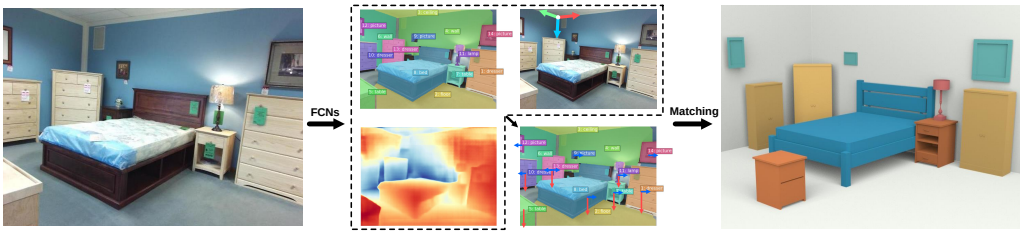
Semantic Modeling of Indoor Scenes with Support Inference from a Single Photograph

Yinyu Nie[1], Jian Chang*[1], Ehtzaz Chaudhry[1], Shihui Guo[2], Andi Smart[3], and Jianjun Zhang[1]

[1]*National Centre for Computer Animation, Bournemouth University, U.K,*
[2]*School of Software, Xiamen University, P.R.China*
[3]*Centre for Innovation and Service Research, University of Exeter Business School, U.K.*
**Corresponding Email: JChang@bournemouth.ac.uk*



(a) (b) (c)

Figure 1: Semantic modeling of an indoor scene. With a single indoor photograph (a), three parallel fully convolutional networks are adopted to extract instance masks, a depth map and a layout edge map for support inference (b). Using the support relationships as constraints, a whole semantic indoor scene (c) can be automatically reconstructed.

Abstract

We present an automatic approach for semantic modeling of indoor scenes based on a single photograph, instead of relying on RGB-D cameras. Without using hand-crafted features, we guide indoor scene modeling with feature maps extracted by Fully Convolutional Networks (FCNs). Three parallel FCNs are adopted to generate object instance masks, a depth map and an edge map of the room layout. Based on these high-level features, support relationships between indoor objects can be efficiently inferred in a data-driven manner. Constrained by the support context, a global-to-local model matching strategy is followed to retrieve the whole indoor scene. We demonstrate that the proposed method can efficiently retrieve indoor objects including situations where objects are badly occluded. This approach enables efficient semantic-based scene editing.

Keywords: semantic modelling, indoor scene reconstruction, fully convolutional network, support inference

1 Introduction

By digitizing real-world surroundings into virtual environments, scene modeling techniques enable people to quickly access a realistic 3D representation of their living world. This is of particular importance in practitioner environments where there is a desire to digitize physical artefacts while adhering to strict limits on capital expenditure. This includes applications in a multitude of cultural heritage sites, for example, historic buildings and stately homes. The modeling approach presented in this paper forms a preliminary experimentation on a €7.8m EU funded project (VISTA AR) to enhance visitor experiences using digital innovation. This preliminary experimentation addresses the complexity of 3D structures, the cluttered surroundings and the complicated spatial interrelationships between indoor objects. Modeling these complicated indoor scenes in a semantic sense is still a challenging problem.

With LiDAR technology, active cameras have played an irreplaceable role in 3D acquisition and

modeling [1]. Bunches of RGB-D image datasets [2, 3, 4] have emerged for contemporary 3D reconstruction. These datasets have enabled the Fully Convolutional Network (FCN) to be an innovative tool in scene understanding [5]. Unlike traditional feature extractors, FCN inherits the feature learning property from convolutional networks and offers a fast end-to-end solution for general computer vision problems, e.g. object detection [6], instance segmentation [7] and depth estimation [8].

In this paper, without using RGB-D cameras and hand-designed features, three parallel FCNs are applied to a single photograph to accomplish semantic modeling (see Figure 1). We generate object instance masks [7], a depth map [8] and an edge map of the room layout [9] respectively. Furthermore, support inference is provided to offer support context between neighboring objects (e.g. lamp on a nightstand and picture on a wall). With learning a priori from the indoor scene dataset SceneNN [2], we present an efficient approach to infer support context in a hierarchical structure and to guide our model matching step. In summary, the main contributions of this work are:

1. We propose a novel approach for indoor scene modeling based entirely on FCNs.
2. We provide a data-driven support inference approach to achieve hierarchical modeling, and have demonstrated that this approach shows great effectiveness in modeling badly occluded objects.

2 Related work

Geometric modeling (e.g. SLAM techniques) focuses on recovering the 3D geometry of surroundings. Semantic modeling, however, is used to retrieve both geometry and semantics of the whole scene [1]. There are two stages in semantic modeling: semantic segmentation; object modeling. Semantic segmentation separates a scene to labeled objects, and the object modeling is to recover their 3D content.

2.1 Semantic segmentation

For indoor scene segmentation, traditional methods usually learn a Conditional Random Field (CRF) or a Markov Random Field (MRF) model from a training database to optimize object labels by minimizing a customized energy function [10, 11]. With the wide availability of RGB-D cameras, the depth information is used to provide additional geometric clues. To learn features for object segmentation, Gupta et al. [12] proposed a geocentric embedding method, for RGB-D images, to encode geometric clues in a scene. Lai et al. [13] proposed a sparse coding method to learn features from RGB-D frames, and built an MRF model to label 3D points. Furthermore, Yang et al. [14] provided a view suggestion method to segment indoor point cloud in an interactive manner. Qi et al. proposed a network architecture, PointNet++ [15], which directly acts on point cloud to output semantic labels.

However, consumer-level RGB-D cameras have noisy depth maps, especially for absorptive or reflective surfaces [1]. Convolutional networks provide opportunities to extend this work and have shown an overwhelming performance in world-renowned competitions. In our work, with an FCN, only a single RGB image is required to segment indoor objects.

2.2 Objects modeling

Modeling methods from segmented objects can be divided into two categories: with depth information; without depth information (including images or sketches [16]). Using an RGB-D image, Guo et al. [17] rendered a composed scene from an RGB-D image by region matching. Shen et al. [18] provided a part assembly method to search part primitives to assemble a whole model. For modeling without depth information, most methods extract low-level features [19, 20] or CNN features [21] from images for geometry reasoning to support object modeling. Zhang et al. [19] adopted Canny Edge Detector to filter out line segments for reasoning the room layout and used cuboid detection to guide the object modeling. Inspired by insights that normal maps and edge information can convey geometric clues, Liu et al. [20] represented objects by a normal-based graph descriptor to build a similar model in the database. Again, extending this work, as 3D information can generally provide more geometric details, we estimate the depth map from a single RGB image using an FCN to guide the support inference and model matching.

3 Overview

The pipeline of our algorithm is presented in Figure 2. With only one indoor photo, our goal is to retrieve a 3D scene with informative semantic context. We produce object masks [7], depth maps [8], and room layout edge maps [9] using three FCNs with different architectures to guide object modeling. In object segmentation, a novel FCN architecture [7] is adopted for training on the NYU v2 [3] dataset, so that we can segment 40 common categories at the instance level. Combining the depth map with instance masks, the segmented point cloud of the scene can be calculated given

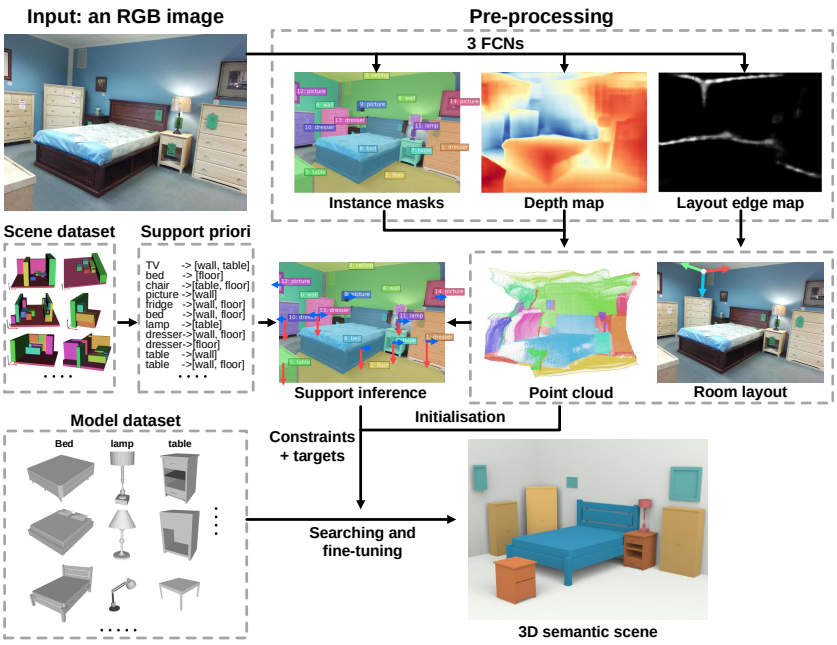


Figure 2: Pipeline of our method

the camera parameters. Meanwhile, room layout is estimated from the layout edge map to provide a baseline for the subsequent support inference.

In support inference, we decide support between objects a priori. Parsing the SceneNN [2] dataset, and combining it with the point cloud and room layout allows the support relationships between objects to be inferred. We build the support context as a hierarchical structure. Beginning with the layout frame (floors and walls), every piece is modeled on the basis of its supporting objects (e.g. if a lamp is supported by a table, the table should be built first). In our experiments, this kind of hierarchical constraint ensures a robust modeling result.

In object modeling, we build each object with a search-to-match strategy using a model database (e.g. Google 3D warehouse). The segmented point cloud is used to estimate an initial position and size for each object. We set the orientation angle, the translation vector and 3D scales of each model as optimization variables. The Intersection over Union (IoU) ratio between the model's perspective projection area and its mask is used as the maximization target. With two optimization steps (global searching and local matching), the whole scene is built following the derivation of the support hierarchy.

4 Pre-processing

In this section, we discuss the methods involved in object segmentation, depth estimation, and room layout estimation.

4.1 Object segmentation

We adopt the FCN architecture proposed by Li et al. [7] to segment a scene into instance-level objects. It offers an end-to-end solution with a great performance in instance segmentation. To use it for indoor scenes, the NYU v2 dataset is utilized. This offers 1449 indoor images with 40 fully segmented labels at the instance level. We use the official training/test split to evaluate the network. The mAP^r score [22] reaches 29.95% and 19.13% at IoU threshold of 0.5 and 0.7 respectively. We conduct training on the whole dataset (1449 images) to improve its performance. Figure 3b shows the segmentation result on an image (Figure 3a) from the dataset SUN-RGBD [4].

The FCN presents promising instance masks with most objects detected, except zig-zagged areas on the mask margin. Therefore in the prediction phase, we append a post-processing layer at the end of the FCN to refine the edges. The Grabcut method [23] is adopted using the FCN masks as probable foreground and the other areas as probable background. The refined result is shown in Figure 3c.

4.2 Depth estimation

For depth estimation, we adopt the network proposed by Laina [8]. It also has an FCN architecture based on residual learning. Without any post-processing, only a small amount of training data is required. As this model contains fewer parameters, it runs quickly in forward propaga-

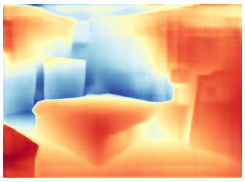
(a) Original image

(b) Initial result from FCN

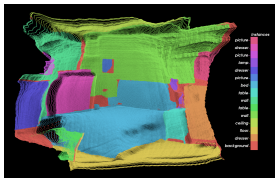
(c) Refined result from Grab-cut.

Figure 3: Object instance segmentation on an indoor image.

tion. Since the model is trained on the benchmark dataset NYU v2 where Microsoft Kinect is used, we adopt the technical parameters of Kinect [24] to retrieve the point cloud (see Figure 4). Figure 4b presents the segmented point cloud using the object masks. This clearly illustrate that the depth map is noisy especially for the margin area of the image. Therefore, support inference is considered to compensate for the geometric information.



(a) Depth map



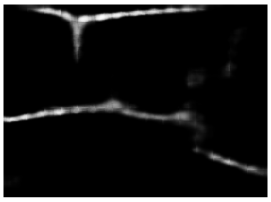
(b) Point cloud

Figure 4: Point cloud retrieval

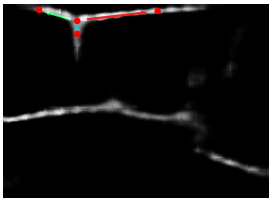
4.3 Layout estimation

Layout estimation is intended to provide a unified reference coordinate system for the further support inference and object modeling. Unlike the general room layout estimation [25] where a 3D parametric box is used to estimate the room layout, only a corner of the box is required to construct the reference system. In this part, we extract the edge map of the room layout following the work by Mallya et al. [9] (see Figure 5a), where structured edge detection forests and an FCN are used to provide a probability map of layout edges. Their experiments present robust results in occluded cases.

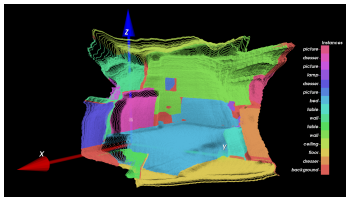
From the edge map, we adopt the RANSAC algorithm to search for a robust room corner (see Figure 5b and a detailed description in Appendix (B)). With the room corner and the point cloud, extrinsic parameters of the camera can be estimated by fitting the corner with an orthogonal system. We transform the point cloud into the new reference system, then align its x-y plane to the floor (the lowest plane) and its z-axis upwards (see Figure 5c). By the layout estimation, the ceiling, two walls and the floor can be determined. Therefore in the segmentation step, we do not require floors, ceilings and walls to be accurately segmented.



(a) Edge map of room layout



(b) Identified room corner

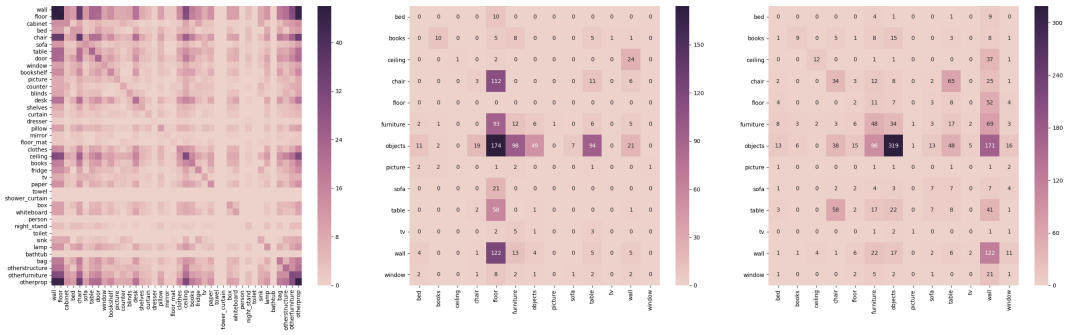


(c) Aligned point cloud

Figure 5: Room layout estimation

5 Support inference

The layout information and the segmented point cloud above are used for support inference between object instances. Three support types are defined: 1. support from below; 2. support from behind; 3. support from the top. It should be noted that we generally only consider the first two support types as they are able to explain most scenarios. We first build basic support rules a priori at the object category level from the SceneNN [2] dataset, which contains 50 sophisticated scenes with the same semantic labels defined in NYU v2. The object co-occurrence map of the dataset is illustrated in Figure 6a, where the color intensity indicates the frequency of two co-occurred objects. To infer support relationships at a general category level, we merge the 40 object categories with the class mapping provided with NYU v2 [3] resulting in 13 general categories. It is based on our observation that if object A is supported by object B, it could also be supported by others with a similar semantic label to object B (e.g. lamps can be supported by both desks and tables). The frequency



(a) Objects co-occurrence map (b) Cases with support from below (c) Cases with support from behind

Figure 6: Support priori from SceneNN dataset

of two co-occurred instances that have some support relationship (support type 1 or 2) are counted. The results are illustrated in Figure 6b and Figure 6c, where the block color represents the number of cases when object i (in row) is supported by object j (in column) from below or behind. These two matrices inform the support relationships a priori.

For each instance in a scene, we query the support matrices to recommend other instances which could have a support relationship with. Taking the dresser (see No.10 instance in Figure 3c) as an example, as the Figure 7 shows, it belongs to the furniture category. From querying the support matrices, the furniture category is likely to be supported from below (by floors, furniture, etc.) and from behind (by walls, furniture, etc.). A subsequent search is undertaken to identify additional instances that belonging to these categories. The point cloud is subsequently used to verify whether they are indeed close to each other from below/behind and to exclude wrong instances when these are identified. Using the prior information can not only improve searching efficiency, it also avoids judgment mistakes that could occur when only using the noisy point cloud. To handle sophisticated cases, we usually use the priori to recommend all potential categories in querying. The inferred support context behind Figure 3 can be built as a hierarchical structure (see Figure 8).

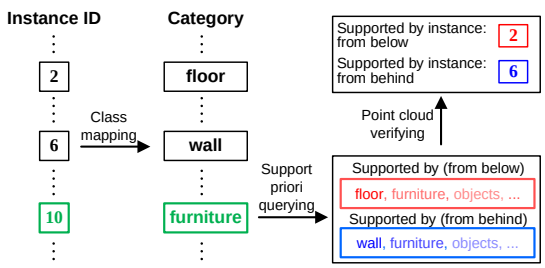


Figure 7: Searching instances with a support relationship.

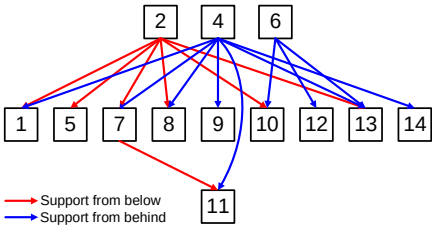


Figure 8: Support hierarchy

6 Object modeling

A global-to-local approach is followed in this section to both search and fine-tune models in the database for semantic modeling. Our optimization variables include the size, position and orientation of models and to match them with object masks to retrieve an indoor scene. Models with the most similar shape are firstly identified by the global searching step, and secondly fine-tuned. Before matching, models in the database should be categorized by labels and pre-processed with z-axis upwards, x-axis front-toward with their center to the origin. We build this database with Google 3D Warehouse.

6.1 Matching problem building

As the point cloud is noisy, it is difficult to use this information to estimate the object orientation. However, this data provides insights regarding the position and height clues. Therefore, we utilize the point cloud to initialize the model position and scales (see Figure 9 where height size is initialized by the point cloud). As the segmented mask provides edge and contour clues, we use it as the optimization target to obtain the object orientation, and subsequently refine the position and 3D scales.

In this routine, the floor and walls are firstly built by the layout estimation. We denote the point cloud of the target object by $\mathbf{P}_i$, its segmented image mask by $Mask_i$. $\mathbf{M}_i$, $\mathbf{M}_i{}^{\mathrm{H}}$ and $\mathbf{M}_i{}^{\mathrm{L}}$ re-

spectively represents the 3D model for matching, the supporting model behind and the one below. $i = 1, 2, ..., n$, n is the number of object instances in the scene. The 3D model scales $\mathbf{S}$, the spatial position $\mathbf{p}$ and the orientation angle θ around z-axis are set as optimization variables. With the camera parameters estimated in the layout estimation and point cloud estimation, the operator for projecting the 3D model onto the image plane can be calculated and we denote it by $\texttt{Proj}(*)$. Then we build the matching problem as to minimise

$$\begin{aligned}
&\max_{\theta,\mathbf{S},\mathbf{p}} \texttt{IoU}\{\texttt{Proj}[\mathbf{R}(\theta)\cdot\mathbf{S}\cdot\mathbf{M}_i+\mathbf{p}], Mask_i\},\\
&\mathbf{R}=\begin{bmatrix}\cos(\theta) & -\sin(\theta) & 0\\ \sin(\theta) & \cos(\theta) & 0\\ 0 & 0 & 1\end{bmatrix},\\
&\mathbf{S}=\begin{bmatrix}s_1 & 0 & 0\\ 0 & s_2 & 0\\ 0 & 0 & s_3\end{bmatrix}, \mathbf{p}=\begin{bmatrix}p_1\\ p_2\\ p_3\end{bmatrix},
\end{aligned} \tag{1}$$

where IoU score is used to measure optimization performance. We aim at that, on the image plane, the projection area of the optimal model can match with the corresponding mask. Besides, there are three types of constraints: 1. from supporting objects below; 2. from supporting objects behind; 3. from point cloud data.

1. From supporting objects below
   This type of constraints is to ensure the matched model placed on the upper surface of its supporting model (see Equation (2)).

$$\begin{aligned}
&\text{mean}[\mathbf{R}(\theta)\cdot\mathbf{S}\cdot\mathbf{M}_i+\mathbf{p}]_{x,y} >= \min[{\mathbf{M}_i}^{\text{L}}]_{x,y}\\
&\text{mean}[\mathbf{R}(\theta)\cdot\mathbf{S}\cdot\mathbf{M}_i+\mathbf{p}]_{x,y} <= \max[{\mathbf{M}_i}^{\text{L}}]_{x,y}\\
&\min[\mathbf{R}(\theta)\cdot\mathbf{S}\cdot\mathbf{M}_i+\mathbf{p}]_{z|x,y} = \max[{\mathbf{M}_i}^{\text{L}}]_{z|x,y}
\end{aligned} \tag{2}$$

2. From supporting objects behind
   The orientation of the target object generally has high relevance with its supporting object behind, and they should be attached close. Here we denote the orientation angle of its supporting model ${\mathbf{M}_i}^{\text{H}}$ by ${\theta_i}^{\text{H}}$. The constraints are written as:

$$\begin{aligned}
&\theta \in \{{\theta_i}^{\text{H}} + k\cdot\pi/4 | k = 0, 1, \ldots, 7\}\\
&\texttt{dist}(\mathbf{M}_i, {\mathbf{M}_i}^{\text{H}})_{x,y,z} < \mathbf{d}_1
\end{aligned}, \tag{3}$$

   where $\texttt{dist}$ is to get the shortest distance in x,y and z axis between $\mathbf{M}_i$ and ${\mathbf{M}_i}^{\text{H}}$, and $\mathbf{d}_1$ is the threshold vector. For those objects that are not supported by any others from behind, we restrain the search domain of θ by $\theta \in \{k\cdot\pi/4 | k = 0, 1, \ldots, 7\}$. Especially, this type of constraint will not be used if a bidirectional support relationship exists between two objects.
3. From point cloud data
   The point cloud is used to initialize model position and 3D scales. The height scale of the model can be initialized by

$$s_3^* = \frac{\max(\mathbf{P}_i)_z - \max({\mathbf{M}_i}^{\text{L}})_z}{\max(\mathbf{M}_i)_z - \min(\mathbf{M}_i)_z}. \tag{4}$$

   To deal with cases when point cloud is partly occluded, $\max(\mathbf{P}_i)_z - \max({\mathbf{M}_i}^{\text{L}})_z$ is used to estimate the real height of the target object. We set the geometric center of the point cloud $\mathbf{P}_i$ as ${\mathbf{p}_i}^{\text{c}}$, and the constraints are designed as the following

$$\begin{aligned}
&|\mathbf{p} - {\mathbf{p}_i}^{\text{c}}| < \mathbf{d}_2\\
&s_1 \in [{\rho_1}^{\text{L}}\cdot s_3^*, {\rho_1}^{\text{U}}\cdot s_3^*]\\
&s_2 \in [{\rho_2}^{\text{L}}\cdot s_3^*, {\rho_2}^{\text{U}}\cdot s_3^*]\\
&s_3 \in [{\rho_3}^{\text{L}}\cdot s_3^*, {\rho_3}^{\text{U}}\cdot s_3^*]
\end{aligned}, \tag{5}$$

   where $\mathbf{d}_2$ is to ensure that the model $\mathbf{M}_i$ overlaps the point cloud $\mathbf{P}_i$, and $\{({\rho_k}^{\text{L}}, {\rho_k}^{\text{U}}) | k = 1, 2, 3\}$ are used to adjust the model scales.

The optimization problem described above is built for both global model searching and local fine-tuning. Following the support hierarchy, every supported objects should be built after their supporting objects.

6.2 Global searching

The global searching step generally requires an efficient method to find out the model with a similar semantic shape in the whole parametric space. Here we adopt the Dividing Rectangles (DIRECT) algorithm [26] to solve the nonlinear optimization problem (1) and find the model with the highest IoU score. The DIRECT algorithm is a deterministic-search method, which can efficiently handle global optimization problems with bound constraints. It starts by scaling the search domain to a hypercube then subdivide it into smaller hyperrectangles step by step to find the global optima. Since we only use it to search an appropriate model for the next local matching, only a few iterations are required.

6.3 Local matching

After the model is identified, the BOBYQA algorithm [27] is followed to decide the final size, position and orientation. This derivative-free approach performs an iteratively constructed quadratic approximation for the objective function, where bound constraints are acceptable. In practice, we use the optima from the global searching to initialize the optimization variables and keep the constraints unchanged. The target object is built after

the algorithm converges.

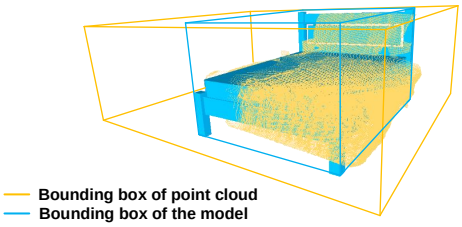
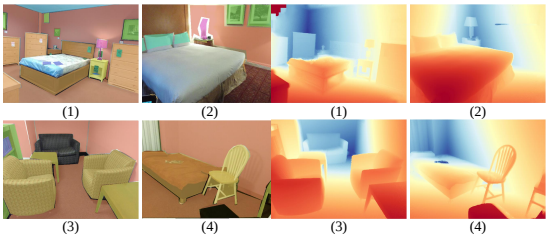


Figure 9: Point cloud and the matched model



(a) Instance masks (b) Depth maps

Figure 10: Ground truth data

7 Experiment and discussions

We present the modeling performance on a variety of indoor images from SUN-RGBD [4]. The whole algorithm is implemented on Ubuntu 16.04 with a GTX 1080 GPU and Intel Xeon CPU E5-1650 0 @ 3.20GHz x 12. The modeling results are shown in Figure 11. The parameters involved in our algorithm are presented in Appendix (A). The performance analysis, comparisons and limitations of our method are also discussed below.

7.1 Performance analysis

Since the ground-truth masks in SUN-RGBD dataset are only segmented at the class-level (see Figure 10a), a numerical comparison is not discussed here. Our segmentation results (Figure 11b) show that most objects in scenes are segmented with their masks refined. Figure 10b presents the corresponding ground-truth depth maps. The errors of the predicted depth maps (see Figure 11c) are evaluated by rel, rms and log10 scores [8] in Table 1, which shows that they are noisier than the test results on the NYU v2 dataset [8]. Although a noisy depth map would lead errors in initializing model positions and scales, we have demonstrated that, with support constraints, these depth maps are sufficient for retrieving semantic scenes. Figure 11d gives the room corner searching results. With the searched room corner as a reference system, the object models (Figure 11f) are built by matching their projection areas (Figure 11e) with the corresponding masks (Figure 11b).

Image ID	rel	rms	log10
(1)	3.509	1.427	0.644
(2)	3.512	0.995	0.644
(3)	4.497	1.089	0.735
(4)	4.738	1.083	0.744

Table 1: Depth estimation errors

Time consumption details are listed in Table 2. Since the grab-cut algorithm processes masks by sequence, the quantity of objects determines the time cost in segmentation. The time consumed in layout estimation is distinct between cases as its efficiency is correlated to the sparsity of the layout edge map. For the modeling step, we calculate the average time cost of matching with a single model, and take the summation for all objects involved. Taking data loading, transferring and all the other factors into consideration, a semantic scene with dozens of objects can be built within five minutes.

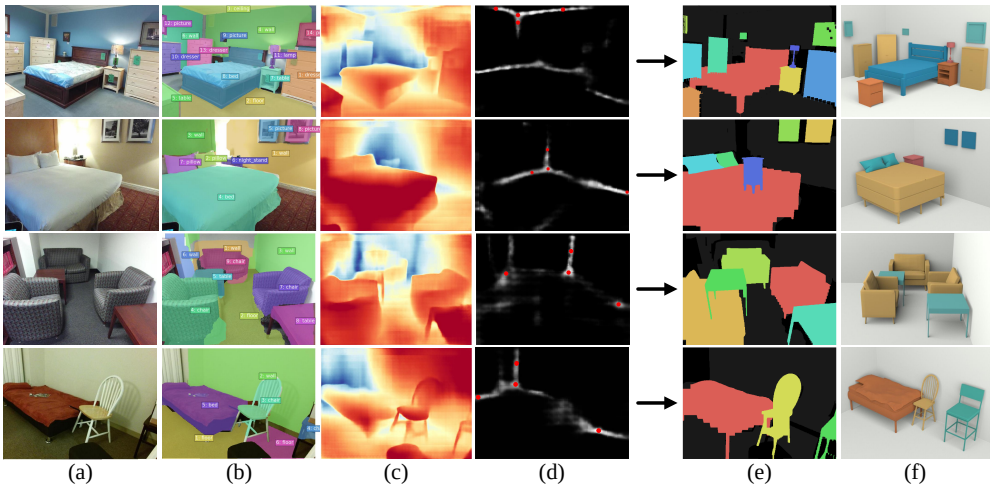
From a visual point of view, Figure 11f illustrates that our algorithm can retrieve plausible indoor objects even for badly occluded ones. This is mainly because we use the top point and the supported model of an object to deduce the height size. For occluded objects we can also obtain their spatial scope (see Figure 11e).

ID	Num	(a)	(b)	(c)	(d)
(1)	14	7.396	0.318	27.718	78.668
(2)	8	4.548	0.254	0.2398	78.137
(3)	9	5.713	0.304	1.1725	141.745
(4)	6	3.918	0.270	14.415	64.967

Table 2: Time consumption details (in seconds). ID: ID of test images; Num: Number of segmented objects; (a): Image segmentation; (b): Depth estimation; (c): Layout estimation; (d): Scene modeling

7.2 Comparisons and limitations

We have compared our method with two closely related works [19, 20]. There are some similarities within the modeling approach. All of our works have a single RGB image using a model database as the input and with the scene modeling completed in a data-driven manner. However, several differences exist. Firstly, our work benefits from high-level features with the three trained FCNs. There are fewer parameters in our method compared with hand-crafted methods that require features to be defined beforehand. Besides, in the segmentation part, different from [20], we do



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

Figure 11: Semantic modeling results. (a) Test images; (b) Instance masks; (c) Depth maps; (d) Layout edge maps; (e) 2D Projections of matched models; (f) Retrieved semantic scenes

not require users to give a semantic label for objects. Also unlike [19] where only main objects are segmented, more objects like pictures, blinds can be segmented in our work (see Figure 12). Secondly, these methods do not provide support semantics between objects. Offering the support context along with the reconstruction can provide cues for retrieving more objects that are supported by others (see Figure 13). In addition, we mainly handle scene modeling from photographs. For recovering scenes from rendered images, we need to loosen the constraints from point cloud and append extra placement priori.

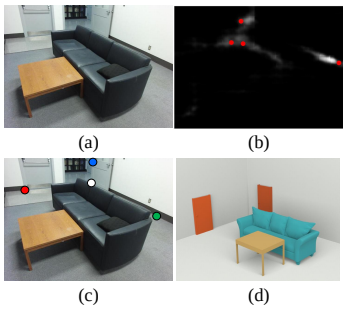
There are some limitations in our work. Although we have tested that our method appears robust in handling noisy inputs, it could possibly fail when the pre-processing step does not work well. The weakest part is the layout edge map generation. For images whose layout edge map is not clear or the layout frame is occluded as Figure 14a shows, the corner searching algorithm could fail (see Figure 14b). In these cases, however, only a few manual interactions are required. As Figure 14c presents, four points on the original image are picked to correct the result. This can be used to improve the final performance (see Figure 14d). For cases with an extremely complicated support context, which cannot be parsed with some dataset, a novel support inference method, based on point cloud and image features, is required. This will be our focus in future work.

(a) Input (b) Zhang et al. [19] (c) Our method

Figure 12: Segmentation comparison

(a) Input (b) Liu et al. [20] (c) Our method

Figure 13: Result comparison



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

Figure 14: Modeling with manual interactions

8 Conclusions

We propose a novel indoor scene modeling method with only a single photograph. Three FCN architectures are blended to produce different feature maps. We have shown that these high-level features can provide informative geometric clues and instance-level semantics for objects. Based on these features, support relationships between instances can be reasonably estimated in a data-driven manner. This offers an effective hierarchical constraint for the model matching, enabling our method to reconstruct objects with noisy inputs.

(a) Input (b) Cottage (c) Palace

Figure 15: Furniture replacement

The experiments show that we can retrieve reliable geometry with detailed support context for indoor scenes even when poorly occluded objects exist. In the future, based on semantic scenes, scene-editing applications (e.g. furniture replacement) will be developed for VR devices, which enables people to experience realistic 3D indoor scenes with different decoration styles (see Figure 15).

Acknowledgements

The research leading to these results has been partially supported by VISTA AR project (funded by the Interreg France (Channel) England), the China Scholarship Council and Bournemouth University.

References

[1] Chen K, Lai YK, and Hu SM. 3d indoor scene modeling from rgb-d data: a survey. *Computational Visual Media*, 1(4):267–278, 2015.

[2] Hua BS, Pham QH, Nguyen DT, Tran MK, Yu LF, and Yeung SK. Scenenn: A scene meshes dataset with annotations. In *3D Vision (3DV), 2016 Fourth International Conference on*, pages 92–101. IEEE, 2016.

[3] Silberman N, Hoiem D, Kohli P, and Fergus R. Indoor segmentation and support inference from rgbd images. *Computer Vision–ECCV 2012*, pages 746–760, 2012.

[4] Song S, Lichtenberg SP, and Xiao J. Sun rgb-d: A rgb-d scene understanding benchmark suite. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 567–576, 2015.

[5] Long J, Shelhamer E, and Darrell T. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015.

[6] Dai J, Li Y, He K, and Sun J. R-fcn: Object detection via region-based fully convolutional networks. In *Advances in neural information processing systems*, pages 379–387, 2016.

[7] Li Y, Qi H, Dai J, Ji X, and Wei Y. Fully convolutional instance-aware semantic segmentation. *arXiv preprint arXiv:1611.07709*, 2016.

[8] Laina I, Rupprecht C, Belagiannis V, Tombari F, and Navab N. Deeper depth prediction with fully convolutional residual networks. In *3D Vision (3DV), 2016 Fourth International Conference on*, pages 239–248. IEEE, 2016.

[9] Mallya A and Lazebnik S. Learning informative edge maps for indoor scene layout prediction. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 936–944, 2015.

[10] Xiong X and Huber D. Using context to create semantic 3d models of indoor environments. In *BMVC*, pages 1–11, 2010.

[11] Silberman N and Fergus R. Indoor scene segmentation using a structured light sensor. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pages 601–608. IEEE, 2011.

[12] Gupta S, Girshick R, Arbeláez P, and Malik J. Learning rich features from rgb-d images for object detection and segmentation. In *European Conference on Computer Vision*, pages 345–360. Springer, 2014.

[13] Lai K, Bo L, and Fox D. Unsupervised feature learning for 3d scene labeling. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 3050–3057. IEEE, 2014.

[14] Yang S, Xu J, Chen K, and Fu H. View suggestion for interactive segmentation of indoor scenes. *Computational Visual Media*, 3(2):131–146, 2017.

[15] Qi CR, Yi L, Su H, and Guibas LJ. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *arXiv preprint arXiv:1706.02413*, 2017.

[16] Xu K, Chen K, Fu H, Sun WL, and Hu SM. Sketch2scene: sketch-based co-retrieval and co-placement of 3d models. *ACM Transactions on Graphics (TOG)*, 32(4):123, 2013.

[17] Guo R, Zou C, and Hoiem D. Predicting complete 3d models of indoor scenes. *arXiv preprint arXiv:1504.02437*, 2015.

[18] Shen CH, Fu H, Chen K, and Hu SM. Structure recovery by part assembly. *ACM Transactions on Graphics (TOG)*, 31(6):180, 2012.

[19] Zhang Y, Liu Z, Miao Z, Wu W, Liu K, and Sun Z. Single image-based data-driven indoor scene modeling. *Computers & Graphics*, 53:210–223, 2015.

[20] Liu M, Guo Y, and Wang J. Indoor scene modeling from a single image using normal inference and edge features. *The Visual Computer*, pages 1–14, 2017.

[21] Izadinia H, Shan Q, and Seitz SM. Im2cad. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2422–2431. IEEE, 2017.

[22] Hariharan B, Arbeláez P, Girshick R, and Malik J. Simultaneous detection and segmentation. In *European Conference on Computer Vision*, pages 297–312. Springer, 2014.

[23] Rother C, Kolmogorov V, and Blake A. Grabcut: Interactive foreground extraction using iterated graph cuts. In *ACM transactions on graphics (TOG)*, volume 23, pages 309–314. ACM, 2004.

[24] Konolige K and Mihelich P. Technical description of kinect calibration. *Tech. Rep., Willow Garage*,

2011.

[25] Hedau V, Hoiem D, and Forsyth D. Recovering the spatial layout of cluttered rooms. In *Computer vision, 2009 IEEE 12th international conference on*, pages 1849–1856. IEEE, 2009.

[26] Jones DR, Perttunen CD, and Stuckman BE. Lipschitzian optimization without the lipschitz constant. *Journal of Optimization Theory and Applications*, 79(1):157–181, 1993.

[27] Powell MJ. The bobyqa algorithm for bound constrained optimization without derivatives. *Cambridge NA Report NA2009/06, University of Cambridge, Cambridge*, 2009.

Appendix

(A) Parameter decision

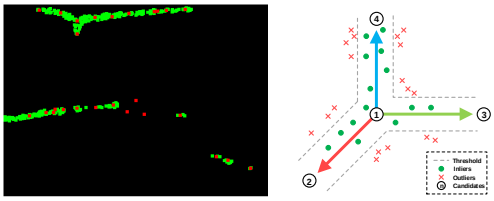
In image segmentation, we keep training configurations following the suite of Li. [7]. In the room corner searching step (see Algorithm 1 in Appendix (B)), we set the maximal iteration number as 1000, the goal of inlier number as $0.7 *$ number of edge pixels, distance threshold as 10.

In object modeling, we set $\mathbf{d}_1 = [0.5, 0.5, 0.5]^{\mathbf{T}}$ (in meters, the same below) for normal objects. For those supported by a wall, $\mathbf{d}_1$ is set as $[0.2, \infty, \infty]^{\mathbf{T}}$ or $[\infty, 0.2, \infty]^{\mathbf{T}}$ depending on the orientation of the wall. $\mathbf{d}_2$ is set as $[1.0, 1.0, 0.5]^{\mathrm{T}}$ as the point cloud is noisier in horizontal plane than in the vertical direction (see Figure 9). For model scales, we set ${\rho_1}^{\mathrm{L}} = {\rho_2}^{\mathrm{L}} = {\rho_3}^{\mathrm{L}} = 0.8$, ${\rho_1}^{\mathrm{U}} = {\rho_2}^{\mathrm{U}} = 1.2$, and ${\rho_3}^{\mathrm{U}} = 1.0$. While for objects whose top part is occluded (see the rightmost chair in the fourth case in Figure 11a), the point cloud could underestimate the model height size. We hence change the lower bounds to ${\rho_1}^{\mathrm{L}} = {\rho_2}^{\mathrm{L}} = {\rho_3}^{\mathrm{L}} = 1.0$, and the upper bounds to ${\rho_1}^{\mathrm{U}} = {\rho_2}^{\mathrm{U}} = {\rho_3}^{\mathrm{U}} = 2.0$ or more. In the global searching step, the maximal iterations number is limited to 50, while in the local matching, generally we do not set the maximal iteration number to ensure convergence, the only stopping criteria is set as when the absolute tolerance reaches 10^{-3}.

(B) Room corner searching method

Based on the edge map (see Figure 5a), we reserve pixels with a probability score higher than the median value (see green dots in Figure 16a). Then the Harris corner detector is adopted to find all possible corners on the map (see red dots on Figure 16a). A relative smaller block size (3x3) is used in the Harris detector to produce extra candidates for searching four points to find the room corner. We utilize the RANSAC method to get the optimal four-point set among the corner candidates (see Figure 5b). With the point cloud, the rotation matrix and the translation vector of the room corner can be estimated as the camera's extrinsic parameters. The pseudo codes of our designed RANSAC algorithm is described in Algorithm 1.

In Algorithm 1, based on our observations, we claim four points forming a room corner when one of them is covered by the convex hull of the others (e.g. in Figure 16b, point 1 is covered by the triangular comprised by point 2, 3 and 4). Here we down-sample the edge map by the sampling interval as 30 pixels to improve efficiency. The whole algorithm only costs several seconds.



(a) Candidates for corner searching (b) Inliers and outliers in RANSAC searching

Figure 16: Searching the optimal corner on an edge map of the room layout

Algorithm 1 RANSAC algorithm for searching the room corner

Data: Point candidates (red dots), Edge pixels (green dots) (see Figure 5b);
Result: The optimal four points set $\mathbf{S}^*$ which forms a corner;
Initialization: Set the maximal number of iterations as max_iter, current step as i, goal of inlier number as $inliers^{\mathrm{g}}$, current inlier number as $inliers$, current best inlier number as $inliers^{\mathrm{b}} = 0$, distance threshold as $dist\_thresh$, current best points set as $\mathbf{S}^{\mathrm{b}} = \{\}$;
while $i \leq max\_iter$ & $inliers^{\mathrm{b}} \leq inliers^{\mathrm{g}}$ **do**
    Randomly pick out a four points set $\mathbf{S} = \{\mathbf{O}, \mathbf{X}, \mathbf{Y}, \mathbf{Z}\}$ from the point candidates without replacement;
    if $\mathbf{S}$ forms a corner **then**
        Calculate the $inliers$ as the number of pixels on the edge map whose the shortest distance from $\{\overrightarrow{\mathbf{OX}}, \overrightarrow{\mathbf{OY}}, \overrightarrow{\mathbf{OZ}}\}$ is smaller than $dist\_thresh$ (see Figure 16b);
        $i = i + 1$;
        if $inliers > inliers^{\mathrm{b}}$ **then**
            $inliers^{\mathrm{b}} = inliers$;
            $\mathbf{S}^{\mathrm{b}} = \mathbf{S}$;
        end if
    end if
end while
$\mathbf{S}^* = \mathbf{S}^{\mathrm{b}}$;