OPD: Single-view 3D openable part detection Supplemental Materials

Hanxiao Jiang, Yongsen Mao, Manolis Savva, and Angel Xuan Chang

Simon Fraser University
3dlg-hcvc.github.io/OPD/

In this supplement to the main paper we provide implementation details (Appendix A), additional quantitative and qualitative results (Appendix B), as well as details on the dataset statistics and construction (Appendix C).

A Additional implementation details

A.1 Training loss details

Detection and Segmentation Losses. Our architecture uses the Mask R-CNN [3] loss which is composed of losses for the RPN module, part detection and segmentation losses for each proposed region of interest: $L_{\text{Mask R-CNN}} = L_{\text{rpn}} + L_{\text{det}} + L_{\text{seg}}$. We refer the reader to the original paper for details on the implementation of these losses.

Motion Losses. We extend the Mask R-CNN network with extra heads for motion prediction. To train the motion prediction heads, we add additional loss terms to the loss associated with each ROI. We construct the motion loss L_m as a weighted sum of cross-entropy loss for the motion type (L_c), and smooth L1 losses for regressing the motion axis (L_a) and motion origin (L_o): $L_m = \lambda_c L_c + \lambda_a L_a + \lambda_o L_o$. The motion loss terms for each RoI i are given by:

$$\begin{aligned} L_{c_i} &= L_{\text{CE}}(\hat{c}_i, c_i) \\ L_{a_i} &= L_{\text{smoothL1}}(\hat{a}_i, a_i) \\ L_{o_i} &= L_{\text{smoothL1}}(\hat{o}_i, o_i)\mathbb{1}\{c_i = \text{rotation}\} \end{aligned} \tag{1}$$

where $\hat{c}_i$ is the predicted motion type and c_i is the ground truth motion type, $\hat{a}_i$ is the predicted axis and a_i is the ground truth axis, $\hat{o}_i \in \mathbb{R}^3$ is the predicted origin and o_i is the ground truth origin. We set $\lambda_c = 1, \lambda_a = 8, \lambda_o = 8$ for our experiments. For **OpdRcnn-C**, the motion axis and origin are in camera coordinates. The overall loss is given by $L_{\text{CC}} = L_{\text{Mask R-CNN}} + L_m$. **OpdRcnn-O** has the same additional loss as OpdRcnn-C for motion parameters, but with additional smooth L1 loss for the extrinsic matrix L_{ext}.

$$L_{\text{OC-S}} = L_{\text{CC}} + L_{\text{ext}_s} = L_{\text{CC}} + \lambda_{\text{ext}} L_{\text{smoothL1}}(\hat{e}_s, e_s) \tag{2}$$

We represent the extrinsic matrix as a vector e_s of length 12 (9 for rotation, 3 for translation). The extrinsic matrix e_s is predicted by taking the features for the entire images directly from the backbone network. For OpdRcnn-O, the motion axis and motion origin are in the canonical object coordinate instead of the camera coordinate. We set $\lambda_{\text{ext}} = 15$ for our experiments.

A.2 Coordinate system details

We use three coordinate systems in our experiments: i) camera coordinates; ii) canonical object coordinates (equivalent to world coordinates for synthetic data); and iii) ANOCs (anistropically-scaled normalized object coordinates), our adaptation of the normalized object coordinates (NOCs).

Camera Coordinates. Our task is to predict the motion parameters from a single-view image so camera coordinates are a natural coordinate system. We evaluate all motion parameters in camera coordinates. The input point clouds for ANCSH and our OpdPN baseline are also represented in camera coordinates.

Canonical Object Coordinates. Inspired by the canonical coordinates used in the ANCSH [4] approach. We use canonical object coordinate in our OpdRcnn-O model to predict the motion axis and motion origin in a more consistent frame of reference. To obtain a canonical object coordinate frame we either rely on existing alignments of objects to a canonical pose (for OPDSynth), or annotate a semantically-consistent oriented bounding box (OBB) with a consistent front and up axis for each object (for OPDReal).

ANOCs (anistropically-scaled NOCs). The ANOCs coordinate system further normalizes the canonical object coordinates. We use the dimensions of the bounding box of each object to normalize each dimension to $[-0.5, 0.5]$. This makes it easier to define the candidate motion origins for the RandMot and MostFreq baselines.

A.3 OpdPN baseline architecture

The OpdPN baseline uses a PointNet++ [5] architecture to process a single-view 3D point cloud of the object. We use a set of part category labels corresponding to the openable part types with one additional label representing any other parts that are not articulated. We also predict an instance segmentation id to separate part instances. These ids do not have a natural ordering defining correspondences between parts in different objects in the dataset, so we instead match the predicted instance id with instances in the ground truth using GIoU [6] and the Hungarian algorithm. We use a mIoU loss for the part category, instance id and motion type. For the motion axis and motion origin we use an MSE loss.

A.4 Re-implementation of ANCSH

We re-implemented the ANCSH approach by Li et al. [4] in PyTorch. Table 1 reports the results of our re-implementation against the original reported results. We see that our re-implementation gives comparable results with the original, with small variations (performance under some metrics improved while it is slightly worse along some other metrics). We performed this comparison as a sanity check experiment to confirm that our re-implementation is consistent with the results reported by the authors.

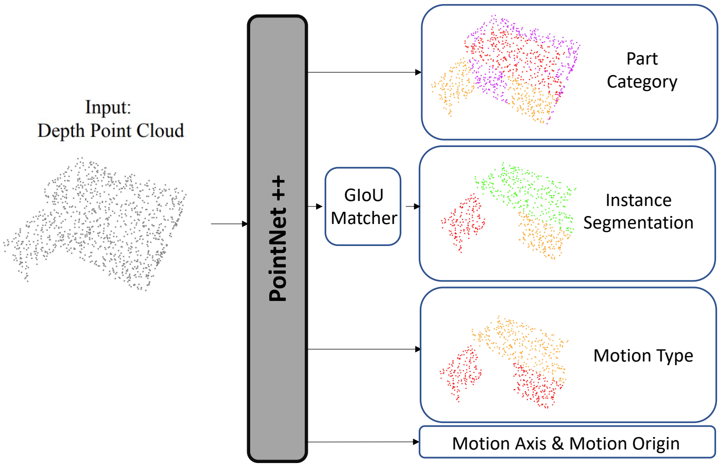

Fig. 1: Network structure for OpdPN. Given the single-view point cloud, the network predicts part category, instance segmentation, motion type, motion axis and motion origin. For the part category, the base part of the object is one category. For other predictions we only consider predicting the moving parts. For the motion axis and motion origin, we assign the motion to each moving part, and all points in that moving part have the same ground truth motion type, motion axis and motion origin. We use GIoU [6] and the Hungarian algorithm to match part instances from the predictions with the ground truth for the evaluation.

B Additional results

B.1 Part detection and segmentation performance

In Table 2, we report the standard COCO metrics for segmentation including AP(averaged over IoU 0.50 to 0.95 thresholds at a increment of 0.5), AP_{50} (for IoU=0.5), AP_{75} (for IoU= 0.75). We also report the mean average precision for part detection over the 2D bounding boxes (AP^{bb}, $\text{AP}^{\text{bb}}{}_{50}$, $\text{AP}^{\text{bb}}{}_{70}$). All results are on the validation set for the *pretrained* part detector.

B.2 Motion parameter performance by motion type

In Table 3, we present the results using our best model on the OPDSynth test set. We include a breakdown of the motion parameter estimation by motion type for translation (t) and rotation (r). Note that the motion origin is only valid for rotation. Our OpdRcnn-O outperforms OpdRcnn-C in all cases for motion axis prediction but slightly underperforms for origin prediction.

B.3 Part motion estimation error metrics

We also evaluate motion parameter estimation by computing the angle error for axis predictions, and normalized distance error for origin predictions following

Table 1: Comparison between our re-implementation of ANCSH [4] and the results reported in the original paper on the original eyeglasses dataset.

	Part-based Metrics			Joint Parameters	
Method	Rotation Err ↓	Translation Err ↓	3D IoU % ↑	Angle Err ↓	Distance Err ↓
Li et al. [4]	3.7, 5.1, 3.7	**0.035**, **0.051**, **0.057**	**87.4**, 43.6, 44.5	2.2, **2.3**	**0.019**, **0.014**
Our implementation	**2.8**, **2.8**, **3.5**	0.039, 0.053, 0.072	87.0, **45.6**, **45.5**	**2.1**, 2.5	0.023, 0.024

Table 2: Part detection and segmentation results on the val set. $\mathrm{AP}^{\mathrm{bb}}$ is the mAP for the 2D bounding box, and AP is the mAP for the instance segmentation.

		Detection			Segmentation		
Input	Model	$\mathrm{AP}^{\mathrm{bb}}$	$\mathrm{AP}^{\mathrm{bb}}_{50}$	$\mathrm{AP}^{\mathrm{bb}}_{75}$	AP	AP_{50}	AP_{75}
RGB	OpdRcnn-C	50.5	74.7	55.5	45.1	67.1	50.2
	OpdRcnn-O	**50.6**	**75.3**	**56.1**	**45.5**	**67.9**	**50.6**
D	OpdRcnn-C	**44.5**	69.4	**48.2**	**38.8**	60.6	**42.0**
	OpdRcnn-O	44.1	**70.5**	46.8	38.3	**61.3**	40.9
RGBD	OpdRcnn-C	**48.6**	**73.6**	**52.5**	**42.3**	**65.3**	**45.7**
	OpdRcnn-O	47.1	73.3	50.9	41.1	64.0	44.6

prior work [4, 8]. As we noted in the main paper (Section 3.2), these error metrics are only computed for matched parts, and do not consider that the number of matched parts may differ between models (e.g., if a model detects only one part the error metric will only take that part into account). For models that predict both parts and their motion parameters, predicting more parts may be penalized for attempting to predict motion parameters of challenging parts.

We consider two ways of computing the error metrics: 1) we compute the micro-averaged mean of errors for detected parts (with `maxDet=100`) that are matched to ground truth parts at IoU of 0.5); and 2) we compute the average error across different IoU thresholds, different area and different `maxDet` to determine the matching between the prediction and GT. In this setting, we compute the average error for each motion type and then compute the macro-average (across motion types) to obtain the final average error. We report the error as well as the average number of matched parts for the two settings for both the OPDSynth (Tables 4 and 5), and OPDReal (Tables 6 and 7). We find that there is no noticeable difference between the two settings, and that the micro-averaged error at IoU=0.5 is reflective of the overall error that sweeps across multiple IoUs.

From Tables 4 and 6, we see that OpdRcnn-C and OpdRcnn-O have comparable number of matched parts with OpdRcnn-O having lower motion parameter errors (the trend holds across the different inputs). In contrast, OpdPN has the lowest axis error but also has fewer matched parts. Because we trained ANCSH on only one structure ('one-door'), ANCSH has the lowest number of matched parts. It also does not predict any motion parameters for the translation

Table 3: Comparison against baselines on the OPDSynth test set with metrics broken down by motion type.

		Part-averaged mAP % ↑			Motion-averaged mAP % ↑				
Input	Model	**PDet**	+**M**	+**MAO**	**MDet**	+**MA**	+**MA** (t)	+**MA** (r)	+**MO** (r)
RGBD	RandMot	5.1	1.4	0.2	6.5	0.7	1.0	0.9	0.7
	MostFreq	69.4	66.1	27.8	73.6	61.6	61.3	62.4	22.1
	OpdRcnn-C	**69.6**	67.4	41.6	**75.4**	55.9	57.6	54.9	**74.6**
	OpdRcnn-O	69.4	**67.9**	**49.7**	75.3	**66.3**	**65.0**	**68.6**	73.4
D	RandMot	4.8	1.2	0.1	5.7	0.6	0.9	0.8	0.8
	MostFreq	66.7	64.1	25.5	69.9	58.5	58.5	59.8	19.4
	OpdRcnn-C	**67.9**	**66.6**	38.6	**72.9**	52.7	54.2	52.3	**71.9**
	OpdRcnn-O	66.7	65.0	**47.3**	71.9	**62.3**	**60.9**	**65.3**	70.7
RGB	RandMot	4.7	1.2	0.1	5.6	0.6	0.8	0.8	0.6
	MostFreq	66.0	63.5	27.3	71.8	60.5	61.2	60.3	20.9
	OpdRcnn-C	**67.2**	**66.0**	38.3	**75.3**	53.5	56.9	50.7	70.7
	OpdRcnn-O	66.0	64.8	**46.8**	73.9	**63.6**	**65.0**	**63.2**	**71.1**

Table 4: Error metrics for matched instances for OPDSynth test set (micro-averaged) with the predicted part matched to the ground-truth at IoU of 0.5 and matching motion type.

		Error ↓				#Matched ↑		
Input	Model	**A**	**A** (t)	**A** (r)	**O**	**A**	**A** (t)	**A** (r)/**O**
RGBD	RandMot	59.71	59.06	60.21	0.38	44713	19598	25115
	MostFreq	11.08	3.57	16.33	0.32	44896	18468	26428
	OpdRcnn-C	9.4±0.02	6.7±0.13	11.5±0.09	0.1±0	46019.3±59.73	20123.8±11.93	25895.5±68.12
	OpdRcnn-O	6.9±0.07	4.1±0.08	9.0±0.1	0.1±0	46250.4±58.47	20133.4±82.60	26117±88.61
D (PC)	ANCSH [4]	10.41	-	10.41	0.09	6935	-	6935
	OpdPN	6.59	3.38	9.11	0.09	19672	8666	11006
D	OpdRcnn-C	9.6±0.08	6.3±0.09	12±0.11	0.1±0	45055±60.28	19249.6±52.59	25805.4±57.31
	OpdRcnn-O	7.1±0.10	4.1±0.10	9.3±0.14	0.1±0	45537.2±57.13	19494±102.17	26043.2±73.82
RGB	OpdRcnn-C	9.7±0.05	6.7±0.07	12.1±0.09	0.1±0	46282±92.65	20424±81.72	25858±22.91
	OpdRcnn-O	7.4±0.09	4.1±0.04	9.9±0.17	0.1±0	46545±128.38	20486±73.61	26059±81.78

Table 5: Error metrics for matched instances for OPDSynth test set (motion averaged) with matches determined by sweeping over different IoU thresholds.

		Motion-averaged						
		Error ↓				#Matched ↑		
Input	Model	**A**	**A** (t)	**A** (r)	**O**	**A**	**A** (t)	**A** (r)/**O**
RGBD	RandMot	59.63	59.06	60.2	0.38	22493	19853	12566
	MostFreq	9.96	3.57	16.36	0.32	22505	18563	13223
	OpdRcnn-C	9.1±0.03	6.7±0.13	11.5±0.09	0.1±0.00	23113.3±27.6	20307.3±15.38	12959.8±34.08
	OpdRcnn-O	6.6±0.06	4.1±0.08	9.0±0.10	0.1±0.00	23239.2±31.33	20335.2±81.92	13071.8±44.58
D (PC)	ANCSH [4]	10.36	-	10.36	0.09	6975	-	6975
	OpdPN	6.25	3.38	9.12	0.09	9862	8705	5510
D	OpdRcnn-C	9.2±0.08	6.3±0.09	12.0±0.11	0.1±0.00	22625.4±23.91	19432.0±51.54	12909.2±28.68
	OpdRcnn-O	6.7±0.10	4.1±0.10	9.3±0.14	0.1±0.00	22888.4±30.77	19713±105.68	13031.8±36.28
RGB	OpdRcnn-C	9.4±0.05	6.7±0.07	12.1±0.10	0.1±0.00	23256.8±44.0	20638.6±75.84	12937.6±11.57
	OpdRcnn-O	7.0±0.08	4.1±0.04	9.9±0.17	0.1±0.00	23398.6±68.99	20718.2±81.51	13039.4±41.66

Table 6: Error metrics for the OPDReal test set (micro-averaged) with the predicted part matched to the ground truth at IoU of 0.5 and matching motion type.

		Error ↓				#Matched ↑		
Input	Model	**A**	**A** (t)	**A** (r)	**O**	**A**	**A** (t)	**A** (r)/**O**
RGBD	RandMot	59.82	59.62	60.12	0.38	10088	5990	4098
	MostFreq	13.99	7.67	22.40	0.30	9738	5562	4176
	OpdRcnn-C	15.21	16.07	13.84	0.10	9942	6119	3823
	OpdRcnn-O	9.84	8.83	11.44	0.14	10076	6191	3885
D	OpdPN	7.33	7.67	6.99	0.08	4461	2265	2196
	OpdRcnn-C	22.37	26.70	15.56	0.12	9485	5800	3685
	OpdRcnn-O	13.76	13.67	13.89	0.17	9417	5738	3679
RGB	OpdRcnn-C	14.93	15.60	13.84	0.12	9916	6151	3765
	OpdRcnn-O	10.32	9.32	11.91	0.16	10225	6270	3955

Table 7: Error metrics for matched instances for OPDReal test set (motion averaged) with matches determined by sweeping over different IoU thresholds.

		Motion-averaged						
		Error ↓				#Matched ↑		
Input	Model	**A**	**A** (t)	**A** (r)	**O**	**A**	**A** (t)	**A** (r)/**O**
RGBD	RandMot	59.88	59.65	60.12	0.38	5074	6050	2049
	MostFreq	15.03	7.65	22.40	0.30	4887	5597	2088
	OpdRcnn-C	14.97	16.11	13.84	0.10	5003	6184	1911
	OpdRcnn-O	10.14	8.84	11.44	0.14	5077	6268	1943
D	OpdPN	7.33	7.67	6.99	0.08	2233	2268	1099
	OpdRcnn-C	21.14	26.72	15.56	0.12	4785	5881	1844
	OpdRcnn-O	13.81	13.71	13.92	0.17	4758	5833	1842
RGB	OpdRcnn-C	14.72	15.61	13.84	0.12	4984	6202	1883
	OpdRcnn-O	10.60	9.29	11.91	0.16	5146	6337	1978

motion type. In the next section, we examine in more detail the ANCSH performance on only the 'one-door' kinematic structure.

B.4 Comparison against ANCSH

Table 8: Comparison of OpdRcnn-O against ANCSH [4] and OpdPN baselines. 'Complete set' means evaluation on all objects in the test set. The 'one-door set' includes only objects with one door exhibiting rotational motion. OpdPN is trained on objects with no more than 5 parts in our train set. ANCSH is trained on objects in the train set with one door exhibiting rotational motion. The 'one rotating door' objects are the most frequent structure in our OPDSynth dataset. OpdRcnn-O is trained with all objects in the train set for the 'complete' test, and trained on only single rotating door objects.

		Part-averaged				Motion-averaged									
		mAP % ↑				mAP % ↑			Error ↓				#Matched ↑		
Input	Model	**PDet**	+**M**	+**MA**	+**MAO**	**MDet**	+**MA**	+**MAO**	**A**	**A** (t)	**A** (r)	**O**	**A**	**A** (t)	**A** (r)/**O**
Complete Set															
RGBD	RandMot	5.0	1.3	0.2	0.1	6.2	0.7	0.3	59.63	59.06	60.2	0.38	22493	19853	12566
RGBD	MostFreq	**69.4**	66.1	49.2	27.8	73.6	61.6	38.8	9.96	3.57	16.36	0.32	22505	18563	13223
D (PC)	OpdPN	20.4	19.3	14.0	13.6	22.0	18.1	17.6	**6.25**	**3.38**	9.12	0.09	9862	8705	5510
D (PC)	ANCSH	2.7	2.7	2.3	2.1	3.9	3.1	2.8	10.36	-	10.36	0.09	6975	-	6975
RGB	OpdRcnn-O	67.5	66.4	51.5	47.9	75.1	64.6	62.2	6.96	4.2	9.72	0.11	23401	20682	13060
D	OpdRcnn-O	67.6	65.8	52.5	48.6	72.2	63.2	60.8	6.54	4.1	**8.98**	0.11	22945	19910	12990
RGBD	OpdRcnn-O	**69.4**	**67.9**	**53.5**	**49.7**	**75.3**	**66.3**	**63.7**	6.47	3.91	9.02	0.10	23247	20288	13103
One-Door Set															
RGBD	RandMot	14.6	4.3	1.3	0.0	3.6	0.8	0.0	60.58	-	60.58	0.36	3224	-	3224
RGBD	MostFreq	**96.3**	**96.3**	41.4	6.6	**96.3**	41.4	6.6	30.17	-	30.17	0.32	4775	-	4775
D (PC)	OpdPN	71.1	71.1	45.6	40.0	71.1	45.6	40.0	11.55	-	11.55	0.11	4075	-	4075
D (PC)	ANCSH	84.2	84.2	**75.2**	**70.0**	84.2	**75.2**	**70.0**	**5.92**	-	**5.92**	**0.06**	4465	-	4465
RGB	OpdRcnn-O	90.0	90.0	65.9	54.8	90.0	65.9	54.8	11.31	-	11.31	0.14	4582	-	4582
D	OpdRcnn-O	94.8	94.8	69.2	61.2	94.8	69.2	61.2	12.77	-	12.77	0.14	4715	-	4715
RGBD	OpdRcnn-O	**96.3**	**96.3**	73.7	63.4	**96.3**	73.7	63.4	11.61	-	11.61	0.13	4775	-	4775

In the main paper we evaluated the ANCSH approach of Li et al. [4] on a dataset including objects with varying number of parts and motion types. This puts this approach at a disadvantage as it was designed such that each trained model can only operate on a fixed kinematic chain. Thus, to evaluate ANCSH in a setting that is closer to its assumption of a fixed kinematic chain, we construct a 'one-door dataset' with includes 243 instances with one rotating door part from our original 683 models. More specifically, we pick 172 models from train, 32 models from val and 39 models from test. We train ANCSH, OpdPN, and OpdRcnn on the train set of the one-door dataset and evaluate on the test set of the 'one-door dataset'.

Table 8 shows the results on the one-door dataset. Unlike the results on the complete set, ANCSH performs well on this subset of the data because the strong assumption of a single kinematic structure is satisfied. In addition, ANCSH is given prior knowledge about the rest (closed) state of the object

Table 9: Analysis of performance of OpdRcnn-O given ground truth 2D bounding box, part category and object pose on the validation set of OPDSynth. We compare the performance of using the predicted part vs using the ground truth 2D bounding box and part category (GT Box2DPart), the ground truth object pose (GT Pose), and the combination (GT Box2DPartPose). We see that having access to the ground truth object pose (GT Pose) is important for accurate motion prediction.

		Part-averaged mAP % ↑				Motion-averaged mAP % ↑		
		PDet	+**M**	+**MA**	+**MAO**	**MDet**	+**MA**	+**MAO**
RGBD	OpdRcnn-O	72.5±0.34	70.6±0.29	51.7±0.62	47.1±0.59	75.4±0.07	61.6±0.32	59.0±0.32
	GT Box2DPart	99.0±0.00	90.9±0.16	50.6±0.36	45.4±0.27	89.7±0.15	58.1±0.32	54.7±0.28
	GT Pose	73.1±0.10	71.0±0.05	60.5±0.06	59.4±0.05	75.2±0.08	67.0±0.14	66.2±0.09
	GT Box2DPartPose	99.0±0.00	90.6±0.37	65.5±0.24	63.8±0.17	89.5±0.19	73.3±0.26	72.0±0.30
D	OpdRcnn-O	69.3±0.35	67.5±0.33	50.7±0.55	45.1±0.50	72.5±0.26	59.1±0.36	55.9±0.49
	GT Box2DPart	99.0±0.00	89.7±0.30	50.6±0.09	45.2±0.17	88.9±0.26	57.8±0.19	54.3±0.26
	GT Pose	70.1±0.21	68.3±0.22	59.0±0.14	57.9±0.13	73.3±0.11	65.2±0.09	64.4±0.10
	GT Box2DPartPose	99.0±0.00	89.3±0.23	64.8±0.19	63.1±0.09	88.4±0.33	72.7±0.29	71.5±0.29
RGB	OpdRcnn-O	74.2±0.34	72.4±0.32	52.4±0.31	47.3±0.40	79.1±0.24	62.6±0.40	59.6±0.45
	GT Box2DPart	99.0±0.00	91.3±0.14	51.8±0.22	46.7±0.23	90.9±0.11	60.8±0.17	57.0±0.25
	GT Pose	75.5±0.07	73.6±0.09	61.0±0.14	59.8±0.08	79.8±0.08	70.5±0.06	69.5±0.04
	GT Box2DPartPose	99.0±0.00	91.4±0.19	64.2±0.16	62.4±0.17	90.3±0.16	73.7±0.23	72.3±0.18

during training, and requires additional annotation that our method does not need. Notably, under this setting, ANCSH has a much more accurate rotation origin prediction than our methods. While our methods have slightly less accurate motion parameter estimation, they have more accurate part detection and our structure-agnostic approach handles arbitrary variations in kinematic structure with the same model.

B.5 Additional analysis

Experiments with ground truth. To investigate what parts of the problem are challenging, we conduct experiments using ground truth bounding boxes and part labels (GT Box2DPart), as well as ground truth object pose (GT Pose) and their combination (GT Box2DPartPose). Table 9 summarizes the result of using ground-truth for OpdRcnn-O on OPDSynth for RGBD, D, and RGB.

For the ground truth 2D bounding boxes, we use them as proposals to extract image features for the box head and mask head in MaskRCNN (dropping all detection and segmentation losses). To make sure there is no gap between our training and inference, we also finetune our final model with features extracted from the ground truth bounding boxes. As expected, when using the ground-truth bounding boxes and ground truth part label (GT Box2DPart), the part detection is close to perfect. As noted in the main paper, having the ground truth object pose (GT Pose) is more important for motion parameter estimation as seen in the increase in +**MA** between (GT Pose) and (GT Box2DPart), with further improvement when both are provided (GT Box2DPartPose). The

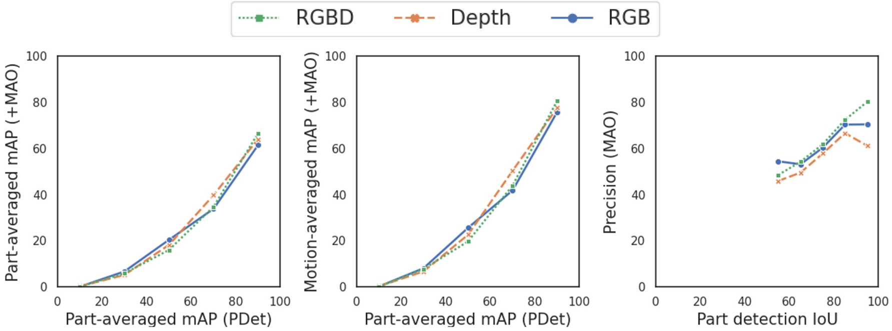

Fig. 2: Plots showing correlation between part detection performance and part motion estimation performance. All results are based on the OpdRcnn-O model evaluated on OPDSynth validation set. Left: part-averaged motion estimation performance against detection performance (**PDet**). Middle: motion-averaged motion estimation performance against detection performance (**PDet**). Right: precision of motion parameter estimation against part detection IoU. These three plots show that motion parameter estimation performance is correlated with part detection performance, with small differences between input modalities.

+**MA** metric is about the same when using the predicted bounding box vs ground truth.

Correlation of part detection and motion estimation performance. We perform a more detailed analysis of the correlation between openable part detection performance and motion estimation performance. We create plots of aggregated motion estimation performance against different buckets of part detection performance as measured by part-averaged mAP (**PDet**) (see Figure 2 left and middle). We also evaluate within the buckets divided by the IoU for each pair of GT and prediction. Figure 2 clearly shows that for buckets with better part detection, motion parameter estimation (+**MAO**) is also higher, which indicates that better openable part detection at the image level contributes to better motion parameter prediction. Although the second setting evaluates on different buckets, it is still computed at the image level, which cannot show the direct relationship between detection and motion prediction for each part. Therefore, we design a third evaluation setting for motion prediction at the instance level. In this setting, instead of using mAP we use precision of the motion parameter estimation on matched parts (IoU > 0.5 and part category matches). Figure 2 (right) shows the results for the motion in different IoU buckets. We see a strong correlation between the detection and the motion prediction.

Motion thresholds. From the plot in Figure 2 (right) we can see that combining depth and RGB information provides a non-trivial benefit in terms of motion parameter precision when the part detection is good (RGBD results are significantly better than RGB for higher detection IoU values). From Table 10, we can see that when the motion threshold is stricter, the depth input and RGBD input have better motion parameter estimation results even if their detection is worse than RGB. We hypothesize that depth information contributes to more precise motion prediction when the part detection performance is reasonable.

Table 10: Results for OPDRCNN-O with different threshold for motion axis (τ_{axis}) and motion origin (τ_{origin} times the diagonal).

			Part-averaged mAP % ↑				Motion-averaged mAP % ↑					
Input	τ_{axis}	τ_{origin}	**PDet**	+**M**	+**MA**	+**MAO**	**MDet**	+**MA**	+**MAO**	+**MA** (t)	+**MA** (r)	+**MO** (r)
RGB	10	0.25	75.3	73.4	53.6	48.8	80.0	64.0	61.2	70.0	58.5	66.4
	10	0.10	75.3	73.4	53.6	35.9	80.0	64.0	48.8	70.0	58.5	32.2
	5	0.25	75.3	73.4	34.5	32.6	80.0	43.6	42.3	47.1	40.4	66.4
	5	0.10	75.3	73.4	34.5	25.2	80.0	43.6	34.3	47.1	40.4	32.2
	1	0.25	75.3	73.4	0.9	0.9	80.0	1.2	1.1	0.7	1.6	66.4
	1	0.10	75.3	73.4	0.9	0.7	80.0	1.2	0.9	0.7	1.6	32.2
D	10	0.25	70.5	68.7	51.7	46.4	73.4	60.3	57.6	63.1	58.9	64.6
	10	0.10	70.5	68.7	51.7	32.0	73.4	60.3	43.8	63.1	58.9	29.1
	5	0.25	70.5	68.7	35.9	33.3	73.4	44.8	43.2	48.6	41.9	64.6
	5	0.10	70.5	68.7	35.9	24.4	73.4	44.8	33.8	48.6	41.9	29.1
	1	0.25	70.5	68.7	0.9	0.8	73.4	1.1	1.1	1.4	0.9	64.6
	1	0.10	70.5	68.7	0.9	0.7	73.4	1.1	1.0	1.4	0.9	29.1
RGBD	10	0.25	73.3	71.0	53.6	48.5	75.4	62.8	60.0	64.7	61.7	67.9
	10	0.10	73.3	71.0	53.6	35.4	75.4	62.8	47.8	64.7	61.7	35.8
	5	0.25	73.3	71.0	37.1	34.6	75.4	46.2	44.7	48.1	44.9	67.9
	5	0.10	73.3	71.0	37.1	26.8	75.4	46.2	36.5	48.1	44.9	35.8
	1	0.25	73.3	71.0	1.5	1.5	75.4	2.0	2.0	1.8	2.2	67.9
	1	0.10	73.3	71.0	1.5	1.2	75.4	2.0	1.7	1.8	2.2	35.8

B.6 Additional qualitative results

Figure 3 shows a qualitative comparison between ANCSH, OPDPN and OPDRCNN-O. We see that our OPDRCNN-O approach detects openable parts much more reliably, and overall provides more accurate motion parameter estimates for the detected parts.

C Dataset details

We provide additional statistics on the part and structure variation (Appendix C.1) found in OPDSynth and OPDReal, and details on how we rendered or selected images for the two datasets(Appendices C.2 and C.3).

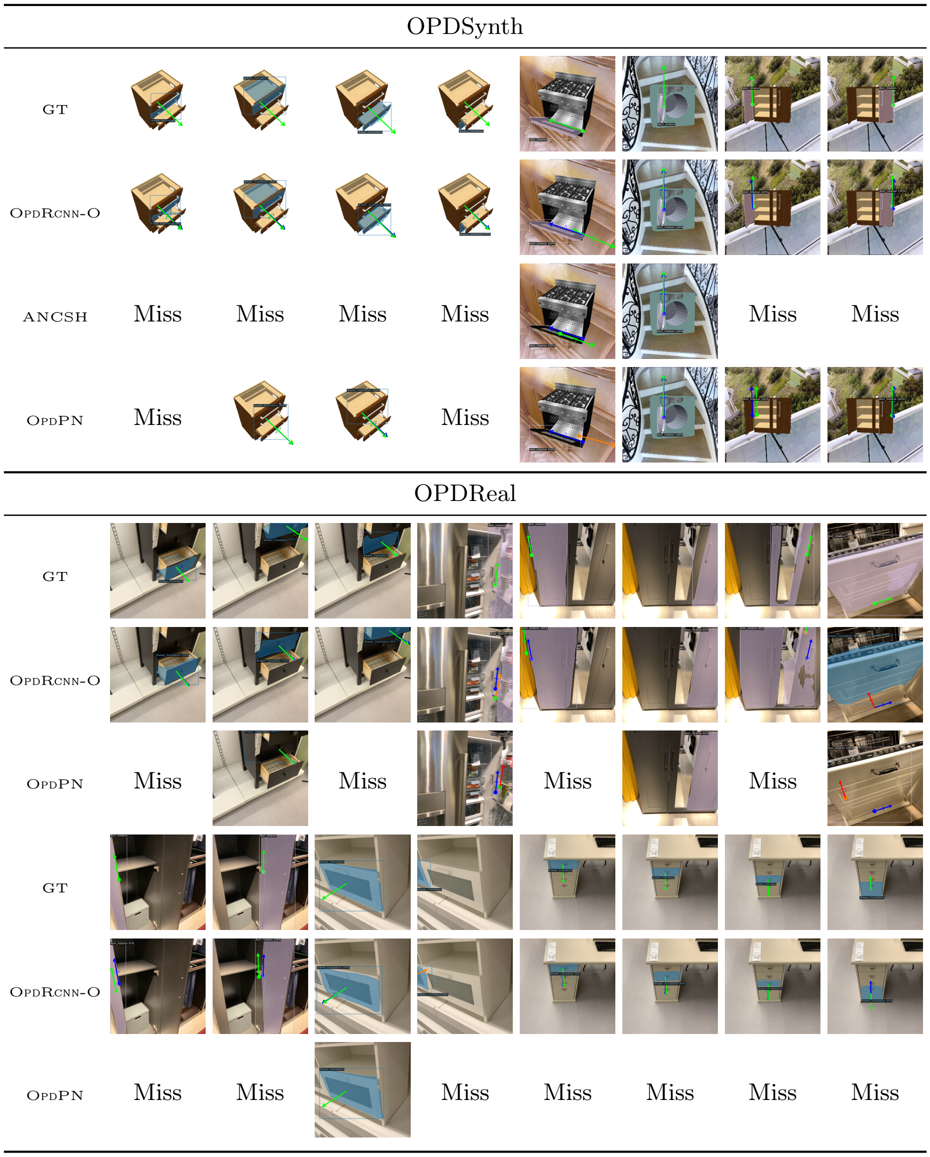

Fig. 3: Qualitative results comparing our approach against the ANCSH and OpdPN baselines. Structure of the figure is the same as in the main paper. Both ANCSH and OpdPN fail to detect many of the openable parts, in particular for the more challenging OPDReal dataset. In contrast, our OpdRcnn-O approach detects more parts and provides more accurate motion parameter estimates.

Table 11: Openable part labels from PartNet-Mobility [9] for each object category we use in our experiments.

Category	Part labels
Storage	cabinet_door, drawer, drawer_box, cabinet_door_surface, handle, glass, other_leaf, door
Table	drawer, drawer_box, handle, cabinet_door, cabinet_door_surface, shelf, keyboard_tray_surface
Fridge	door, door_frame, display_panel, control_panel, glass
Microwave	door
Washer	door
Dishwasher	door, door_frame, display_panel
Bin	cover, lid, frame_vertical_bar, opener, cover_lid, other_leaf, drawer
Oven	door, door_frame
Safe	door
Box	rotation_lid, lid_surface, countertop, drawer
Suitcase	lid

C.1 Dataset statistics

Part and motion statistics In Table 12 we report the total numbers of different part types, as well as number of images of different part types and number of motion types observed across all images.

Openable part structure variation statistics We note that it is possible for objects belonging to the same object category to have variation in the structure (number of `doors`, `drawers`, and `lids`). Figure 4 shows the statistics of the structure variation of objects in our datasets.

C.2 OPDSynth details

Consistent part labeling Our OPDSynth dataset is based on synthetic objects from the PartNet-Mobility dataset. As the initial part labels for these objects may be inconsistent, we developed a two-pass approach to identify and label all openable parts. In the first pass, we identify all part labels that may correspond to openable parts. For each object category we identify the set of openable part labels from the set of all part labels. For instance, for the object category of ‘box’, we include ‘rotation lid’, ‘lid_surface’ as openable parts, but not ‘base_body’ or ‘handle’. After collecting all part labels for each model category, we select an example model for each part label in that model category. Then through verifying the corresponding example model, we determine if we want to include this part label or not. From the accepted part labels (Table 11), we see that the semantic meaning of these part labels are all relevant to drawers, doors or lids. After the first pass, we get 740 models and 1441 parts over 11 categories. The second pass is to verify all the parts selected from the first pass manually. We designed a user interface to show the part mobility and help annotators judge if it is a valid part which can be opened and closed. We also relabel the parts with consistent labels from three main categories (`drawer`, `door` and `lid`). The annotation process took approximately 30 minutes to obtain 1343 valid parts and reassign them to the consistent set of openable part labels.

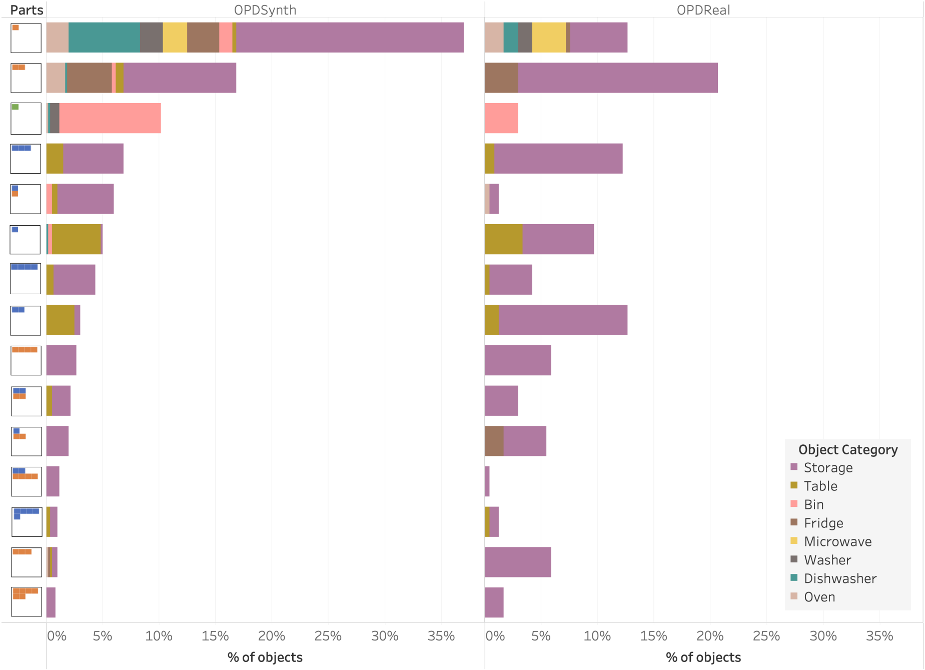

Fig. 4: Part composition distribution for our OPDSynth (left) and OPDReal (right) datasets. Each row represents a particular part composition with the icon at the left indicating the number of doors (orange), drawers (blue), and lids (green). The plot bar colors indicate the distribution over object categories. The top 15 part compositions are plotted sorted by number of objects with that part composition in the OPDSynth dataset. We see that our datasets exhibit a diverse set of part compositions with varying numbers of parts of each type. These compositions are also distributed across several object categories.

Table 12: Statistics of the distribution over part types and motion types in our datasets. The first three columns report the number of part instances (i.e. counting distinct openable parts across all objects). The second set of three columns reports the number of images with parts of that type across all objects. The last two columns report the number of images with a part exhibiting the specific motion type, across all parts and objects.

		# parts			# part images			# motion images	
		`drawer`	`door`	`lid`	`drawer`	`door`	`lid`	translation	rotation
OPDSynth	train	363	508	89	16882	141180	16880	171705	155175
	val	79	94	20	28265	22635	4605	29200	26305
	test	75	97	18	28425	25640	4825	28665	30225
OPDReal	train	304	268	3	27598	17695	212	27540	17965
	val	78	79	2	7324	5311	152	7520	5267
	test	74	65	2	7002	4685	110	7002	4795

View selection We render RGBD images of the object using a perspective camera, with a 50° vertical field-of-view at 256 × 256 resolution. For the RGB color we use Phong shading and a hemisphere light. We vary the position and distance of the camera so that we get mostly views above and in front of the object. To select specific views we sample the elevation θ, azimuth ϕ and distance d independently. For each, we use a Bates distribution $B_k(a, b) = a + \frac{b-a}{k}\sum_{i=1}^{k} u_i$, that is the sum of k standard uniform random variables u_i scaled to the range (a, b), with a, b set as described below. We first sample a categorical variable v to determine if we want a camera viewpoint: 1) in a mostly above and frontal view with probability 0.6; 2) in a wider vertical distribution for the camera allowing for the elevation θ to range from slightly below the object to above with probability 0.2; or 3) in a wider horizontal distribution for the camera allowing for a larger range for the azimuth ϕ and distance d. Note that the camera directly in front of the object is at $\phi = 0$. The three cases are summarized below:

1. Above, frontal view:
   $P(v = 1) = 0.6$, $\theta \sim B_2(30, 70), \phi \sim B_2(-60, 60), d \sim B_2(1.8, 2.8)$
2. Slightly below above:
   $P(v = 2) = 0.2$, $\theta \sim B_3(-35, 35), \phi \sim B_2(-60, 60), d \sim B_2(1.8, 2.8)$
3. Slightly below above, wider azimuth/distance distribution:
   $P(v = 3) = 0.2$ $\theta \sim B_3(-35, 35), \phi \sim B_3(-90, 90), d \sim B_2(1.6, 3.1)$

Image rendering We render single-view RGB and depth images from OPDSynth. For each object, we render a total of $5 + 20 \cdot \text{num\_parts}$ views each with different motion states for each part. In one motion state all parts are at the min value of their motion range. Then, we pick four random states for each moving part except the min value (one of which must be the max value of the range), while other parts stay at the min value of their motion range. We augment the images using

RGBD backgrounds from the Matterport3D [1] dataset by randomly selecting from the 'straight ahead' and 'downward tilt' camera views. Each image has four random backgrounds resulting in a total of $25 + 100 \cdot \text{num\_parts}$ images.

C.3 OPDReal details

Data capture and reconstruction We used iPad Pro 2021 devices to capture RGB-D video scans of articulated objects in indoor environments. We focused on object categories that overlap with OPDSynth and have openable parts. Each scan focused on a single object instead of capturing the entire environment. We take multiple scans (~ 3) of each object. In total, three student volunteers collected 863 scans covering 294 different objects across 8 object categories. From these, we obtained 763 polygonal meshes for 284 different objects (some scans failed to produce high quality reconstructions). We obtained polygonal mesh reconstructions from these scans using the Open3D [11] implementation of RGB-D integration and Waechter et al. [7]'s implementation of texturing.

Annotation We adapted the 3D annotation tool from ScanNet [2] to work with textured meshes and used it to annotate object parts ('door', 'drawer', 'lid', or 'base'). For the object articulation, we use the annotation interface from Xu et al. [10]. Each articulatable part is annotated with the motion type ('rotation' or 'translation'), motion axis and rotation origin, as well as motion ranges. We developed another interface to indicate the semantic orientation (front) of the oriented bounding box for the object. The annotation was done by student volunteers. The semantic part annotation took the longest with an average of 7.18 minutes per object, and 5483 minutes in total (across 5 volunteers). Articulated part annotation took an average of 3.24 minutes per object and 2473 minutes in total. The annotation of semantic OBBs was much faster with an average of ~ 20 seconds taken per scan. Articulation and semantic OBB annotation was done by one of the authors.

Frame selection For OPDReal, we selected frames from each scan to form our image dataset. We rescale both the depth and color frames to a 256 x 256 resolution using a center-crop strategy. When selecting frames, we sample one frame every second ensuring that at least 1% of pixels belong to an openable part and at least 20% of parts are visible.

References

1. Chang, A., Dai, A., Funkhouser, T., Halber, M., Niessner, M., Savva, M., Song, S., Zeng, A., Zhang, Y.: Matterport3D: Learning from RGB-D data in indoor environments. In: Proceedings of the International Conference on 3D Vision (3DV) (2017)
2. Dai, A., Chang, A.X., Savva, M., Halber, M., Funkhouser, T., Nießner, M.: ScanNet: Richly-annotated 3d reconstructions of indoor scenes. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 5828–5839 (2017)
3. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask R-CNN. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV), pp. 2961–2969 (2017)
4. Li, X., Wang, H., Yi, L., Guibas, L., Abbott, A.L., Song, S.: Category-level articulated object pose estimation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2020)
5. Qi, C.R., Yi, L., Su, H., Guibas, L.J.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space. arXiv preprint arXiv:1706.02413 (2017)
6. Rezatofighi, H., Tsoi, N., Gwak, J., Sadeghian, A., Reid, I., Savarese, S.: Generalized intersection over union. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2019)
7. Waechter, M., Moehrle, N., Goesele, M.: Let there be color! — Large-scale texturing of 3D reconstructions. In: Proceedings of the European Conference on Computer Vision, Springer (2014)
8. Wang, X., Zhou, B., Shi, Y., Chen, X., Zhao, Q., Xu, K.: Shape2Motion: Joint analysis of motion parts and attributes from 3D shapes. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 8876–8884 (2019)
9. Xiang, F., Qin, Y., Mo, K., Xia, Y., Zhu, H., Liu, F., Liu, M., Jiang, H., Yuan, Y., Wang, H., Yi, L., Chang, A.X., Guibas, L., Su, H.: SAPIEN: A simulated part-based interactive environment. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 11097–11107 (2020)
10. Xu, X., Charatan, D., Raychaudhuri, S., Jiang, H., Heitmann, M., Kim, V., Chaudhuri, S., Savva, M., Chang, A.X., Ritchie, D.: Motion annotation programs: A scalable approach to annotating kinematic articulations in large 3d shape collections. In: 2020 International Conference on 3D Vision (3DV), pp. 613–622, IEEE (2020)
11. Zhou, Q.Y., Park, J., Koltun, V.: Open3D: A modern library for 3D data processing. arXiv:1801.09847 (2018)