PartNet: A Large-scale Benchmark for Fine-grained and Hierarchical Part-level 3D Object Understanding Supplementary Material

Kaichun Mo[1] Shilin Zhu[2] Angel X. Chang[3] Li Yi[1] Subarna Tripathi[4] Leonidas J. Guibas[1] Hao Su[2]
[1]Stanford University [2]University of California San Diego [3]Simon Fraser University [4]Intel AI Lab
https://cs.stanford.edu/~kaichun/partnet/

A. Overview

This document provides additional dataset visualization and statistics (Sec B), hierarchical template design details and visualization (Sec C), and the architectures and training details for the three shape segmentation tasks (Sec D), to the main paper.

B. More Dataset Visualization and Statistics

We present more visualization and statistics over the proposed PartNet dataset.

B.1. More Fine-grained Segmentation Visualization

Figure 5 and 6 show more visualization for fine-grained instance-level segmentation annotations in PartNet. We observe the complexity of the annotated segmentation and the heterogeneous variation of shapes within each object category.

B.2. More Hierarchical Segmentation Visualization

Figure 7, 8 and 9 show more visualization for example hierarchical instance-level segmentation annotations in PartNet. We visualize the tree-structure of the hierarchical segmentation annotation with the 2D part renderings associated to the tree nodes.

B.3. Shape Statistics

We report the statistics for the number of annotations, unique shapes and shapes that we collect multiple human annotations in Figure 1.

B.4. Part Statistics

We report the statistics for the number of part semantics for each object category in Figure 2. We also present the statistics for the maximum and median number of part instances per shape for each object category in Figure 3. We report the statistics for the maximum and median tree depth for each object category in Figure 4.

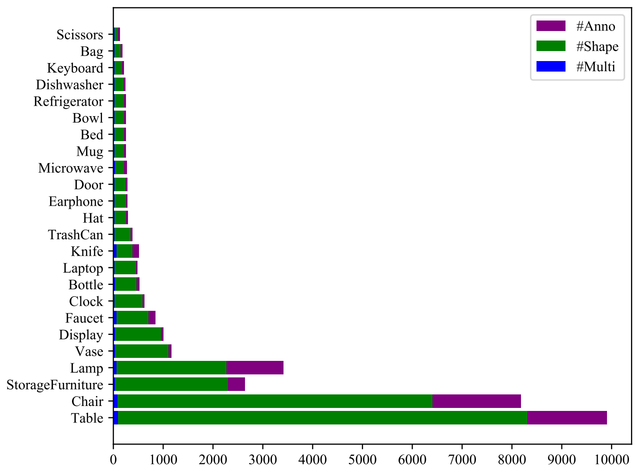

Figure 1. **PartNet shape statistics.** We report the statistics for the number of annotations, unique shapes and shapes that we collect multiple human annotations.

C. More Template Design Details and Visualization

We provide more details and visualization for the expert-defined hierarchical templates to guide the hierarchical segmentation annotation and the template refinement procedure to resolve annotation inconsistencies.

C.1. Template Design Details

We design templates according to the rules of thumb that we describe in the main paper. We also consulted many online references[1] that describe object parts (often for manufacturing and assembly) and previous works that relate language to the shapes [2] as guides for the design of our

[1]*E.g.* http://www.props.eric-hart.com/resources/parts-of-a-chair/.

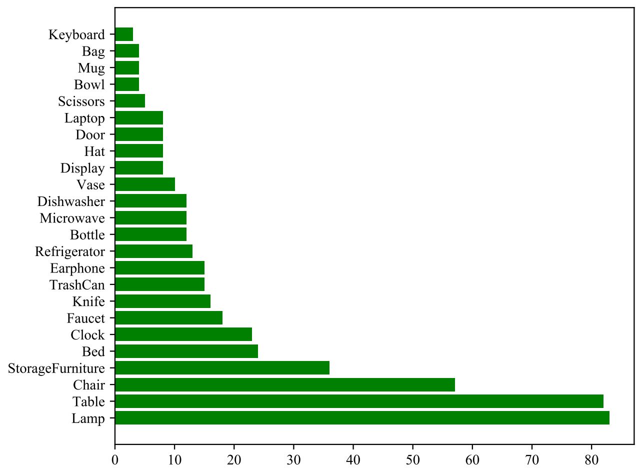

Figure 2. **PartNet part semantics statistics.** We report the statistics for the number of part semantics for each object category.

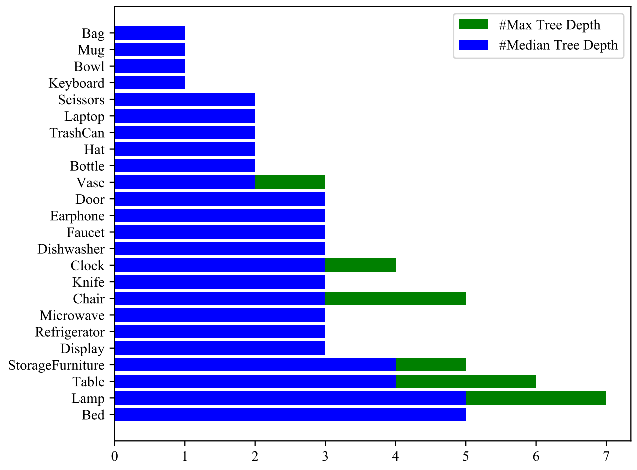

Figure 4. **PartNet tree depth statistics.** We report the statistics for the maximum and median tree depth for each object category.

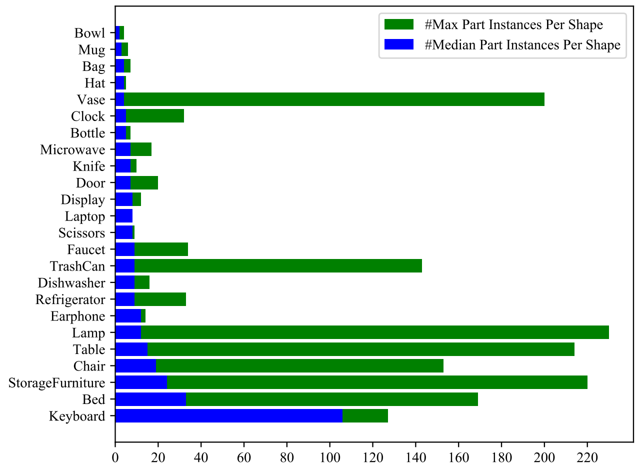

Figure 3. **PartNet part instance statistics.** We report the statistics for the maximum and median number of part instances per shape for each object category.

template. To ensure that our templates cover most of the shape variations and part semantics of each object category, we generated a t-SNE [5] visualization for the entire shape space to study the shape variation. We trained an auto-encoder based on the shape geometry within each object category to obtain shape embeddings for the t-SNE visualization.

Although we try to cover the most common part semantics in our templates, it is still not easy to cover all possible object parts. Thus, we allow annotators to deviate from the templates and define their own parts and segmentation structures. Among all the annotated part instances, 1.3% of them are defined by the annotators. In the raw annotation, 13.1% of shapes contained user-defined part labels.

Our analysis shows that our template designs are able to cover most of the ShapeNet [1] shapes. Of the 27,260 shapes we collected in total, our annotators successfully labeled 26,671 of them, giving our templates a coverage rate of at least 97.8% for ShapeNet shapes. While template coverage is a potential issue, the remaining 2.2% were not annotated mainly due to other issues such as poor mesh quality, classification error, error during mesh splitting, *etc.*

We design hierarchical templates that cover both the coarse-level part semantics and fine-grained part details down to the primitive level, *e.g. chair back vertical bar* and *bed base surface panel*. Most primitive-level parts are atomic such that they are very unlikely to be further divided for end applications. If an application requires different segmentation hierarchy or level of segmentation than the ones we already provide in our template, developers and researchers can try to build up their own segmentation based upon the atomic primitives we obtain in PartNet.

Moreover, we try our best to make the shared part concepts among different shapes and even different object categories share the same part labels. For example, we use the part label *leg* for *table*, *chair*, *lamp base*, *etc.* and the part label *wheel* for both *chair swivel base wheel* and *refrigerator base wheel*. Such part concept sharing provides rich part correspondences within a specific object category and across multiple object categories.

C.2. Template Refinement Details

Fine-grained shape segmentation is challenging to annotate due to the subtle concept gaps among similar part semantics. Even though we provide detailed textual and visual explanation for our pre-defined parts, we still observe some annotation inconsistencies across multiple annotators. To quantitatively diagnose such issues, we reserve a small subset of shapes for which we collect multiple human annotations. We compute the confusion scores among the pre-defined parts across the multiple annotations and conduct careful template refinement to reduce the part ambiguity.

	Avg	Bag	Bed	Bott	Bowl	Chair	Clock	Dish	Disp	Door	Ear	Fauc	Hat	Key	Knife	Lamp	Lap	Micro	Mug	Frid	Scis	Stora	Table	Trash	Vase
O$_{avg}$	69.8	54.8	70.0	87.5	87.7	59.0	62.1	67.3	85.2	64.4	74.1	69.9	86.8	77.0	75.0	44.6	61.6	71.0	91.0	65.3	88.7	68.0	40.1	51.4	72.6
O$_{avg}$	19.0	29.3	17.4	6.9	8.1	26.7	24.6	19.1	17.8	15.2	15.6	17.6	9.5	17.9	14.6	29.0	27.1	19.3	10.6	27.7	9.0	21.3	29.3	23.3	19.5
R$_{avg}$	83.3	82.1	76.2	89.3	91.7	77.8	91.1	81.5	94.0	77.0	83.0	84.7	89.3	89.6	77.8	72.7	78.3	84.4	91.7	85.1	90.2	77.1	71.4	71.0	92.3
R$_{std}$	10.4	11.1	9.2	6.0	7.4	15.2	7.0	7.2	2.8	11.2	13.5	8.6	10.3	3.5	14.9	17.3	14.6	9.1	9.7	6.2	8.6	12.8	22.6	13.2	7.5

Table 1. **The average confusion scores and the standard deviations for multiple annotations (%).** We report the average confusion scores and the standard deviations by calculating over the entries on the diagonal of the confusion matrix for each object category using the small subset of shapes that we collect multiple human annotations. Rows **O** and **R** respectively refer to the scores before and after the template refinement process.

There are primarily three sources of such inconsistencies: boundary ambiguity, granularity ambiguity and part labeling ambiguity. Boundary ambiguity refers to the unclear boundary between two parts, which is also commonly seen in previous works [3, 10]. For example, the boundary between the bottle neck and the bottle body is not that clear for wine bottles. Granularity ambiguity means that different annotators have different understanding about the segmentation granularity of the defined parts. One example is that, for a curvy and continuous chair arm, one can regard it as a whole piece or imagine the separation of armrest and arm support. The most common type of ambiguity in our dataset is the part labeling ambiguity. The fine-grained part concepts, though intended to be different category-wise, may apply to the same part on a given object. For example, a connecting structure between the seat and the base of a chair can be considered as chair seat support or chair base connector.

We study the mutual human agreement on the multiple annotation subset. We consider the parts defined at the leaf node level of segmentation on the hierarchy and compute the confusion matrix across multiple human annotations[2]. The ideal confusion matrix should be close to the diagonal matrix without any part-level ambiguity. In our analysis, we observe human disagreement among some of our initial part definitions. To address the ambiguity, we either merge two similar concepts with high confusion scores or remove the hard-to-distinguish parts from evaluation. For example, we find our annotators often mix up the annotation for regular tables and desks due to the similarity in the two concepts. Thus, we merge the desk subtype into the regular table subtype to address this issue. In other cases, some small parts such as the buttons on the displays are very tricky to segment out from the main display frame. Since they may not be reliably segmented out, we decided to remove such unclear segmentation from evaluation.

Table 1 compares the annotation consistency before and after the template refinement process. We compute the confusion matrices at the most fine-grained segmentation level. After the template refinement, the data consistency score is 83.3% on average, having 13.5% improvement over the raw annotation. The template refinement process improves the annotation consistency by a clear margin. This also reflects the complexity of the task in terms of annotating fine-grained part concepts. Future works may investigate how to further design better templates with less part ambiguities.

[2]We consider the entire path labels as histories to the leaf nodes when computing the confusion matrix.

C.3. More Visualization of Hierarchical Templates

Figure 10, 11 and 12 show more visualization for the expert-designed hierarchical templates after resolving the data inconsistency and conducting template refinements. We show the lamp template in the main paper.

D. Tasks and Benchmarks

In this section, we provide more details about the architectures and training details for the benchmark algorithms. We also present additional evaluation metrics, shape mean Intersection-over-Union (shape mIoU) and shape mean Average-Precision (Shape mAP), and report the quantitative results using these metrics.

D.1. Fine-grained Semantic Segmentation

More Architecture and Training Details We follow the default architectures and training hyper-parameters used in the original papers: PointNet [6], PointNet++ [7], SpiderCNN [9] and PointCNN [4], except the following few modifications:

- Instead of training one network for all object categories as done in the four prior works, we train separate networks for each object category at each segmentation level. This is mainly to handle the increase in the number of parts for fine-grained part segmentation. Originally, there are only 50 parts for all 16 object categories using the coarse ShapeNet Part dataset [10]. Now, using PartNet, there could be 480 different part semantics in total. Also, due to the data imbalance among different object categories, training a single network may overfit to the big categories.

- We change the input point cloud size to 10,000. The original papers usually sample 1,000, 2,000 or 4,000 points and input to the networks. PartNet suggests to use at least 10,000 to guarantee enough point sampling over small fine-grained parts, *e.g.* a door handle, or a small button.

	Avg	Bag	Bed	Bott	Bowl	Chair	Clock	Dish	Disp	Door	Ear	Fauc	Hat	Key	Knife	Lamp	Lap	Micro	Mug	Frid	Scis	Stora	Table	Trash	Vase
P1	71.8	59.3	39.6	81.0	78.5	81.8	67.1	78.9	88.2	71.1	68.0	67.5	58.5	65.6	66.5	46.5	96.5	75.0	84.2	**79.6**	86.5	55.9	85.6	66.7	76.3
P2	50.1	–	21.3	–	–	52.4	–	60.0	–	47.1	–	–	–	–	–	43.5	–	**64.3**	–	**63.9**	–	48.8	50.0	–	–
P3	48.2	–	13.0	55.3	–	44.8	37.8	55.2	79.0	38.8	47.5	55.5	–	–	40.0	34.7	–	54.5	–	**53.2**	–	47.4	42.5	46.4	74.0
Avg	63.4	59.3	24.6	68.2	78.5	59.7	52.4	64.7	83.6	52.3	57.8	61.5	58.5	65.6	53.2	41.6	96.5	64.6	84.2	**65.6**	86.5	50.7	59.4	56.5	75.2
P⁺1	**76.8**	72.7	54.7	85.8	78.5	**84.5**	**74.1**	**81.9**	90.7	**73.5**	**77.8**	**73.6**	64.2	62.5	**75.0**	**65.5**	96.6	80.3	90.9	72.1	87.5	61.2	**86.7**	71.5	**81.4**
P⁺2	**54.7**	–	34.8	–	–	**54.9**	–	60.6	–	**57.0**	–	–	–	–	–	**56.8**	–	63.0	–	58.4	–	52.9	**53.6**	–	–
P⁺3	**53.4**	–	25.1	**61.0**	–	**49.6**	**46.1**	52.5	81.0	**48.0**	**56.1**	60.4	–	–	**49.1**	**46.0**	–	54.3	–	50.7	–	50.6	**47.0**	**54.7**	**75.1**
Avg	**68.1**	72.7	38.2	**73.4**	78.5	**63.0**	**60.1**	65.0	85.8	**59.5**	**67.0**	**67.0**	64.2	62.5	**62.0**	**56.1**	96.6	65.9	90.9	60.4	87.5	54.9	**62.4**	**63.1**	**78.2**
S1	73.9	**72.9**	**55.9**	**86.1**	**83.4**	83.8	72.1	73.3	90.4	60.4	70.6	71.5	**71.6**	64.6	42.1	59.1	**97.1**	78.6	91.6	68.7	77.0	**64.2**	83.8	**74.4**	79.5
S2	53.3	–	**37.8**	–	–	53.6	–	**65.3**	–	55.0	–	–	–	–	–	41.4	–	62.1	–	62.6	–	49.8	51.7	–	–
S3	48.0	–	27.2	52.8	–	44.7	44.2	51.1	77.2	40.7	47.5	53.7	–	–	27.3	35.7	–	54.4	–	52.4	–	**53.1**	43.3	48.0	62.3
Avg	65.1	**72.9**	40.3	69.4	**83.4**	60.7	58.1	63.2	83.8	52.0	59.0	62.6	**71.6**	64.6	34.7	45.4	**97.1**	65.0	91.6	61.2	77.0	**55.7**	59.6	61.2	70.9
C1	75.5	72.0	55.3	83.6	75.0	83.9	65.6	81.8	**91.9**	68.1	74.5	71.1	66.8	**70.4**	68.1	55.6	**97.1**	**83.1**	**92.7**	78.9	**92.6**	58.8	85.5	67.7	71.8
C2	52.1	–	36.6	–	–	52.9	–	63.4	–	54.9	–	–	–	–	–	42.4	–	64.1	–	57.7	–	**54.4**	42.7	–	–
C3	49.6	–	**29.1**	58.7	–	47.7	36.2	**55.3**	**81.5**	40.4	55.8	**60.7**	–	–	26.4	34.4	–	**58.7**	–	50.8	–	52.3	37.4	50.8	67.0
Avg	66.3	72.0	**40.3**	71.2	75.0	61.5	50.9	**66.8**	**86.7**	54.5	65.2	65.9	66.8	**70.4**	47.2	44.1	**97.1**	**68.6**	**92.7**	62.5	**92.6**	55.2	55.2	59.2	69.4

Table 2. **Fine-grained semantic segmentation results (shape mIoU %).** Algorithm **P**, **P⁺**, **S** and **C** refer to PointNet [6], PointNet++ [7], SpiderCNN [9] and PointCNN [4], respectively. The number **1**, **2** and **3** refer to the three levels of segmentation: coarse-, middle- and fine-grained. We put short lines for the levels that are not defined.

- We reduce the batch sizes for training the networks if necessary. Since we use point cloud size 10,000, to fit the training in NVIDIA TITAN XP GPU 12G memory, we need to adjust the training batch size accordingly. For PointNet [6], PointNet++ [7], SpiderCNN [9] and PointCNN [4], we use batch size of 24, 24, 2 and 4 respectively.

- We only input 3D coordinates as inputs to all the networks for fair comparison. Although the 3D CAD models in ShapeNet [1] usually provide additional features, *e.g.* opacity, point normals, textures and material information, there is no guarantee for the quality of such information. Thus, we choose not to use them as the inputs. Also, only using pure geometry potentially increase the network generalizability to unseen objects or real scans [6]. PointNet++ [7] and SpiderCNN [9] by defaults take advantage of the point normals as additional inputs. In this paper, we remove such inputs to the networks. However, point normals can be estimated from the point clouds. We leave this as a future work.

Shape mIoU Metric and Results We introduce the shape mean Intersect-over-Union (Shape mIoU) evaluation metric as a secondary metric to the Part-category mIoU metric in the main paper. Shape mIoU metric considers shapes as evaluation units and measures how an algorithm segment an average shape in the object category. In contrast, Part-category mIoU reports the average performance over all part semantics and indicates how an algorithm performs for any given part category.

Shape mIoU is widely used on ShapeNet Part dataset [10] for 3D shape coarse semantic segmentation [6, 7, 9, 4]. We propose a slightly different version for fine-grained semantic segmentation. For each test shape, we first compute the IoU for each part semantics that either presents in the ground-truth or is predicted by the algorithm, and then we calculate the mean IoU for this shape. We remove the ground-truth unlabeled points from the evaluation. Finally, we calculate the Shape mIoU by averaging mIoU over all test shape instances.

We benchmark the four algorithms using Shape mIoU in Table 2. Besides the Shape mIoU scores for each object category at each segmentation level, we also report the average across levels for each object categories and further calculate the average over all object categories.

We observe that PointNet++ [7] achieves the best performance using the Shape mIoU metric, while PointCNN [4] performs the best using the Part-category mIoU metric. The Part-category mIoU metric considers all part semantics equally while the Shape mIoU metric considers all shapes equally. We observe an unbalanced counts for different part semantics in most object categories, *e.g.* there are much more chair legs than chair wheels. To achieve good numbers on Part-category mIoU, a segmentation algorithm needs to perform equally well on both frequent parts and rare parts, while the Shape mIoU metric bias over the frequently observed parts.

D.2. Hierarchical Semantic Segmentation

We describe the architecture and training details for the three baseline methods we propose for hierarchical semantic segmentation in the main paper. All three methods use PointNet++ [7] segmentation network as the network backbone. The difference of the three methods is mainly on the training and inference strategies to enforce the tree knowledge to the final prediction.

	Avg	Bed	Bott	Chair	Clock	Dish	Disp	Door	Ear	Fauc	Knife	Lamp	Micro	Frid	Stora	Table	Trash	Vase
Bottom-Up	65.9	42.0	74.3	63.8	64.1	**66.3**	84.2	61.4	70.0	**74.2**	67.1	62.7	**63.0**	60.8	57.8	65.7	62.8	80.9
Top-Down	65.9	42.0	73.7	62.3	**65.5**	64.0	85.5	63.1	71.1	73.5	**68.8**	63.3	62.7	58.8	57.6	66.2	63.0	79.3
Ensemble	**66.6**	**42.9**	**74.4**	**64.3**	**65.5**	62.7	**85.8**	**63.7**	**71.7**	74.0	66.7	**63.4**	61.9	**61.5**	**60.6**	**67.5**	**64.0**	**82.2**

Table 3. **Hierarchical segmentation results (shape mIoU %).** We present the hierarchical segmentation performances for three baseline methods: bottom-up, top-down and ensemble. We conduct experiments on 17 out of 24 categories with tree depth bigger than 1.

The Bottom-up Method The bottom-up method learns a network to perform segmentation at the most fine-grained leaf part semantics. We use the PointNet++ [7] segmentation network with a softmax activation layer as the network architecture. At inference time, we use the ground-truth tree hierarchy to gather the prediction for the parent nodes. The parent node prediction is the sum of all its children node predictions. Even though we only train for the leaf node parts, the parent history is implicitly encoded. For example, we define vertical bars for both chair back and chair arm, but they are two different leaf node parts: *chair back vertical bar* and *chair arm vertical bar*.

In the ground-truth annotation, all the points in the point cloud belong to the root node. Each point is assigned a path of labels from the root node to some intermediate node in the tree. The paths for most points are all the way down to the leaf levels while some points may not. For example, a point on a bed blanket (removed from evaluation since it cannot be distinguished without color information) may be assigned with labels $\{$*bed*, *bed unit*, *sleeping area*$\}$ in the ground-truth annotation. The part *sleeping area* is not a leaf part. For such cases, we introduce an additional leaf node *other* for each parent node in the tree and consider them in the training.

The Top-down Method The top-down method learns a multi-labeling task for all the part semantics in the tree, considering both the leaf nodes and the parent nodes. Compared to the bottom-up method, the top-down method takes advantage of the tree structures at training time.

Assuming there are T tree nodes in the hierarchy, we train a PointNet++ [7] segmentation network for a T-way classification for each point. We apply a softmax activation layer to enforce label mutual exclusiveness. For a point with the ground-truth labels $\{y_1, y_2, y_3\}$ and prediction softmax scores $\{s_i | i = 1, 2, \cdots, T\}$, we train the labels using a multi-labeling cross-entropy loss

$$L = -\log(s_{y_1}) - \log(s_{y_2}) - \log(s_{y_3}) \qquad (1)$$

to increase the values of all the three label predictions over the rest labels.

The Ensemble Method The ensemble method trains multiple neural networks at different levels of segmentation as defined in the fine-grained semantic segmentation task. The key idea is that conducting segmentation at the coarse-, middle- and fine-grained levels separately may learn different features that work the best at each level. Compared to the bottom-up method that we only train at the most fine-grained level, additional signal at the coarse level helps distinguish the coarse-level part semantics more easily. For example, the local geometric features for both chair back vertical bars and chair arm vertical bars may be very similar, but the coarse-level semantics may distinguish chair backs and chair arms better.

During the training, we train 2$\sim$3 networks at multiple levels of segmentation. At the inference time, we perform a joint inference considering the prediction scores from all the networks. We use a path-voting strategy: for each path from the root node to the leaf node, we calculate the average log-likelihood over the network prediction scores after applying the softmax activations, and select the path with the highest score as the joint label predictions.

Shape mIoU Metric and Results Similar to Sec D.1, we define Shape mIoU for hierarchical segmentation. The mIoU for each shape is calculated over the part semantics in the entire hierarchical template that are either predicted by the network or included in the ground-truth. The unrelated parts are not taken into consideration. Table 3 shows the quantitative evaluation for the three baseline methods. We observe similar performance for the three methods, with the ensemble method works slightly better.

D.3. Instance Segmentation

More Architecture and Training Details To train our proposed method, we use batch size 32, learning rate 0.001, and the default batch normalization settings used in the PointNet++ [7].

For SGPN [8], we use two-stage training as suggested by the authors of [8]. We first pretrain the PointNet++ semantic segmentation branch using batch size 32 and learning rate 0.001, with the default batch normalization as in PointNet++. And then, we jointly train for the semantic segmentation, similarity score matrix and confidence scores with batch size 1 and learning rate 0.0001. As suggested in the original SGPN paper, for the first five epochs of the joint training, we only turn on the loss for training the similarity scores matrix. The rest training epochs are done with the full losses switched on. We have to use batch size 1 because the input point cloud has the size of 10,000 and thus the similarity score matrix forms a $10,000 \times 10,000$ ma-

	Avg	Bag	Bed	Bott	Bowl	Chair	Clock	Dish	Disp	Door	Ear	Fauc	Hat	Key	Knife	Lamp	Lap	Micro	Mug	Frid	Scis	Stora	Table	Trash	Vase
S1	72.5	62.8	38.7	76.7	83.2	91.5	41.5	**81.4**	91.3	71.2	81.4	82.2	71.9	23.2	**78.0**	60.3	**100**	76.2	94.3	60.6	74.9	55.0	80.1	76.1	87.1
S2	50.2	–	22.7	–	–	51.1	–	**78.7**	–	43.3	–	–	–	–	–	49.1	–	68.6	–	42.9	–	51.9	43.7	–	–
S3	50.2	–	17.5	66.5	–	42.3	40.7	59.3	83.9	29.0	60.2	61.6	–	–	55.0	37.6	–	53.7	–	30.6	–	45.1	37.8	50.0	82.0
Avg	64.2	62.8	26.3	71.6	83.2	61.6	41.1	73.1	87.6	47.8	70.8	71.9	71.9	23.2	66.5	49.0	**100**	66.2	94.3	44.7	74.9	50.7	53.8	63.0	84.6
O1	**80.3**	**78.4**	**62.2**	**80.8**	**83.8**	**94.9**	**74.6**	**81.4**	**94.3**	**76.1**	**87.1**	**86.5**	**77.8**	**44.5**	76.6	**65.0**	**100**	**79.5**	**95.3**	**79.0**	**87.6**	**62.7**	**88.1**	**82.3**	**89.0**
O2	**60.5**	–	**29.4**	–	–	**64.7**	–	75.4	–	**61.1**	–	–	–	–	–	**56.8**	–	**78.2**	–	**61.7**	–	**57.4**	**59.4**	–	–
O3	**57.7**	–	**22.1**	**68.3**	–	**58.4**	**53.7**	**67.5**	**84.8**	**38.0**	**62.4**	**66.8**	–	–	**63.5**	**45.8**	–	**54.0**	–	**45.0**	–	**52.6**	**52.5**	**58.7**	**86.4**
Avg	**72.2**	**78.4**	**37.9**	**74.6**	**83.8**	**72.7**	**64.2**	**74.8**	**89.5**	**58.4**	**74.8**	**76.6**	**77.8**	**44.5**	**70.1**	**55.8**	**100**	**70.6**	**95.3**	**61.9**	**87.6**	**57.6**	**66.7**	**70.5**	**87.7**

Table 4. **Instance segmentation results (shape mAP %, IoU threshold 0.5).** Algorithm **S** and **O** refer to SGPN [8] and our proposed method respectively. The number **1**, **2** and **3** refer to the three levels of segmentation: coarse-, middle- and fine-grained. We put short lines for the levels that are not defined.

trix, which occupies too much GPU memory. Our proposed method is more memory-efficient, compared to SGPN. We also observe that our training is much faster than SPGN. We train all the networks until convergence.

Shape mAP Metric and Results We define Shape mean Average-Precision (Shape mAP) metric as a secondary metric to the Part-category mAP metric in the main paper. Similar to the Shape mIoU scores we use in Sec D.1 and D.2, Shape mAP reports the part instance segmentation performance on an average shape in a object category. It averages across the test shapes, instead of averaging across all different part semantics, as benchmarked by Part-category mAP in the main paper.

To calculate Shape mAP for a test shape, we consider the AP for the part semantics that occur either in the ground-truth or the prediction for the given shape and compute their average as the mean AP score. Then, we average the mAP across all test shapes within a object category. Table 4 reports the part instance segmentation performance under the Shape mAP scores. We see a clear performance improvement of the proposed method over SGPN.

References

[1] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. ShapeNet: An information-rich 3D model repository. *arXiv preprint arXiv:1512.03012*, 2015. 2, 4

[2] Angel X Chang, Rishi Mago, Pranav Krishna, Manolis Savva, and Christiane Fellbaum. Linking WordNet to 3D shapes. In *Global WordNet Conference*, 2018. 1

[3] Xiaobai Chen, Aleksey Golovinskiy, and Thomas Funkhouser. A benchmark for 3D mesh segmentation. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 2009. 3

[4] Yangyan Li, Rui Bu, Mingchao Sun, and Baoquan Chen. PointCNN: Convolution on $\mathcal{X}$-transformed points. *Advances in neural information processing systems (NIPS)*, 2018. 3, 4

[5] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008. 2

[6] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. PointNet: Deep learning on point sets for 3D classification and segmentation. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, volume 1, page 4, 2017. 3, 4

[7] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. PointNet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems*, pages 5099–5108, 2017. 3, 4, 5

[8] Weiyue Wang, Ronald Yu, Qiangui Huang, and Ulrich Neumann. SGPN: Similarity group proposal network for 3D point cloud instance segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2569–2578, 2018. 5, 6

[9] Yifan Xu, Tianqi Fan, Mingye Xu, Long Zeng, and Yu Qiao. SpiderCNN: Deep learning on point sets with parameterized convolutional filters. *European Conference on Computer Vision (ECCV)*, 2018. 3, 4

[10] Li Yi, Vladimir G Kim, Duygu Ceylan, I Shen, Mengyan Yan, Hao Su, Cewu Lu, Qixing Huang, Alla Sheffer, Leonidas Guibas, et al. A scalable active framework for region annotation in 3D shape collections. *ACM Transactions on Graphics (TOG)*, 35(6):210, 2016. 3, 4

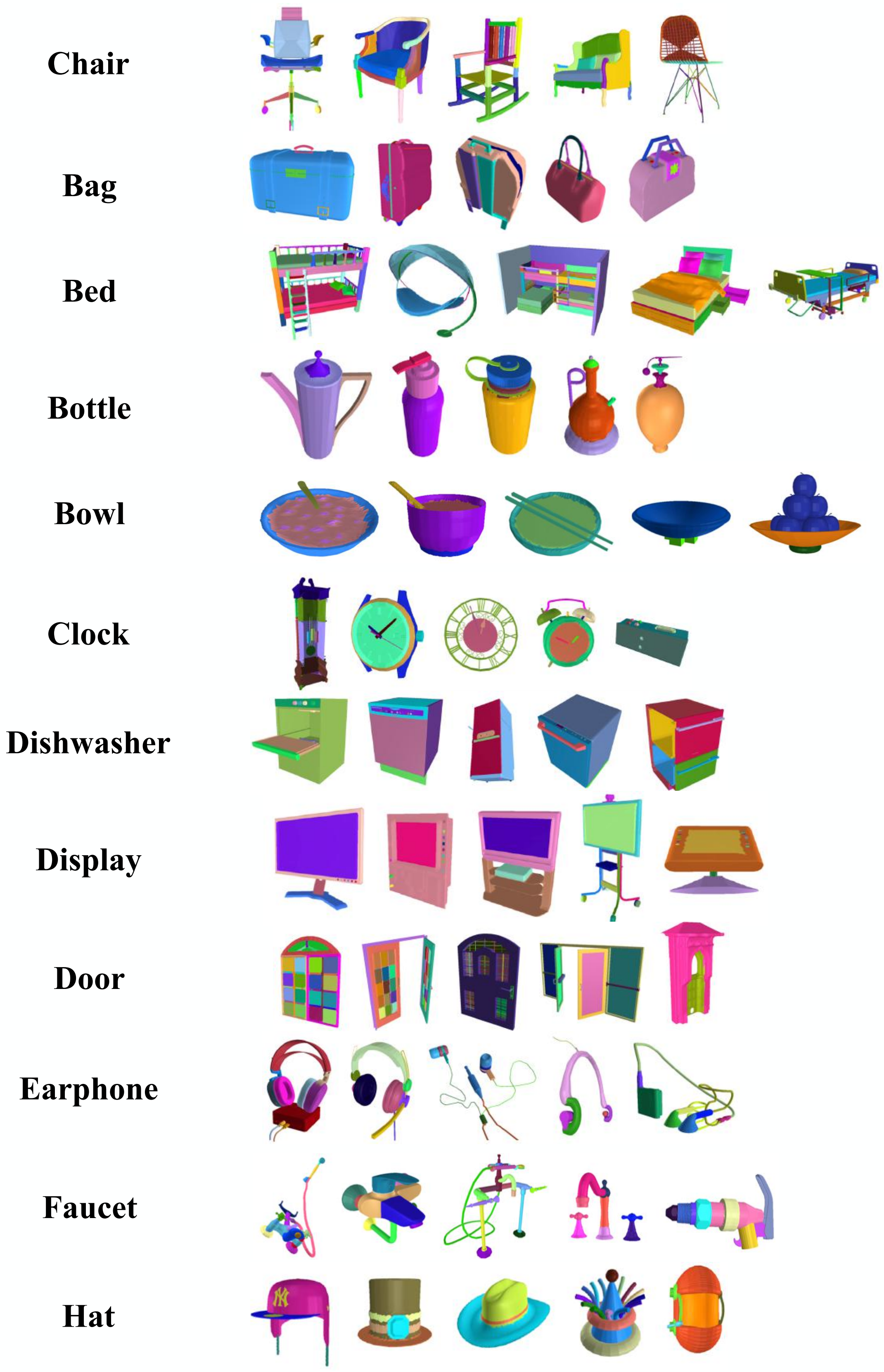

Figure 5. **Fine-grained instance-level segmentation visualization (1/2).** We present visualization for example fine-grained instance-level segmentation annotations for chair, bag, bed, bottle, bowl, clock, dishwasher, display, door, earphone, faucet, and hat.

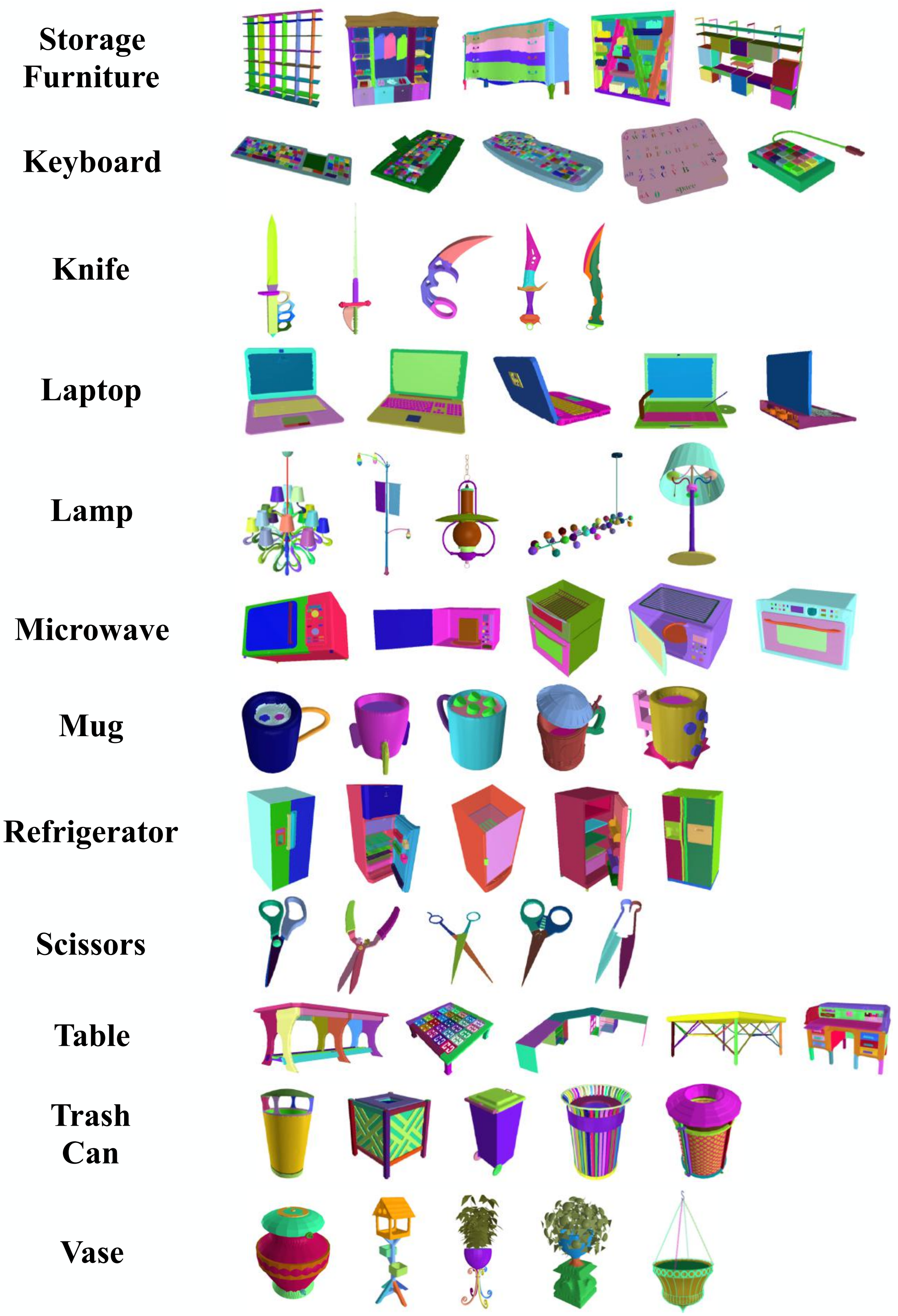

Figure 6. **Fine-grained instance-level segmentation visualization (2/2).** We present visualization for example fine-grained instance-level segmentation annotations for storage furniture, keyboard, knife, laptop, lamp, microwave, mug, refrigerator, scissors, table, trash can, and vase.

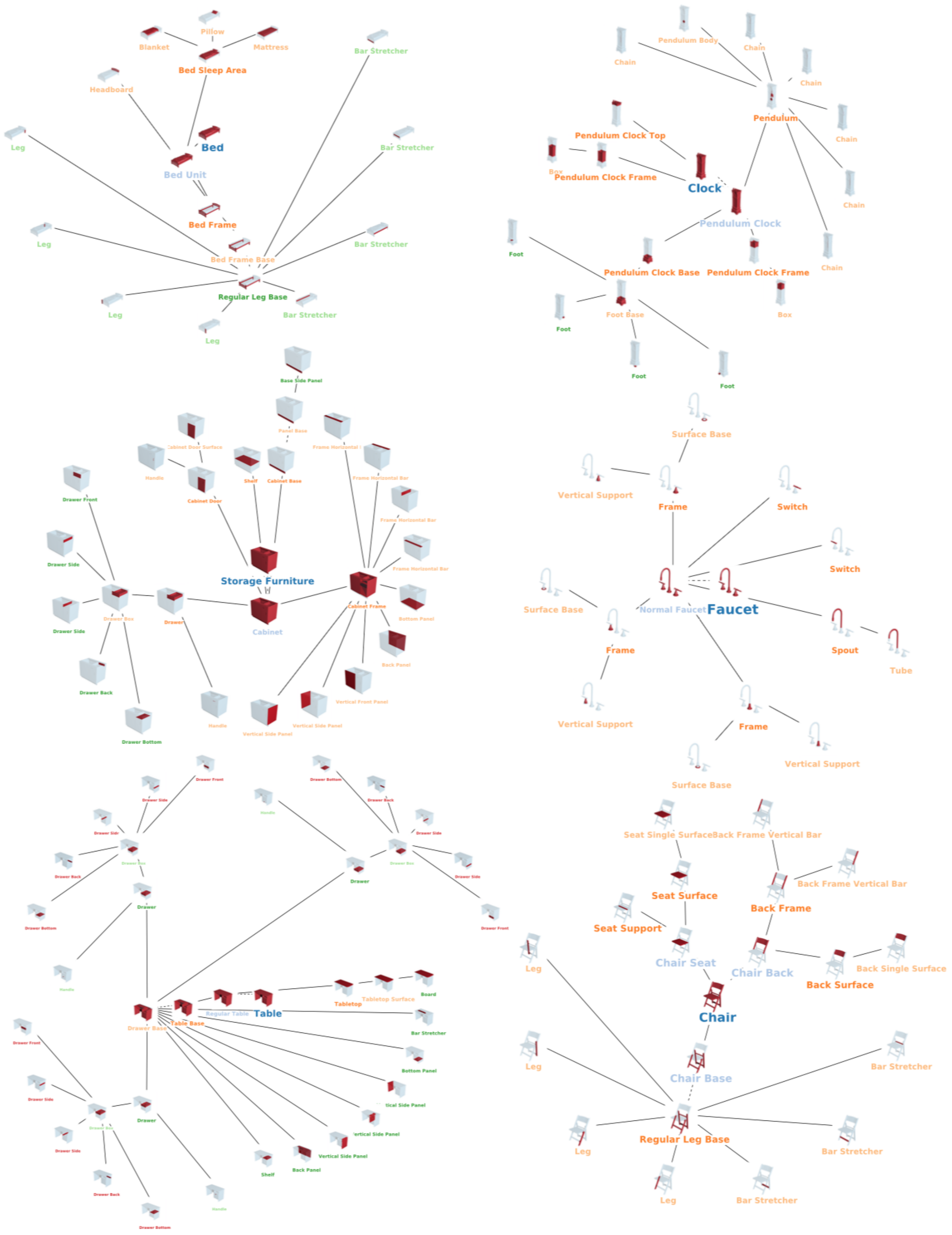

Figure 7. **Hierarchical instance-level segmentation visualization (1/3).** We present visualization for example hierarchical instance-level segmentation annotations for bed, clock, storage furniture, faucet, table, and chair. The lamp examples are shown in the main paper. The And-nodes are drawn in solid lines and Or-nodes in dash lines.

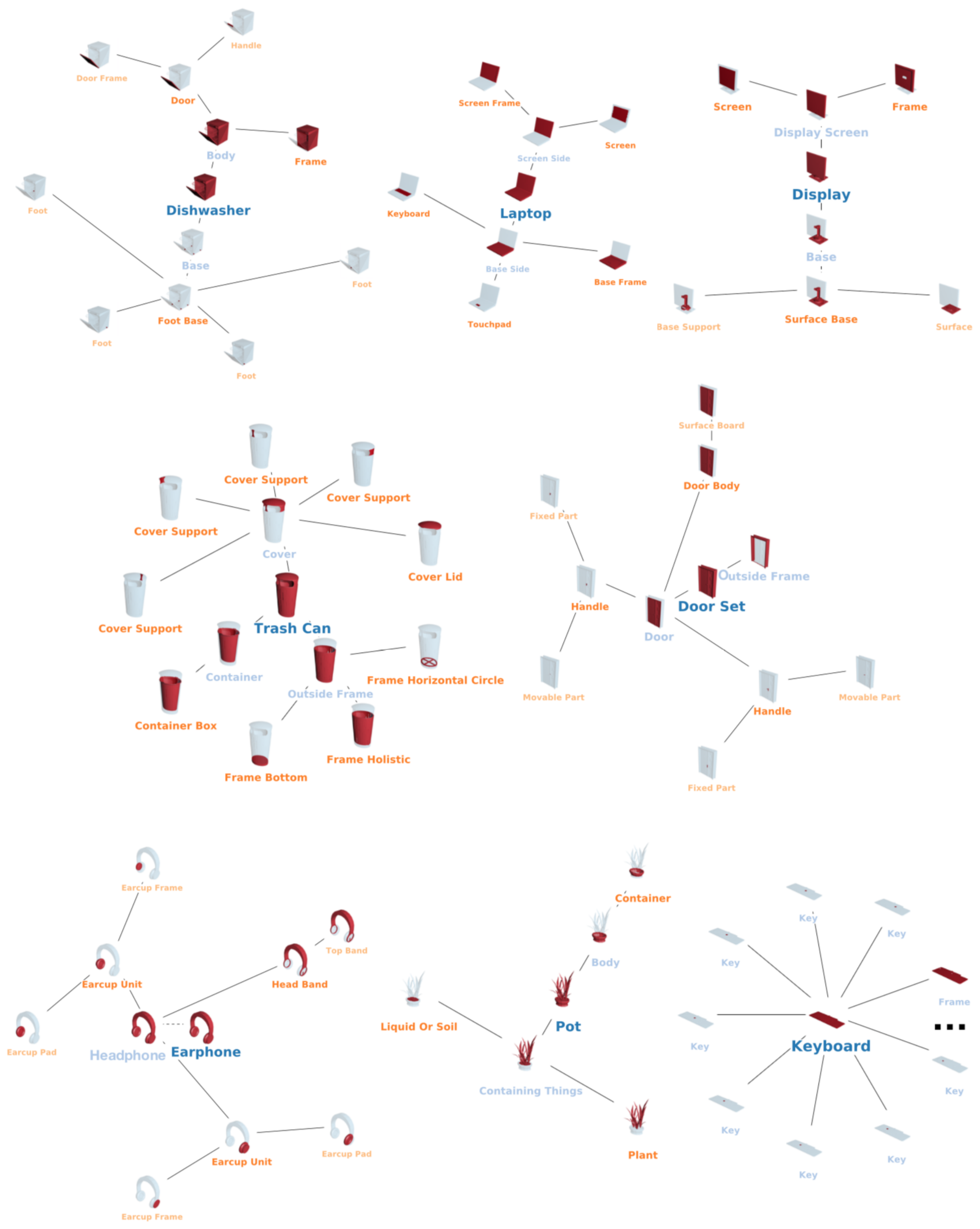

Figure 8. **Hierarchical instance-level segmentation visualization (2/3).** We present visualization for example hierarchical instance-level segmentation annotations for dishwasher, laptop, display, trash can, door (door set), earphone, vase (pot), and keyboard. The lamp examples are shown in the main paper. The And-nodes are drawn in solid lines and Or-nodes in dash lines.

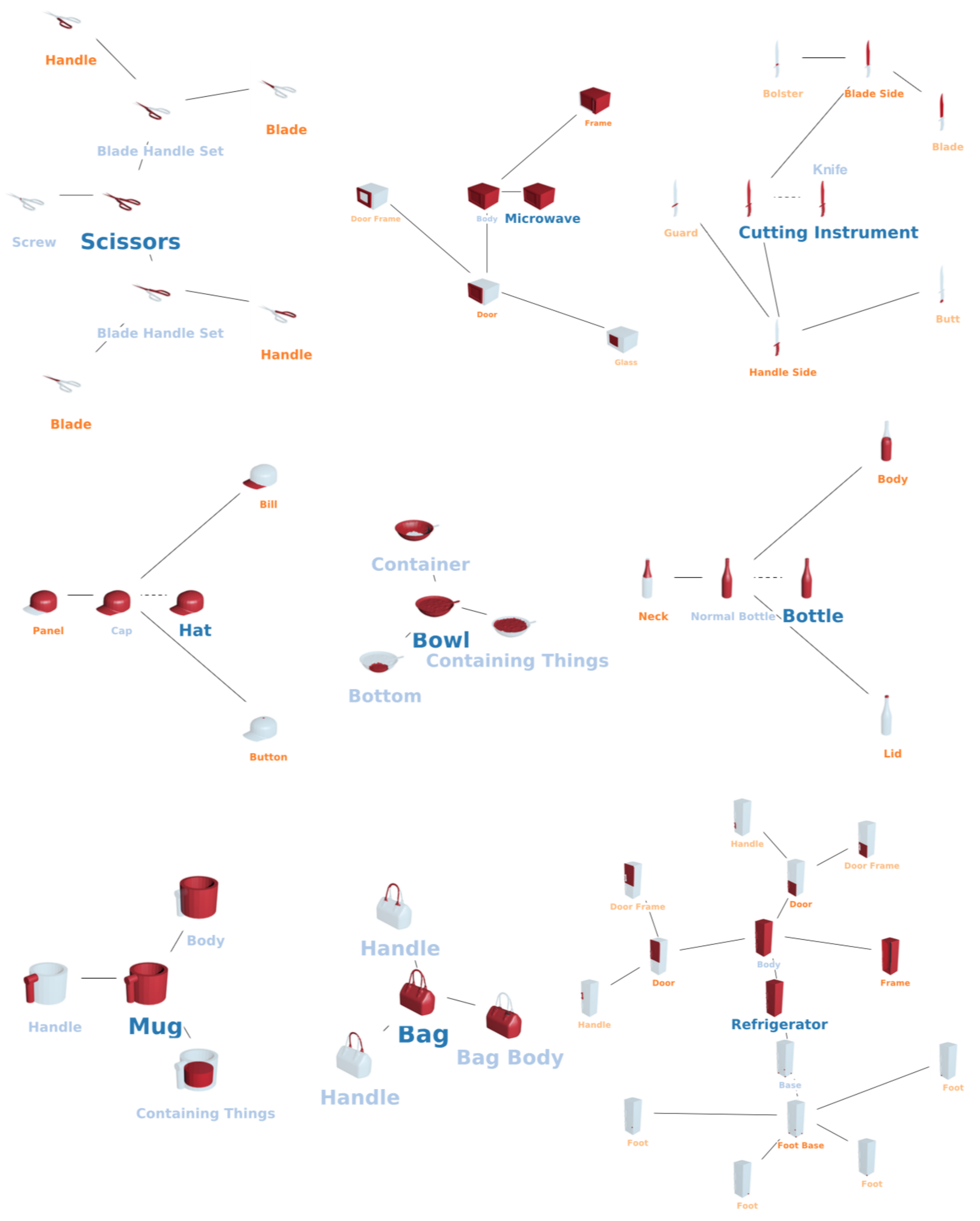

Figure 9. **Hierarchical instance-level segmentation visualization (3/3).** We present visualization for example hierarchical instance-level segmentation annotations for scissors, microwave, knife (cutting instrument), hat, bowl, bottle, mug, bag, and refrigerator. The lamp examples are shown in the main paper. The And-nodes are drawn in solid lines and Or-nodes in dash lines.

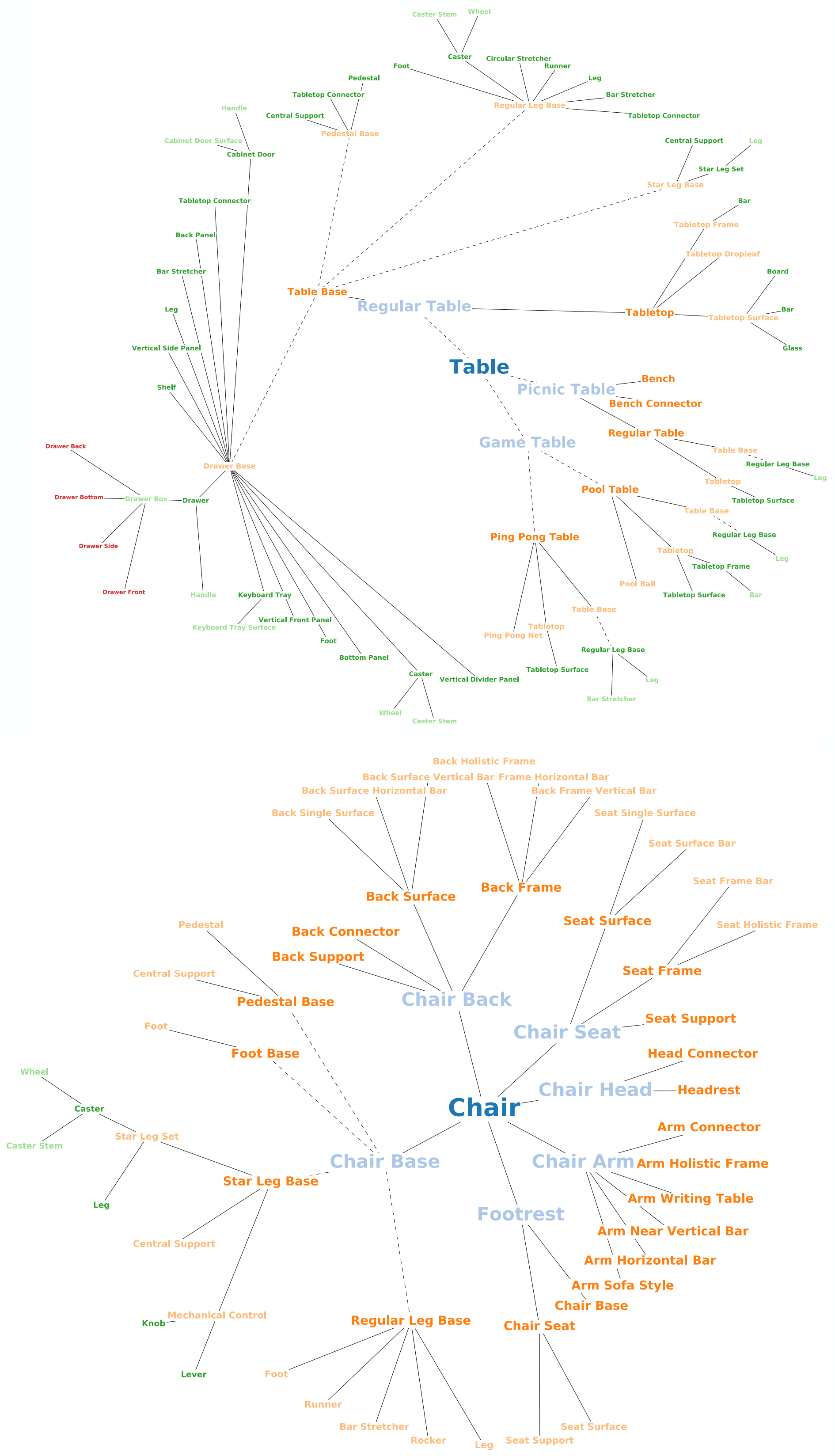

Figure 10. **Template visualization (1/3).** We present the templates for table and chair. The lamp template is shown in the main paper. The And-nodes are drawn in solid lines and Or-nodes in dash lines.

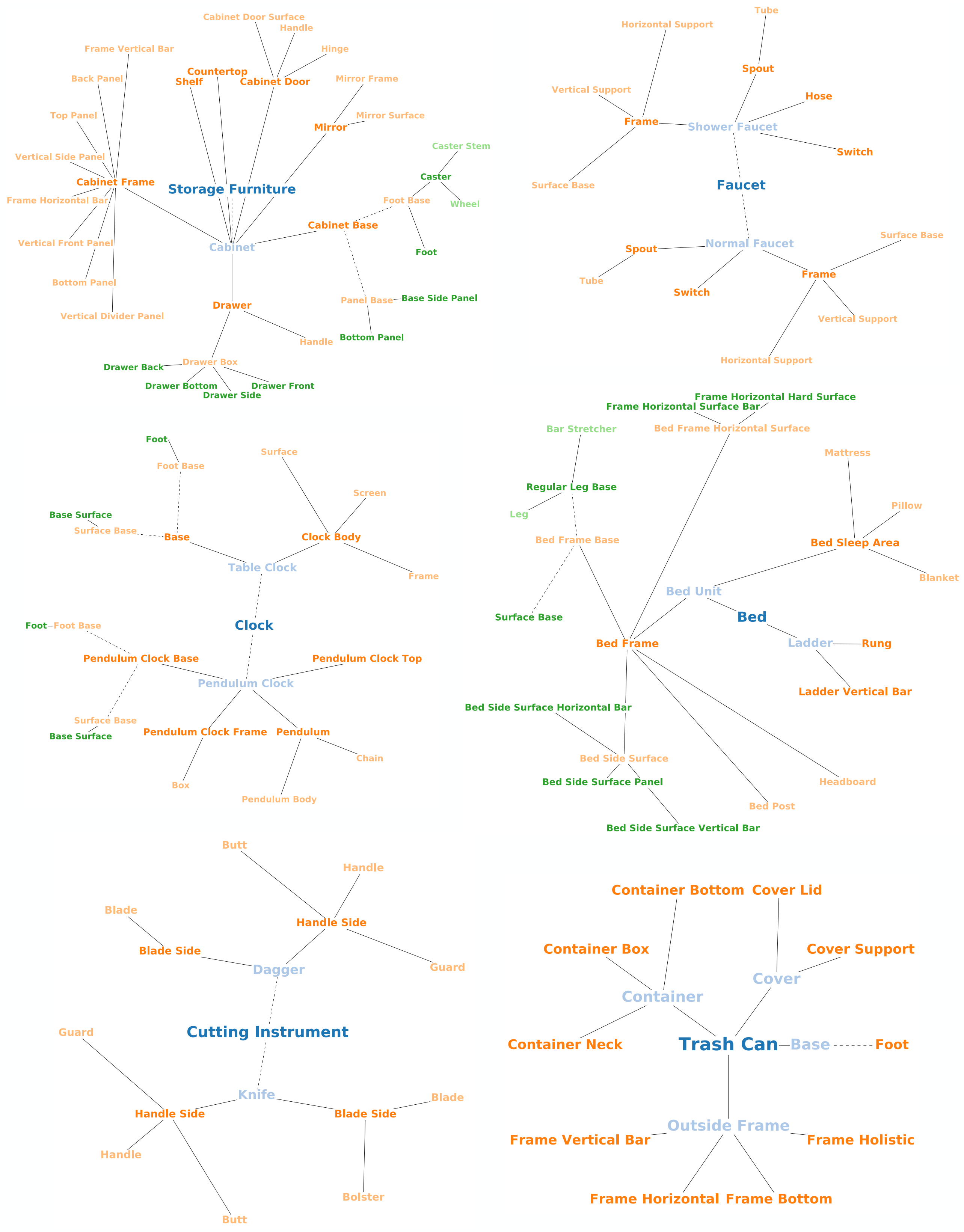

Figure 11. **Template visualization (2/3).** We present the templates for storage furniture, faucet, clock, bed, knife (cutting instrument), and trash can. The lamp template is shown in the main paper. The And-nodes are drawn in solid lines and Or-nodes in dash lines.

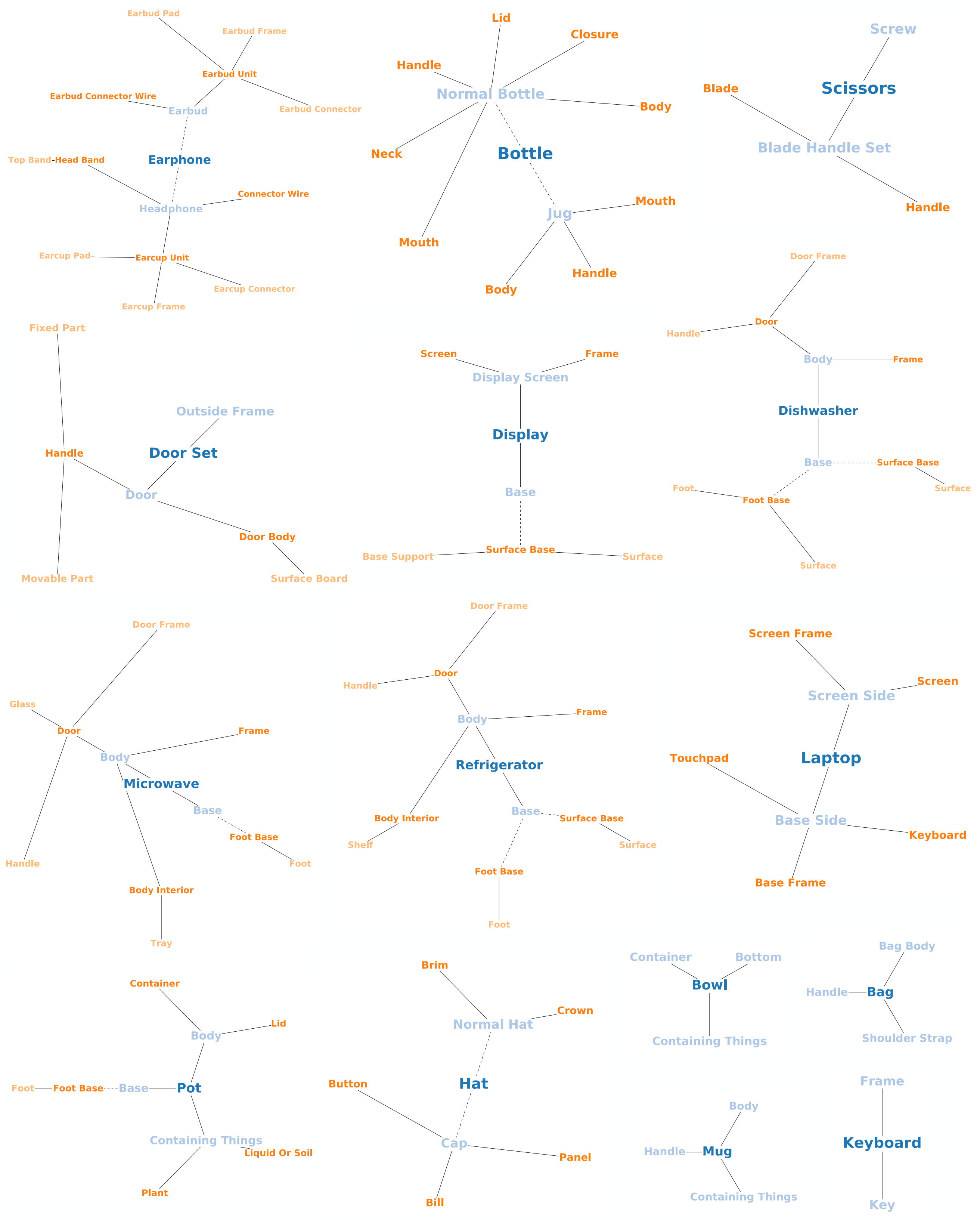

Figure 12. **Template visualization (3/3).** We present the templates for earphone, bottle, scissors, door (door set), display, dishwasher, microwave, refrigerator, laptop, vase (pot), hat, bowl, bag, mug, and keyboard. The lamp template is shown in the main paper. The And-nodes are drawn in solid lines and Or-nodes in dash lines.