A. Supplementary Material

A.1. 360-image Completion Module

Network Architecture We use the same network architecture as in [49]. For simplicity, we remove the image warping process and the recurrent module.
Qualitative Results We show qualitative results in Figure 5.

A.2. 2D Layout Completion Module

Network Architecture The 2D layout module takes in partial scan and output its 2D layout completion. This module consists of two parts, feature extraction module and layout completion module. The feature extraction module is responsible to convert observed partial scans into a partial 2D feature grid. The layout completion module then takes in such partial feature grid and hallucinate a complete layout. The feature extraction module consist of two network works together. The first is a PointNet-style network that operate on point cloud input directly, the second network is a convolutional image backbone for which we use ResNet18 with ImageNet pre-trained weights. The feature from these two networks are combined together to assign a feature for each point. In order to project the 3D points into 2D grid, we simultaneously predict a floor plane (parameterized as plane equation) using max pooled per-point feature. Using the predicted floor plane equation, we can determine which 2D cell one 3D point should rest in. In order to accommodate the variation in height dimension, we further bin the height axis into 4 bins at height $h < -0.1m$, $-0.1m \leq h \leq 0.7m$, $0.7m \leq h \leq 1.5m$, $1.5m < h$ respectively. We average pool the features inside each bin to get a 2D feature map where only cell that has 3D points above it has feature (others have all zero feature vector). Then we pass this feature map through a convolutional encoder-decoder structure to get final 2D layout prediction. Please refer to Figure 10a for details.
Qualitative Results We show qualitative results in Figure 8.

A.3. Planar Patch Completion Module

Data Generation We estimate the ground truth plane using the whole room point cloud. We use RANSAC to estimate planes, and discard plane that has multiple disconnected components. We also leverage the semantic segmentation label available on point cloud to discard plane that joins two points from different categories. Examples of resulting plane estimation can be found at Figure 7. For each perspective image, we render an index map that tells which plane each pixel belongs to. The total planes we are predicting consist of two parts: planes that appears in the perspective image over certain threshold (500 pixels in our experiments), and structural planes that consist of walls, ceiling and floors.

Network Architecture The planar patch module firstly extracts partial point cloud derived from depth image and then randomly sampled 8192 points as the input. To integer more information, we separated our network into two branches. The first branch predicts the plane's center and normal showing in the partial point cloud. The second branch predicts the room's structural plane center and normal which might not show in the partial point cloud. In the first branch, we use the PointNet-style network to predict a point-wise output. Each point of output contains the predicted plane normal, relative distance to the plane center, feature vector, and semantic label. In the second branch, we use another PointNet-style network to directly predict the structural plane normal and center. This branch capture precise global information of room. Please refer to Figure 10b for details.
Qualitative Results We show qualitative results in Figure 6.

A.4. Local Relation Network

Network Architecture The input to our local relation network is 16384 points where the first 8192 points belongs to the source scan, and the later 8192 points belong to the target scan. We have 10 feature channels as input in total: we augment the xyz position with color, normal, and an 0,1 indicator indicates whether this point belongs to source or target. The input goes through a PointNet-style network to extract a per-point feature of dimension 1088. In order to provide the feature more local information, we also added an image branch similarly as in the 2D layout completion network to extract a 32 dimension convolutional per point feature. Then, using the sampled relations between source and target, we concatenate the corresponding feature vector (resulting a feature vector of 2*(1088+32) dimension) and output a relation prediction after 3 layers of fully connected layers. In addition to relation prediction, we also added per point semantic segmentation task in order to equip the feature with semantic information. We found adding the semantic prediction task increase the relation prediction's average accuracy by 5%. Please refer to Figure 10c for details.

Training details of local module. Randomly sampled pairs have unbalanced distribution of relations. In order to reduce the data in-balance, we manually force equal portion of relations during training. We found this strategy works well in practice.
Qualitative Results We show qualitative results in Figure 9.

A.5. Visualization of Feature Correspondences

We show the visualization of three representation's correspondences in Figure 11.

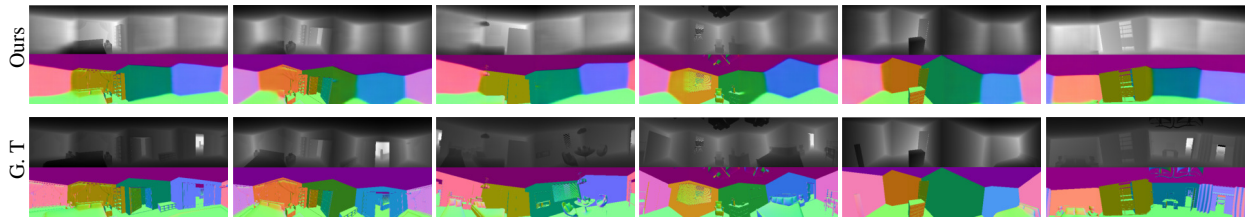


Figure 5: 360 image completion results. First row is completed normal/depth, second row is ground truth.

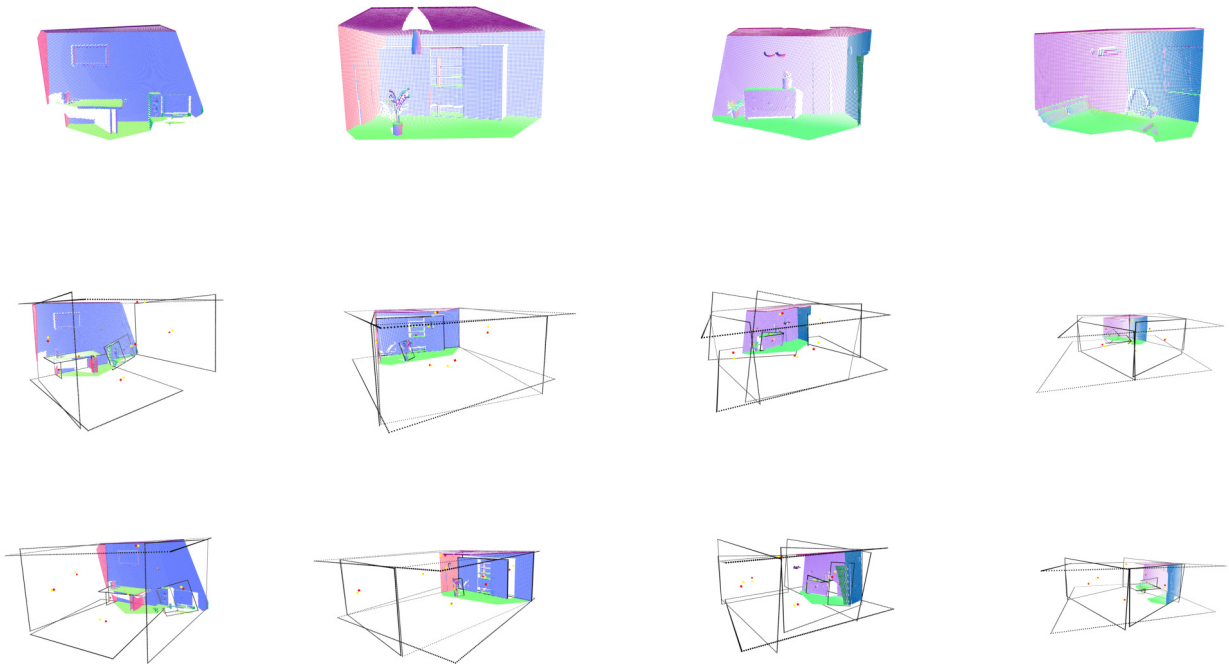


Figure 6: Planar patch completion results. Each column shows one example. The first row is the input point cloud of network. The second and third rows are two views of planar patch completion results. The red points in the figure show the ground truth of plane center and the yellow points show the prediction of our network.

A.6. Details of Spectral Matching Module

In this section, we provide the technical details on the definition of the consistency matrix used in spectral matching. Specifically, consider two feature correspondences (p_1, p_2) and (p'_1, p'_2). With $\boldsymbol{p}_i$, $\boldsymbol{n}_i$, and $\boldsymbol{d}_i$ we denote the position, the normal, and the descriptor of p_i, respectively. Likewise, with $\boldsymbol{p}'_i$, $\boldsymbol{n}'_i$, and $\boldsymbol{d}'_i$ we denote the position, the normal, and the descriptor of p'_i, respectively. For each pair (p_i, p'_i), we consider four geometric quantities that are invariant under rigid motions, i.e., one distance and three angles:

$$l_i = \|\boldsymbol{p}_i - \boldsymbol{p}'_i\|, \theta_{i,1} = \text{angle}(\boldsymbol{n}_i, \boldsymbol{n}'_i)$$

$$\theta_{i,2} = \text{angle}(\boldsymbol{n}_i, \frac{\boldsymbol{p}_i - \boldsymbol{p}'_i}{\|\boldsymbol{p}_i - \boldsymbol{p}'_i\|}), \theta_{i,3} = \text{angle}(\boldsymbol{n}'_i, \frac{\boldsymbol{p}_i - \boldsymbol{p}'_i}{\|\boldsymbol{p}_i - \boldsymbol{p}'_i\|}).$$

With this setup, we define the consistency score between $p_1p'_1$ and $p_2p'_2$ as

$$c(p_1p'_1, p_2p'_2) = \exp\Big(- \frac{\|\boldsymbol{d}_1 - \boldsymbol{d}'_1\|^2 + \|\boldsymbol{d}_2 - \boldsymbol{d}'_2\|^2}{2\gamma_1^2} - \frac{(l_1 - l_2)^2}{2\gamma_2^2} - \frac{(\theta_{1,2} - \theta_{1,1})^2}{2\gamma_3^2} - \frac{(\theta_{2,2} - \theta_{2,1})^2}{2\gamma_4^2} - \frac{(\theta_{3,2} - \theta_{3,1})^2}{2\gamma_5^2} \Big) \quad (9)$$

where $\gamma_i, 1 \leq i \leq 5$ are hyper-parameters to be learned from data.

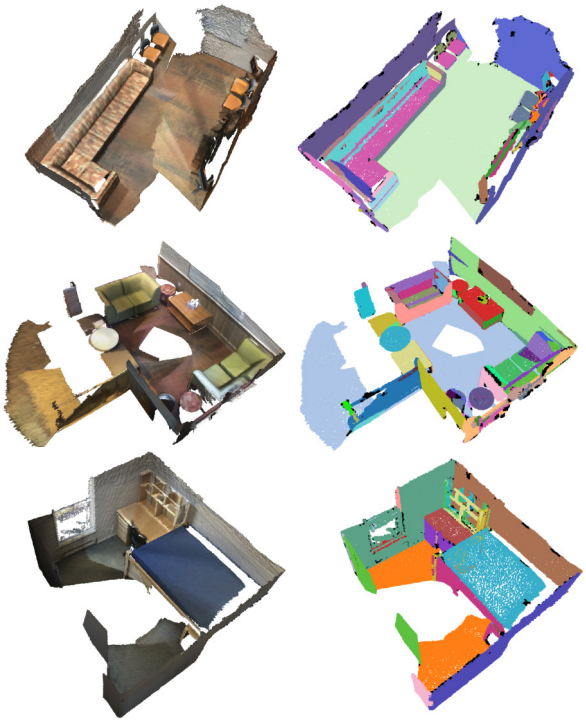


Figure 7: RANSAC plane fitting results. Left column is room point cloud, right column is plane fitting visualization. All points belongs to same plane are colored with the same unique color.

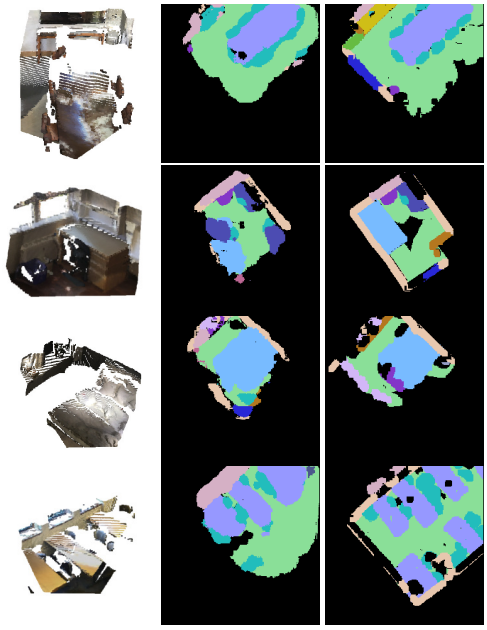


Figure 8: Layout completion results. The left column is the input scan, middle column is inferred layout completion. The right column is ground truth.

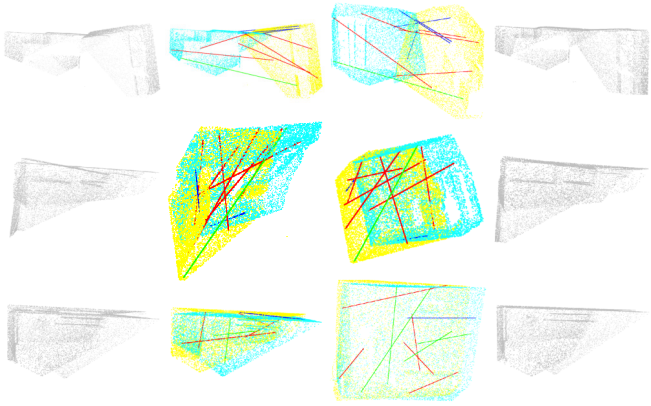


Figure 9: Local module results. The first column is the two input scans, where there is noticeable mis-alignment error. The second column is visualization of detected geometric relation. Blue: coplane, red: perpendicular, green: parallel. The last column is the output of local module.

B. Additional Technical Details of The Training Procedure

In this section, we provide additional technical details on the training procedure mentioned in the main paper.

Back-propagation through implicitly defined functions. Training l_4 requires us to compute the derivatives with respect to the optimal solution to an objective function. Here we represent a general formulation. Without losing generality, we assume we have an objective function $f(\boldsymbol{x}, \alpha)$, where α denotes its hyper-parameters and/or input parameters, and where $\boldsymbol{x}$ denotes the variables to be optimized. Consider the optimal solution

$$\boldsymbol{x}^\star(\alpha) := \underset{\boldsymbol{x}}{\operatorname{argmin}}\, f(\boldsymbol{x}, \alpha). \tag{10}$$

Our goal is to compute the derivatives of $\boldsymbol{x}^\star$ with respect to α. To this end, notice that $\boldsymbol{x}^\star(\alpha)$ is a critical point of f. This means

$$\frac{\partial f}{\partial \boldsymbol{x}}(\boldsymbol{x}^\star, \alpha) = 0. \tag{11}$$

Computing the derivatives of both sides of (11) to α. We arrive at

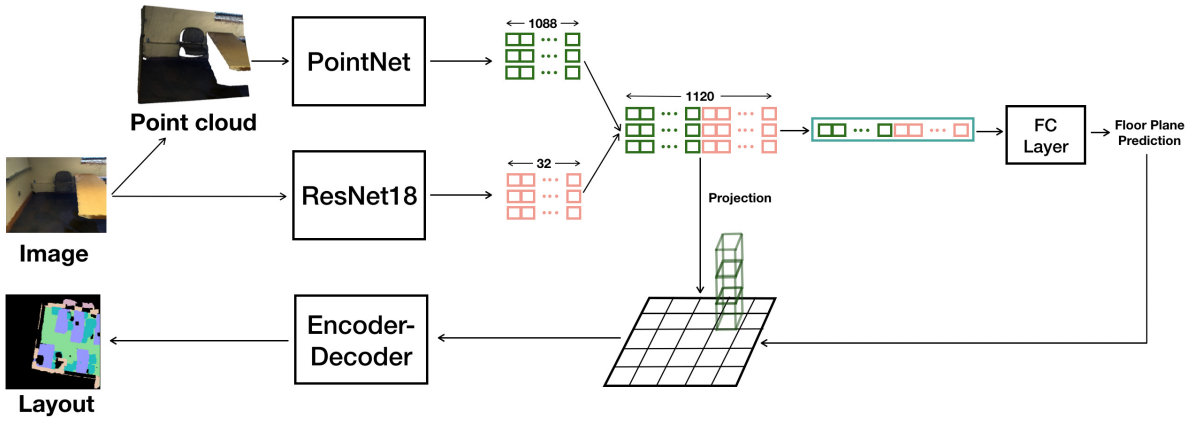
$$\frac{\partial^2 f}{\partial^2 \boldsymbol{x}}(\boldsymbol{x}^\star, \alpha) \cdot \frac{\partial \boldsymbol{x}^\star(\alpha)}{\partial \alpha} + \frac{\partial^2 f}{\partial \boldsymbol{x} \partial \alpha}(\boldsymbol{x}^\star, \alpha) = 0.$$

In other words

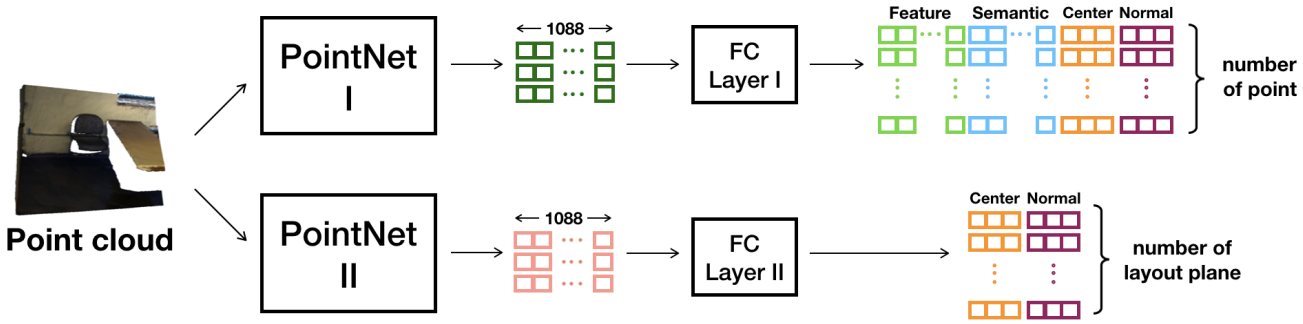
$$\frac{\partial \boldsymbol{x}^\star(\alpha)}{\partial \alpha} = -\left(\frac{\partial^2 f}{\partial^2 \boldsymbol{x}}\right) \cdot \frac{\partial^2 f}{\partial \boldsymbol{x} \alpha} \tag{12}$$

(12) allows us to optimize network parameters where the objective function involves $\boldsymbol{x}^\star$. Note that in this paper, $\boldsymbol{x}^\star \in \mathbb{R}^6$, so computing the second order derivatives are manageable.

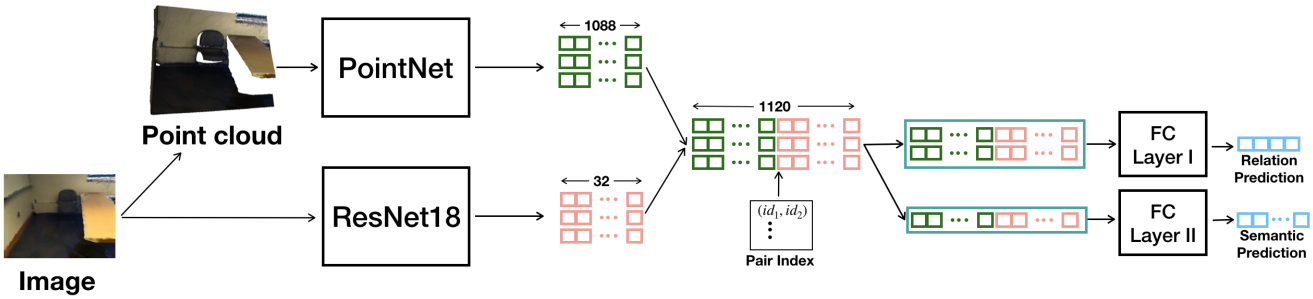
Training details.



(a) 2D layout completion module



(b) plane completion module



(c) local module

Figure 10: Network architectures for different modules.

360-image Completion Module We use the same setting as [49] without image warping process and recurrent module.

Planar Patch Completion Module We trained the first branch of network with 60 epochs and the second branch with 30 epochs. The learning rate of both networks is 0.0002. *2D Layout Completion Module/Local Module* We train 60 epoch with initial learning rate 0.0002.

C. Additional Technical Details of the Experimental Setup

Confusion matrix of the prediction module. Table 5 shows the confusion matrix of the prediction module. Note that although the prediction module does not deliver accurate predictions. However, since we maintain a broad set of point-pairs, the predicted pairs still contain suffi-

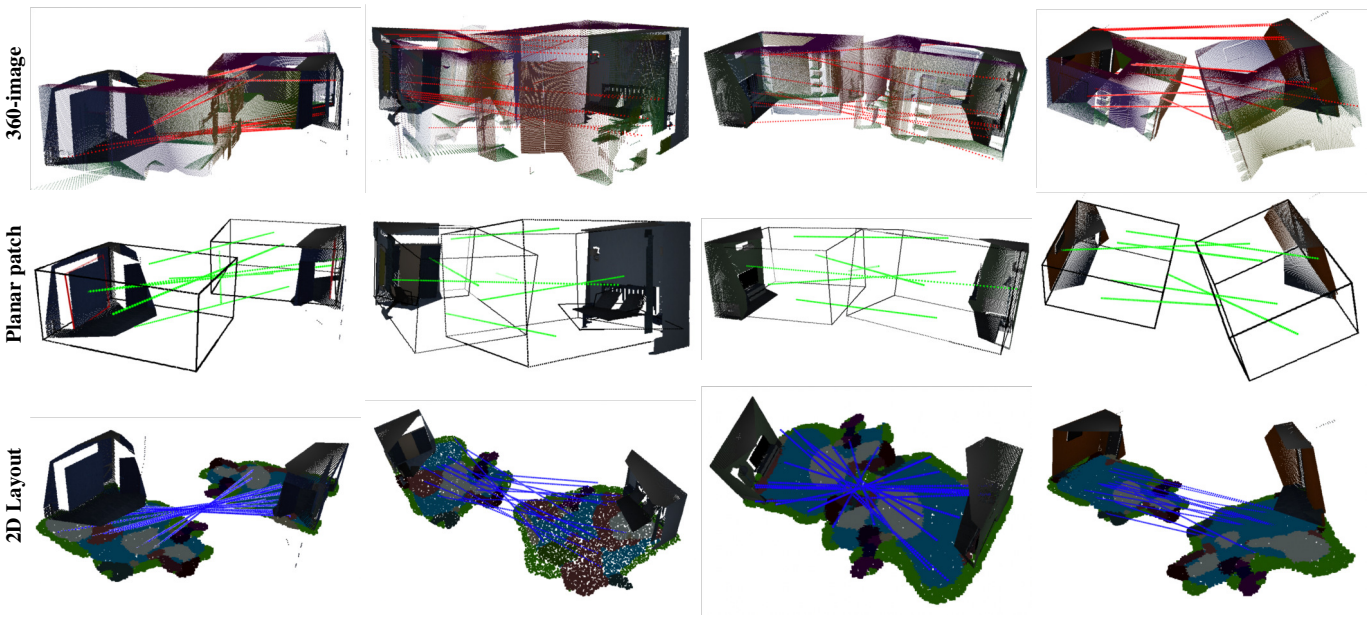


Figure 11: Visualization of correspondences on three representations. Each column contains one example. The first row shows 360-image representation. The second row shows planar patch. The third row shows 2D-layout.

	No-relation	Co-planar	Perpend.	Parallel
No-relation	0.806	0.831	0.077	0.034
Co-planar	0.667	0.307	0.019	0.007
Perpend.	0.568	0.064	0.314	0.054
Parallel	0.628	0.065	0.006	0.301

Table 5: Confusion matrix of the prediction module.

cient constraints for regressing the underlying relative pose. To extract correct relations, our approach utilizes robust reweighted non-linear squares and the fact that we have an initial pose to begin with, which also helps to prune wrong predictions, e.g., points with similar normals but are predicted to be perpendicular.