

SEG&STRUCT: The Interplay Between Part Segmentation and Structure Inference for 3D Shape Parsing — Supplementary Material

Jeonghyun Kim¹ Kaichun Mo² Minhyuk Sung^{1†} Woontack Woo^{1†}
¹KAIST ²Stanford University

In the supplementary material, we first introduce more details about the architecture design in our framework (Section A). Next, we demonstrate the implementation details and training setup for our framework (Section B). Finally, we provide more experimental results to evaluate the proposed architecture including ablation studies (Section C).

A. Architecture Design

A.1 Representation of Structure Hierarchy

The structure used in our framework has two kinds of relationships: \mathbf{H} and \mathbf{R} , where $\mathbf{H} \subset P^2$ describes the directed connectivity between a parent node and its sibling nodes and \mathbf{R} is a $M \times M$ matrix that describes the entire part relations between the nodes under the same parent node in the structure. In the following script, we use (m_i, m_j, τ) to describe an edge between m_i and m_j with a set of types of part relation $\tau \in \mathcal{T}$. The type of part relations \mathcal{T} covers translational, reflective, rotational symmetry, and adjacency, denoted as $\{\tau^{trs}, \tau^{ref}, \tau^{rot}, \tau^{adj}\}$. Same as StructureNet [6], we assume that a part-relation τ only exists between the nodes in the same subset of the tree and one pair of nodes can have multiple types of relations among \mathcal{T} . The part relation τ is encoded as one-hot vector. We do not predict transformation parameters for symmetry relations.

A.2 Parsing-based Structure Encoder

Structure Tree Construction. Given the shape point cloud $A \in \mathbb{R}^{N \times 3}$, we first parse the input into a set of part segments $B = \{b_l\}_{l \in L}$, where a part segment $b_l = (X_l, y_l)$ contains a set of points in the corresponding part region X_l and a semantic label y_l . By taking a point cloud of input shape A , our backbone ψ decomposes it into a set of leaf part instances. Based on the part segmentation output, we get a set of leaf nodes $\{m_l\}_{l \in L}$ for L number of the parts, which will be used for the structure tree construction following. Here, as mentioned in our paper, we encode each leaf part segment b_l into a 128-dimensional part feature vector $\mathbf{x}_l \in \mathbb{R}^{128}$. The feature is encoded by a part feature

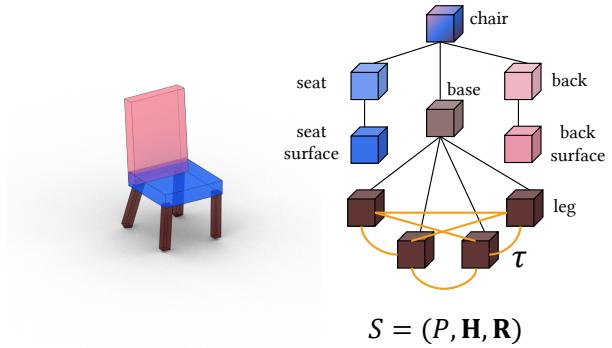


Figure 1. **Structure Hierarchy Representation.** We represent the part structure (left) in the tree hierarchy (right). For each subset in the tree, a parent node and sibling nodes are connected hierarchically in \mathbf{H} (black edges) and the sibling nodes have relationships τ in \mathbf{R} (orange edges). The part node in P comprises the oriented bounding box parameters θ and the semantic label.

encoder f_{part} , which uses *PointNet++* [8] with four set abstraction layers and two linear layers where each layer is followed by ReLU activation and batch normalization.

$$\mathbf{x}_l = f_{part}(X_l) \quad (1)$$

In the tree construction step, we denote a tree node as $\bar{m}_i = (\mathbf{x}_i, X_i, y_i)$ for $i \in M$ to distinguish it from the final part node $m_i = (\theta_i, y_i)$. Based on the part node extracted in the previous step, we operate node grouping recursively until it reaches to root node \bar{m}^{root} .

To group a subset of nodes at each level of the hierarchy, we take a heuristic approach based on their semantic labels and spatial distances. Since the semantic label y_i describes the level of the tree where the part node belongs to by itself as defined in StructureNet [6], we construct a structure tree based on them. For instance, a leaf part with a semantic label named *leg* guarantees that it has to be grouped to a *base* node (Fig. 1). We refer the readers to the StructureNet paper for more details about the part annotation for the hierarchy.

In general, the node grouping based on semantic prior works fine. For some cases where depending on the seman-

[†]Co-corresponding Authors.

tic information yields an ambiguity, we apply the simple heuristics based on the spatial information of the part points X_l . For example, *arms* in a *chair* can be separated according to the global center coordinates relative to the center of the input shape. Here, we can approximate the center of each node by its point cluster X_l by averaging their global coordinates.

Hierarchical Feature Encoding. Once the hierarchical connectivity \mathbf{H} is established, we calculate the encode a parent node’s feature vector \mathbf{x}_r by aggregating the features of its sibling nodes $\{\mathbf{x}_c\}_{c \in C_r}$, where C_r is the number of the sibling nodes. For the parent feature encoding, we use a feature encoder f_{child} with one single linear layer that takes a set of features from child nodes::

$$\mathbf{x}_r = f_{child}(\{\mathbf{x}_c\}) \quad (2)$$

In this process, we encourage the parent features to have a broader view of the subset’s sibling nodes. We recursively iterate this process until it approaches to the root feature \mathbf{x}^{root} . Finally, we treat this root feature as an *global context* throughout the entire structure, both in the decoding step and merge prediction network later. As aggregate the node features, we also gather the part point cloud to define a part region for each parent node.

After the construction step, we get all the part nodes and hierarchical connectivity (\bar{P}, \mathbf{H}) , where the part nodes $\{\bar{m}_i\} \in \bar{P}$ involve the global context. We further demonstrate how can we take advantage of these context features and the hierarchy prior to support the structure decoding step in the following.

A.3 Multi-level Context-Aware Structure Decoding

In this section, we introduce the part structure decoder \mathcal{G} using multi-level context across the structure hierarchy in *top-down* manner. The goal of our part structure decoding is two-fold: regressing the part bounding box θ_i and classifying the part relationship among the nodes (m_i, m_j, τ) . Though it is possible to directly decode them directly from segmentation output, we found this yields an inconsistent aligned part structure. This is mainly due to the nature of 3D shape, the parts in the structure are strongly co-related not only horizontally, but also the vertical (hierarchical) way.

To tackle this, we propose a hierarchical message passing g_h , to learn the structural context based on the *hierarchy prior*. Our previous segmentation-based structure construction enables us to map the part regions in the geometry space into the hierarchical representation. Based on this, we build an association between part segments and part nodes in the form of a *skip connection* connecting the corresponding parts (Fig 2). By fully exploiting both hierarchical connectivity and the association from the skip connection, we perform two types of message passing: 1) global context learning and 2) local part relation learning.

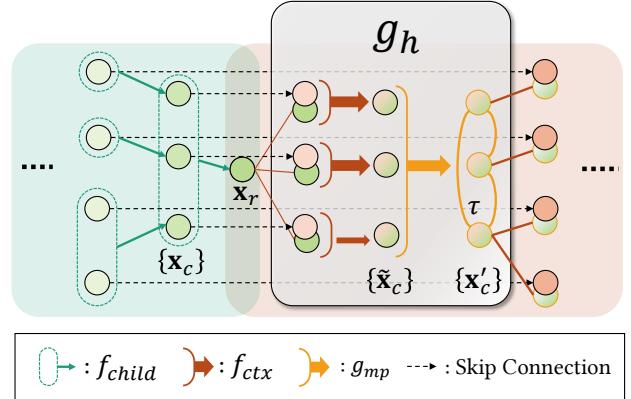


Figure 2. **The illustration of Hierarchical Message Passing.** Given the constructed hierarchy and all the features encoded using f_{child} , we perform hierarchical message passing g_h to make features learn a multi-level context in the structure. In global context learning f_{ctx} , the parent feature \mathbf{x}_r and the sibling feature connected via a skip connection \mathbf{x}_c are aggregated. Then, the local part message passing g_{mp} classifies a part relation τ and performs inter-part communication between connected sibling nodes. The multi-level context-aware node feature \mathbf{x}'_c is again used for the recursive message passing to its sibling subset.

Global Context Learning. The main purpose of this step is to make our part features aware of the global context across the constructed hierarchy in *vertical* way. Here, the feature vector of each node \mathbf{x}_i is updated through parent feature aggregation before learning inter-part relationships. This ensures that the part feature not only contains the local relationship between neighboring part nodes but also covers the global context across the different levels of structures. Starting from the root node feature \mathbf{x}^{root} , we propagate each level of global context by decoding it down to the sibling nodes.

To do this, we use \mathbf{x}_c using f_{ctx} , a single linear layer that consumes a concatenated feature vector of child and parent nodes $[\mathbf{x}_c; \mathbf{x}_r] \in \mathbb{R}^{256}$:

$$\tilde{\mathbf{x}}_c = f_{ctx}([\mathbf{x}_c; \mathbf{x}_r]) \quad (3)$$

Here, we consider $\tilde{\mathbf{x}}_c$ as a global context-aware feature, which supports the part-relation learning and the bounding box decoding further.

Local Part Relation Learning. After aggregating the global context to each part node, we perform a *local* message passing to learn an inter-part communication for each subset *horizontally*.

Inspired by StructureNet [6], we aim to make the part nodes learning predefined local part relations, i.e. symmetry and adjacency by classifying the accurate type of part relation $\tau \in \mathcal{T}$. For \mathcal{T} , we consider three types of symmetry $\{\tau^{trs}, \tau^{rot}, \tau^{ref}\}$ and the adjacency τ^{adj} as mentioned earlier. With the accurately predicted part relations, the co-

related part features can achieve more consistent part geometry prediction further, e.g. preserving a *reflective* symmetry between *arms* in a *chair*. Analogous to StructureNet, we connect the whole part nodes inside the subset to each other and compute a set of corresponding edge feature vectors for each pair of nodes through the message passing. To compute the edge feature $\mathbf{y}_{ij} \in \mathbb{R}^{256}$, we use g_{edge} , a two-layer multi-layer perceptron that takes two context-aware feature vectors $\{\tilde{\mathbf{x}}_i, \tilde{\mathbf{x}}_j\}$:

$$\mathbf{y}_{ij} = g_{edge}([\tilde{\mathbf{x}}_i; \tilde{\mathbf{x}}_j]) \quad (4)$$

Then we classify which types of relationships exist between them among the part-relations \mathcal{T} for the number of types $|\mathcal{T}|$ given the edge feature \mathbf{y}_{ij} .

$$(p_{(\bar{m}_i, \bar{m}_j, \tau_1)}, \dots, p_{(\bar{m}_i, \bar{m}_j, \tau_{|\mathcal{T}|})}) = \sigma(g_{\tau}(\mathbf{y}_{ij})) \quad (5)$$

where σ and p are a sigmoid and a probability function, respectively. Note that we do not predict any parameters for each relation here. Here, we treat edge prediction as a classification problem where we predict the probability of edge types $\in [0, 1]$.

Finally, we perform an inter-part message passing, adopted from StructureNet [6]. We prune out the edges with a probability lower than the edge threshold, i.e. 0.5, to keep a valid set of edges. Next, we perform two iterations of message passing for each part feature with the connected neighbors in the subset, using the edge features with the classified relationships. Lastly, we aggregate the features from each iteration to get the finalized node feature \mathbf{x}'_i from a set of connected parts $\{\bar{m}_k\}_{k \in K_i}$ where K_i is the number of parts connected. We refer the readers to the paper of StructureNet [6] for more details about the message-passing network.

Through hierarchical message passing across the subsets and the overall structure, we encourage the final part representation \mathbf{x}'_i to incorporate much broader context across the structure beyond a geometry-dependent feature vector.

Bounding Box Decoding. Finally, we regress the bounding box parameters $\theta_i = (\mathbf{t}_i, \mathbf{s}_i, q_i)$ for each part node in the hierarchy. Here, we use g_{box} built with a series of linear layers to output the bounding box parameters. Empirically, we observe that the decoder often fails to perform accurate regression if it works on the feature vector alone. To address this, we utilize the part points X_i for each node to support the box decoding, which we gathered during the tree construction. For the translation vector \mathbf{t}_i , we first compute the rough center coordinates $\hat{\mathbf{t}}_i$ by averaging the global coordinates of X_i . The decoder g_{box} predicts the offset vector $\mathbf{o}_i \in \mathbb{R}^3$ to slightly tune the center coordinate as $\mathbf{t}_i = \hat{\mathbf{t}}_i + \mathbf{o}_i$. By doing so, we can achieve a much precise estimation of box parameters, not only the box center but the others as well. Similarly, we have seen that it is

enough to approximate the scaling vector \mathbf{s}_i by examining the volume of part points along the rotation axis prediction done for unit quaternion q . Our final formulation for box parameter decoding is as follows:

$$(\mathbf{t}_i, \mathbf{s}_i, q_i) = g_{box}(\mathbf{x}'_i, X_i) \quad (6)$$

Finally, we get a part node $m_i = (\theta_i, y_i)$ for all the nodes in the structure and complete the part structure inference.

A.4 Training and Losses.

Structure Inference Network. The parameters for the encoder \mathcal{F} and the decoder \mathcal{G} are supervised both at the structure decoding step through backpropagation. As mentioned in our paper, our overall loss design for structure inference network is mostly brought from StructureNet [6] since ours and StructureNet share the equivalent objective, inferring the structure output. Similar to StructureNet, our total loss \mathcal{L}_{total} is computed by the following steps. First, a correspondence map $\mathbf{M} \subset P \times \hat{P}$ between the predicted parts \hat{P} and the target parts P is established to supervise network parameters based on the geometric distance between the parts using the hungarian algorithm [4]. Second, the geometry reconstruction error is computed by summing up \mathcal{L}_{box} and \mathcal{L}_{norm} , a bounding box fitting distance and an additional normal error for the precise orientation regression, respectively. Third, we compute an edge prediction loss \mathcal{L}_{edge} , where we consider the edge prediction as a binary classification problem, and the classifier is supervised using cross-entropy loss.

Finally, we calculate a structure consistency loss \mathcal{L}_{cons} to make the relationship at parent nodes consistent with their sibling subsets. For more detail, we refer the readers to the StructureNet paper [6]. In summary, we compute our total loss as follows:

$$\mathcal{L}_{total} = \lambda_1 \mathcal{L}_{box} + \lambda_2 \mathcal{L}_{norm} + \mathcal{L}_{edge} + \mathcal{L}_{cons} \quad (7)$$

where we define the coefficients λ_1 and λ_2 as 20 and 10, respectively.

Segmentation Refinement Network. To train part segmentation refinement network \mathcal{M} , we use focal loss [5] as discussed in our main paper. We define the focal loss F taking a probability score $p_s \in [0, 1]$ and a target label p :

$$F(p_s, p) = -\alpha_t (1 - p_t)^\gamma \log(p_t) \quad (8)$$

where p_t is p_s if $p = 1$ and p_t is $1 - p_s$ otherwise. α_t is a weight and γ is a focusing parameter. Empirically, we opt to set α as 0.15 and γ as 2. The merge loss \mathcal{L}_{merge} is then calculated with ground-truth assignment $\bar{\mathbf{C}}$ as follows:

$$\mathcal{L}_{merge} = \sum_{(i, j) \in \bar{\mathbf{C}}} F(p_{(m_i, m_j)}^{merge}, \mathbb{1}_{\bar{\mathbf{C}}}(i, j)) \quad (9)$$

where $\mathbb{1}_{\bar{\mathbf{C}}}$ is an indicator function that gives 1 for the valid candidate pair in $\bar{\mathbf{C}}$ and the others 0.

Table 1. **Comparison on Structure Inference.** Same as our paper, AP means part prediction accuracy (%) computed by average precision with IoU threshold 0.25, and EE means edge prediction error calculated by one minus F1-score of edge prediction outputs. Please note that the second and third baselines do not measure EE since they do not predict any part relationships. The columns for key components describe which prior knowledge or the level of message passing each method takes. The bold text means the best results for each column.

Id	Method	Key Components			Categories						Avg			
		Prior		Message Passing		Chair		Table		Storagefurn.				
		Seg.	Hier.	Skip.	Local	global	AP (%)	EE (↓)	AP (%)	EE (↓)	AP (%)	EE (↓)		
1	$\mathcal{F}_s + \mathcal{G}_{SN}$		✓		✓	✓	5.03	0.6824	2.02	0.8272	1.07	0.6491	2.71	0.7196
2	$\psi + \text{PCA}$	✓					37.32	-	20.96	-	17.75	-	25.34	-
3	$\psi + g_{box}$	✓					46.66	-	25.89	-	19.96	-	30.83	-
4	$\psi + g_{box} + g_{mp}$	✓			✓		48.39	0.8576	24.19	0.8835	19.96	0.8933	30.85	0.8781
5	$\mathcal{F} + \mathcal{G}_{SN}$	✓	✓		✓	✓	10.79	0.4211	1.28	0.7863	1.95	0.5191	4.68	0.5755
6	$\mathcal{F} + \mathcal{G} - f_{ctx}$	✓	✓	✓		✓	47.21	0.3006	22.91	0.4597	19.25	0.6664	29.79	0.4756
7	$\mathcal{F} + \mathcal{G} - g_{mp}$	✓	✓	✓	✓	✓	47.34	0.3448	26.41	0.5024	21.20	0.6867	31.65	0.5113
8	$\mathcal{F} + \mathcal{G}$ (Ours)	✓	✓	✓	✓	✓	48.41	0.2727	26.36	0.4400	21.57	0.6934	32.11	0.4687

B. Implementation Details

We implement our framework in PyTorch [7]. The training for each category is performed until convergence with batch size 16, mostly requiring 1-2 days for structure inference network and less than two hours for segmentation refinement network on a single GeForce RTX 3090 and an Intel Xeon Silver 4210R CPU. We use the Adam optimizer [3] with the initial learning rate as 0.5^{-3} for inference network and 10^{-4} for refinement network, decayed by 0.8 per 500 steps.

C. More Experimental Results

C.1 Ablation Study on Structure Inference Network

To demonstrate the effectiveness of the key components in our proposed hierarchical message passing g_h , we perform sets of ablation studies. For other methods not explained here, we refer readers to our main paper.

Baselines. Here, we built another type of baseline, which directly decodes the bounding box from extracted part segments without having hierarchical priors or key components used in our method. Based on our backbone ψ and the part feature encoder f_{part} , the box decoder baseline parses the input shape into leaf part instances, encodes part features, and directly predicts bounding box parameters $\{\theta_l\}_{l \in L}$. Here, we prepare three kinds of box decoder baselines (2nd – 4th rows in Table 1) directly consuming the output from backbone ψ : a PCA-based bounding box estimator, a box decoder g_{box} , and the box decoder with message passing network g_{mp} . Since these box encoders do not predict and learn any part relations except the last one, the edge prediction error is not measured.

Note that the last one uses the reduced version of our part-relation learning where the relationships are learned across all the leaf instances, which makes the edge prediction a lot more difficult. Then, we show how the level of feature updates in the local part-relation learning and global context learning affects the performance of structure infer-

ence for each. To this end, we built baselines by subtracting each component of g_h from our framework $\mathcal{F} + \mathcal{G}$, represented as the one without local part message passing g_{mp} and global context encoding f_{ctx} .

Results. We observe that our method based on hierarchical message passing beats the other baselines quantitatively in Table 1. For the baselines with different levels of structural context, the mean part prediction accuracy also increases. We find that learning global context even helps to improve the local part relation classification on average, which supports the necessity of our hierarchical message passing. While the box decoder with leaf parts message passing shows compatible part prediction accuracy to ours, it extremely suffers to predict the precise part relationships with the highest edge prediction error.

We illustrate how this negatively affects to the prediction of the globally consistent structure in Figure 3. As we gradually added the key component of our method, the leaf parts are coherently arranged by learning hierarchical relationships in the structure (from left to right). Despite the precise quality of the part segmentation output, the compared baselines fail to capture co-relations between parts across the structure. For the box decoder baselines in the third and the fourth column, some parts pop out and degrade the overall assembly quality. Even with part relation learning, the *symmetry* between nodes is easily corrupted (see the *chair* cases). The cases of shapes with a cluttered set of parts (see the *table* cases) get much severe where the *adjacency* between parts is broken resulting in the scattered output.

For the baselines subtracting the key component of hierarchical message passing, we also found that incorporating all the context information achieves the most plausible results. Although the edge prediction quality seems to have a relatively small margin in numbers (Table. 1), we observe there is a more clear improvement in visuals. While the symmetry between parts is preserved better than the box decoder baselines, we observe that missing one of the structural contexts still yields flawed prediction with

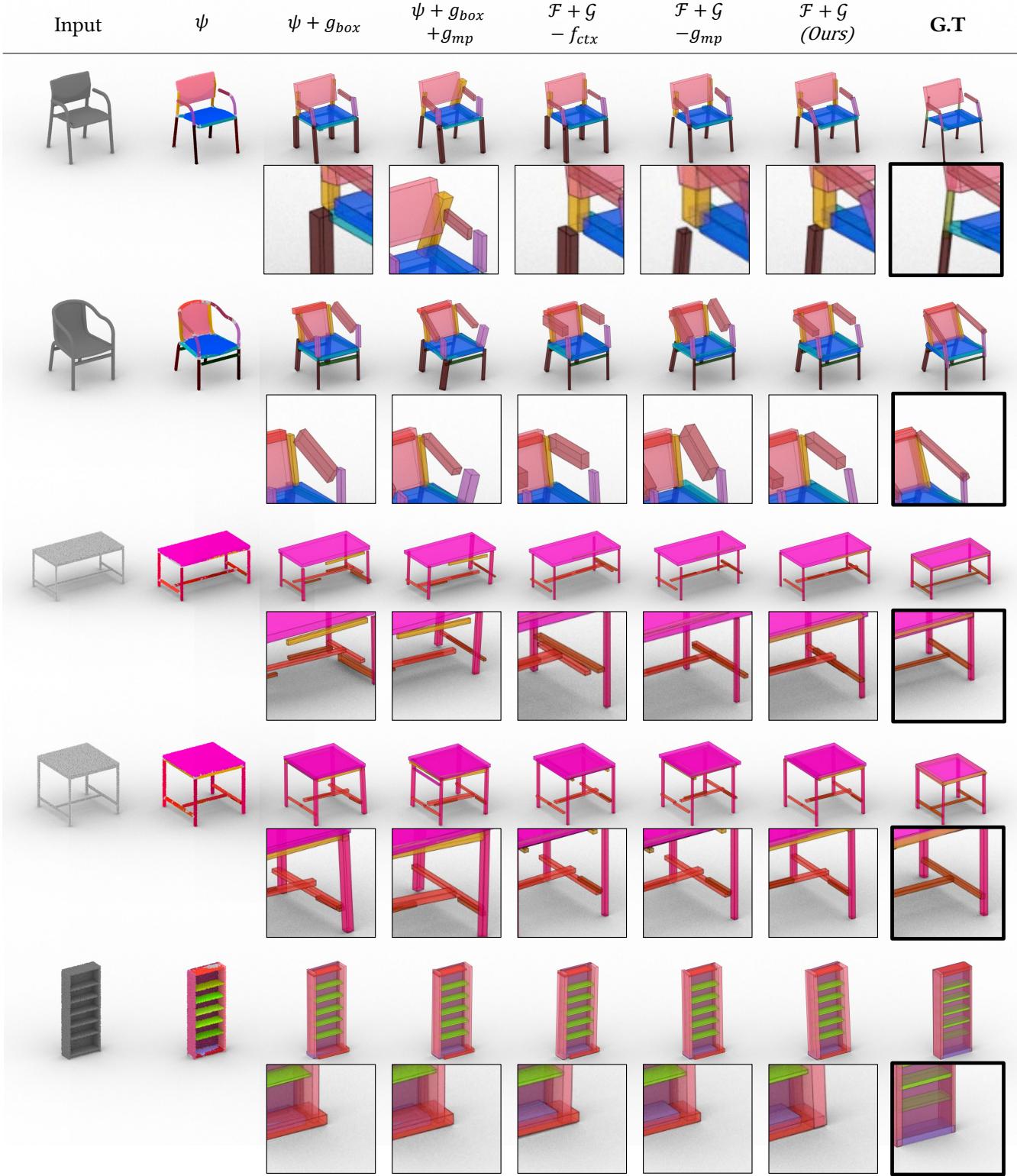


Figure 3. **The illustration of Ablation Study on Hierarchical Message Passing.** The zoom-in images for each case describe the continuous improvements of the structure output more clearly. As our key components for hierarchical message passing are applied, the overall arrangement of part structure improves gradually.

Table 2. **Comparison to Unsupervised Method.** Same as our paper, we use mean average precision (mAP) with an IoU threshold 0.5. Note that the second column indicates which kind of supervision is used for each method.

Method	Supervision	Chair
BSP-Net [1]	None (Unsup.)	19.91
PointGroup [2]	Segmentation	40.70
SEG&STRUCT (Ours)	Segmentation, Structure	40.81

broken adjacency. As each context is added, we observe the part structure accomplish *globally-aligned* arrangement even with the cluttered set of parts. For example, in the first chair case, the adjacency and alignment between a leg (brown box) and a vertical frame (yellow box) become more consistent.

C.2 Comparison to Unsupervised Part Segmentation Method

To demonstrate the impact of the supervision used in our framework, we compare ours with BSP-Net [1], an unsupervised method that parses 3D shapes into a set of volumetric primitives. Since BSP-Net abstracts a raw geometry to produce *super-segments*, rather than part bounding boxes, we evaluate the performance on part segmentation task only.

As BSP-Net does not predict any semantics for primitives, we assign semantics per points using ground-truth semantic labels following the original paper [1] and treat predicted primitives as part instances. For evaluation, we use mean average precision (mAP) with an IoU threshold 0.5, as same as our paper. In Table 2, we demonstrate the result of the quantitative evaluation for part segmentation in *Chair* category. Ours clearly outperforms the unsupervised method, while fully exploiting the supervision of part segmentation and structure both.

C.3 Limitation and Discussion

We observe that our segmentation refinement method suffers from the imperfect supervision given by the noisy annotations. For similar shapes, there are noisy annotations that make our network hard to predict the correct merge operation. In Figure 4, we illustrate these failure cases caused by the noisy annotations. For example, given almost the same shapes in *Chair* category, the leg part (dark brown) is hard to be distinguished from the foot part (orange) even for the annotators. When these confusing labels are found in the part boundaries, the network cannot clearly describe which part is falsely segmented and also examine the direction of the merge process correctly. Unfortunately, we found these failure cases also happen across the other categories. Since our framework solves a supervised problem for the part segmentation, these noisy annotations largely

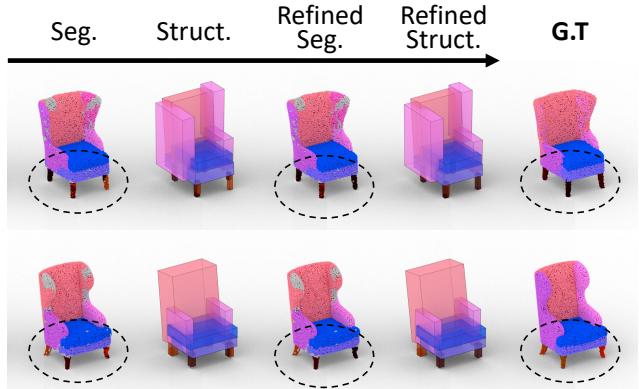


Figure 4. **Failure Cases.** Both shapes in two rows share the same Chair category. However, noisy labeling on the same region by human annotators, the orange color for *foot* part and the dark brown color for *leg* part, confuses the network to predict correct merge operations and yields false outputs.

restrain the improvement on the structure-to-segmentation refinement task and the further refinement on the part structures.

C.4 More Qualitative Results

In this section, we provide more visuals of the qualitative evaluation on both tasks: *Segmentation-to-Structure Inference* and *Structure-to-Segmentation Refinement*. As mentioned in our paper, we test our method on the three largest categories from PartNet (e.g., chair, table, and storage furniture), and report the results for each category in the following figures.

For the first three figures (Figure 5, 6, 7), we illustrate more results of segmentation-to-structure inference network per category, comparing our method to other baselines discussed in our paper. Next, in Figure 8, 9, 10, we depict the results of structure-to-segmentation refinement including the updated structure from the refinement output. **Please see the following pages for more qualitative results.**

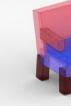
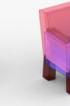
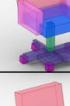
Input								
ψ								
$\mathcal{F}_s + \mathcal{G}_{SN}$								
$\mathcal{F} + \mathcal{G}_{SN}$								
$\mathcal{F} + \mathcal{G}$ (Ours)								
G.T								
Input								
ψ								
$\mathcal{F}_s + \mathcal{G}_{SN}$								
$\mathcal{F} + \mathcal{G}_{SN}$								
$\mathcal{F} + \mathcal{G}$ (Ours)								
G.T								

Figure 5. More Results of Structure Inference: Chair

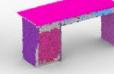
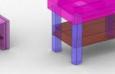
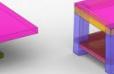
Input								
ψ								
$\mathcal{F}_s + \mathcal{G}_{SN}$								
$\mathcal{F} + \mathcal{G}_{SN}$								
$\mathcal{F} + \mathcal{G}$ (Ours)								
G.T								
Input								
ψ								
$\mathcal{F}_s + \mathcal{G}_{SN}$								
$\mathcal{F} + \mathcal{G}_{SN}$								
$\mathcal{F} + \mathcal{G}$ (Ours)								
G.T								

Figure 6. More Results of Structure Inference: Table

Input								
ψ								
$\mathcal{F}_s + \mathcal{G}_{SN}$								
$\mathcal{F} + \mathcal{G}_{SN}$								
$\mathcal{F} + \mathcal{G}$ (Ours)								
G.T								
Input								
ψ								
$\mathcal{F}_s + \mathcal{G}_{SN}$								
$\mathcal{F} + \mathcal{G}_{SN}$								
$\mathcal{F} + \mathcal{G}$ (Ours)								
G.T								

Figure 7. More Results of Structure Inference: Storagefurniture

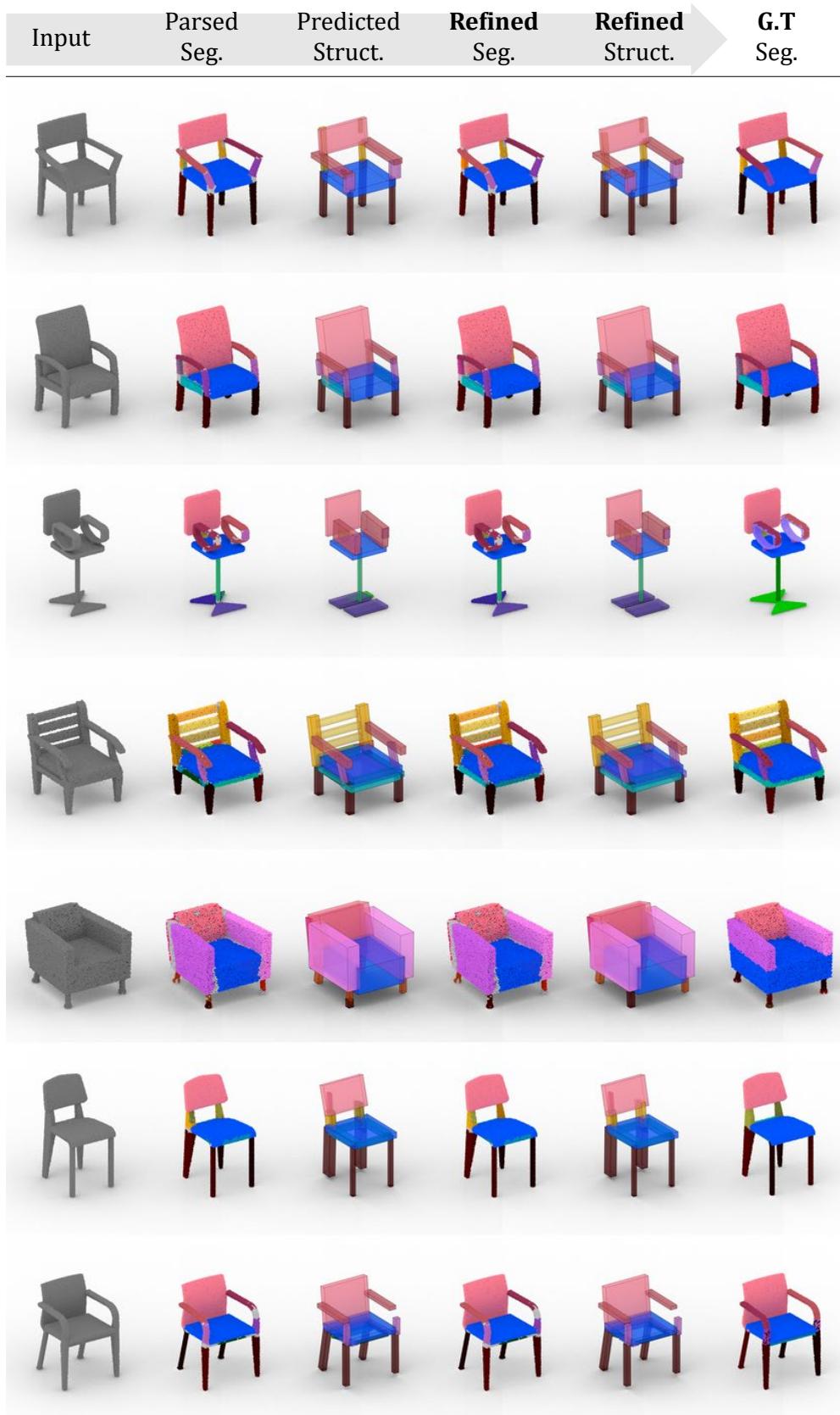


Figure 8. More Results of The Interplay between Part Segmentation and Structure Inference: Chair



Figure 9. More Results of The Interplay between Part Segmentation and Structure Inference: Table

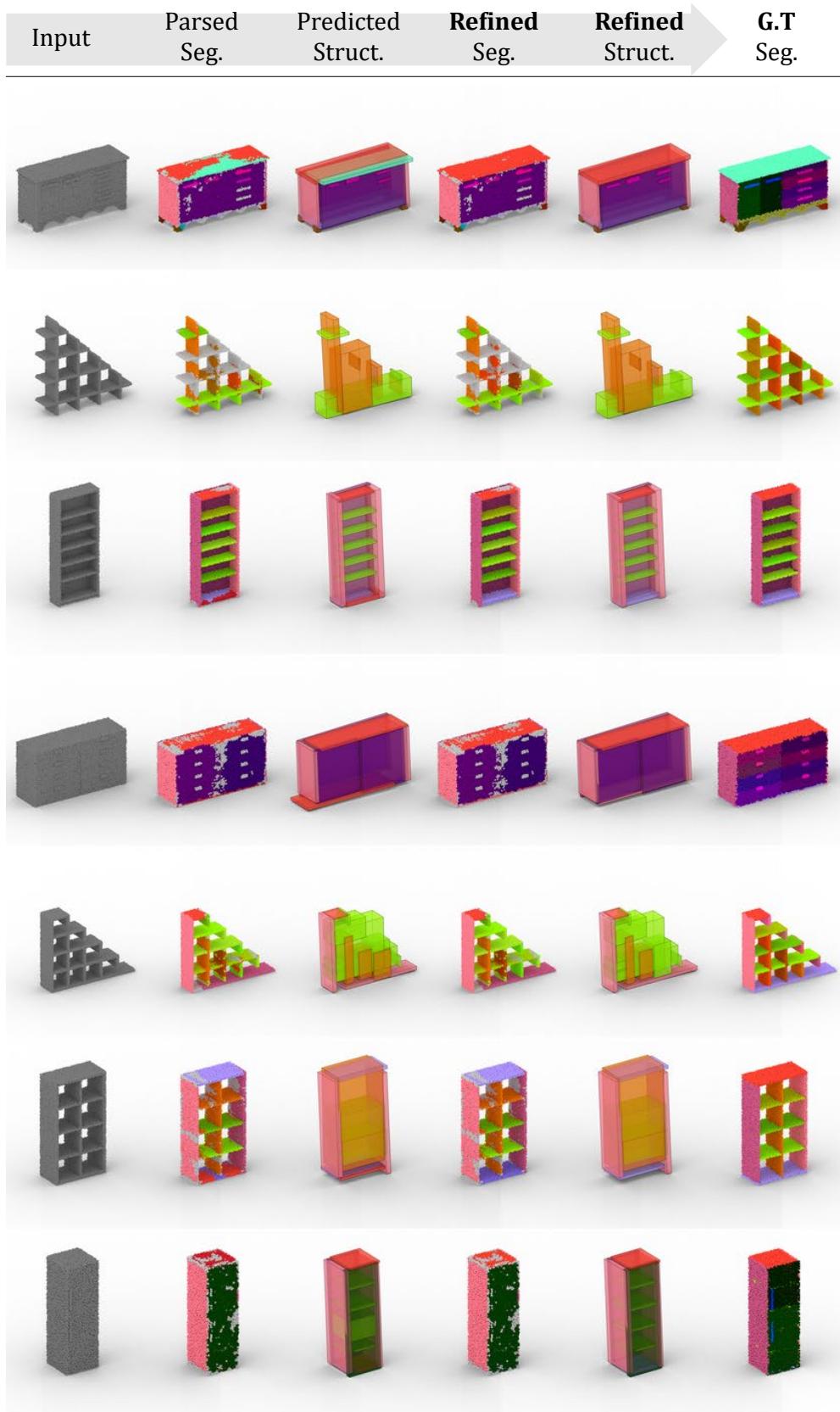


Figure 10. More Results of The Interplay between Part Segmentation and Structure Inference: Storagefurniture

References

- [1] Zhiqin Chen, Andrea Tagliasacchi, and Hao Zhang. BSP-Net: Generating compact meshes via binary space partitioning. In *CVPR*, 2020.
- [2] Li Jiang, Hengshuang Zhao, Shaoshuai Shi, Shu Liu, Chi-Wing Fu, and Jiaya Jia. PointGroup: Dual-set point grouping for 3D instance segmentation. In *CVPR*, 2020.
- [3] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [4] Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 1955.
- [5] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *ICCV*, 2017.
- [6] Kaichun Mo, Paul Guerrero, Li Yi, Hao Su, Peter Wonka, Niloy J. Mitra, and Leonidas J. Guibas. StructureNet: Hierarchical graph networks for 3D shape generation. *ACM TOG*, 2019.
- [7] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.
- [8] Charles R Qi, Li Yi, Hao Su, and Leonidas J Guibas. PointNet++: Deep hierarchical feature learning on point sets in a metric space. In *CVPR*, 2017.