Hierarchical Parsing of 3D Scenes via Recursive Context Propagation

Allen Wu
Princeton University
Princeton NJ 08540
zhelunw@princeton.edu

Abstract

Many tasks in 3D scene analysis and synthesis involve learning and inference of contextual information. Contextual analysis faces the coupled problems of object extraction and relation inference. Context can be defined only when objects are accurately detected. Object extraction, on the other hand, is best be conducted with the help of context. A natural solution thus seems to be interleaving the two problems and iteratively improving object detection and contextual relation inference. We propose a method for inferring a hierarchical parse of a 3D scene. We then utilize a recursive neural network (RvNN) architecture that leverages these hierarchical parses to aggregate and propagate contextual information, refining the object identities and their relations in an iterative process. With this approach, we demonstrate that we can improve object detection performance in both reconstructed and synthetic 3D scenes relative to baselines that use hand-crafted hierarchy construction algorithms, or no hierarchical information.

1. Introduction

In the real world we need to identify physically present objects to successfully navigate. As humans, we rely on our understanding of both the 3D shape and the semantics of the environment to interact with the objects in the environment. For autonomous robots, an understanding of real 3D scenes is essential to perform common tasks such as turning on a TV, or bringing a bottle from the kitchen table.

With such examples as motivation, our goal is to demonstrate that addressing the two problems of object detection and relation inference jointly can lead to improved performance. The two problems of object extraction and relation inference are often viewed as sequential tasks, but they are actually deeply correlated. A key observation was made in our paper on the dataset we studied, SUNCG [5], a large-scale dataset of 3D scenes with dense occupancy and semantic annotations. For many rooms, there are patterns of how the objects are arranged. They often aggregate in pairs or groups. Some are obvious, such as table and sofa; desk and chair; toilet and bathtub; computer and shelves etc. (see Figure 2) So naturally, there are relations between groups of objects that can be utilized for us to identify objects and in this paper, we are trying to describe those relations in a hierarchical way.

Such a hierarchical representation is useful in two ways. First, we can group furniture and other objects in the room in an organized, hierarchical way. This is useful if we want to further classify objects by adding extra annotations in the room to different functional roles and relationships (e.g., dining table or bedside lamp). Second, we are able to guess unknown or partially observed objects in the dataset by learning relationships between the hierarchical structures and spatial relations.

We wish to build such a hierarchical structure of a scene where frequently co-occurring objects are grouped into similar sub-hierarchies, even across different scenes. However, this task is quite challenging, especially for real world scenes where the objects are partially observed and their categories are unknown. Without knowing object categories, how can we build a scene hierarchy and utilize its contextual information to help scene analysis? In this work, we achieve this in two stages. First, we train a model that can predict the co-occurrence probability between any two objects without knowing their category labels. Second, we leverage the model to build a scene hierarchy in a greedy manner.

This prediction model is learned from the dataset without any heuristic rules. After the hierarchical structure has been built, we combine it with a recursive neural network based object detection method to improve object detection performance. The results indicate that hierarchies built by our method are not only meaningful and explainable, but also beneficial for context propagation and object detection.

This paper is structured as follows. We start by discussing related work (Section 2) and how we find the starting point of this research project. In Section 3, we explain how to build the co-occurrence model. Then in Section 4, we show how our model can be combined with a recursive

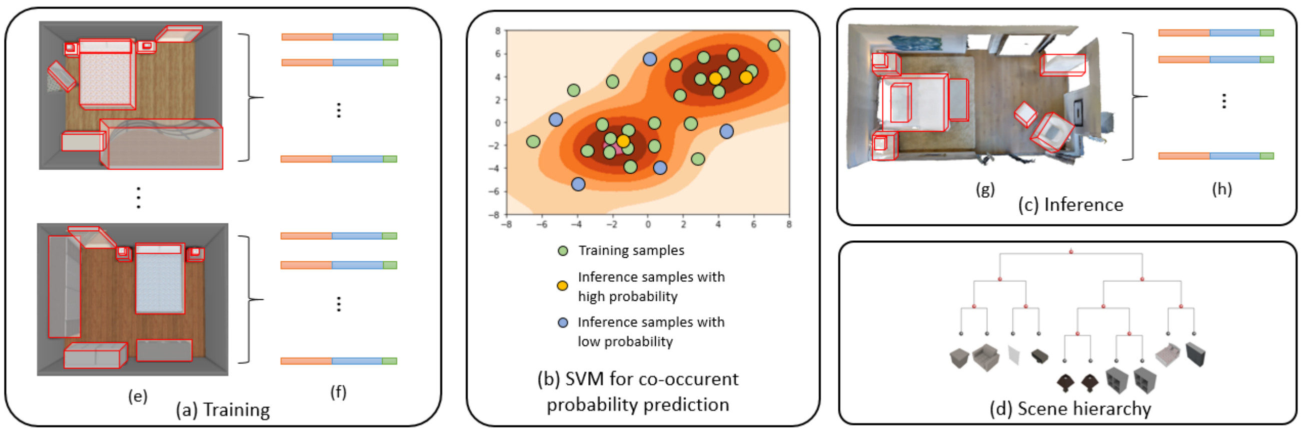

(e)
(a) Training
(f)
Training samples
Inference samples with high probability
Inference samples with low probability
(b) SVM for co-occurent probability prediction
(g)
(c) Inference
(h)
(d) Scene hierarchy

Figure 1: An overview of our pipeline. We train an one-class SVM (b) to predict co-occurrent probability for a pair of objects. During training (a), training samples are collected by randomly selecting object pairs from rooms in SUNCG (e). Each object pair is represented by a feature vector (f). The bar in orange and blue represent the PointNet feature of the two objects, respectively, while the bar in green represents their spatial layout feature. During inference (c), object proposals are extracted from the input scene (g). Feature vectors are computed (h) and co-occurrent probability for every two objects are predicted by the learnt SVM (b). A scene hierarchy is generated by using the SVM predictions in a greedy manner.

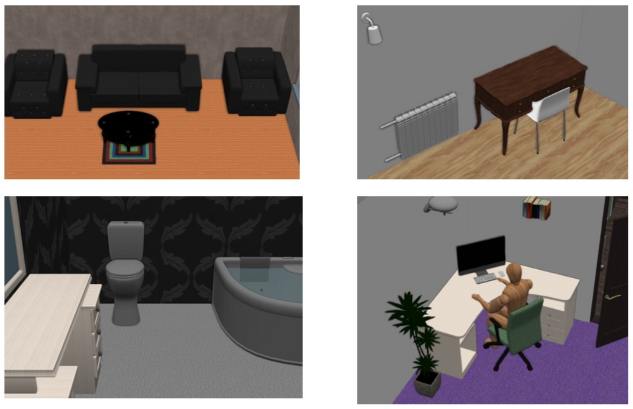


Figure 2: Some Co-occurrent Patterns in SUNCG dataset

neural network architecture and trained for improving object detection. Section 5 concludes the work.

2. Related Work

Hierarchical representation has always been a hot area in indoor scene context understanding, and we are trying to apply it with a different flavor of interweaving in order to better solve the indoor object recognition problem.

Originally, we were inspired by [6], which used Gaussian Mixed Model(GMM) and a Siamese network both processed by PointNet to jointly train a retrieval and embedding network, which is then used for identifying the probability of two parts of an object next to each other and thus reconstruct an object from partial pieces of an object.

Our hierarchical parsing and object detection approach is based on the GRASS [2] model which was applied to hierarchically parse and reconstruct 3D object geometry.

This is a really illuminating point by using both positive samples and negative samples as this will help the network differentiate samples with higher confidence as there is a larger margin when training on both positive samples and negative samples as opposed to only positive samples. Also sharing weights across these two networks means fewer parameters to train for, which in turn means less data required and less tendency to overfit.

In our paper, we also use PointNet to extract features and retrieve relations between different objects instead of object parts. Additionally, we are trying to learn the pattern and thus help us reconstruct the missing or unidentified objects in the scene. However, we are not using negative samples to train our network.

3. Method

We define scene hierarchy generation as the problem of taking as input the set of detected objects and the output is a binary tree representing a hierarchical grouping of the objects (see Figure 1). The leaves of the output tree are the detected objects and the internal nodes represent subgroupings of the objects. The root of the tree represents the entire room.

To form the hierarchy, we define an affinity score $A(n_i, n_j)$ between two nodes n_i and n_j. We want this affinity score to capture the co-occurrence probability of two nodes. To do so, we build an object-object co-occurrence

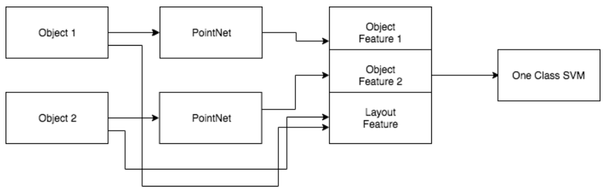


Figure 3: Details of the co-occurrence model

model (Section 3.1) by training a one-class SVM to estimate the co-occurrence probability of two object nodes $P(n_i, n_j)$. Then, for any two nodes n_i and n_j, we compute the affinity as the average of all pairs of objects in the two nodes:

$$A_s(n_i, n_j) = \frac{1}{|n_i||n_j|} \sum_{o_i \in n_i, o_j \in n_j} P(o_i, o_j)$$

. Using the affinity score between nodes, we greedily build the hierarchy bottom-up (see Section 3.3).

3.1. Co-occurrence Model

In order to build the co-occurrence model, we take as input pairs of objects that appear in the same room. Each object is represented as a 3D point cloud with an axis-aligned bounding box (AABB), and a category label. The 3D point cloud is essentially a set of points $\{p_i\} = \{x_i, y_i, z_i, r_i, g_i, b_i\}$ with their 3D (x, y, z) position and color (r, g, b). In addition, we also want the co-occurrence model to take into account the spatial relationship between the two objects. Thus, the input will be composed of three parts: features for the two objects, and features capturing the spatial layout of the two objects. The output of the model will be a value between 0 and 1, indicating the probability that these two objects co-occur in the room.

Based on the above, we design a one-class SVM that takes as input an object descriptor for the two object nodes $D_o(n_i)$ and $D_o(n_j)$ and a spatial relationship descriptor $D_s(n_i, n_j)$ between the two objects. Using a one-class SVM allows us to estimate the probability that a given object pair is expected to co-occur using only positive samples for training. Figure 3 shows the architecture of the co-occurrence model. Given two objects, the PointNet features [4] are extracted. We concatenate these two vectors along with a layout descriptor [3]. We then use the concatenated feature vector to train an one-class SVM that outputs the co-occurrence probability between two objects (see Figure 1b).

We consider two variants for the feature descriptor of an object D_o: i) an one-hot vector representation of the category label, and ii) features extracted using PointNet [4]. PointNet is a neural network model that is designed to consume a raw point cloud (set of points) and learn both global and local point features. PointNet has been shown to effectively learn the informative features from a point cloud and use these features for segmentation and labeling. So with the ability of summarizing an input point cloud, we are motivated to use the feature extracted by PointNet. The one-hot vector allows us to see how well our model can perform if we had access to the category label, while using the PointNet features allows us to estimate the co-occurrence probability of two objects by appearance only (e.g. without access to the category label).

The spatial relationship descriptor D_s captures the spatial relationship between the bounding boxes of two object nodes n_i and n_j. Following [3], we define $D_s(n_i, n_j)$ between two nodes n_i and n_j as a 7-dimensional vector:

$$\begin{aligned} D_s(n_i, n_j) = [&n_i.z_{min} - n_j.z_{min}, \\ &n_i.z_{min} - n_j.z_{max}, \\ &n_i.z_{max} - n_j.z_{min}, \\ &\|n_i.\text{box.center} - n_j.\text{box.center}\|, \\ &\textbf{Dist}(n_i.\text{box}, n_j.\text{box}), \\ &\textbf{Vol}(n_i.\text{box} \cap n_j.\text{box})/\textbf{Vol}(n_i.\text{box}), \\ &\textbf{Vol}(n_i.\text{box} \cap n_j.\text{box})/\textbf{Vol}(n_j.\text{box})] \end{aligned}$$

Intuitively, the first three components describe the relative height of the two objects and capture the vertical relationship between them. The fourth component is the Euclidean distance between the centroids of n_i and n_j. The fifth is the distance between the two bounding boxes defined as the closest distance between all pairs of points on those two bounding boxes. The last two components represent the amount of overlap between the two regions.

The overall co-occurrence model can be written as,

$$P(n_i, n_j) = f(D_o(n_i), D_o(n_j), D_s(n_i, n_j))$$

. where $D_o(n_i)$ and $D_o(n_j)$ are the features of object n_i and object n_j, respectively. $D_s(n_i, n_j)$ is the spatial descriptor of the layout between the two objects (n_i, n_j). $P(n_i, n_j)$ is the co-occurrence probability of the object pair (n_i, n_j). If $P(n_i, n_j) > 0.5$, we will take it as ”inlier”, indicating these two are likely to be co-occurrent, otherwise, it is an ”outlier” indicating they are not co-occurrent.

3.2. Training the Model

To train the one-class SVM, we use individual rooms extracted from the SUNCG dataset [5]. In our experiments, we focus on four types of rooms: bedroom, living room, office, and toilet. We follow the train/test split used in [5]. For each room type, we filter out bad rooms whose floor area is larger than 50m^2 and randomly select 9000 rooms for training and 1000 rooms for testing from the rest. We further randomly select 45,000 object pairs from the 9000 rooms for training and 5000 object pairs from the 1000 rooms for testing. Table 1 shows the number of objects that occur in the different room types.

	average # of objects per room	min # of objects per room	max # of objects per room
bedroom	17.7	7	45
office	19.3	4	43
toilet	7.9	1	44
living room	17.4	5	56

Table 1: Number of objects for different room type

Given the variety of data in SUNCG, we might learn some infrequent co-occurrences between objects, but the effect is marginal because the one-class SVM is capable of fitting into the majority pattern of the training and ignoring those deviant sample points.

To create the training data, we first sample points from the SUNCG object models to create point clouds of all SUNCG models. Since real-world scans are often incomplete, we create incomplete versions of the models by randomly sampling a plane and remove all points on one side of the plane. This is intended to approximate occlusions in real scans. A completeness degree is assigned to each incomplete point cloud (e.g. a point cloud with 20% of points discarded will have a completeness degree of 80%). We use point clouds with a completeness degree of 80% to 90% for training so that our network can handle incomplete scans.

There is no restriction on the distance or any other spatial alignment restraints between the objects, we sample the pairs randomly from all rooms, so the patterns we learn from the SVM is unprejudiced and general.

We used `sklearn.svm.SVC` as our one-class SVM to train over the entire training set, and we simply use `SVC.score` to churn out the score for each test data descriptor. The score will pass through sigmoid function $\sigma(x) = \frac{1}{1+e^{-x}}$ and return us a value between 0 and 1. We then take this value as the likelihood of the pair co-occurring in the same room. All SVM parameters are default values.

3.3. Scene Hierarchy

Once we have trained the object-object co-occurrence model, we can compute the affinity score between any two nodes by taking the average of the co-occurrence probability between all pairs of objects across the two nodes. To build the scene hierarchy of the room, we cluster nodes greedily in a bottom-up fashion, starting with the detected objects as the leaf nodes, until only one node remains.

The algorithm for building the hierarchy is as follows:

1. Compute the affinity of every two nodes. Initially, those are the co-occurrence probabilities based on the point clouds of the objects.
2. Select the two nodes with the highest co-occurrence probability to form a merged node and delete those two nodes.
3. Compute the affinities of the newly generated node with other existing nodes.
4. Repeat step 2 and 3 until there is only one node left.

Examples of output hierarchies are shown in Figure 6.

4. Experiments and Results

4.1. Co-occurrence model

To validate that the one-class SVM works well for both positive and negative examples, we create two groups of test data: a positive set that is drawn from pairs of objects that occur in the rooms, and a negative set that is created by swapping out an object from a room with another object from another room of the same type. When replacing the object, we make sure that the bounding box center of the inserted object is aligned with the object that is switched out. While it is possible that the switch will result in a reasonable placement, this switching procedure will most likely introduce unreasonable pairs, which can serve as good contrast examples.

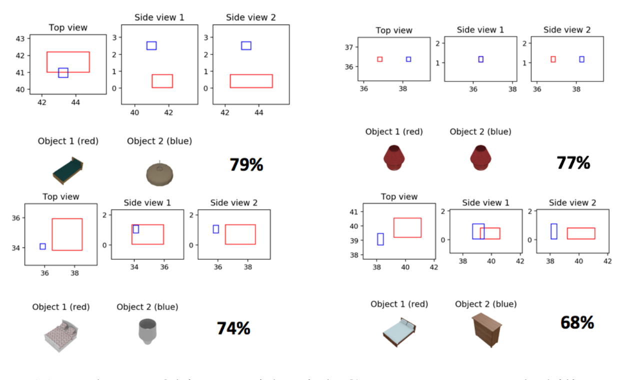
Qualitative examples To qualitatively evaluate our co-occurrence model, we visualize co-occurrence probabilities predicted by our model. We show high and low co-occurrence object pairs for each of the four room types in Figure 4. For each pair, visualize the bounding boxes of the objects as seen from the top, and two sides.

Now let us go through those four types of rooms and see if our SVM can distinguish between high and low probability object pairs. In Figure 4a where it shows bedroom objects with high co-occurrence probability, we found bed and pendant light are paired together with 79% confidence, two lamps juxtaposed together with 66%, a bedside lamp with a bed of 74% and bed with cupboard of 68%. They are all kind of reasonable.

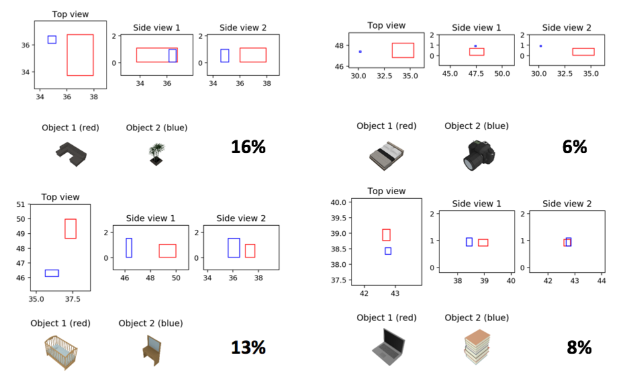
For bedroom objects with low co-occurrence probability in Figure 4b, we have sofa and plant, bed with camera, cradle with dress table and laptop with books.

For living room in Figure 4c, sofa and table, television and TV stand, big sofa by a small sofa, and sofa with TV are highly co-occurrent. In Figure 4d, it shows that lamps are not usually placed near the window, stereo is not common to be 2 meters by 2 meters sideway placed from a TV stand. Also the trash can should not be placed near the dresser.

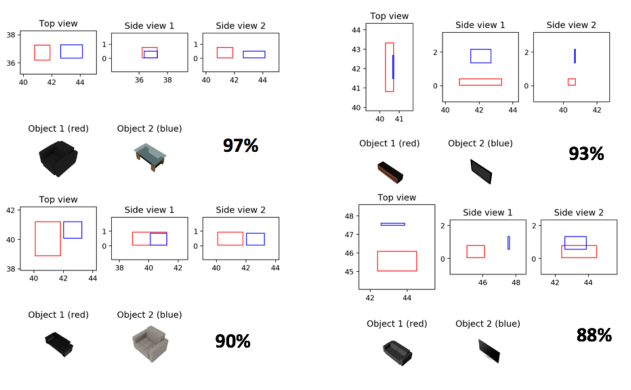
In an office, desk and chair are usually placed together, computer usually on top of the desk, two chairs close to each other and chair with an laptop are highly co-occurrent objects as shown in Figure 4e. On the other hand, in Figure 4f, the sofa is not placed near a piano, and the dining table not near desk and so on.



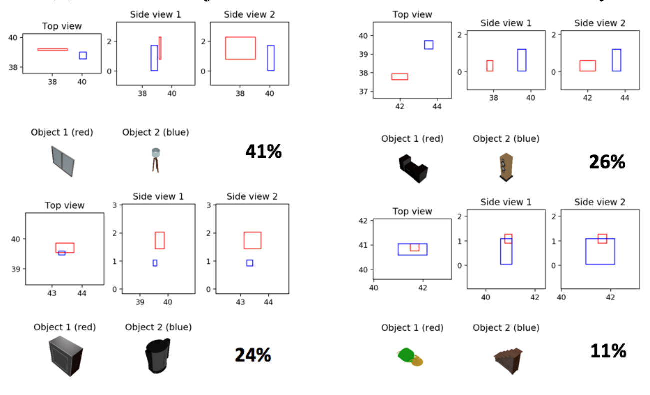
(a) Bedroom Objects With High Co-occurrence Probability



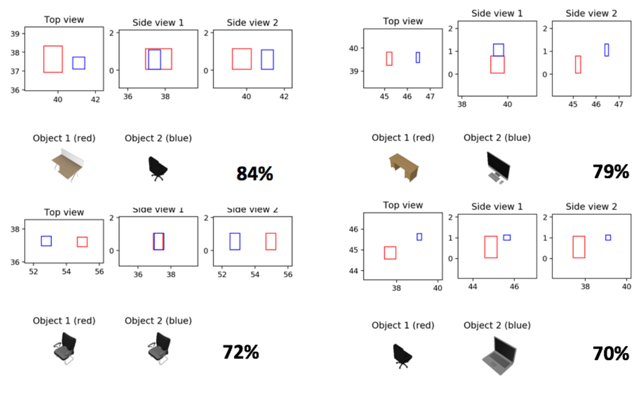
(b) Bedroom Objects With Low Co-occurrence Probability



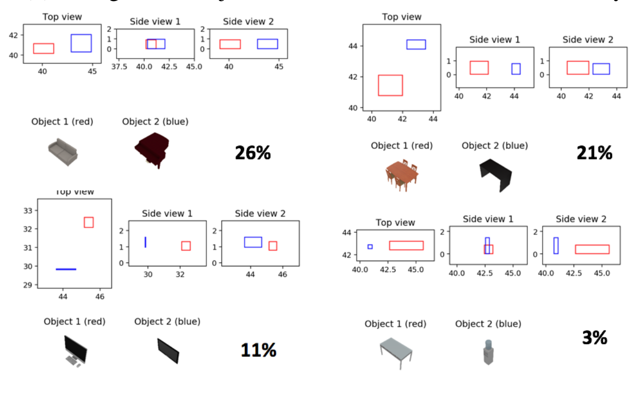
(c) Living Room Objects With High Co-occurrence Probability



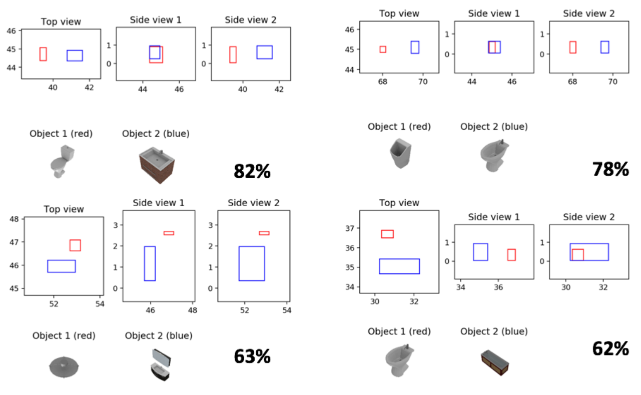
(d) Living Room Objects With Low Co-occurrence Probability



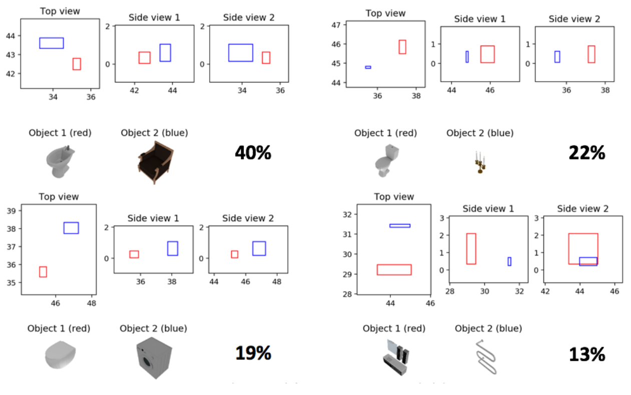
(e) Office Objects With High Co-occurrence Probability



(f) Office Objects With Low Co-occurrence Probability



(g) Restroom Objects With High Co-occurrence Probability



(h) Restroom Objects With Low Co-occurrence Probability

Figure 4: Predicted co-occurrence probabilities for different room types

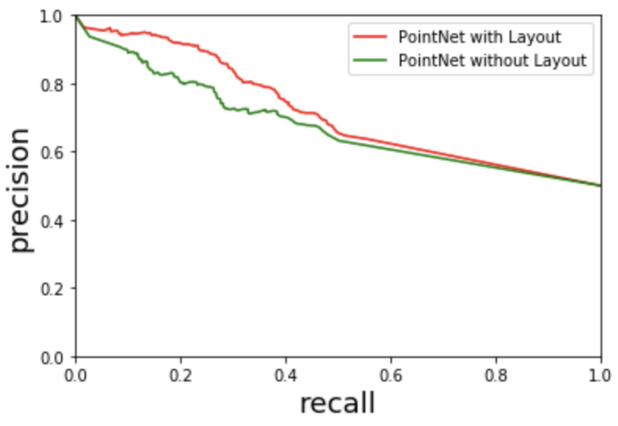


Figure 5: Precision-Recall Curve Comparison

In restrooms, the toilet is usually placed along with the sink, standing toilet and flushing toilet together, and light-sink pair as in Figure 4g. Likewise, it is uncommon to have a chair in the restroom, or washing machine as in Figure 4h.

Quantitative evaluation For a quantitative evaluation, we compute the precision-recall relation of the trained SVM on the prepared test set (see Figure 5). With a one-hot encoding (not shown), we are able to distinguish between high and low probability cases perfectly. Using PointNet features, including the spatial layout features can improve the overall precision at lower recall.

We took a positive set of 5000 pairs and a negative set of 5000 pairs. As we can see on Figure 5, when recall rate goes to 1, the precision goes to 0.5. In order to generate the graph, We used the trained SVM to score test data. And then we scan the threshold from 0 to 1 with step 0.01 by specifying those object pairs with higher likelihood than threshold be taken as positive and negative otherwise. Then we compare the predicted labels with the ground-truth labels as we know which are from the positive sets and which are from the negative set. In this way, we get to know the true positives(TP), false positives(FP), true negatives(TN) and false negatives(FN), and further we get the PR curve. Just for reference,

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

So this explains why when threshold is 0, we just predict all pairs as true, so the precision rate is 0.5 and the recall rate is 1.

4.2. Hierarchy Results

Figure 6 shows example hierarchies for different room types generated by our method (using PointNet features and the layout features). When generating the hierarchy we exclude objects that are supported by the walls (e.g. doors and windows) and ceiling (e.g. ceiling lights and ceiling fans). As shown in Figure 6, the generated hierarchies have reasonable groupings such as the grouping of laptop with desk, and then with chair (2nd row left, and 4th row right). The network also learns to give high co-occurrence probabilities to pairs of objects that are the same such as two nightstands (1st row left), four bar stools (2nd row right), and two chairs (4th row right). An example of a less ideal grouping would be the grouping of the two chairs (2nd row right) with the couch area instead of the desk area.

4.3. Integration with Object Detection Model

To further validate the usefulness of our hierarchy, we investigate whether it can be used to improve object detection. We hypothesize that by leveraging an RvNN (recursive neural network) based encoder and decoder that can capture the hierarchical organization of the room, we can effectively capture contextual cues for improved object detection.

To do so, we propose a novel hierarchical object detection model based on RvNNs that will take as input an hierarchy of incomplete object point clouds and output predicted categories for each object and bounding box offsets (to correct for noisy/incomplete input objects).

4.3.1 Hierarchical object detector

The architecture of the RvNN object detection model is illustrated in Figure 7 (a). There are two components: an encoder to aggregate information in a bottom-up manner and a decoder to parse the aggregated information in a top-down manner.

For the encoder, the hierarchy generated by our method is applied. Each node (both leaf node and internal node) contains a fixed-length feature vector. The leaf nodes are nodes for detected objects. The input of leaf nodes for the encoder is a feature vector containing both PointNet features and the AABB parameters for the object. The PointNet features contain the descriptor of the object itself, while the AABB parameters encode its location in world coordinates as well as its physical size. Internal nodes are aggregated nodes for multiple objects with a latent fixed-length feature vector that is learned. We compute this feature vector by using the network shown in Figure 7 (b). The network takes as input both the feature of the two children and

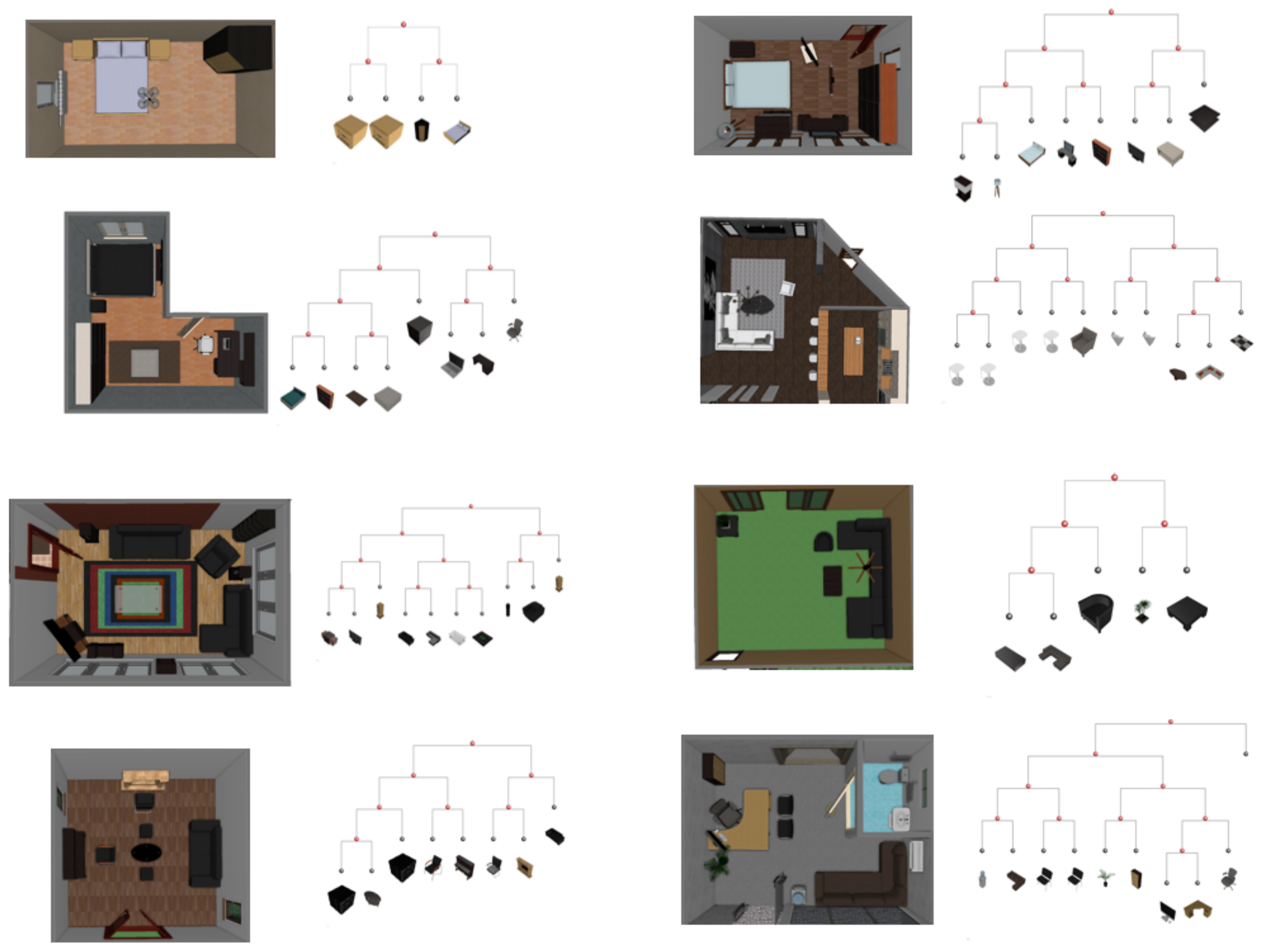


Figure 6: A gallery of the hierarchies generated by our method

	Chair	Table	Cabinet	Cushion	Sofa	Bed	TV	mAP
No hierarchy	0.67	0.50	0.10	0.85	0.19	0.81	0.33	0.49
Random hierarchy	0.72	0.63	0.16	0.85	0.36	0.79	0.42	0.56
Hierarchy with pre-defined rules	0.75	0.60	0.13	0.90	0.37	**0.84**	0.40	0.57
Our hierarchy	**0.78**	**0.68**	**0.23**	**0.91**	**0.47**	0.78	**0.43**	**0.61**

Table 2: Average Recognition Precision on Matterport3D

their spatial descriptor (as described in Section 3.1) and outputs a merged feature vector. The complete encoding is accomplished by repeatedly collapsing pairs of nodes into a merged node. The final output of the encoder is a root node with a fixed-length feature vector which contains the contextual information of the entire scene.

For the decoder, the same hierarchy is applied. Starting from the root node, we recursively parse the root node feature into a set of leaf nodes. During decoding, internal nodes are split into two children using the network in Figure 7 (c). For leaf nodes, we use an object classifier and an AABB parameter regressor, as shown in 7 (d), to predict the object category label and to compute the size and location of the object, respectively.

4.3.2 Evaluation

We tested the object detection method by using our hierarchy on two dataset: Matterport3D [1] and SUNCG [5]. Matterport3D contains 3D scene data captured throughout 90 properties with a Matterport Pro Camera. For Matter-

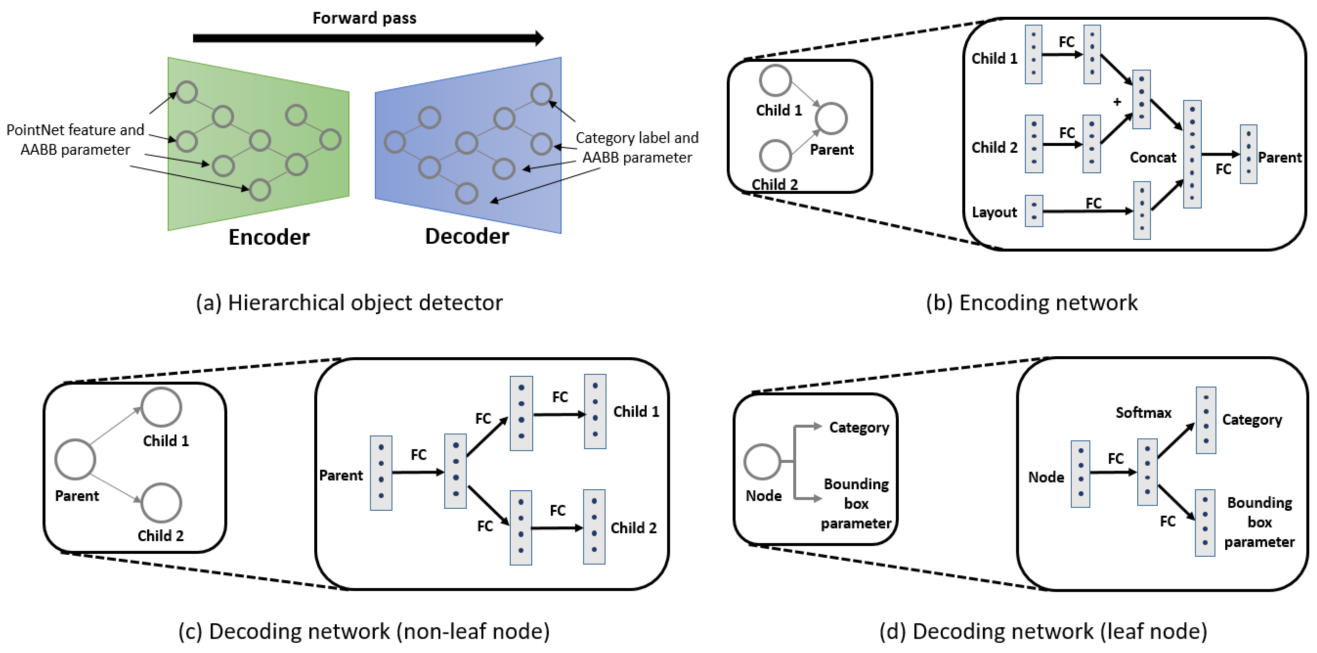

Forward pass
PointNet feature and AABB parameter
Encoder
Decoder
Category label and AABB parameter
(a) Hierarchical object detector
Child 1
Child 2
Parent
FC
+
Layout
Concat
(b) Encoding network
Parent
Child 1
Child 2
FC
(c) Decoding network (non-leaf node)
Node
Category
Bounding box parameter
Softmax
FC
(d) Decoding network (leaf node)

Figure 7: Our model for object detection based on RvNN and autoencoder

	Chair	Table	Cabinet	Cushion	Sofa	Bed	TV	mAP
No hierarchy	0.68	0.38	0.22	0.69	0.71	0.58	0.68	0.56
Random hierarchy	0.72	0.47	**0.46**	0.76	0.88	0.85	0.94	0.73
Hierarchy with pre-defined rules	0.73	0.54	0.38	0.82	0.90	0.84	**0.96**	0.74
Our hierarchy	**0.81**	**0.58**	0.43	**0.88**	**0.93**	**0.88**	0.95	**0.78**

Table 3: Average Recognition Precision on SUNCG

port3D, we train our object detection model on 225 rooms and test on 60 rooms following the train/test split in [1]. SUNCG includes a large amount of synthetic 3D houses manually designed by people. For SUNCG, we randomly select 5000 rooms whose floor area is smaller than 50 m^2. 4000 rooms and 1000 rooms are used for training and testing, respectively.

The metric for evaluation is average precision with intersection over union (IOU) at 0.25 for each category and mean average precision for all categories. We demonstrate the improved performance of object detection using hierarchies produced by our method relative to three baselines: 1) No hierarchy: We directly use PointNet features to predict object category and AABB without using the RvNN based model. We add four FC layers in the network to increase the number of network parameter to make the comparison fair; 2) Hierarchy (random): We use our object detection model with randomly generated hierarchies; 3) Hierarchy (rules): We use our RvNN-based scene parsing model with hierarchies generated by pre-defined rules. The pre-defined rules encourage objects with a close distance and similar size to group first.

The results are shown in Table 2 and Table 3. It is clear that using our hierarchy outperforms all baselines on both datasets. The reason is that our method is able to detect frequently co-occurring object pairs and organize them with a meaningful representation. Another observation is that the baseline object detection method with no hierarchy is significantly inferior to the ones that incorporate some hierarchy. This demonstrates that hierarchical representation is useful in aggregating contextual information and helpful for object detection.

5. Conclusion

In this paper, we presented a method to learn how to group detected objects into a scene hierarchy. We showed that by using an one-class SVM, we can learn to predict the co-occurrence probabilities for pairs of objects and form a meaningful hierarchy. By incorporating the hierarchy into a novel RvNN-based model for object detection, we show that our hierarchy performs better than simpler baseline hierarchies. By leveraging hierarchical context, we are able to improve over just using PointNet features for object detection in 3D point clouds.

Acknowledgements

This project was done as part of a larger project, in collaboration with Yifei Shi and Angel Chang.

References

[1] A. Chang, A. Dai, T. Funkhouser, M. Halber, M. Nießner, M. Savva, S. Song, A. Zeng, and Y. Zhang. Matterport3D: Learning from RGB-D data in indoor environments. *arXiv preprint arXiv:1709.06158*, 2017.

[2] J. Li, K. Xu, S. Chaudhuri, E. Yumer, H. Zhang, and L. Guibas. Grass: Generative recursive autoencoders for shape structures. *ACM Transactions on Graphics (TOG)*, 36(4):52, 2017.

[3] T. Liu, S. Chaudhuri, V. G. Kim, Q. Huang, N. J. Mitra, and T. Funkhouser. Creating consistent scene graphs using a probabilistic grammar. *ACM Transactions on Graphics (TOG)*, 33(6):211, 2014.

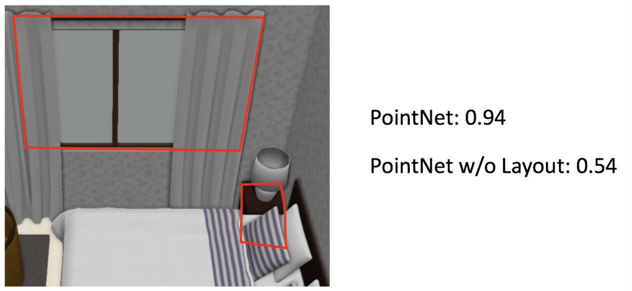
[4] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. PointNet: Deep learning on point sets for 3D classification and segmentation. *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 1(2):4, 2017.

[5] S. Song, F. Yu, A. Zeng, A. X. Chang, M. Savva, and T. Funkhouser. Semantic scene completion from a single depth image. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2017.

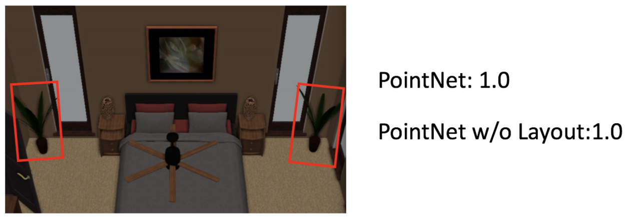
[6] M. Sung, H. Su, V. G. Kim, S. Chaudhuri, and L. Guibas. ComplementMe: weakly-supervised component suggestions for 3D modeling. *ACM Transactions on Graphics (TOG)*, 36(6):226, 2017.

A. PointNet + Layout vs. PointNet Score Comparison

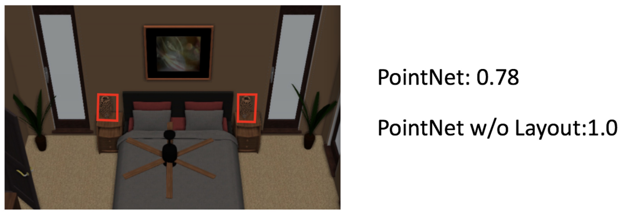
On the next page, there are some results comparing how specifically layout feature could give us a boost in evaluating co-occurrence scores.



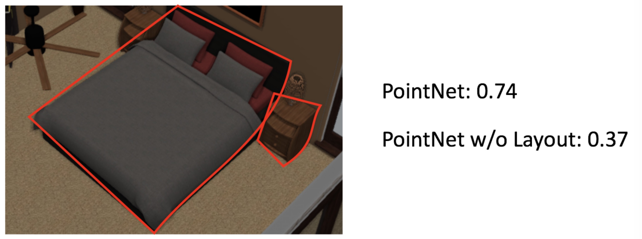
(a)



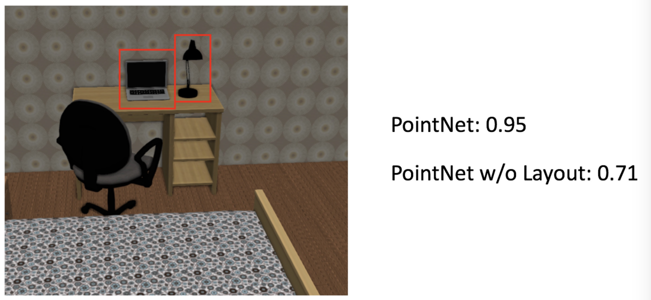
(b)



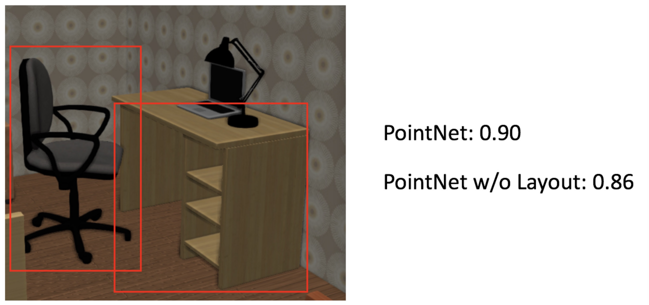
(c)



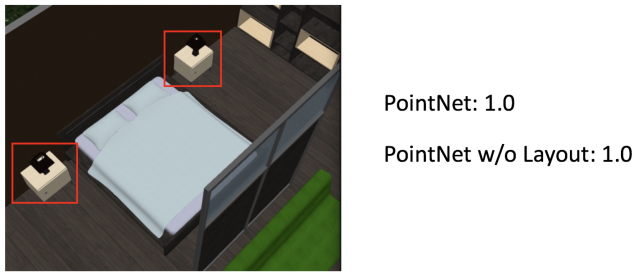
(d)



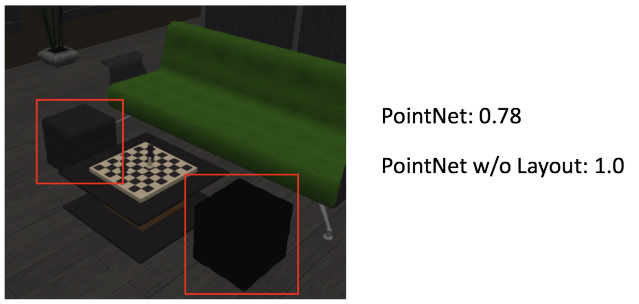
(e)



(f)



(g)



(h)

Figure 8: Visualization of predicted object co-occurrence probabilities with PointNet features