Supplementary Material for TO-Scene: A Large-scale Dataset for Understanding 3D Tabletop Scenes

Mutian Xu[1], Pei Chen[1], Haolin Liu[1,2], and Xiaoguang Han[1,2]

[1] School of Science and Engineering, The Chinese University of Hong Kong, Shenzhen
[2] The Future Network of Intelligence Institute, CUHK-Shenzhen

Outline

This supplementary document is arranged as follows:
(1) Sec. A provides more statistics, sample visualizations and discussions of TO-Scene dataset;
(2) Sec. B describes our data acquisition framework more comprehensively;
(3) Sec. C elaborates the implementations and ablations for benchmarking on our TO-Scene;
(4) Sec. D illustrates the details of TO-Real test set and visualizes the test results on it.

A TO-Scene Dataset

In this section, we provide detailed statistics and visualize more examples of three variants in TO-Scene, and discuss the current limitations of TO-Scene.

A.1 Per-category Statistics

Table III enumerates the train/val split statistics of the instance quantities, plus semantic segmentation mIoU (seg) and object detection mAP@0.25 (det) of each class in three variants of TO-Scene, where our TO-Scene contains a rich amount of instances under rational train/val split, and diverse tabletop object categories in all variants. For the description of the benchmark results, see Sec. C.3.

A.2 Example Visualizations

Fig. I presents a larger set of examples in three variants of TO-Scene. For each sample, the surface mesh and separate object instance labels are illustrated. It clearly shows the difference across three variants, as well as the outstanding diversity of tabletop objects. In TO_Crowd, more occlusions and reconstruction errors can be seen (e.g., the right column in the third row of TO_Crowd).

A.3 Discussions

There are two major limitations of the current TO-Scene dataset:

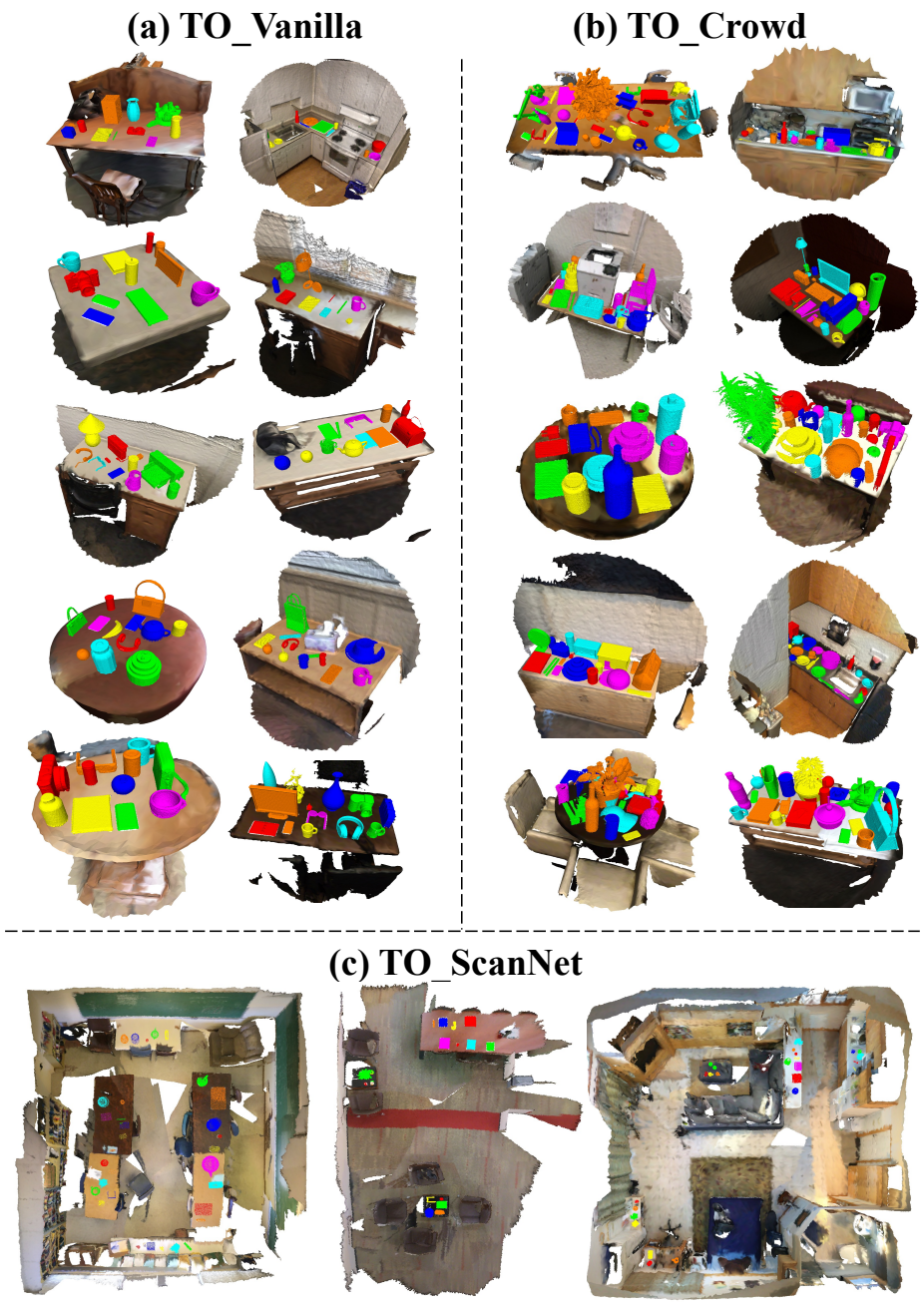

Fig. I: A variety of example scenes in three variants of TO-Scene. Each object instance shown with a different randomly assigned color.

(1) The *texture* of the tabletop object is currently not available due to the lack of the high-quality textured object CADs, but we believe our accessible Web UI opens the chance to replenish our dataset, or build similar datasets for different applications.

(2) *Stacking, collision and suspension* are very common in real-world tabletop scenes. For instance, toothbrushes and toothpastes are in cups, cables and wires are connected to monitors and cellphones, and they are placed randomly on tabletops and colliding with other objects. These scenarios are hard to be designed by our current framework. However, compared with our acquisition method, manually scanning and labelling real tabletop objects with stacking and collision will become *more harsh*. Moreover, our framework firstly makes it possible to build a large-scale dataset of tabletop scenes, through a new insight of "*mixing objects and scenes* by crowd-sourcing". Inspired by TO-Scene, future works may design to improve our scalable framework and UI, to create more complex scenes realizing stacking, collision or suspension.

B Acquisition Framework

This section states more details for specific steps in our data acquisition framework.

B.1 Transfer UI

We use two open-source platforms, Three.js * and Node.js †, to develop an easy-to-use Web UI to achieve the object transfer process. As shown in Fig. II, the operation is basically consisted of a few clicks on UI.

Source load and selection. At the beginning of each transfer, when an user presses the button "Load" at the lower right corner to load new data, a table will be automatically picked from ScanNet [3] and loaded. Then a Bird's-Eye-View (BEV) of the loaded table will be shown at left, accompanied with a 3D view of its surroundings at right. At the same time, the semantic classes of small objects are listed at the top of the page, which are also randomly selected from suitable classes in ModelNet [8] and ShapeNet [1] matched with the type of the table (pre-defined in UI according to commonsense knowledge). An user may observe the context around the table and choose one suitable category of the objects, and the "Instances" column will be provided at left for picking an object instance of the chosen class.

Object placement. Notably, when placing an object, the user does *not* need to perceive the *height* to place it in 3D. The only thing to do is just clicking at somewhere on the table in BEV, then the selected object will

* `https://threejs.org`
† `https://nodejs.org`

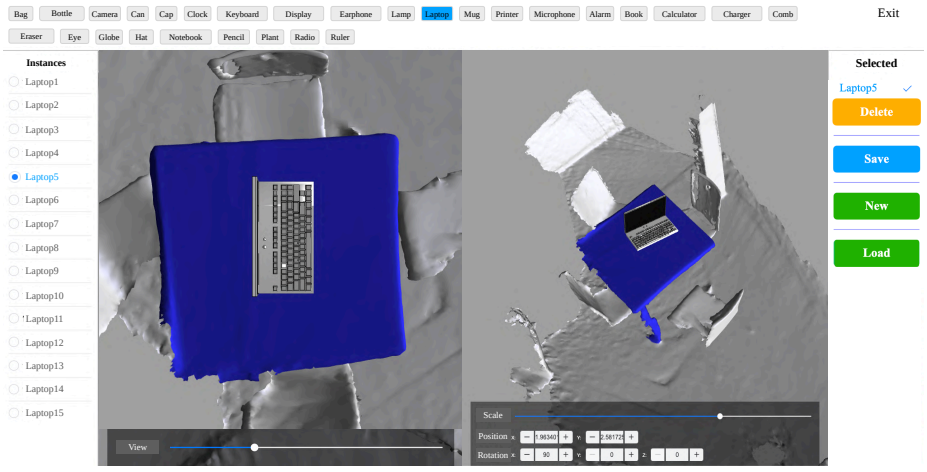

Fig. II: Our web-based crowdsourcing interface for transferring CADs onto real tabletops, where the user chooses "Laptop" from the category list on the top, then pick "Laptop5" from "Instances" column at left, and click somewhere on Bird's-Eye-View (BEV) (left) for placing it above 3D tables (right).

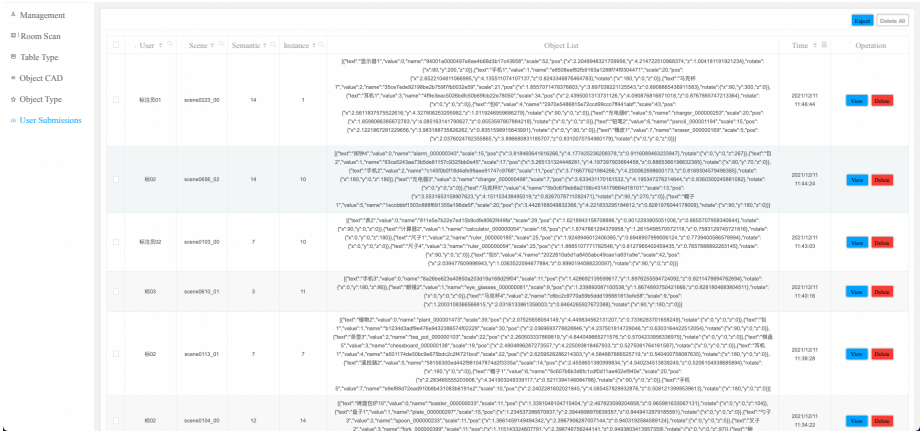

Fig. III: The backend management system in our UI, where the user submission interface is shown here.

be automatically placed on the position as the user suggests. We integrate a precise algorithm inside the UI to achieve this, where the height is calculated by the maximum height of the local region on the table within the suggested location to place the object, which ensures the object is transferred as precise as possible, with no gap nor intersection between the table and object.

Transfer completion. Simultaneously with each click on BEV, the produced object placement will also be shown in 3D scenes at right, where user is able to drag the 3D scene at right to check the transferred object from comprehensive views, so that its position is able to be adjusted by some simple re-clicks on BEV. Moreover, the toolbar listed at the bottom of the page assist the users to *fine-tune* the scale, position and orientation (operated by modifying the pitch, yaw and roll angles) of the object, arranging the objects in a way that closer to real life scenarios. After finish transferring, the calibration, alignment and other parameters that can re-realize the final transferred result are all saved in a configuration file indicated by the sudo-mappings between different tables and CADs. The average transfer time on a table per crowd worker is 1.8 min for TO_Vanilla, and 4.5 min for TO_Crowd.

Backend management. To enable high scalability of our data acquisition framework, and continual transparency into the transfer progress, we also endow our UI with a backend management system to track and organize the data (see Fig. III). The left column provides the icon to manage (i.e., revise or delete) the source data of room scans paired with table categories [3], and object CADs coupled with object classes [8,1], as well as the user submissions of transferred tabletop scenes. This management significantly guarantees the data quality.

B.2 Real-scene Simulation

To eliminate the domain gap between synthetic tabletop objects and real-world scans, we simulate the CADs into realistic data.

We run Blender [2] on several CPUs to render CAD objects into averaged 50-100 RGB-D sequences, with mean time 9.8 sec on each tabletop scene.

We select average 26.5 RGB-D frames to reconstruct each scene by implementing TSDF fusion [10] [‡] in a multiprocessing manner, where the mean reconstruction time per sample is 50 sec.

C Benchmark Tasks

This section presents more experimental details and ablations for benchmarking the state-of-the-arts on our dataset.

[‡] `https://github.com/andyzeng/tsdf-fusion-python`

C.1 Implementation

As described in the main paper, we follow the original training strategies and data augmentations of all selected methods. Commonly, we run different deep networks using PyTorch [5] via several Nvidia GeForce RTX 3090 GPUs.

When running experiments on TO_Vanilla and TO_Crowd, the semantic labels (52 categories) of tabletop objects in our dataset are *all* recognized during training for thorough exploitation. For TO_ScanNet, besides keep using all 52-classes labels on tabletop objects, we use NYU2 labels [7] on the big furniture extracted from ScanNet [3], following the original settings in ScanNet.

According to [3] and the state-of-the-arts [9,13,11,4], we do not use RGB input. Additionally, for parsing tabletop scenes of TO_Vanilla and TO_Crowd, taking table classes in one-hot format as an additional input prior may help learning, but here we ignore it for simplification.

C.2 Ablations of Tabletop-aware Learning.

The tabletop-aware learning strategy is proposed to tackle the difficulty of perceiving the small tabletop objects under indoor environments, which is especially challenging on TO_ScanNet with lots of big furniture surroundings. Thus, we explore more about our tabletop-aware learning strategy on TO_ScanNet for 3D semantic segmentation task.

λ on L_{dis}. As stated in Sec. 4 of the main paper, by introducing tabletop-object discriminator, the loss can be written as: $L_{total} = L_{seg\,or\,det} + \lambda L_{dis}$, where λ is the weight on the discriminator loss L_{dis}. We apply feature vector of tabletop-object discriminator on Point Transformer [13], and conduct experiments under different λ. As listed in Table I, the model performs best when choosing 0.01 as λ, which provides a decent learning balance. The result also degrades under 1.0 λ, which may caused by the over optimization on the tabletop-object discriminator.

Sampling methods. A derived dynamic sampling is proposed in the main paper. To make a more complete comparison, we also employ random sampling and grid sampling (i.e., voxelization) on Point Transformer [13]. Table II summarizes the results, where our dynamic sampling improves baseline with 1.17%↑. Grid sampling has no obvious effect, while random sampling hurts the performance.

Tasks are still challenging. Although under the help of our tabletop-aware learning strategies, the tasks are still difficult because:

(**1**) *Discriminator itself is challenging*: Discriminator is not 100% accurate especially on the joint areas of objects and tables (Paper Fig. 7 (b)), and will also misguide the learning of downstream networks, leaving the tasks difficult.

(**2**) *Clustering is not easy even based on perfect discriminator*: We extract tabletop objects from scenes (i.e., equivalent to perfect discrimination) then use DBSCAN, a popular point cloud cluster method. However, it performs not well

Table I: Ablations on λ.

λ	Segmentation mIoU (%)
0.0 (Baseline)	67.17
1.0	66.12
0.1	67.96
0.01	**69.09**
0.001	67.58

Table II: Ablations on sampling.

Sampling	Segmentation mIoU (%)
FPS (Baseline)	67.17
grid	67.18
random	65.52
dynamic (Ours)	**68.34**

Fig. IV: Cluster result.

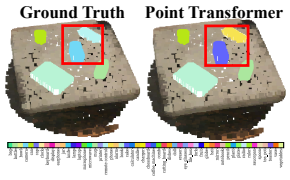

Fig. V: A failure case.

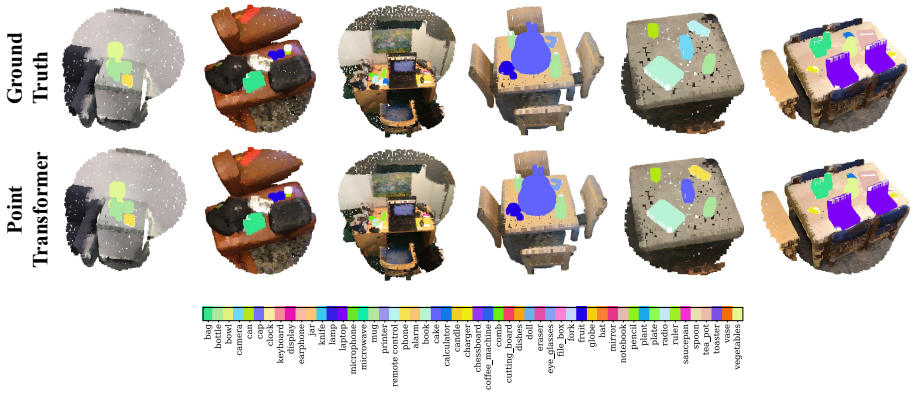

Fig. VI: Visualization of 3D semantic segmentation results on our dataset. The first row is the ground truth, and the second row is the scenes segmented by Point Transformer [13]. Each column indicates a scene. The colors in the bottom colorbar represent categories consistently across all scenes.

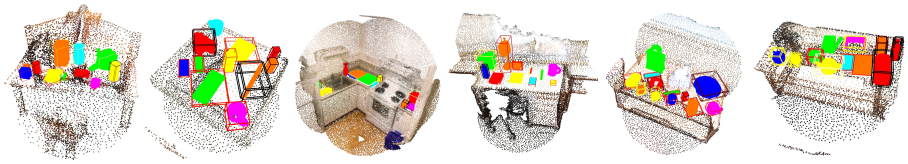

Fig. VII: Visualization of 3D object detection results by VoteNet [6] on our dataset. Each column denotes a scene. Each object instance shown with a different randomly assigned color.

due to the squeeze of objects (Fig. IV). Such failure is naturally expected when introducing big furniture from TO_ScanNet.

(**3**) *Segmentation/detection is not easy even based on perfect clustering*: Generally speaking, an ideal cluster algorithm can cluster the points of a same object, but it can NOT infer instance categories. For example in Fig. V, even applying a powerful deep network on simple TO_Vanilla for semantic segmentation, two objects are yet **misclassified** while the instances are successfully clustered.

C.3 Results.

Table III exhaustively itemizes the train/val split statistics of the instance quantities, plus semantic segmentation mIoU (seg) and object detection mAP@0.25 (det) of each category in three variants of TO-Scene.

Moreover, Fig. VI and Fig. VII intuitively visualize the results on our dataset for semantic segmentation and object detection tasks using Point Transformer [13] and VoteNet [6], respectively. As you can see, our dataset successfully drives the state-of-the-arts to perform stably well on both tasks.

D TO-Real

The main paper introduces TO-Real that is scanned from real world for verifying the practical value of our TO-Scene.

D.1 Data

Here we present more details of TO-Real dataset. To achieve the diversity, we employ 52 people of various ages from different professions to place the objects following their daily habits, where half of them are guided to arrange crowded objects. The whole process are taken place at diverse indoor rooms (e.g., kitchen, conference room, bedroom, bathroom, lobby, living room) in several offices or homes. After finishing the object placement, we employ 10 experts to scan the tabletop scenes using Microsoft Kinect [12] and annotate the data with both point-wise segmentation and object bounding boxes manually, which producing Real_Vanilla and Real_Crowd. Similarly, the experts also manually scan and label the whole rooms holding the tables, yielding Real_Scan. When collecting Real_Scan, expert workers may walk close to tabletops and scan the small tabletop objects from different views, avoiding the severely undesirable data quality of tabletop objects.

Fig. VIII illustrates more samples in TO-Real. For each reconstruction, the surface mesh with colors is shown, as well as a visualization of separate object instance labels are also available to indicate multiple instances. It conspicuously shows that the characteristics of three variants (Real_Vanilla, Real_Crowd, Real_Scan) exactly match their counterparts in TO-Scene shown in Fig. I.

Tab. IV shows the instance distributions of several classes, on which Real_Scan and TO_ScanNet are similar, which minimizes the gap between the two datasets.

Table III: Per-category train/val instance quantities and benchmark results on the three variants of TO-Scene. "seg" denotes segmentation mIoU (%), "det" indicates detection mAP@0.25 (%).

Category	TO_Vanilla			TO_Crowd			TO_ScanNet		
	train/val	seg	det	train/val	seg	det	train/val	seg	det
Tabletop object:									
bag	1.6k/368	94.8	87.8	1.4k/341	95.5	85.8	1.8k/346	91.2	87.3
bottle	3.2k/670	90.1	64.2	2.2k/378	87.9	72.4	3.2k/584	85.3	77.5
bowl	694/93	94.6	75.4	550/84	87.5	77.0	681/84	85.7	69.8
camera	881/183	87.1	81.0	1k/234	85.3	74.7	1k/188	81.1	81.6
can	5.3k/1k	93.6	70.4	4.4k/869	96.2	76.0	6.2k/1k	90.2	82.2
cap	995/225	93.5	87.1	772/207	87.8	88.2	979/235	85.4	90.5
clock	242/40	31.1	28.6	152/39	28.9	18.8	255/45	14.6	20.5
keyboard	169/49	86.4	51.2	260/56	89.3	37.7	163/45	81.0	39.0
display	337/104	94.4	93.9	336/77	91.4	84.8	331/105	89.7	92.5
earphone	576/164	93.2	79.7	859/181	96.7	85.8	670/181	91.8	73.1
jar	335/47	63.9	39.5	262/43	61.7	53.5	334/36	64.6	60.0
knife	88/14	39.1	19.0	103/22	53.3	17.5	85/13	36.9	28.9
lamp	679/149	88.9	84.6	744/150	90.0	85.6	670/129	72.1	83.3
laptop	973/185	94.6	90.3	969/223	96.9	96.8	946/179	95.5	91.8
microphone	26/5	5.6	40.9	21/6	0.0	1.3	34/6	16.3	0.9
microwave	133/26	94.6	96.7	111/21	94.3	83.0	119/26	87.1	93.4
mug	5.4k/1k	95.0	81.7	4.2k/788	96.4	90.1	5.5k/950	92.0	86.6
printer	198/56	89.7	94.3	163/47	87.4	88.5	211/56	84.7	76.8
remote control	467/85	49.0	19.4	385/85	51.7	29.4	558/86	33.5	24.4
phone	4k/815	76.4	24.6	3k/655	68.8	32.7	4.3k/748	61.3	27.6
alarm	783/155	74.3	59.2	714/107	66.0	48.9	743/173	51.4	71.4
book	1.5k/354	81.6	60.0	1.5k/334	67.3	62.7	1.6k/349	68.5	59.5
cake	254/40	83.4	83.6	282/66	76.5	68.2	309/38	85.7	57.1
calculator	1k/239	76.8	47.7	1.1k/197	65.7	38.4	1.2k/223	65.2	33.6
candle	268/66	88.6	62.1	190/35	40.7	53.4	306/62	78.9	75.8
charger	1.7k/394	88.2	34.3	1.7k/347	85.2	36.9	2.2k/438	72.0	39.9
chessboard	121/20	92.1	82.0	112/13	87.2	69.3	115/27	87.9	89.4
coffee machine	118/20	92.2	91.5	100/27	86.8	91.2	108/16	97.4	86.5
comb	1k/239	91.4	40.2	322/51	70.0	32.7	622/142	74.3	27.7
cutting board	126/21	89.9	70.8	102/20	96.3	41.3	108/16	75.6	50.2
dishes	231/26	92.7	53.1	241/40	96.1	70.5	213/23	88.9	88.5
doll	126/24	65.0	50.9	49/17	60.3	61.1	121/31	13.5	27.1
eraser	1k/258	65.8	9.7	1.5k/298	71.4	12.1	1.3k/291	34.0	9.1
eye glasses	2.3k/502	93.4	68.0	1.7k/342	94.8	78.6	2.5k/483	90.4	77.0
file box	168/55	93.0	84.2	117/42	95.8	87.0	198/56	92.1	87.0
fork	357/42	34.5	31.1	361/43	38.2	16.8	375/35	37.5	6.7
fruit	2.7k/466	94.2	76.0	2.5k/463	88.7	77.9	3.2k/449	81.6	75.3
globe	371/88	98.1	87.9	264/52	98.0	95.9	375/74	96.3	94.1
hat	196/43	54.0	76.8	160/32	39.4	52.9	202/41	36.9	67.6
mirror	90/19	59.5	43.5	43/11	73.7	20.0	44/12	21.8	33.3
notebook	1.5k/331	69.7	28.4	1.4k/321	62.4	32.7	1.6k/313	54.5	30.5
pencil	1.1k/278	71.9	14.5	1.5k/344	75.8	15.1	1.3k/289	58.5	6.9
plant	1.5k/306	94.0	87.7	1.3k/251	96.9	89.7	1.4k/288	91.9	89.9
plate	390/52	93.8	34.6	314/57	98.2	47.7	375/38	90.4	67.6
radio	288/51	52.0	50.8	283/47	50.5	60.1	280/37	37.4	27.4
ruler	858/190	73.4	13.5	832/16	67.4	10.0	899/189	55.8	8.9
saucepan	153/23	94.2	73.5	121/25	87.1	57.5	127/16	69.9	75.9
spoon	429/58	60.4	32.6	365/56	46.9	19.9	456/43	49.5	16.9
tea pot	1.1k/201	97.5	89.7	1k/198	95.7	93.4	1.1k/203	94.5	83.4
toaster	93/12	83.2	66.1	62/6	45.6	41.2	91/8	86.9	46.3
vase	1.2k/243	89.5	71.5	638/141	83.4	79.1	1k/188	71.7	79.3
vegetables	73/11	3.89	34.9	85/12	9.0	10.5	79/13	20.1	7.4

Continued at next page

Table III: Per-category train/val instance quantities and benchmark results on the three variants of TO-Scene. "seg" denotes segmentation mIoU (%), "det" indicates detection mAP@0.25 (%). (Continued)

Category	TO_Vanilla			TO_Crowd			TO_ScanNet		
	train/val	seg	det	train/val	seg	det	train/val	seg	det
Big furniture:									
wall	-	-	-	-	-	-	19.6k/4.1k	76.7	-
floor	-	-	-	-	-	-	9.8k/2.1k	94.8	-
cabinet	-	-	-	-	-	-	6.8k/1.3k	59.4	49.3
bed	-	-		-	-	-	943/218	80.3	88.8
chair	-	-	-	-	-	-	15.9k/3.8k	86.7	82.2
sofa	-	-	-	-	-	-	2.1k/331	75.2	89.5
table	-	-	-	-	-	-	6.5k/1.3k	71.9	61.8
door	-	-	-	-	-	-	6.8k/1.4k	55.6	45.8
window	-	-	-	-	-	-	3.5k/863	59.4	37.1
bookshelf	-	-	-	-	-	-	961/306	63.9	31.7
picture	-	-	-	-	-	-	2.2k/628	20.8	5.2
counter	-	-	-	-	-	-	1k/165	58.5	45.9
desk	-	-		-	-	-	2.9k/567	62.7	70.6
curtain	-	-	-	-	-	-	878/137	58.3	48.9
refrigerator	-	-	-	-	-	-	887/166	61.8	61.2
shower curtain	-	-	-	-	-	-	349/95	71.3	70.4
toilet	-	-	-	-	-	-	522/121	85.9	88.6
sink	-	-	-	-	-	-	1.3k/284	58.1	37.5
bathtub	-	-	-	-	-	-	344/104	82.6	90.0
other furniture	-	-	-	-	-	-	7.5k/1.5k	45.0	41.3

Class	chair	table	window	bookshelf	counter	sink	bottle	camera	cap	fruit
Real	32.6	9.15	6.15	1.54	1.54	1.54	10.23	3.21	3.40	11.22
TO	27.3	10.75	6.02	1.75	1.65	2.27	9.64	3.09	3.08	9.34

Table IV: Instance distributions (%) in Real_Scan and TO_ScanNet.

D.2 Tabletop-aware Learning on TO-Real.

Tab. V shows the improvements (segmentation mIoU) on several categories after applying the proposed strategies (+FV+DS) in real scans.

D.3 Visualization of Segmentation.

Fig. IX visualizes the semantic segmentation result on TO-Real, where Point Transformer [13] is trained on the training set of TO-Scene, and is *directly* tested on TO-Real. Although purely trained on TO-Scene and tested on realistic data, the model still get acceptable result. This intuitively proves the practical value of our TO-Scene dataset.

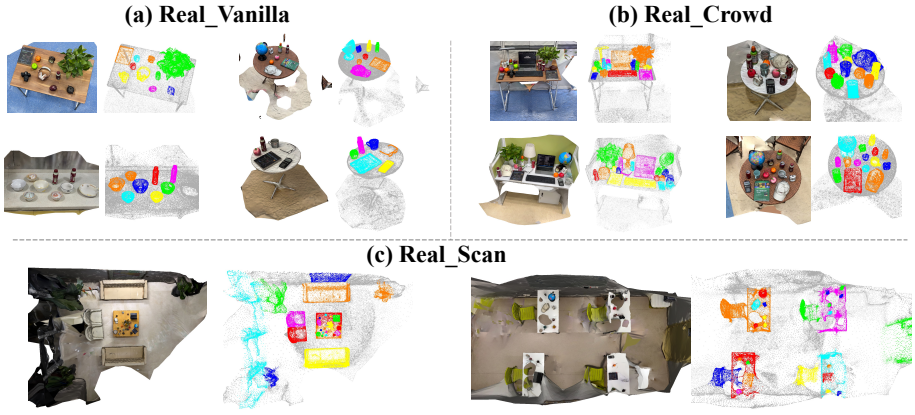

Fig. VIII: A variety of example scenes in three variants of TO-Real. Each sample is shown with the original reconstructed mesh at left, and object instance assigned with random color at right.

Class	bottle	bowl	cap	keyboard	lamp	mug	alarm	book	fruit
+FV+DS	0.43↑	0.44↑	1.02↑	0.54↑	0.45↑	1.36↑	0.74↑	0.94↑	0.65↑

Table V: Per-category mIoU improvements (%) on TO-Real.

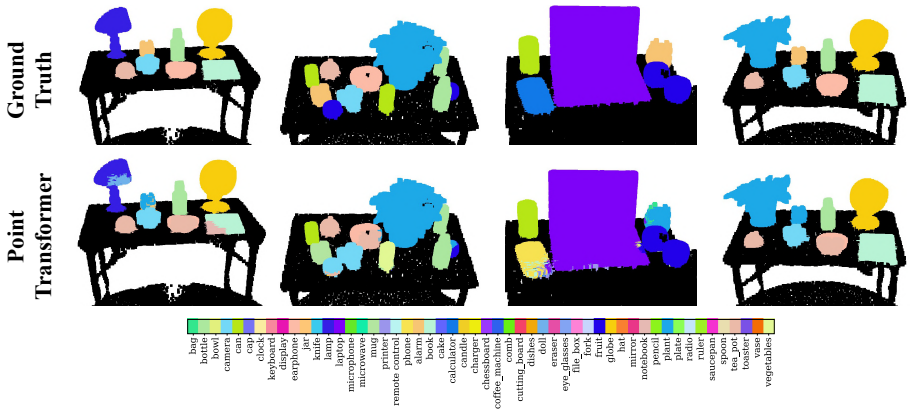

Fig. IX: Visualization of 3D semantic segmentation test results on TO-Real. The first row is the ground truth, and the second row is the TO-Real scenes segmented by Point Transformer [13] that trained on our TO-Scene. Each column indicates a scene. The colors in the bottom colorbar indicate classes consistently across all scenes.

References

1. Chang, A.X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., Xiao, J., Yi, L., Yu, F.: Shapenet: An information-rich 3d model repository. arXiv preprint arXiv:1512.03012 (2015)
2. Community, B.O.: Blender - a 3D modelling and rendering package. Blender Foundation, Stichting Blender Foundation, Amsterdam (2018), `http://www.blender.org`
3. Dai, A., Chang, A.X., Savva, M., Halber, M., Funkhouser, T., Nießner, M.: Scannet: Richly-annotated 3d reconstructions of indoor scenes. In: CVPR (2017)
4. Liu, Z., Zhang, Z., Cao, Y., Hu, H., Tong, X.: Group-free 3d object detection via transformers. In: ICCV (2021)
5. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: Pytorch: An imperative style, high-performance deep learning library. In: NeurIPS (2019)
6. Qi, C.R., Litany, O., He, K., Guibas, L.J.: Deep hough voting for 3d object detection in point clouds. In: ICCV (2019)
7. Silberman, N., Hoiem, D., Kohli, P., Fergus, R.: Indoor segmentation and support inference from rgbd images. In: ECCV (2012)
8. Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., Xiao, J.: 3d shapenets: A deep representation for volumetric shapes. In: CVPR (2015)
9. Xu, M., Ding, R., Zhao, H., Qi, X.: Paconv: Position adaptive convolution with dynamic kernel assembling on point clouds. In: CVPR (2021)
10. Zeng, A., Song, S., Nießner, M., Fisher, M., Xiao, J., Funkhouser, T.: 3dmatch: Learning local geometric descriptors from rgb-d reconstructions. In: CVPR (2017)
11. Zhang, Z., Sun, B., Yang, H., Huang, Q.: H3dnet: 3d object detection using hybrid geometric primitives. In: ECCV (2020)
12. Zhang, Z.: Microsoft kinect sensor and its effect. IEEE MultiMedia (2012)
13. Zhao, H., Jiang, L., Jia, J., Torr, P., Koltun, V.: Point transformer. In: ICCV (2021)