S2O: Static to Openable Enhancement for Articulated 3D Objects

Supplementary Material

In this supplement, we provide additional details and analysis for the two datasets we use in our experiments (Appendix A), method details (Appendices B and C), and additional qualitative and quantitative results (Appendix D). See our paper summary video and additional results video (corresponding to Appendix A.2) for more details and results.

A. Dataset Details

We compare our new Articulated Containers Dataset (ACD) to PM-Openable (Appendix A.1), provide information on how we split ACD into train and val (Appendix A.2), show T-SNE visualization of similarity of shapes for PM-Openable across the train/val/test splits (Appendix A.3), and provide details on the construction of PM-Openable-ext (Appendix A.4).

A.1. ACD statistics

We show examples from ACD in Fig. 8. In Tab. 8 and Fig. 9, we provide statistics of the object categories. Note that PM-Openable provides only coarse level categories of StorageFurniture and Table. To obtain finer object categories, we map the PartNetMobility [51] asset ids to original ShapeNet [3] asset ids to determine the finer classification in Tab. 8 and Fig. 9.

ACD focused on a diverse collection of objects with a variety of part configurations and part shapes, with subclasses of storage furniture and tables which have more part variation than appliances. Figure 10 shows the distribution of part configurations based on the number of drawers, doors, and lids that a object has. Compared to PM-Openable, ACD has a wider range of different number and type of parts for each object. Many PM-Openable objects have just 1 or 2 parts of the same type (71% compared to 39% for ACD). ACD also has more variety in motion types in different object categories such as objects with doors that translate (see Fig. 11). We show counts per part configuration in Fig. 12.

ACD has diverse object and part shapes such as non-rectangular or curved drawers (22), L-shaped desks (14), angled corner cabinets (10), and objects with drawer-like parts that articulate as doors (10). The complex part arrangements and diversity in shapes all make ACD a better dataset for benchmarking generalization of openable part segmentation and motion prediction.

A.2. ACD-train and ACD-val

To study the distribution gap between PM-Openable and ACD as well as its potential for training, we split ACD into train and val by selecting shapes by their original data

Table 8. Comparison of PM-Openable and ACD, with the number of objects and average number of openable parts for each object type. The colors highlight differences between the categories in PM-Openable and Articulated Containers Dataset. We show object categories within the broad category StorageFurniture (light blue), categories within the broad category Table (dark blue), and lastly categories that are newly introduced (green) in Articulated Containers Dataset to add more diversity. Note that the PM-Openable categories for StorageFurniture and Table encompass several more fine-grained categories such as bookcases and wardrobes. Coarse category totals in italics.

		PM-Openable		ACD	
coarse category	category	obj	part/obj	obj	part/obj
StorageFurniture		*335*	2.4	*72*	*3.1*
	CabinetUnit	233	2.1	10	2.0
	Cabinet	83	3.1	39	3.9
	Wardrobe	8	3.3	66	4.0
	ChestOfDrawers	4	6.0	44	5.8
	Sideboard			24	4.1
	Bookcase	5	2.0	16	5.3
	TvStand			17	4.2
	WallUnit			8	8.2
	SinkCabinet			11	2.5
	StorageBench	2	2.0	10	2.3
Table		*77*	*2.6*	*72*	*3.1*
	Table	31	2.1	13	3.8
	Sidetable	12	2.7	1	1.0
	Desk	31	3.1	22	4.7
	Nightstand	3	1.3	36	1.9
Bed				6	2.8
Appliance		*137*	*1.3*	*31*	*1.7*
	Refrigerator	42	1.62	5	2.5
	Dishwasher	41	1	2	1
	Oven	24	1.5	9	2.3
	WashingMachine	17	1	8	1
	Microwave	13	1	6	1.5
Other		*99*	*1.1*	*11*	*1.6*
	Barbecue			3	3.3
	Safe	30	1	3	1
	Trashcan	69	1.1	5	1.0
All		**648**	**2.0**	**354**	**3.8**

source (to ensure that the val set is not too similar to the training set). We select all the shapes from 3D-FUTURE [14] (185) to use for training, and keep shapes from HSSD [24] (147) and ABO [7] (22) for evaluation. This results in a total of 645 training shapes (PM + 3DF), 169 validation shapes in ACD-val and 95 in PM-Openable-val. The additional data allows us to compare training with PM only, with addition of PM-Openable-ext, with ACD 3DF models, or with both added.

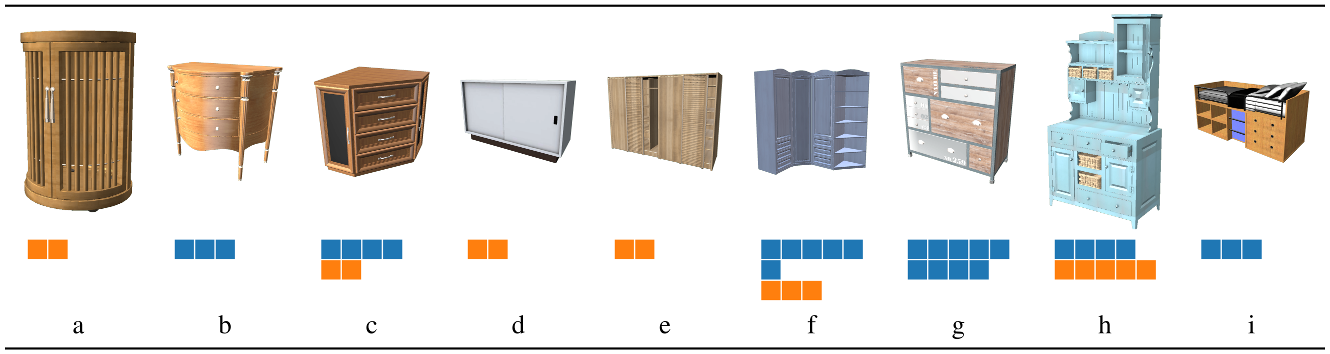

Figure 8. Examples from ACD showing a diversity of part and object shapes – round doors (a), curved drawers (b), corner cabinet (c), different motion types – translational doors (d,e), large objects (e,f), complex part arrangements (f,g,h), and openable parts in non-standard objects – drawers in beds (i). For each object, we also indicate the number of openable drawers (blue) and doors (orange).

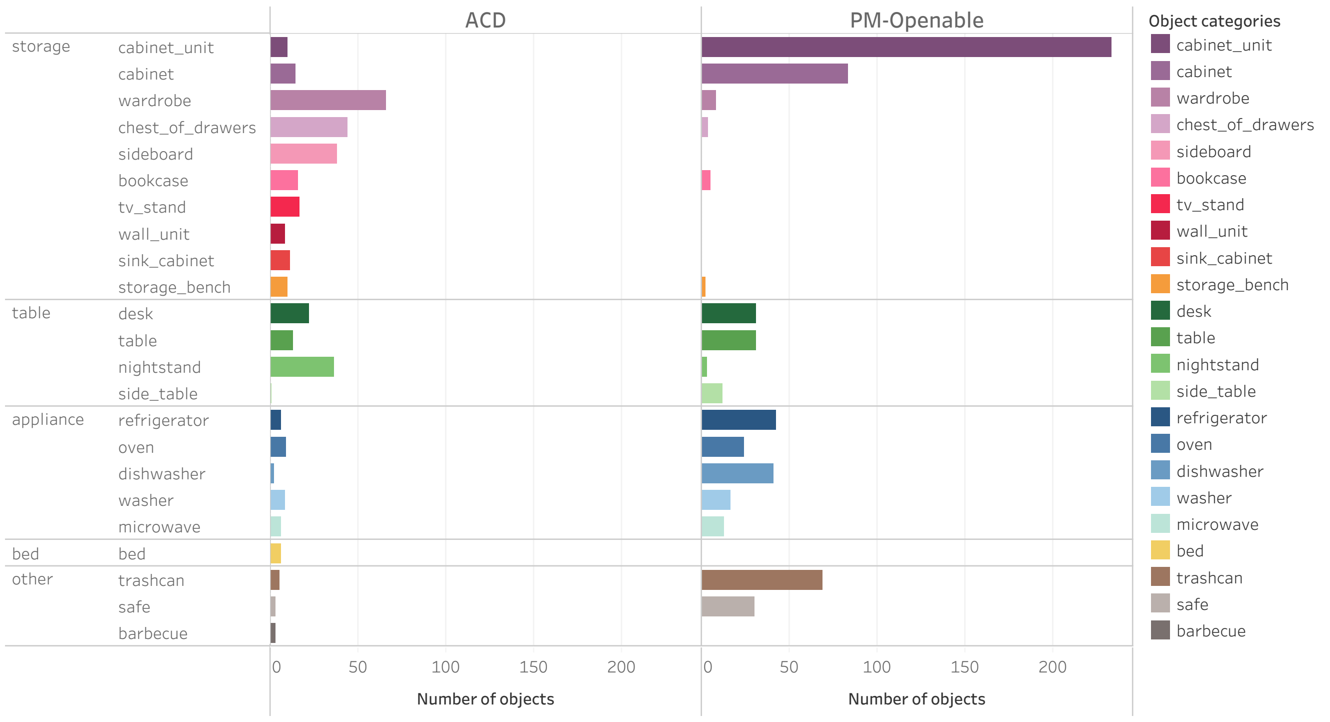

Figure 9. Comparison of the distribution of objects categories in our Articulated Containers Dataset (ACD) vs PM-Openable.

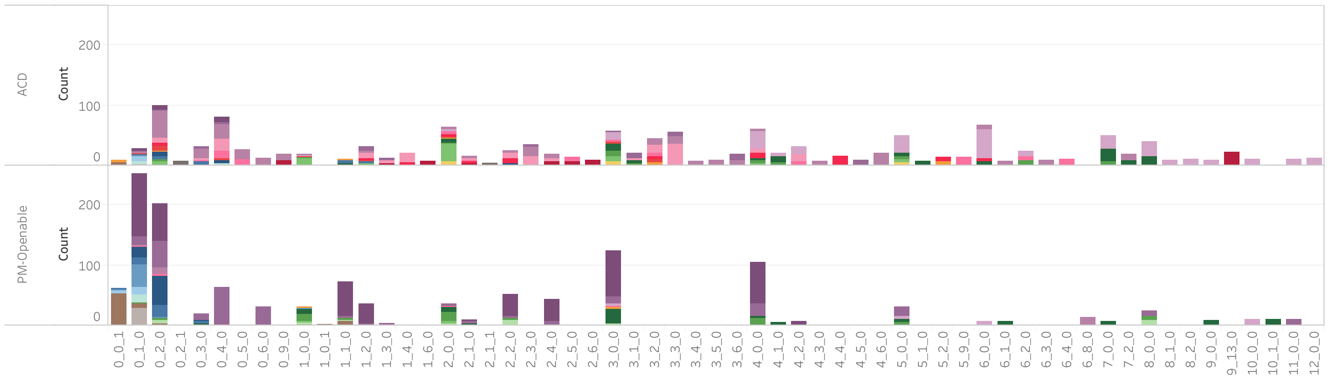

Figure 10. Comparison of the distribution of **part configuration** (drawers, doors, lids) of objects categories in our Articulated Containers Dataset (ACD) vs PM-Openable. For each object, we compute a *part code* that indicates the number of drawers, doors, and lids. For instance, `2_1_0` indicates the object has 2 drawers, 1 door, and 0 lids. In this figure, we plot the number of objects with each different part configuration colored by their object category. Storage furniture is in purple/pink/red, appliances in blue, and tables in greens (Fig. 9 for full legend). While PM-Openable is dominated by objects with just a few openable parts (often just drawers or doors), ACD shows a broader distribution of openable part configurations.

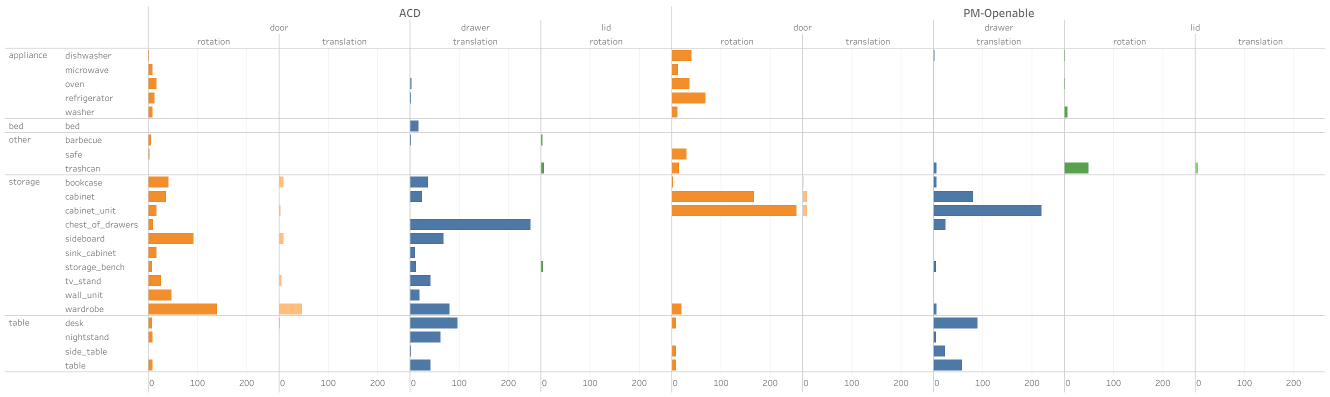

Figure 11. Comparison of the distribution of part **motion type** in objects categories in our Articulated Containers Dataset (ACD) vs PM-Openable. ACD has more diverse motion types for doors (more doors that translate) across a broader set of object categories.

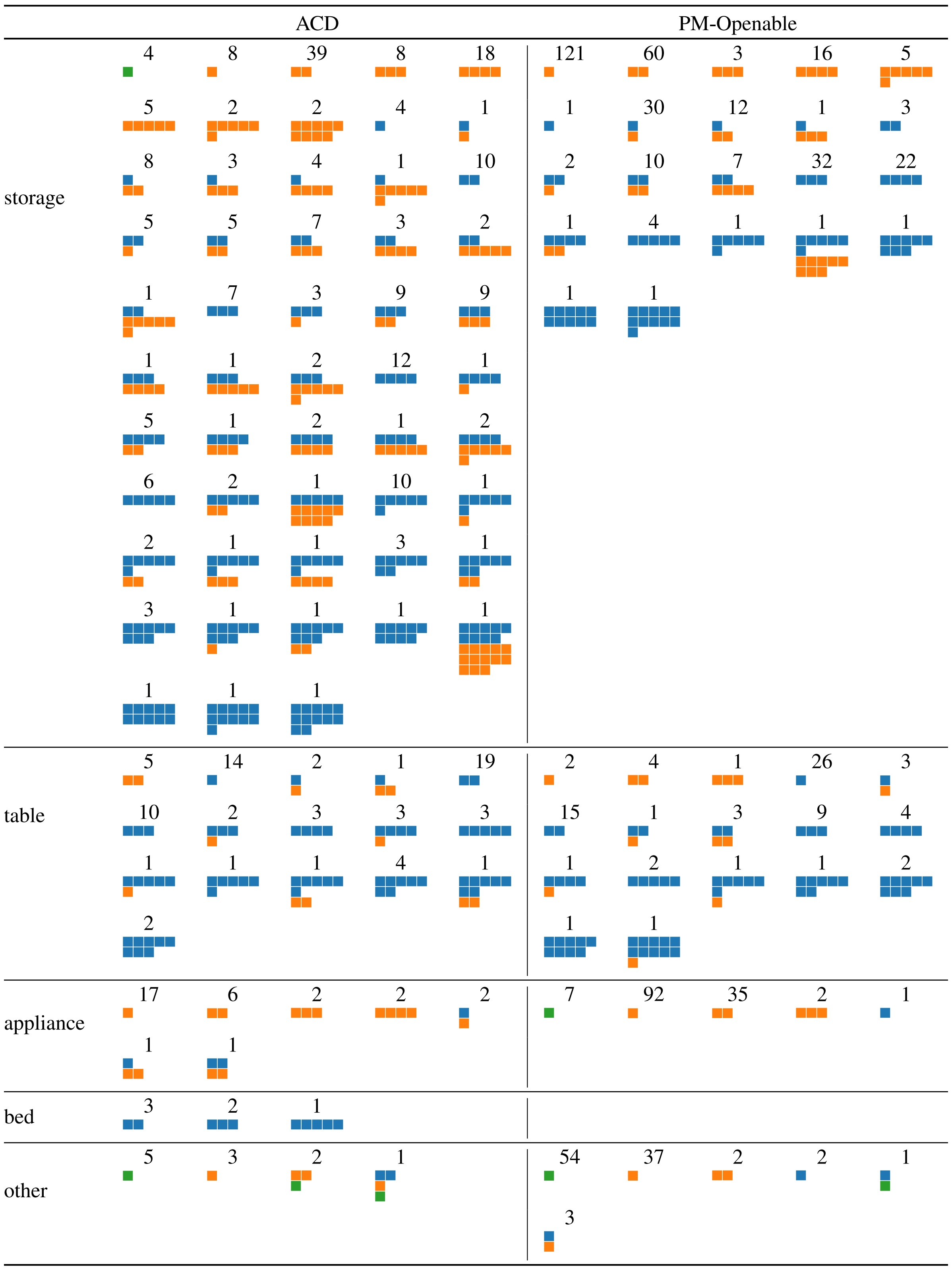

Figure 12. Summary visualization of part configurations for different object categories in the two datasets for our experiments (ACD on left and PM-Openable on right). Each colored block represents one openable part: blue are drawers, orange are doors, and green are lids. The numbers above each icon are the counts of objects with that part configuration.

A.3. Analysis of PM-Openable

To create PM-Openable, we select from PartNet-Mobility openable objects. In total, we obtain 648 objects (out of a total of 2346 objects). These openable containers make up roughly 28% of objects in PartNet-Mobility.

While PM-Openable contains a large number of objects, we find that objects in PM-Openable are highly similar (even across the train/val/test splits). This is especially true for category with the largest number of objects (Storage Furniture). In the main paper Fig. 3, we show examples from PM-Openable that are visually very similar. Here in Figure 13, we visualize IM-NET [5] shape embeddings for storage furniture in PM-Openable across the train/val/test sets. We project the embeddings using T-SNE [48] with the perplexity set to 10. As the figure shows, storage furniture in PM-Openable is highly repetitive within and across the train, val, and test splits. This clustering of highly similar objects across splits is indicative of both a lack of diversity and data leakage between the splits. Observation of these trends was one of our motivations for the construction of ACD. The above statistics and the object similarity analysis from Section 4 of the main paper show that ACD is more diverse and exhibits significantly less similarity between splits.

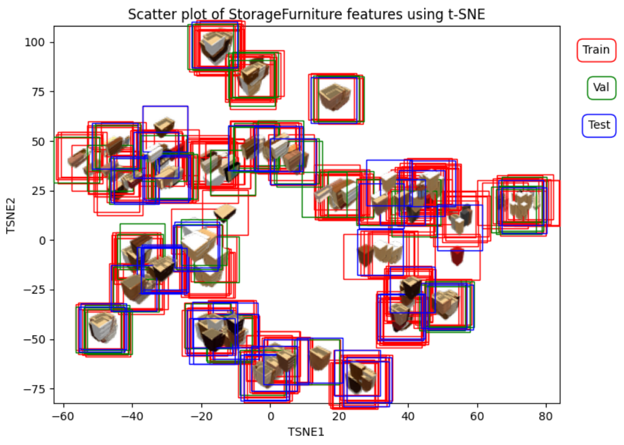

Figure 13. Projection of in PM-Openable storage furniture object embeddings using T-SNE. Note strong clustering of objects spanning train, val, and test splits. This is indicative of the lack of a general lack of diversity in object geometry, and some degree of data leakage between the splits.

A.4. Construction of PM-Openable-ext

The first step required to remove interiors from PM-Openable shapes is to perform an over-segmentation. As PM-Openable objects are originally human-modelled URDFs, they already come segmented in semantic parts. In order to segment them further, we employ connectivity-based segmentation which works relatively well. This is because the original 3D assets from which PM-Openable is constructed were authored by human designers.

Then, we proceed to render the indices of triangles visible from the views sampled uniformly around the shape and keep only the precomputed segments that have at least one triangle visible. This way, we eliminate the interiors while keeping the triangles connected to the outer layer of the shape which preserves some structure rather than having a single triangle layer equivalent to the scan. PM-Openable contains a significant number of shapes with missing countertops, which would exhibit artifacts with a straightforward application of the above algorithm. Therefore, we identify all such shapes and add a countertop surface via a simple algorithm as a pre-processing step. An example of an object before and after the full procedure can be found in Fig. 14.

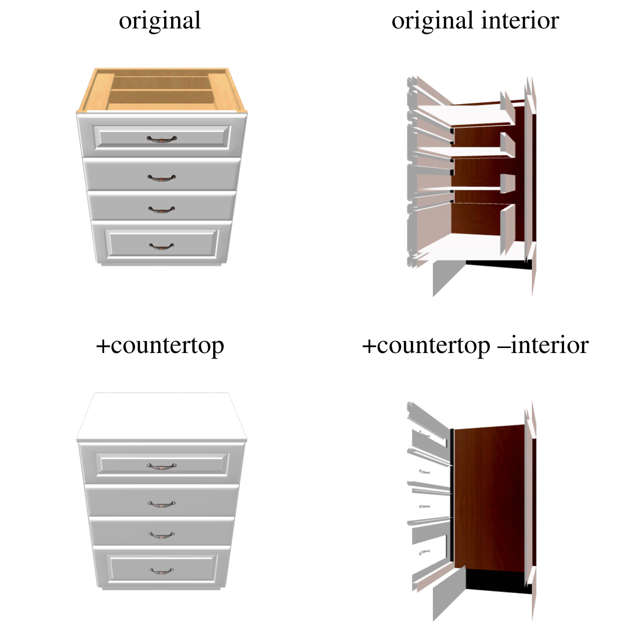

Figure 14. Example of an object in PM-Openable (top) with corresponding object from PM-Openable-ext (bottom) after addition of counterop surface and removal of interior geometry.

B. Part segmentation details

Here we describe baseline methods (Appendix B.1) and implementation details for the part segmentation stage of our framework. We provide details on our FPN for improving PointGroup (Appendix B.2), the point-cloud sampling (Appendix B.3), projecting predictions to the mesh from rendered images and sampled point-clouds (Appendix B.4), implementation and training details (Appendix B.5).

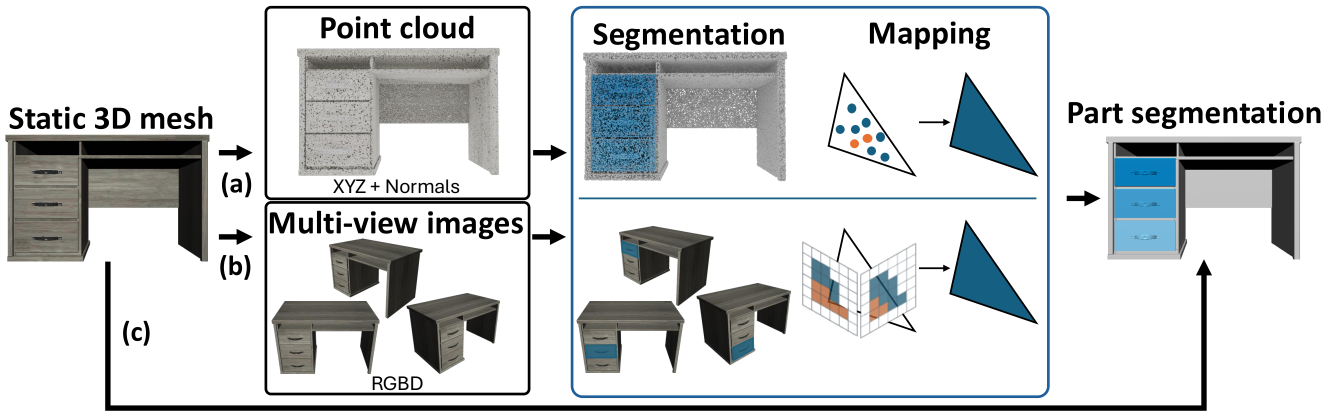

Figure 15. Illustration of our part segmentation experiments on different modalities: a) point cloud-based; b) image-based; and c) 3D mesh-based segmentation methods. The three modalities are all produced from the input mesh.

B.1. Part segmentation baselines

We consider obtaining part segmentation from different types of input: rendered images, sampled-point cloud, or directly on the mesh. Figure 15 illustrates how segmentation can be done for different modalities.

Image. We use OPDFORMER [46] for view-based segmentation as it is the state-of-the-art for openable part detection in images. We sample views from the training viewpoint distribution used by OPDFORMER, and render three RGB and depth views per object. To ensure that we can project the predicted segmentation back onto the mesh, we also render triangle indices for each view. See Appendix B.4 for image to mesh projection details.

Point-cloud. For point-cloud, we considered several baselines including SHAPE2MOTION [49] which is designed for segmenting articulated objects and motion prediction, and two commonly used object segmentation methods: POINTGROUP [21] and Mask3D [43]. See Appendix B.3 for point sampling details and Appendix B.4 for image to mesh projection details. Further, we describe POINTGROUP as it serves as a basis of our methods. The input point cloud is embedded via backbone, followed by a simple MLP. Then, two heads project the features into per-point semantic labels and offsets to centroids of parts. After instances are clustered, scoring module outputs the score per each part, acting as a confidence for predicted part. Finally, non-maximum suppression is employed to obtain the final instances. Semantic head is supervised via cross-entropy loss. The offset head is supervised via a combination of L1 loss comparing the points after predicted and ground truth offsets to the part's centroid are applied and the negative cosine similarity on offset vectors that targets the direction of the predicted offsets specifically. The score branch is supervised via binary cross-entropy with soft labels derived from IoU between predicted and ground truth instances.

Mesh. We use MESHWALKER [26], an RNN-based method that predicts semantic segmentation using random walks on the mesh vertices. This method makes far fewer assumptions such as manifoldness and watertightness on the input mesh compared to other mesh segmentation methods. Since the input 3D object meshes typically do not conform to such requirements, this method is ideal for practical application. Since MeshWalker only outputs semantic categories (i.e. it does not predict instances), we treat each semantic category as one instance.

B.2. FPN details

We introduce a simple Feature Pyramid Network (FPN) as a feature adapter module after the initial PointNeXt feature extraction in PointGroup. Our FPN consists of: pre-FPN convolution, FPN with 3 bottleneck blocks with the first one halving the feature dimension, followed by top-down convolutions per bottleneck block. Features are concatenated afterwards and passed to a post-FPN convolution that reduces them back to the original hidden dimension.

B.3. Point cloud sampling details

For training, we sample 200K points for each annotated part and apply farthest point sampling (FPS) to downsample to 20K points total for each object. During inference, to ensure every triangle is represented, we sample a point cloud with 1M points from the mesh, and add all triangle vertices as points for the methods leveraging triangle-based propagation so that all triangles are covered. To match training, we downsample to 20K points for inference, and predictions are mapped back to the high-res point cloud using k nearest neighbor lookup. We leverage the computed connectivity segmentation to guide the initial sampling of 1M points by distributing the number of samples for each connected component based on the area of the component, with a threshold if the area is too small. This allows to obtain detailed geometry even for small parts (e.g., handles). Such point clouds are, however, non-uniform. One would expect that non-uniformness would disappear after applying farthest point downsampling from 1M to 20k points, however we find that our models are not robust to the changes in sampling.

Then, to propagate predictions to the original point cloud from 20K subset, we apply a k-nearest neighbors lookup to map predictions to all points in the point cloud. We set $k = 3$ in our experiments.

B.4. Projecting predictions to mesh from other modalities

General algorithm. We leverage mapping from other modality to mesh triangles to aggregate per-triangle predictions. We keep instance predictions with confidence greater than a threshold of 0.9. It is possible that predicted part masks overlap when projected to triangles. To reconcile different predictions, we consider all predictions in order of confidence, from highest to lowest. Overlapping masks in a triangle with triangle-area weighted IoU greater than 0.8 are either: 1) merged into one part and we take the label from the prediction with higher confidence; 2) compared via confidence and only the mask with the highest confidence is kept. The former approach is adapted for images due to their multi-view nature, whereas the latter is used with point clouds. Otherwise, we retain two separate parts and assign the triangle to the higher confidence prediction. By reasoning with projected masks (vs individual triangles), we ensure that part predictions on the mesh are not broken into smaller interleaving parts.

Image to mesh propagation. We leverage rendered triangle indices to project predictions from pixels of multiviews to mesh triangles. Note that this approach assumes confidence scores are comparable across views. Triangles not observed in any view are assigned to the *base* part.

Point cloud to mesh propagation. The simplest algorithm uses per-point indices of triangles recorded during sampling process. Semantic instance labels are assigned by majority voting per triangle, which guarantees at least three votes since vertices are included in our point clouds during inference. Additionally, we introduce **topology-aware propagation** that leverages voting for over-segmentation rather than individual triangles. Hence, leveraging per-point over-segment ids and corresponding oversegment-to-triangle mappings for propagation. The over-segmentation is obtained by computing a connectivity segmentation of the mesh. Two triangles are considered connected if they share at least a single identical vertex index. We find that such over-segmentation often yields semantically-meaningful components, though at a finer level (i.e., drawer face, separate sides of drawer boxes, parts of drawer handles). Thus, propagating the labels to such segments yields much more complete segmentation than using separate triangles. This, however, improves the results only for a model that is sufficiently good, while incorporating such procedure for the bad models tends to propagate the errors further. It also eliminates the need of including the vertices in the point clouds during inference as there is no concern of zero points being sampled from a segment.

Table 9. Training parameters for part segmentation.

Method	dim	optimizer	learning rate	batch size	epochs
PG + U-NET	16	ADAM	0.002	4	500
PG + SWIN3D	48	ADAMW	0.006	8	600
PG + POINTNEXT	48	ADAMW	0.002	8	500
MASK3D	128	ADAMW	0.0001	32	600

B.5. Training and implementation details

Implementation details. All point cloud methods use only point coordinates and normals as input features. We use a POINTGROUP (PG) re-implementation[1] for the U-NET backbone, and adapt it to work with POINTNEXT[2]. For PG with Swin3D, we use the Pointcept implementation [8]. For SHAPE2MOTION, we use the re-implementation from Mao et al. [37]. We follow the suggested hyperparameters from the original work and train until convergence (see supplement). For MASK3D, we query the transformer with 10 queries, setting the confidence threshold on predictions to 0.7 during inference. We use a pretrained OPDFORMER-P checkpoint that is trained on RGB-D data. This model takes images of size 256×256 as an input.

Training details. We train all models (except for OPDFORMER, which is pre-trained) on Nvidia RTX 2080Ti, A5000 and A40 GPUs. We train to convergence based ontrain set evaluation metrics, and find that training runs take from 5-12 hours for POINTGROUP variants, 12 hours for MASK3D, 17 hours for SHAPE2MOTION, and roughly four and a half days for MESHWALKER. Table 9 summarizes key training parameters for most of the methods we report in the main paper. Below, we provide training details for the other methods.

MESHWALKER is trained using ADAM optimizer with learning rate of 0.00001 for 600K iterations. All other parameters are set to defaults according to the original paper.

SHAPE2MOTION consists of three modules that are trained in stages. We train the Motion Part Proposal and Motion Attribute Proposal Modules for 500 epochs with batch size 8, then the Proposal Matching Module for 100 epochs with batch size 16, and finally the Motion Optimization Network for 100 epochs with batch size 8. We use learning rate 0.001 and ADAM optimizer for all stages, with other parameters (e.g., loss weights) set according to the original paper.

[1]https://github.com/3dlg-hcvc/minsu3d
[2]https://github.com/guochengqian/PointNeXt

C. Motion prediction details

Here we provided information about the motion prediction baselines (Appendix C.1) and details of our heuristic based motion prediction (Appendix C.2).

C.1. Motion prediction baselines

SHAPE2MOTION (S2M) predicts instance segmentations along with motions. It does not produce part semantic predictions off the shelf. We heuristically infer the part semantic label based on the predicted instance and corresponding mobility parameters. All parts with prismatic motion are labeled as drawers. Parts with revolute motion are split into: 1) vertical axis - labeled as door; 2) non-vertical axis with non-vertical average part normal (computed from normals of part points) - labeled as door; 3) other cases (horizontal axis and vertical average normal) - labeled as lid. For S2M, the input point clouds are downsampled to 4096 points, so we map its predictions first to our subset and then to the full inference point clouds using k nearest neighbor lookups.

C.2. Motion prediction heuristic

For motion prediction, we designed a heuristic-based method (introduced in Section 5.2 of the main paper). Here we provide more details about the heuristics we used.

For the motion type, we utilize the most common motion type for each part semantic category following the statistics from the train set. Specifically, prismatic motion is assigned to drawers, while revolute motion is assigned to doors and lids. For the prismatic joint, the heuristic is simple: we use the given front direction of the openable part as the motion axis direction. For the revolute joint, we assume the motion axis should be on one of the edges of the bounding box of the openable part (four edges on the front face and four edges on the back face).

To determine if the axis is on the front face or the back face, we check its alignment with the base part. We observe that in most cases, the motion axis should also be very close to the edges of the base part. Leveraging this observation, our heuristic determines whether the front face or the back face of the openable part is closer to the edges in the bounding box of the base part.

After selecting the face, we still have four edges from which to choose. To handle this choice, we follow a geometric heuristic that finds the handle position first, and sets the axis line on the opposite edge of the handle position. To determine the handle position, we leverage the assumption that the handle geometry is more complex compared to the whole openable part. There are mainly two cases: the handle is raised (e.g., the door of the cabinet) or concave (e.g., the door of the washing machine). In both cases, there is some asymmetry in the geometry. We detect this asymmetry by binning vertex densities from front to back and then locate the handle position accordingly. We check the number of points in each bin to detect the handle and infer the edge that serves as the motion axis, and applying symmetry-based reasoning to select direction and origin for revolute joints.

Table 10. Ablation of choices made for FPNGROUPMOT construction and their effects on segmentation. OO stands for predicting offsets to origin rather than axis. OSV stands for voting for over-segmentation.

		PM-Openable			ACD		
OO	OSV	P	R	F1	P	R	F1
✗	✗	69.7	62.7	65.8	21.8	5.3	8.5
✓	✗	75.9	65.9	70.4	**24.8**	**5.1**	**8.4**
✓	✓	**77.3**	**66.9**	**71.6**	24.5	4.5	7.5

D. Additional Experiments

We provide additional results including ablation experiments (Appendix D.1) for informing the design of our models, evaluation of point-cloud segmentation directly on point-clouds (Appendix D.2.1), evaluation of mesh segmentation using additional metrics (Appendix D.2.2).

D.1. Ablation Experiments

D.1.1. FPNGROUP ablations

To construct FPNGROUP, we ablated a number of backbones in Table 4. The losses used follow POINTGROUP. Overall, we find that recent backbones provide a significant improvement compared to the original U-NET. We find that using POINTNEXT leads to superior results on PM-Openable but lags behind the SWIN3D when it comes to generalizing on ACD. We find that adding our FPN feature adapter leads to a very significant performance improvement, especially when it comes to generalizing on ACD. Employing our topology-aware voting procedure pushes results on PM-Openable further, while the results on ACD deteriorate slightly. This is expected as voting for over-segmentation while the predictions are not very good could lead to propagating these errors further. This, however, changes with the additional data added into the training split as can be seen in the version trained on PM-Openable-ext in addition. Any further improvements in training data would benefit from such propagation procedure even further. Thus, we select these modifications to be FPNGROUP.

D.1.2. FPNGROUPMOT ablations

In Tables 10 and 11 we provide the ablations for segmentation and motion prediction capabilities of FPNGROUP-MOT.

Add motion prediction to FPNGROUP. We extend our FPNGROUP for motion prediction by adding motion prediction support similar to those used in GAMMA [58], which uses a submodule for segmentation and a separate

Table 11. Ablation of FPNGroupMot design choices and their effects on motion prediction, evaluated on PM-Openable. CE stands for using cross-entropy loss for motion types. OO stands for predicting offsets to origin rather than axis. OSV stands for voting for over-segmentation. EA stands for edge-aware motion predictions post-processing.

					F1 % ↑			Error ↓	
CE	OO	OSV	EA	#	+M	+MA	+MAO	AE	OE
✗	✗	✗	✗	**137**	66.4	55.0	27.5	13.4	0.27
✓	✗	✗	✗	131	66.2	57.0	26.5	9.7	0.27
✓	✓	✗	✗	131	70.2	59.9	34.2	9.8	0.25
✓	✓	✓	✗	133	**71.2**	60.9	35.1	9.7	0.25
✓	✓	✓	✓	133	**71.2**	**61.9**	**50.6**	**9.5**	**0.18**

submodule for motion prediction. Following our FPNGroup, we use two FPNs for preparing features for the two submodules. This is akin to taking GAMMA and replacing the convolutional projection layers with FPNs after the backbone and before each of the submodules.

We also propose a number of additional changes to the initial GAMMA formulation. We simplify the motion type loss from combination the focal loss to cross-entropy. Axis directions and offsets to origin are supervised using the PointGroup losses - L1 on vector norm and negative cosine similarity on vector direction. Moreover, we remove the use of combination of both offsets to part centroid from segmentation submodule and offsets to motion axis from motion submodule that were used for segmentation in GAMMA and stick with only the latter as in the original formulation of PointGroup.

The next modification we add is predicting offsets not to motion axis but to motion origin in motion submodule. While this modification targets improving motion predictions, we find that it is also quite beneficial for segmentation. Since both task are learned together, it is hard to decouple their effects but our hypothesis is that it is easier to learn offsets to the origin which benefits the overall training stability. Finally, we employ topology-aware voting procedure which further pushes the segmentation results. We see minor reduction in performance on ACD, similar and as discussed by Appendix D.1.1.

In summary, the ablation terms are: CE - using a **c**ross-**e**ntropy loss as compared to a focal loss, OO - predicting **o**ffsets to **o**rigin rather than predicting offsets to axis and extracting the origin from there, OSV - **o**ver-**s**egmentation **v**oting, EA - **e**dge-**a**ware post-processing of predictions. Simplifying the loss does not decrease the motion type performance significantly, however leads to improvements in axis error. We also note that the simplified loss leads to better generalization to ACD. We see that further improvements subsequently enhance the motion prediction capabilities. We see a large improvement in MAO F1 score when predicting offsets to origin directly rather than axis from

Table 12. Point cloud segmentation evaluation. Overall, Articulated Containers Dataset (ACD) is much more challenging than PM-Openable. Note that S2M does not output semantic labels, therefore evaluation considers two classes: base vs openable part.

	PM-Openable			ACD		
Method	mAP↑	mAR↑	OC↓	mAP↑	mAR↑	OC↓
PG + U-Net	38.0	44.6	0.236	17.4	22.3	0.357
PG + Swin3D	41.0	51.2	0.209	17.6	25.5	0.339
PG + PX	51.3	55.7	0.198	14.9	22.8	0.357
FPNGroup	**69.3**	**75.3**	**0.099**	21.0	27.8	0.324
FPNGroup*	67.8	72.4	0.117	**28.2**	**33.6**	**0.303**
FPNGroupMot	56.5	60.0	0.142	19.2	24.6	0.338
FPNGroupMot*	51.7	55.5	0.168	18.5	23.0	0.344
Mask3D	34.9	41.5	0.250	15.1	18.3	0.370
S2M	11.2	13.6	0.323	15.2	16.9	0.375

26.5 to 34.2 points. Voting for over-segments further improves the results a little.

Post-processing. Finally, we propose a post-processing step inspired by our mobility heuristic. As the motion axis is extracted by majority voting, we decide to straighten it as in our data, aligning to world coordinate axes. Thus, we select the most dominant direction by absolute value, set it as 1 with corresponding sign and all other values are set to 0. We see that this results in minor improvement in MA F1 score from 60.9 to 61.9 points. Finally, we also post-process the predicted origin. As we voted for over-segmentations, we now likely have much better part bounds than with the initial point cloud predictions. We note that outputs of GAMMA are actually not guaranteed to respect segmentation bounds as motion and segmentation predictions are done independently. We improve on it by computing oriented bounding boxes for the predicted doors and lids and finding the closest corner to the predicted motion origin. Then, we set that corner to be a new origin. This results in a significant improvement in origin quality, as can be seen from MAO F1 score improvement of 15.5 points as well as reduction in origin error.

D.2. Part segmentation

We provide additional results for part segmentation, including evaluation directly on the point-cloud (Appendix D.2.1, as well as a comparison of different approaches (image based, point-cloud based, mesh based) on mesh segmentation (Appendix D.2.2). For point-cloud based methods, we compare Shape2Motion [49], Mask3D [43], PointGroup [21] with our proposed FPNGroup. For PointGroup, we also compare the performance of different backbones: the original U-Net, Swin3D [56], and PointNeXt [40] which is the basis of our FPNGroup. We also report the segmentation performance of our FPNGroupMot which includes a submodule for motion prediction. In these experiments, we use * as an shorthand to indicate that

Table 13. Breakdown per part category for evaluation of point cloud segmentation methods. Segmentation of openable parts is challenging, especially for drawers in the Articulated Containers Dataset data dataset where interior geometry behind the drawer is not typically available as a useful signal.

		Drawer		Door		Lid		Base	
	Method	AP↑	AR↑	AP↑	AR↑	AP↑	AR↑	AP↑	AR↑
PM-Openable	PG + U-NET	15.6	18.4	40.9	43.9	19.6	29.6	76.0	86.7
	PG + SWIN3D	12.6	20.1	51.3	61.1	9.8	30.9	90.1	92.5
	PG + PX	11.4	13.2	52.5	59.6	55.2	58.0	85.9	91.9
	FPNGROUP	55.2	**63.9**	69.7	**77.2**	**64.9**	**66.7**	87.4	93.5
	FPNGROUP*	**59.9**	63.6	**74.5**	76.5	49.0	55.6	**87.9**	**94.2**
	FPNGROUPMOT	52.0	54.4	54.5	58.4	31.4	35.8	87.4	91.2
	FPNGROUPMOT*	55.1	55.4	49.0	51.2	18.6	24.7	84.1	90.9
	MASK3D	30.7	36.7	22.7	27.4	18.4	27.2	67.8	74.6
	S2M	0.0	0.4	0.0	0.4	0.0	0.0	44.6	53.7
ACD	PG + U-NET	0.1	0.4	1.5	4.1	12.5	15.7	55.5	69.2
	PG + SWIN3D	0.0	0.2	2.2	8.6	2.5	15.7	65.7	77.3
	PG + PX	0.0	0.1	2.4	4.6	6.0	20.4	51.3	65.9
	FPNGROUP	0.0	0.2	8.3	13.3	14.1	25.0	61.6	72.9
	FPNGROUP*	**0.9**	**3.6**	**11.5**	**18.0**	**30.5**	**31.5**	**69.9**	**81.5**
	FPNGROUPMOT	0.0	0.2	4.1	8.8	20.2	23.1	52.5	66.2
	FPNGROUPMOT*	0.0	0.5	2.7	5.7	15.5	17.6	55.7	68.4
	MASK3D	0.0	0.0	0.9	3.5	0.9	0.9	58.4	68.9
	S2M	0.0	0.0	0.0	0.1	0.0	0.0	60.9	67.4

the model was trained using both PM-Openable and PM-Openable-ext. By default, only PM-Openable was used in training.

Results consistently show that our FPNGROUP *outperforms other methods*, with the model trained with additional PM-Openable-ext data (FPNGROUP*) having better performance on ACD. We provide more details on these additional experiments and results below.

D.2.1. Point-cloud segmentation

Metrics. We report the **mAP** and **mAR** metrics for point cloud segmentation following prior work [21]. Although these metrics are commonly used, we find that they have flaws: 1) they do not take false positives into account if their confidence is lower than the confidence of the true positive prediction; and 2) mAP and mAR are aggregated across classes and therefore do not weigh each shape equally. Therefore, we report another metric that does not have these issues: **OC-COST** by Otani et al. [38]. OC-COST (OC) evaluates the part segmentation quality for each shape by measuring the cost of correcting the predicted segmentation to match the ground truth segmentation. The metric was originally proposed for images and uses GIoU [42]. We adapt it to shapes by computing GIoU on 3D bounding boxes.

Results. Our FPNGROUP has considerably higher performance than the baseline without FPN, and the best overall performance on PM-Openable (Table 12) with the highest mAP (69.3%) and mAR (75.3%) and lowest OC-COST. As expected, training on the mixed PM-Openable and PM-Openable-ext data gives better performance on ACD, as PM-Openable-ext provides a closer distribution to ACD. At the same time, ACD remains a challenging benchmark due to more diverse and challenging shapes. ACD shapes typically do not have complete interior geometry thus the useful signal of a drawer box attached behind the front is lost. In fact, we see that point cloud based methods mistake drawers for doors on multiple occasions (see Fig. 16). However, after some data exhibiting missing interiors is added, FPNGROUP and FPNGROUPMOT learn to generalize better at segmenting drawers without interiors. Similarly, shelves behind the doors that are present in PM-Openable are mostly missing in ACD which also explains lower performance on this part category. Additional challenges come from more diverse categories (i.e. BBQ grills) and large differences in real-world scale of the objects as, once resized during pre-processing, the openable parts become too small.

Breakdown by part-category. In Tab. 13, we examine the performance by part category. Not surprisingly, the base part is relatively easy (with the highest performance). There is a significant drop in performance for drawers and doors when going from PM-Openable to ACD.

D.2.2. Mesh segmentation

Standard mesh segmentation metrics typically measure the overall class accuracy of the segmentation [26]. As the base part dominates the mesh surface (i.e. most of the surface is not an openable part), it is possible to achieve a high accuracy by only predicting the base. Thus in the main paper, we reported the precision (P), recall (R), and F1 metrics macro-averaged over object instances, measuring how well openable parts are identified at the object level. Here, we report the P/R/F1 micro-averaged over part instances as well as additional accuracy based mesh metrics used in prior work.

Full results with part-level averages In Tab. 14 and Tab. 15, we report the P/R/F1 micro-averaged over part instances. We see the micro-average P/R/F1 tend to trend with the macro-averages, and that the methods follow the same ordering in performance. *Comparison of predicting on point-cloud vs image vs mesh.* In our experiments, we find point-cloud based methods to be the most effective. The mesh-based method (MESHWALKER) cannot reliably identify openable parts. This is likely because these openable parts do not have geometry that protrudes from the object base body to provide a signal that they can be opened (i.e. door handles are often not represented in enough geometric detail). In this work, we only considered one view-based method, OPDFORMER. While it did not do as well as the point cloud methods, a stronger base model (e.g. SAM [25]) and improved aggregation schemes may improve the performance of view-based models.

Additional metrics. In Tab. 16 we report mesh segmentation metrics following prior work in mesh segmentation [23] that focuses on the area of the shape that is se-

Table 14. Mesh segmentation of openable parts. We report the precision (P), recall (R), and F1 for openable parts (base part is excluded) at IoU=0.5. Note that a subset of macro-averaged metrics was reported in the main paper.

		PM-Openable						ACD					
		Micro			Macro			Micro			Macro		
Type	Method	P	R	F1	P	R	F1	P	R	F1	P	R	F1
	S2M	3.2	1.6	2.2	1.2	0.5	0.7	3.7	0.8	1.3	2.2	0.5	0.8
	Mask3D	60.0	42.9	50.0	52.6	33.9	41.1	15.5	4.9	7.5	19.2	5.0	9.2
	FPNGroup	**90.8**	**81.3**	**85.8**	**92.2**	**83.0**	**87.2**	42.3	11.3	17.8	30.0	7.0	11.2
PC	FPNGroup*	88.0	76.4	81.8	85.2	74.8	79.4	**43.1**	**19.0**	**26.4**	**37.0**	**15.5**	**21.9**
	FPNGroupMot	82.6	73.1	77.6	77.3	66.9	71.6	35.9	7.9	13.0	24.5	4.5	7.5
	FPNGroupMot*	81.3	62.1	70.4	74.5	55.1	63.2	33.2	6.4	10.8	24.9	3.7	6.5
Mesh	MeshWalker	1.7	1.6	1.7	1.0	1.0	1.0	1.1	0.7	0.8	1.2	0.7	0.9
View	OPDFormer	0.3	0.5	0.4	0.7	1.2	0.9	1.6	1.0	1.2	1.8	1.0	1.3

Table 15. Mesh segmentation results on PM-Openable and ACD-val, including FPNGroup and FPNGroupMot trained on a subset of ACD. Training on PM-OE +3DF helps FPNGroup with performance on PM-Openable, also providing an improvement on ACD-val. However, having PM-Openable-ext additionally in the training split pushes the performance on ACD-val even further. FPNGroupMot, on the other hand, benefits the most from having only 3DF added to the training split, likely due to need to predict the motion parameters as well. We note that macro averages have been reported in the main paper.

				PM-Openable						ACD-val (HSSD + ABO)					
	Data			Micro			Macro			Micro			Macro		
Method	PM-OE	3DF	Size	P	R	F1	P	R	F1	P	R	F1	P	R	F1
FPNGroup	✓	✗	920	88.0	76.4	81.8	85.2	74.8	79.4	47.2	25.3	33.0	47.1	28.0	34.8
FPNGroup	✗	✓	645	**91.8**	**86.3**	**89.0**	**89.8**	**85.8**	**87.6**	63.5	47.3	54.2	66.6	52.8	58.8
FPNGroup	✓	✓	1105	84.9	77.5	81.0	75.4	69.2	71.8	**65.8**	**49.0**	**56.1**	**75.3**	**58.5**	**65.7**
FPNGroupMot	✓	✗	920	81.3	62.1	70.4	74.5	55.1	63.2	40.0	11.5	17.8	49.3	14.2	21.9
FPNGroupMot	✗	✓	645	82.2	68.7	74.9	72.4	60.4	65.6	58.7	41.6	48.7	62.6	44.2	51.8
FPNGroupMot	✓	✓	1105	86.7	71.4	78.3	79.3	64.7	71.1	47.0	29.9	36.6	55.0	35.3	42.9

mantically accurately labeled and compares the precise instance segmentation to the ground-truth. However, we find the original naming[3] to be misleading and therefore propose the following names: Classification Accuracy (**CA**) for Equation 12 from the paper and Normalized Classification Accuracy (**NCA**) for Equation 13. The metrics are defined as below:

$$\mathrm{CA} = \frac{\sum_i \mathbb{1}[c_i, \hat{c}_i] a_i}{\sum_i a_i}, \quad \mathrm{NCA} = \sum_i \mathbb{1}[c_i, \hat{c}_i] \frac{a_i}{A_{c_i}}$$

where a_i is the area of the ith face, c_i is the ground truth and $\hat{c}_i$ the predicted label, $\mathbb{1}(c, c')$ is the standard indicator function which equals to 1 if the two labels match and 0 otherwise; A_{c_i} in the NCA formulation is the total area of a segment with ground truth label c_i. NCA is effectively the average of the per-label accuracy while CA is the face-area weighted segmentation accuracy. As both NCA and CA are semantic accuracies (e.g. do not account for instances), we follow Kalogerakis et al. [23] and also report the Adjusted Rand Index (**ARI**) which measures agreement between the ground truth and predicted instance segmentations (adjusted for chance groupings).

As the CA and NCA metrics take the base part into account, the accuracy numbers may be inflated since the base part constitutes the largest part of the object. For instance, even when no openable parts are predicted, the CA and NCA metrics will still be high due to high accuracy on predicting the base. Thus, we introduce $\mathrm{CA}_{\mathrm{nb}}$, a variant of CA which only considers openable parts (no base). $\mathrm{CA}_{\mathrm{nb}}$ aggregates matching predictions over the union of GT openable parts and predicted openable parts only. This way, both cases of over- and under-segmentation are considered.

Results. Table 16 shows results of mesh segmentation predictions, suggesting that point cloud-based methods outperform all other methods. We note that as base parts are typically the biggest parts in all of the objects, they significantly influence the metric results. The performance based on CA

[3]Metrics were originally named ‘Classification Error’ and ‘Segment-Weighted Error’

Table 16. Evaluation of mesh segmentation following metrics from Kalogerakis et al. [23]. CA is the face-area weighted semantic accuracy, NCA is the average per-label accuracy, and $\text{CA}_{\text{nb}}$is a variant of CA that excludes the base part. The Adjusted Rand Index (ARI) measures the quality of the instance segmentation. For CA and NCA, the classification accuracy is dominated by the base part, these results overestimate the ability of the different methods to detect and identify the openable parts. As P/R/F1 metrics from the main paper use matches based on ixed area threshold, they do not assess how much of the area gets a correct label assigned. We note that FPNGROUP * is dominant on PM-Openable with these metrics rather than expected FPNGROUP. This might indicate that FPNGROUP * produces more partial segmentations less than the area threshold. On ACD, we see that FPNGROUP * is dominant by quite a large margin across all metrics. Overall, results deteriorate significantly due to the dataset complexity. The * suffix denotes training on a mix of PM-Openable and PM-Openable-ext.

		PM-Openable				ACD			
Type	Method	CA↑	NCA↑	ARI↑	CA_{nb}↑	CA↑	NCA↑	ARI↑	CA_{nb}↑
PC	PG + U-NET	91.7	87.1	0.473	69.7	81.4	53.2	0.143	19.3
	PG + SWIN3D	94.0	90.7	0.549	77.1	84.7	59.4	0.242	30.3
	PG + PX	95.2	91.3	0.511	81.3	80.7	52.0	0.112	17.3
	FPNGROUP	**96.9**	93.4	0.617	85.8	84.2	54.1	0.136	18.7
	FPNGROUP*	96.5	**94.4**	**0.754**	**86.4**	**87.7**	**67.1**	**0.358**	**41.4**
	FPNGROUPMOT	94.4	89.2	0.566	77.6	83.0	51.2	0.096	12.3
	FPNGROUPMOT*	94.2	86.9	0.574	72.9	84.3	53.7	0.133	16.9
	MASK3D	88.3	75.3	0.360	51.4	81.9	52.9	0.150	17.6
	S2M	78.4	55.3	0.030	13.7	82.5	46.0	0.008	3.8
Mesh	MESHWALKER	65.8	52.6	0.185	17.5	72.4	44.3	0.044	6.7
View	OPDFORMER	75.6	56.5	0.058	17.8	77.0	54.7	0.070	21.0

and NCA are significantly higher than when we focus on just the openable parts. We see with CA_{nb}, the impact of the base part is limited. But as CA, NCA, and CA_{nb} do not account for instance segmentation, we also report ARI (a perfect segmentation will achieve an ARI of 1). We find that compared to our metrics (precision, recall, F1 over detected parts), these metrics from prior work are less interpretable (ARI) or do not properly evaluate the instance segmentation and are too heavily influenced by the base part (CA, NCA).

Discussion. Despite these flaws, results from these metrics show mostly the same trend as we report in the main paper, with the FPNGROUP variants being the top performers for point-cloud methods and a large drop in performance as we go from PM-Openable to ACD, showing the challenge of ACD. As well as SHAPE2MOTION being the worst-performing PC-based method. We see that OPDFORMER under-performs compared to other methods (except for MESHWALKER). This is likely due to overprediction by OPDFORMER and these metrics being area-weighted vs considering detected instances. The overprediction is due to the fact that there can be multiple predictions from different viewpoints that may predict either the same or different part instances. While our heuristic for dealing with overlapping masks tries to solve some of these issues, there might still be some "leftover" masks after splitting. The ARI metric gives a good measure of how well the predicted segmentation matches the ground truth

Table 17. Motion prediction precision and recall on PM-Openable. FPNGROUP with our heuristic works the best. When comparing FPNGROUPMOT and heuristic prediction given the same (FPNGROUPMOT) segmentation, we find precision and recall comparable for M and MA but our heuristic clearly outperforming when it comes to MAO. Other baselines perform poorly.

			Precision % ↑			Recall % ↑		
Motion	Segmentation	#	+M	+MA	+MAO	+M	+MA	+MAO
Learned	SHAPE2MOTION	3	1.1	1.1	1.1	0.8	0.8	0.8
Learned	FPNGROUPMOT	133	71.1	61.7	50.7	71.3	62.2	50.5
Heur	MASK3D	78	42.4	37.5	30.5	36.4	31.2	24.5
Heur	FPNGROUPMOT	133	71.0	62.4	58.0	71.3	61.7	57.0
Heur	FPNGROUP	**148**	**79.6**	**75.4**	**60.9**	**78.4**	**70.3**	**59.1**
Heur	GT	182	94.7	88.4	84.5	94.7	88.4	84.5

(OPDFORMER also performs poorly here), but it is less easy to determine why the match is poor, while the separation into precision and recall allows us to determine that OPDFORMER is overpredicting.

We also find that MESHWALKER under-performs significantly compared to other methods. We hypothesize this is due to PM-Openable and our mesh data in general having highly non-uniform vertex distribution with curved parts such as handles being far denser than flat surfaces, severely impacting the random walks on which the method is based.

In terms of performance on ACD, while CA may be not that susceptible to the errors, NCA and ARI show significant performance degradation.

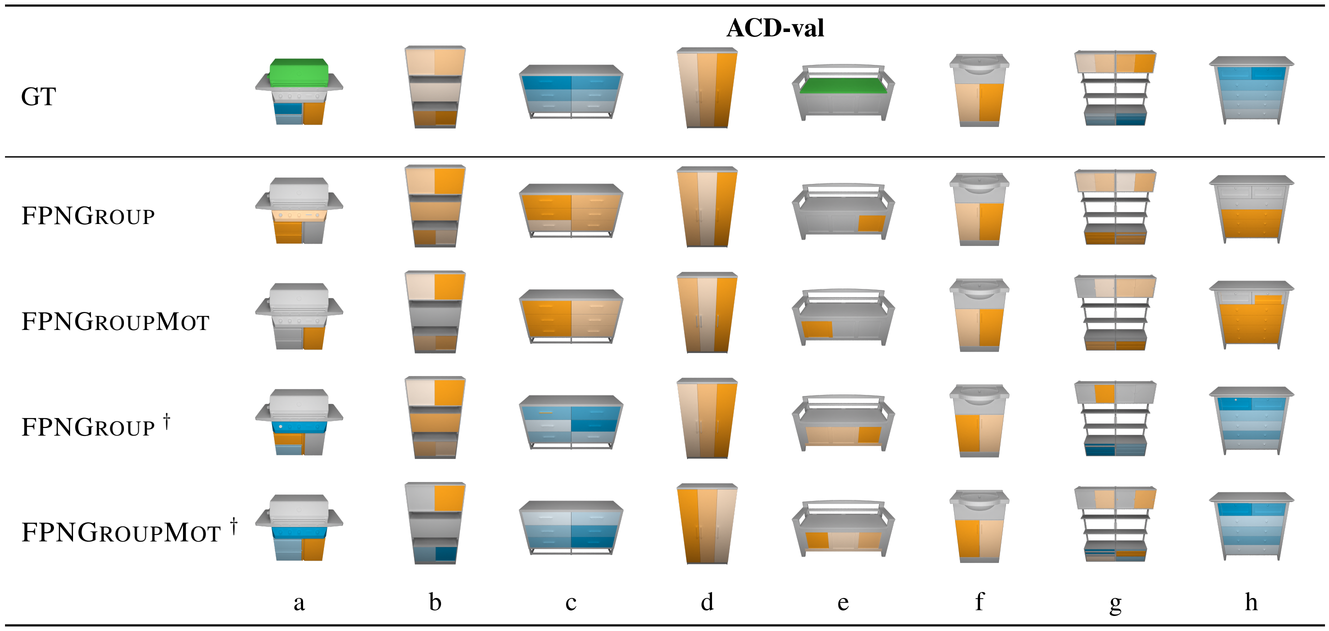

Figure 16. Comparison of our methods trained on PM-O and PM-O + ACD-train (†) variants. We show drawers in blue, doors in orange, and lids in green. As PM-O contains interiors for drawers, models trained on it struggle to generalize on ACD as it lacks interiors. WE notice that the models tend to be predicting doors instead of drawers (c, g, h). Adding ACD-train to the training split, helps both FPNGROUP and FPNGROUPMOT to both separate the instances and identify the drawers (c, g, h). We find that lids are still hard to segment as the data is scarce (a, e). Models also struggle to generalize on the objects that are large in the real-world, as after resizing the openable parts become smaller (g).

Table 18. Motion prediction precision and recall on ACD. Performance overall is quite low, as segmentation remains being the bottleneck. We find FPNGROUP, providing the best segmentation, with our heuristic is the best option in the setup with no additional training.

			Precision % ↑			Recall % ↑		
Motion	Segmentation	#	+M	+MA	+MAO	+M	+MA	+MAO
Learned	SHAPE2MOTION	10	1.1	1.1	0.9	1.2	1.2	0.8
Learned	FPNGROUPMOT	106	13.6	11.9	6.4	0.9	0.8	4.3
Heur	MASK3D	66	8.4	7.8	3.8	5.9	5.0	2.8
Heur	FPNGROUPMOT	106	13.8	9.3	7.2	9.1	6.6	5.1
Heur	FPNGROUP	**151**	**21.3**	**14.2**	**9.5**	**15.2**	**9.8**	**6.4**
Heur	GT	1350	94.9	81.0	65.3	94.9	81.0	65.3

D.3. Motion Prediction

Tables 17 to 19 present precision and recall values for motion prediction, in addition to the F1 scores reported in the main paper. In these experiments, PM-Openable was used in training by default. We use * as an indication that the model was trained using both PM-Openable-ext and † to indicate that ACD-train was also used for training.

We further select the best checkpoints for FPNGROUP and FPNGROUPMOT trained on additional data, according to their segmentation performance on ACD-val as seen in Tab. 3 and study the performance on ACD-val. Overall, the metrics are noticeably improved but we still find that the

Table 19. Motion prediction precision and recall on ACD-val. Overall, our heuristic dominates in terms of precision while recall is comparable when given the same FPNGROUPMOT segmentation. With FPNGROUP segmentation, heuristic-based approach is clearly dominant. We note that there is still quite a bit of room until heuristic reaches its upper-bound, meaning that stronger segmentaiton is required.

			Precision % ↑			Recall % ↑		
Motion	Segmentation	#	+M	+MA	+MAO	+M	+MA	+MAO
Learned	FPNGROUPMOT †	225	44.4	37.4	26.3	39.7	33.4	25.1
Heur	FPNGROUPMOT †	225	44.8	37.3	30.3	40.3	33.5	27.7
Heur	FPNGROUP *†	**265**	**58.4**	**46.1**	**36.9**	**53.0**	**40.8**	**32.1**
Heur	GT	541	96.7	82.4	76.9	96.7	82.4	76.9

Table 20. Motion prediction evaluation on ACD-val. We select the best checkpoints trained on additional data and benchmark learned and heuristic-based motion prediction. We find that FPNGROUP with heuristic outperforms FPNGROUPMOT.

			F1 % ↑			Error ↓	
Motion	Segmentation	#	+M	+MA	+MAO	AE	OE
Learned	FPNGROUPMOT †	225	41.9	35.3	25.7	**11.2**	0.40
Heur	FPNGROUPMOT †	225	42.4	35.3	29.0	11.6	**0.27**
Heur	FPNGROUP *†	**265**	**55.6**	**43.3**	**34.3**	15.1	0.30
Heur	GT	541	96.7	82.4	76.9	13.6	0.18

Table 21. Interior and object reconstruction evaluation. Given a shape from PM-Openable-ext as an input, we evaluate against shapes from PM-Openable. We note that our pipeline allows to preserve the original geometry. We report the chamfer distance over all parts (CD), and F1 scores for drawers. We note that SINGAPO has seen 83% of these shapes during training not accounting for augmented data.

Method	CD	F1
FPNGROUP *† + Heuristic	**1.7**	37.3
SINGAPO	3.3	**69.8**
URDFormer	23.4	0.0
GT + Heuristic	1.6	73.9

same trends hold as in evaluation on full ACD. There is a lot of room until heuristic hits its upper-bound as seen in GT segmentation row. We find that segmentation is the biggest challenge of the pipeline and bottlenecks downstream motion prediction and interior completion.

D.4. Interior completion

For quantitative evaluation, we leverage the correspondence between PM-Openable-ext and PM-Openable: applying interior completion on PM-Openable-ext shapes, and treating PM-Openable as ground truth.

Metrics. To evaluate interior completion, we compare the predicted shape with the ground-truth shape using chamfer distance (CD) across all parts based on 20K sampled points to also take in account how the baselines preserve the original geometry. As we do not target exact match of drawer interiors but rather having a plausible completion we compute macro F1 score for drawers based on bounding box matching. We select a high threshold of 0.8 for matching so that the drawers are sufficiently close to ground truth but allowing for minor variations.

Results. We evaluate interior completion on our validation split. We filter the categories unseen by SINGAPO (Safe, Trashcan), leaving 81 shapes total. We note that SINGAPO's training split, not considering the augmented data, has 67 overlapping shapes with our validation split. The results in Tab. 21 shows lower CD value for our approach, meaning that it excels at preserving the original geometry. In terms of drawer F1 score, we find that segmentation remains being the bottleneck and SINGAPO performs better. With the ground truth segmentation, however, our heuristic dominates all the metrics.

D.5. Application

We find that the improvements introduced by training on ACD-train and PM-Openable-ext allow more reliably generalizing to more complex shapes. This enables automated application of S2O pipeline on shapes not included in ACD. For this matter, we apply it on some shapes from 3D-FUTURE and ABO datasets which were not previously annotated as part of ACD effort. We find that our pipeline is able to make such shapes articulated, as shown in Fig. 17. This enables for automated and more efficient shape annotation. See supplemental videos for more detailed overview of examples.

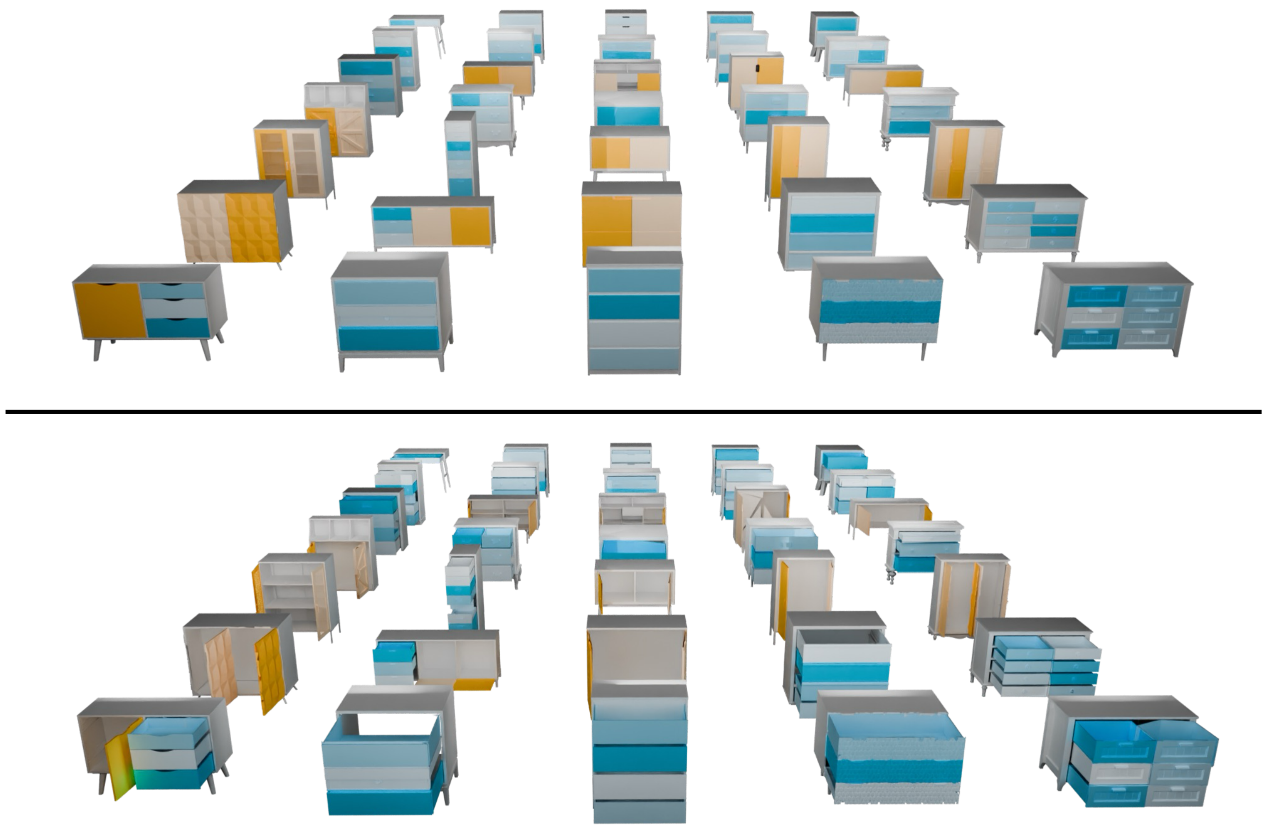

Figure 17. Compilation of results after applying our pipeline on 3D-FUTURE [14] and ABO [7] shapes, outside of annotated ACD subset. The top shows the openable part segmentations with doors in shades of orange and drawers in shades of blue. The bottom shows the same objects with their openable parts articulated to a more open state.