharing the same cuboid face label, points

Figure 3. Face labels and their spatial relationships in a cuboid, and their extension to a polycuboid. (a) A cuboid with color-coded face and edge components. (b) Spatial relationship graph of cuboid components, whose nodes and edges correspond to faces and edges of a cuboid, respectively. (c) A polycuboid with an additional adjacency type of concave edge marked in black.

are translated along their shift vectors, aligning them near the centers of the corresponding polycuboid faces. The shifted points are then clustered using the DBSCAN algorithm [4] to identify individual polycuboid faces. Small polycuboid faces below a predefined size threshold (5cm in our implementation) are filtered out to remove noise.

3.2. Polycuboid Reconstruction

We reconstruct polycuboid instances by aggregating detected faces and identifying cuboids that are compatible with the face configuration. We first infer spatial relationships among the detected faces through a Graph Convolutional Network (GCN) [30] to group them into individual polycuboid instances. Each polycuboid instance is then reconstructed as a rectilinear mesh by assembling cuboids based on the grouped faces.

Polycuboid instance construction To infer the spatial relationships among the detected faces, we first analyze these relationships within a polycuboid. Since a cuboid is the building block of a polycuboid, the relationships can be derived from those of a cuboid. A cuboid consists of six faces, each maintaining a fixed adjacency relationship with its neighboring faces. For example, in Fig. 3a, the red face is always adjacent to the front side of the purple face. Likewise, a cuboid has twelve distinct spatial relationships among faces, which correspond to twelve cuboid edges. These relationships can be represented as a graph (Fig. 3b).

To represent spatial relationships among faces within a polycuboid, we incorporate two additional adjacency types. *Concave* adjacency refers to two faces that meet at an inward angle, forming a non-convex structure (Fig. 3c). *Disconnected* adjacency refers to two faces that are spatially separated and do not touch each other. With these two additional types, the spatial relationship between any pair of detected faces can be categorized into one of fourteen types.

We infer the spatial relationships of detected faces using a graph representation. In the graph, each node corresponds to a detected face, and edges are connected to the k-nearest neighbor (k-NN) faces based on Euclidean distances. Given this graph as the input, we use a GCN to infer the final face label for each node and the adjacency type for each edge. Note that the face labels determined in the previous polycuboid face detection step may be erroneous due to noise, partial scans, and rotational ambiguity. These errors can be corrected by gathering information from neighboring faces through a GCN.

For each node in the graph, we store 256 points randomly sampled from 3D points belonging to the corresponding detected face. We also concatenate the features F to the sampled points, which have been computed by the backbone network during the face labeling step. Similarly, for each edge in the graph, we store the concatenation of points and their features from the two connected nodes. We then use two separate PointNet [19] encoders to extract features for nodes and edges, respectively. We use a similar approach to [30] to train and infer a GCN using the graph.

Our GCN predicts the probability vectors for six possible face labels for node and fourteen possible adjacency types for edge, denoted by $\mathbf{n} \in \mathbb{R}^6$ and $\mathbf{e} \in \mathbb{R}^{14}$, respectively. The PointNets and GCN are trained using a combination of two cross-entropy losses defined as

$$\mathcal{L}_{graph} = \frac{1}{N_n}\sum CE(\mathbf{n}_i, \mathbf{n}_i^*) + \frac{1}{N_e}\sum CE(\mathbf{e}_i, \mathbf{e}_i^*), \quad (3)$$

where N_n is the number of nodes and N_e is the number of edges, and $\mathbf{n}^*$ and $\mathbf{e}^*$ are the ground-truth labels of nodes and edges, respectively. The trained GCN can robustly infer the final face labels and their spatial relationships to form polycuboid instances.

Using the inference results of the GCN, we group detected faces into polycuboid instances. For each edge in the graph, we verify whether its adjacency type and the face labels of the two connected nodes conform to the predefined cuboid label configuration in Fig. 3b. We then extract subgraphs that remain after removing edges with invalid configurations, obtaining valid polycuboid instances.

Rectilinear mesh reconstruction Finally, we reconstruct a compact polycuboid mesh for each polycuboid instance. As a polycuboid instance may be slightly rotated relative to the global frame, we first align its local frame with the global frame before applying our reconstruction process. After reconstruction, we transform the mesh back to its original orientation.

To generate a polycuboid mesh, we fit a bounding box to the polycuboid instance and construct a 3D non-uniform grid by slicing the bounding box with planes fitted to the detected faces (Fig. 4ab). A subset of 3D boxes in the grid is then selected to determine the shape of the polycuboid instance. We perform the box selection by checking the label configuration of detected faces surrounding each box. Only

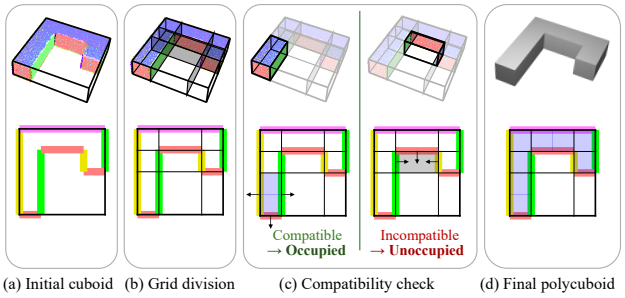

Figure 4. Illustration of our polycuboid reconstruction process. From a 3D non-uniform grid fitted to a polycuboid instance, we select valid 3D boxes that are inside the polycuboid based on the inferred face labels.

the boxes with compatible configurations are classified as inside the polycuboid instance.

For checking the compatibility of face configuration, we introduce a heuristic score function. Boxes with positive scores are classified as parts of the polycuboid instance, while those with negative scores are excluded. The final polycuboid mesh is reconstructed from the outer surface of the selected 3D boxes. Details on the score function are provided in the supplementary material.

While polycuboid meshes provide a compact abstraction of the scene, more details may be required for certain applications. Finer geometric details can be obtained by subdividing the 3D non-uniform grid and reapplying the reconstruction process. We set the subdivision interval to 0.1m in our experiments. For visualization, each box is colored using the average color of the surrounding points of its faces.

3.3. Indoor Scene Polycuboid Reconstruction

Indoor scenes typically consist of objects placed in the room layout, forming a nested structure. To avoid potential ambiguity among object- and room-level structures, we divide the scene into two parts: the room layout and the objects within it. To separate the room layout from the input point cloud, we employ an off-the-shelf point cloud segmentation method [10]. We then apply our pipeline to the room layout and object parts separately (Fig. 5).

4. Experiments

4.1. Datasets

We build a synthetic *Polycuboid* dataset for training our networks and conduct qualitative and quantitative evaluations on three distinct real world datasets: *ScanNet [2], Replica [26]*, and *iPhoneCapture.*

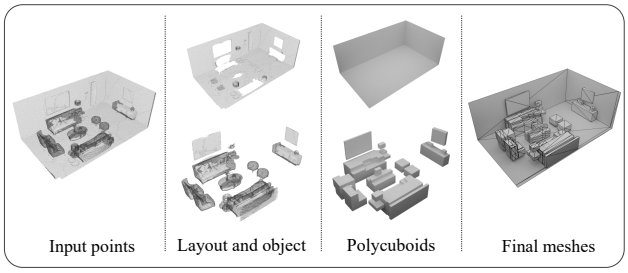

Figure 5. Illustration of our indoor scene reconstruction pipeline. The input point cloud is first separated into layout and object points. These two point sets are independently reconstructed into polycuboid meshes using our polycuboid fitting method, which are then merged to produce the final output.

Polycuboid dataset We introduce a synthetic polycuboid dataset designed to replicate the 3D structures of indoor scenes and real-world scanning conditions. The dataset contains 1,800 training and 200 validation scenes, each composed of a mixture of polycuboid and cuboid meshes. To provide diverse contextual information, we generate two types of configurations; 1) Polycuboids with random arrangements, 2) Polycuboids with contextual arrangements, derived from bounding boxes provided in the ScanNet dataset. We then sample 3D points at 1cm intervals on each face of the meshes. To mitigate domain gaps caused by a scanning process, inspired by DoDA [3], we simulate scanning noise by adding Gaussian noise and creating holes by randomly removing faces and their corresponding points. For the face component detection task, each point is assigned a label corresponding to one of six face labels and an offset vector pointing to its respective face center. For the graph prediction task, we reconstruct individual faces from the sampled points and build a k-nearest neighbors (k-NN) graph based on these faces. In this graph, each node is assigned one of six labels, while each edge is given one of fourteen possible labels. Details on data generation are provided in the supplementary material.

ScanNet ScanNet dataset [2] consists of over 1,500 3D indoor scene scans, providing annotated bounding boxes for objects in the scene. We use the validation set for our evaluation. Since points corresponding to layout categories are too incomplete for reconstruction, we only use points classified as object categories for evaluation.

Replica Replica dataset [26] features highly photo-realistic digitization of real indoor scenes, represented with clean and dense meshes. Among 18 different scenes, excluding FRL apartment scenes, we use a total of 9 scenes consisting of 5 office rooms, a hotel room, and 3 rooms of apartments for evaluation.

iPhoneCapture We demonstrate the effectiveness of our polycuboid reconstruction on scenes captured by a mobile phone. We used Polycam [18], a commercial 3D scanning app, to capture two common real-world room types, a living room and a bedroom, with an iPhone Pro 12. We present two demos to showcase the potential of our method in practical applications, such as virtual room tour and room editing (Figs. 7 and 10).

4.2. Implementation Details

We train the transformer [10] and GCN [30] using the Polycuboid dataset. The transformer is trained with the AdamW [14] optimizer at a learning rate of 0.006, a weight decay of 0.05, and a batch size of 4 over 100 epochs, using a voxel size of 0.02m. The GCN is trained with the Adam optimizer at an initial learning rate of 0.0001, adjusted at epochs 30 and 60, with a weight decay of 0.5. GCN training runs for 80 epochs with a batch size of 8, using a k-NN graph with k set to 5. All training is conducted on a single NVIDIA GeForce RTX 4090 GPU.

4.3. Results and analysis

Baseline Our framework is designed for scene-scale 3D abstraction of an input point cloud, in contrast to most existing studies that focus on object-scale abstraction. We compare our method with MonteBoxFinder (MBF) [21] as a scene-scale abstraction work, which employs cuboid fitting for input point clouds representing indoor scenes.

Reconstruction fidelity We evaluate the reconstruction fidelity of our method by utilizing the Chamfer Distance (CD) metric which quantifies the discrepancy between the input point cloud and the reconstructed polycuboids. Even though real world data includes severe holes (e.g., walls and furniture with large hidden parts) and moderate noise, our method effectively handles these challenges and produce polycuboids faithfully recovering the underlying geometry of input point clouds, as illustrated in Figs. 6 and 7. In contrast, MBF, which relies on a heuristic cuboid fitting approach, often produces thin planar boxes that fail to accurately represent objects, as shown in Fig. 1.

Our approach achieves better quantitative results for non-layout objects, while MBF slightly outperforms in metrics for layout parts. This is likely due to structured noise around transparent materials, like windows, in layout categories, causing our method to generate slightly larger polycuboids. Despite this, our method produces visually plausible results for layout points, as shown in Figs. 5 and 7.

Regarding the level of polycuboid details, we provide abstraction results at a higher resolution in the last two columns of Fig. 6. This resolution can be adjusted by the user for different purposes, depending on the application.

Table 1. Quantitative evaluation of reconstruction fidelity. We measure Chamfer Distance (CD) on ScanNet and Replica datasets.

	ScanNet	Replica	
	non-layout	non-layout	layout
MBF[21]	0.066	0.078	**0.047**
Ours	**0.040**	**0.044**	0.081

Table 2. Quantitative evaluation of object-level representation. We measure precision and recall scores on ScanNet validation set.

	IoU 25		IoU 50	
Method	Recall	Precision	Recall	Precision
MBF[21]	0.41	0.11	0.11	0.03
Ours	**0.64**	**0.15**	**0.43**	**0.10**

More qualitative and quantitative results can be found in the supplementary material.

Object-level representation We assess how effectively our fitted polycuboids approximate scenes at the object level in the ScanNet dataset. We compute precision and recall scores on detected polycuboids using ground truth bounding boxes at two Intersection over Union (IoU) thresholds, 25 and 50, respectively. As shown in Table. 2, our method achieves higher scores than MBF, demonstrating that our approach more accurately decomposes the scene, with each polycuboid more likely representing a single object. As MBF tends to over-decompose the scene, its recall score drops significantly as the IoU threshold increases. In contrast, our method demonstrates robust performance, maintaining higher recall scores even at a stricter IoU threshold.

Potential applications When combined with off-the-shelf texture mapping features provided by Blender, polycuboid geometry allows a visually plausible appearance of the scene to be obtained with a small number of polygons. To showcase the potential of our method in practical scenarios, we present two applications.

- **Virtual room tour** We can navigate the reconstructed rooms for seamless virtual exploration experience, as shown in Fig. 7, where polycuboid-based reconstruction provides complete and lightweight geometry. With texture mapping, realistic rendering can be achieved, closely resembling the original scanned data.
- **Room editing** Each polycuboid represents a scene object such as a sofa or a TV shelf. The scene can be easily edited by manipulating the polycuboids, which would be useful for interior design and AR/VR applications.

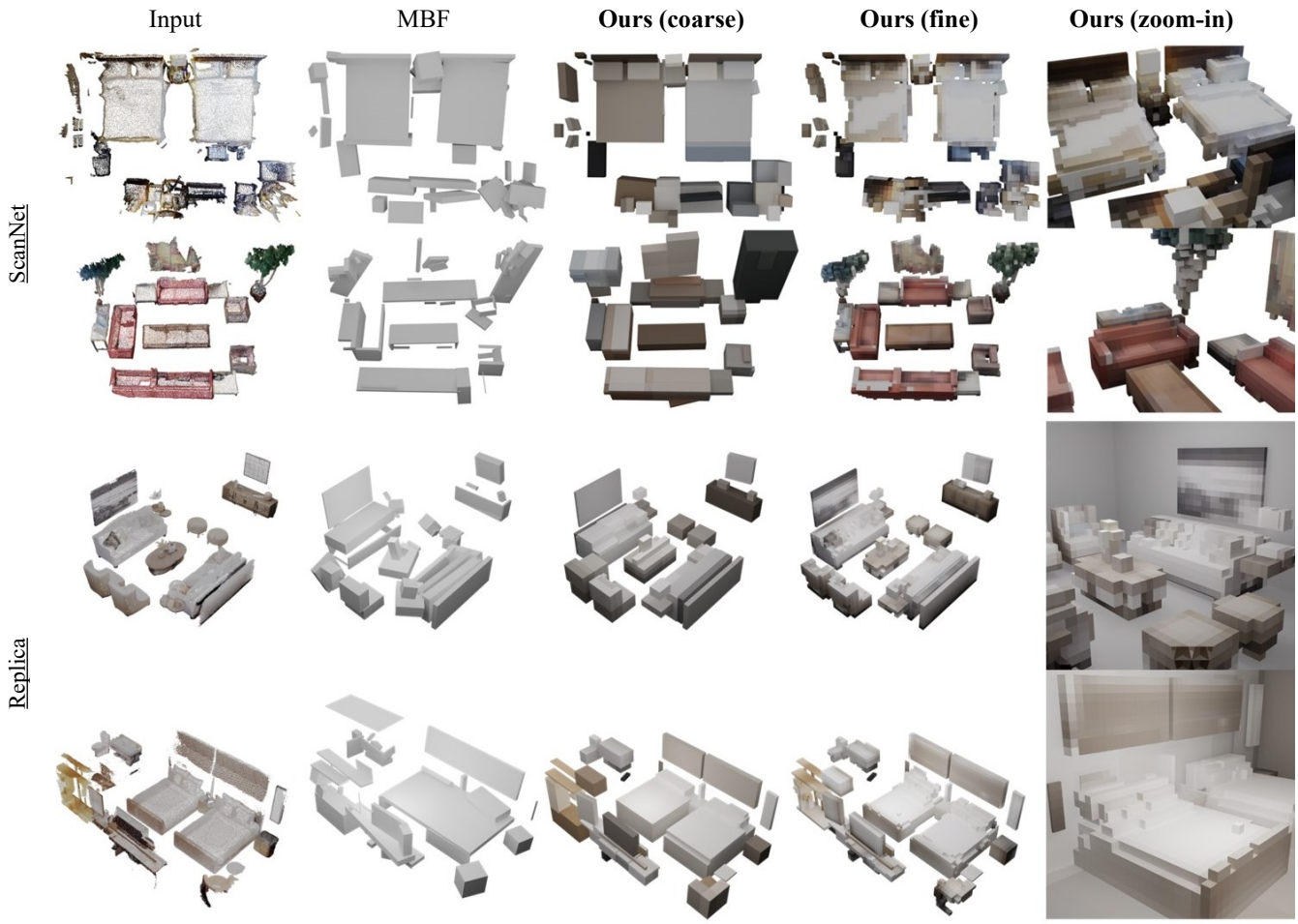

Figure 6. Qualitative comparison on ScanNet and Replica datasets. Columns from left to right show the input point clouds, results from MBF[21], results from our method in coarse and fine detail levels, and zoomed-in views of the fine level results.

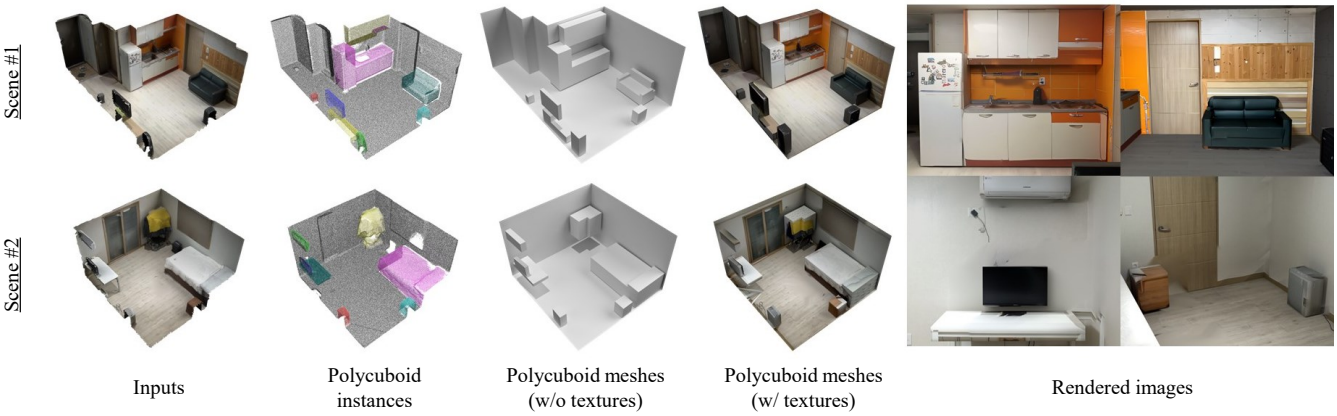

Figure 7. Qualitative evaluations on room data captured using an iPhone. Columns from left to right show the input point clouds, point sets covered by detected polycuboid instances encoded with different colors, reconstructed polycuboids without and with textures, and rendering results.

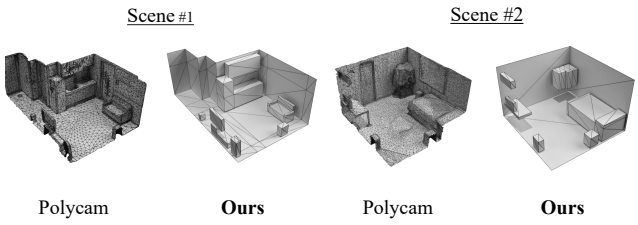

Figure 8. Comparison of reconstruction efficiency with a commercial application [18] on data captured using an iPhone. Our method provides the benefit of a lightweight representation.

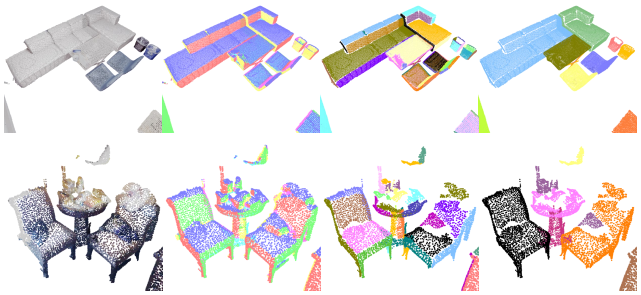

Figure 9. Visualization of the intermediate result of each module in our framework on ScanNet data. The first column represents the input point clouds. In the remaining three columns, color encodings visualize estimated face labels, detected faces, and detected polycuboid instances, respectively.

4.4. Discussion

Generalization ability on arbitrary shapes Despite being trained solely on the synthetic Polycuboid dataset, our framework generalizes effectively to real-world datasets with various shapes and scanning conditions. We visualize the intermediate result from each module, including face detection and polycuboid instance construction. As shown in Fig. 9, the transformer network consistently assigns cuboid face labels to input points, demonstrating robustness in detecting object faces even with complex and noisy geometries. In addition, our GCN provides concrete inference results needed for robust reconstruction of polycuboid instances.

Evaluation for lightweight representation In Fig. 8, for scene #1, Polycam [18] generates 15,011 vertices and 25,688 faces, while our polycuboid meshes have just 182 vertices and 328 faces. Similarly, for scene #2, Polycam [18] produces 27,802 vertices and 48,196 faces, compared to our 94 vertices and 156 faces. Our method uses over 90 times fewer vertices and 60 times fewer faces, producing exceptionally lightweight yet complete meshes, compared to the already simplified outputs of Polycam [18]. This reduction in mesh topology would significantly improve computational efficiency for downstream applications such as VR and AR.

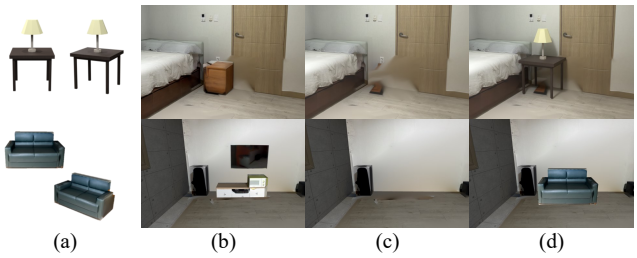

Figure 10. Room editing application for object removal and rearrangement. (a) New objects to be placed into the scene, bedside table and sofa. (b) Original scene reconstructed with polycuboids. (c) Scene after object removal. (d) Scene with newly added object.

Limitation Due to the rectilinear nature of polycuboids, geometric discrepancies arise when abstracting objects with highly curved surfaces, which can potentially degrade visual fidelity. A promising future work is to detect regions with high geometric errors and refine the polycuboids in those regions to enhance representation accuracy. Also, our framework currently relies solely on geometric cues for polycuboid fitting. Incorporating semantic cues could further improve object-level abstraction and enhance reconstruction quality.

5. Conclusion

In this paper, we proposed a framework for fitting polycuboids to indoor scenes based on geometric properties and configurations, without requiring high-level understanding of object structures. In our framework, cuboid face labels and their spatial relationships, inferred from a transformer and a GCN, facilitate the reconstruction of polycuboid instances, enabling a lightweight and editable representation of indoor scenes. We demonstrate that our framework generalizes well to real-world datasets and has strong potential for practical applications. We believe that leveraging primitive geometries, such as polycuboids, can benefit a wide range of scene representation and editing tasks.

Acknowledgements

This work was supported by NRF grants (RS-2023-00280400, RS-2024-00451947) and IITP grants (ICT Research Center, RS-2024-00437866; AI Innovation Hub, RS-2021-II212068; AI Graduate School Program, RS-2019-II191906) funded by Korea government (MSIT), and POSCO DX Company Ltd. (23DO1109).

References

[1] Denis Chekhlov, Andrew P Gee, Andrew Calway, and Walterio Mayol-Cuevas. Ninja on a plane: Automatic discovery of physical planes for augmented reality using visual slam. In *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 153–156. IEEE, 2007. 1

[2] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3D reconstructions of indoor scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5828–5839, 2017. 2, 5

[3] Runyu Ding, Jihan Yang, Li Jiang, and Xiaojuan Qi. Doda: Data-oriented sim-to-real domain adaptation for 3D semantic segmentation. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXVII*, pages 284–303. Springer, 2022. 5

[4] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, page 226–231. AAAI Press, 1996. 4

[5] Zhenbang He, Yunhai Wang, and Zhanglin Cheng. Manhattan-world urban building reconstruction by fitting cubes. In *Computer Graphics Forum*, pages 289–300. Wiley Online Library, 2021. 2

[6] Jingwei Huang, Angela Dai, Leonidas J Guibas, and Matthias Nießner. 3Dlite: towards commodity 3D scanning for content creation. *ACM Trans. Graph.*, 36(6):203–1, 2017. 1

[7] Hao Jiang and Jianxiong Xiao. A linear approach to matching cuboids in rgbd images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2171–2178, 2013. 2

[8] Li Jiang, Hengshuang Zhao, Shaoshuai Shi, Shu Liu, Chi-Wing Fu, and Jiaya Jia. Pointgroup: Dual-set point grouping for 3D instance segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and Pattern recognition*, pages 4867–4876, 2020. 3

[9] Yunho Kim and Honguk Woo. Integrating a deep learning-based plane detector in mobile ar systems for improvement of plane detection. In *Proceedings of the 8th International Conference on Computing and Artificial Intelligence*, pages 597–602, 2022. 1

[10] Xin Lai, Jianhui Liu, Li Jiang, Liwei Wang, Hengshuang Zhao, Shu Liu, Xiaojuan Qi, and Jiaya Jia. Stratified transformer for 3D point cloud segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8500–8509, 2022. 3, 5, 6

[11] Lingxiao Li, Minhyuk Sung, Anastasia Dubrovina, Li Yi, and Leonidas J Guibas. Supervised fitting of geometric primitives to 3D point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2652–2660, 2019. 2

[12] Dahua Lin, Sanja Fidler, and Raquel Urtasun. Holistic scene understanding for 3D object detection with rgbd cameras. In *Proceedings of the IEEE international conference on computer vision*, pages 1417–1424, 2013. 1

[13] Weixiao Liu, Yuwei Wu, Sipu Ruan, and Gregory S Chirikjian. Robust and accurate superquadric recovery: A probabilistic approach. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2676–2685, 2022. 2

[14] I Loshchilov. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 6

[15] Masashi Mishima, Hideaki Uchiyama, Diego Thomas, Rin-ichiro Taniguchi, Rafael Roberto, Veronica Teichrieb, et al. RGB-D slam based incremental cuboid modeling. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, pages 0–0, 2018. 2

[16] Liangliang Nan and Peter Wonka. Polyfit: Polygonal surface reconstruction from point clouds. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2353–2361, 2017. 2

[17] Despoina Paschalidou, Ali Osman Ulusoy, and Andreas Geiger. Superquadrics revisited: Learning 3D shape parsing beyond cuboids. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10344–10353, 2019. 2

[18] Polycam. Polycam, 2022. 6, 8

[19] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3D classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017. 4

[20] Charles R Qi, Or Litany, Kaiming He, and Leonidas J Guibas. Deep hough voting for 3D object detection in point clouds. In *proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9277–9286, 2019. 3

[21] Michaël Ramamonjisoa, Sinisa Stekovic, and Vincent Lepetit. Monteboxfinder: Detecting and filtering primitives to fit a noisy point cloud. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXVIII*, pages 161–177. Springer, 2022. 1, 2, 6, 7

[22] Sipu Ruan, Karen L Poblete, Hongtao Wu, Qianli Ma, and Gregory S Chirikjian. Efficient path planning in narrow passages for robots with ellipsoidal components. *IEEE Transactions on Robotics*, 39(1):110–127, 2022. 1

[23] Sipu Ruan, Xiaoli Wang, and Gregory S Chirikjian. Collision detection for unions of convex bodies with smooth boundaries using closed-form contact space parameterization. *IEEE Robotics and Automation Letters*, 7(4):9485–9492, 2022. 1

[24] Tianjia Shao, Aron Monszpart, Youyi Zheng, Bongjin Koo, Weiwei Xu, Kun Zhou, and Niloy J Mitra. Imagining the unseen: Stability-based cuboid arrangements for scene understanding. *ACM Transactions on Graphics*, 33(6), 2014. 1, 2

[25] Gopal Sharma, Difan Liu, Subhransu Maji, Evangelos Kalogerakis, Siddhartha Chaudhuri, and Radomír Měch. Parsenet: A parametric surface fitting network for 3D point clouds. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VII 16*, pages 261–276. Springer, 2020. 2

[26] Julian Straub, Thomas Whelan, Lingni Ma, Yufan Chen, Erik Wijmans, Simon Green, Jakob J. Engel, Raul Mur-Artal, Carl Ren, Shobhit Verma, Anton Clarkson, Mingfei Yan, Brian Budge, Yajie Yan, Xiaqing Pan, June Yon, Yuyang Zou, Kimberly Leon, Nigel Carter, Jesus Briales, Tyler Gillingham, Elias Mueggler, Luis Pesqueira, Manolis Savva, Dhruv Batra, Hauke M. Strasdat, Renzo De Nardi, Michael Goesele, Steven Lovegrove, and Richard Newcombe. The Replica dataset: A digital replica of indoor spaces. *arXiv preprint arXiv:1906.05797*, 2019. 2, 5

[27] Chun-Yu Sun, Qian-Fang Zou, Xin Tong, and Yang Liu. Learning adaptive hierarchical cuboid abstractions of 3D shape collections. *ACM Transactions on Graphics (TOG)*, 38(6):1–13, 2019. 1, 2

[28] Shubham Tulsiani, Hao Su, Leonidas J Guibas, Alexei A Efros, and Jitendra Malik. Learning shape abstractions by assembling volumetric primitives. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2635–2643, 2017. 1, 2

[29] Thang Vu, Kookhoi Kim, Tung M. Luu, Thanh Nguyen, and Chang D. Yoo. Softgroup for 3D instance segmentation on point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2708–2717, 2022. 3

[30] Johanna Wald, Helisa Dhamo, Nassir Navab, and Federico Tombari. Learning 3D semantic scene graphs from 3D indoor reconstructions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3961–3970, 2020. 4, 6

[31] Chao Wang and Xiaohu Guo. Efficient plane-based optimization of geometry and texture for indoor rgb-d reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 49–53, 2019. 1

[32] Yuwei Wu, Weixiao Liu, Sipu Ruan, and Gregory S Chirikjian. Primitive-based shape abstraction via nonparametric bayesian inference. In *European Conference on Computer Vision*, pages 479–495. Springer, 2022. 2

[33] Siming Yan, Zhenpei Yang, Chongyang Ma, Haibin Huang, Etienne Vouga, and Qixing Huang. Hpnet: Deep primitive segmentation using hybrid representations. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2753–2762, 2021. 2

[34] Kaizhi Yang and Xuejin Chen. Unsupervised learning for cuboid shape abstraction via joint segmentation from point clouds. *ACM Transactions on Graphics (TOG)*, 40(4):1–11, 2021. 1, 2

[35] Yichao Zhou, Haozhi Qi, Yuexiang Zhai, Qi Sun, Zhili Chen, Li-Yi Wei, and Yi Ma. Learning to reconstruct 3D manhattan wireframes from a single image. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7698–7707, 2019. 2