Supplementary Material

A. OVERVIEW

This supplementary material offers additional technical details, dataset visualization, and qualitative results to support the main paper. In this overview, we provide specific information on how our dataset generator and sample dataset will be managed. We offer SynTable as an open-source contribution to the academic community, with the hope that it will prove useful for various research endeavors. Section B provides additional technical details to clearly explain our dataset generation methodology. Section C provides a comprehensive elaboration of the evaluation metrics employed. Section D illustrates how occlusion order accuracy is calculated and the validity of the metric. Furthermore, Section E delineates the process of generating an occlusion order directed graph from the occlusion order adjacency matrix to classify objects in three distinct order layers. Section F showcases some qualitative inference results of UOAIS-Net on the OSD-Amodal dataset. Section G presents additional results from our experiments in Section 5 of the main paper. These experiments were conducted by training and evaluating AmodalMRCNN, ORCNN, and ASN on the same datasets as UOAIS-Net. The results observed from the experiments are consistent with our claims in the main paper and further demonstrate the capability of SynTable to improve the performance of a variety of different UOAIS models.

A.1. Video Demonstration of SynTable-Sim Generation Process

In addition to this document, we include a demonstration video as part of our supplementary material to demonstrate in detail the process of generating a custom synthetic dataset using SynTable. We refer readers to the demonstration video for a detailed visualization of the dataset generation process to enhance their understanding of our work. The video can be found at `https://www.youtube.com/watch?v=zHM8H58Kn3E`.

A.2. SynTable Pipeline Source Code

We also include our Python source code to install and run the SynTable pipeline as part of our supplementary material. We provide a ReadMe file to guide users on how to use the pipeline. Our Python source code can be found at `https://github.com/ngzhili/SynTable`. We will actively monitor all issues raised by future users through the GitHub repository and also via email so that users will benefit from our work as much as possible.

A.3. Management of the SynTable-Sim Dataset

All the CAD models of the objects used in our SynTable-Sim dataset, as well as the dataset itself, are hosted in the Zenodo open repository, free for all to download. The DOI of our dataset is `10.5281/zenodo.10565517`. The dataset can be accessed at `https://doi.org/10.5281/zenodo.10565517`.

A.4. Authors' Statement

We, the authors of this manuscript, hereby affirm our responsibility for the content and ethical conduct of the research presented in this work. By submitting this manuscript for publication, we declare that:

1. We acknowledge that we are solely responsible for the accuracy, originality, and ethical integrity of the content presented in this manuscript. Any violation of copyright, ethical standards, or the rights of individuals is entirely our responsibility.
2. We confirm that all data presented in this manuscript are either original or used with appropriate permissions and in accordance with applicable licenses. Any third-party data, figures, or other content used in this work have been appropriately credited, and we have obtained the necessary permissions or licenses for their use.
3. We affirm that the manuscript complies with all relevant laws, regulations, and ethical guidelines. In case of any legal claims or disputes related to the content of this manuscript, we, as the authors, assume full responsibility.
4. We attest that all listed authors have made substantial contributions to the conception, design, data acquisition, analysis, and interpretation of the work. All authors have reviewed and approved the final version of the manuscript and agree to its submission for publication.

B. METHOD

Our dataset generation pipeline is illustrated using the diagram in Figure 2 in the main paper. We use a YAML file to store the parameters and configurations of the scenes to be rendered. The objects and settings required to render the scene are retrieved based on the instructions in the configuration file. We collectively term the objects, materials, and light sources used in our pipeline as assets. Thereafter, the tabletop scene with objects floating above the table is rendered in Isaac Sim. We then run a physical simulation to drop the rendered objects onto the table. For every view within a scene, camera viewpoints and lighting conditions are re-sampled. Subsequently, ground truth annotations are obtained and systematically recorded to create the dataset.

B.1. Preparing Each Scene

The method to prepare each scene is shown in Figure 4. A table is randomly sampled from the assets in Omniverse Nucleus and is rendered at the center of a room. The texture and materials of the table, ceiling, wall, and floor are randomized for every scene to ensure domain randomization. The objects are added to the scene with randomized x, y, and z coordinates and orientations. We randomly sample (with replacement) N_{lower} to N_{upper} objects to render for each scene. By default, $N_{lower} = 1$, $N_{upper} = 40$. Each object is initialized with real-life dimensions, randomized rotations and coordinates, allowing for diverse object arrangements across scenes. Each object also has mass and collision properties so that they can be dropped onto the tabletop in our physics simulation.

B.2. Physical Simulation of Each Scene

Upon completing the scene preparation, the rendered objects are dropped onto the table surface using a physics simulation. The simulation is paused after t seconds ($t = 5$ by default). During the simulation, any objects that rebound off the tabletop surface and fall outside the spatial coordinate region of the tabletop surface (i.e., either below the table or beyond the width and length of the table) are automatically removed. This is necessary to prevent the inclusion of extraneous and irrelevant objects outside the specified tabletop region during the annotation process from different viewpoints.

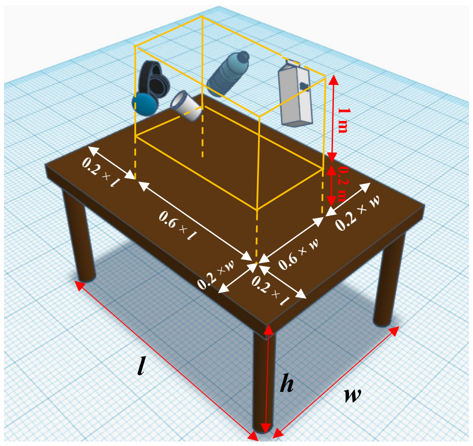


Fig. 4: Initialization of objects with randomized coordinates and rotations. The initial position of the objects in the scene is randomized but constrained to be within the dimensions of the 3D orange box. The orange box is 0.2 m above the tabletop. The roll, pitch, and yaw of each object are also randomly sampled within the range of 0° to 360°.

B.3. Sampling of Camera Viewpoints

To capture annotations for each scene from multiple viewpoints, we enhance the approach by Gilles *et al.* [21]—which only uses fixed viewpoint positions—by introducing a feature that captures V number of viewpoints at random positions within two concentric hemispheres, as illustrated in Figure 5. V can be set by the user. The radii of the two concentric hemispheres are uniformly sampled within the range r_{view_lower} m to r_{view_upper} m, where r_{view_lower} and r_{view_upper} are defined in Equations 2 and 3. Users may also set fixed values for r_{view_lower} and r_{view_upper} should they wish to do so.

$$r_{view\_lower} = max\left(\frac{w}{2}, \frac{l}{2}\right) \tag{2}$$

$$r_{view\_upper} = 1.7 \times r_{view\_lower} \tag{3}$$

The hemisphere's spherical coordinates are parameterized using three variables r_{view}, u, and v. To generate the camera coordinates in the world frame, we first obtain the radius of the hemisphere r_{view} by uniform sampling between r_{view_lower} and r_{view_upper}. Next, we uniformly sample $u, v \in [0,1]$, then substitute all the sampled values into Equations 4, 5 and 6 to compute the cartesian coordinates of the camera.

$$x = r_{view} \sin(\arccos(1-v))\cos(2\pi u) \tag{4}$$

$$y = r_{view} \sin(\arccos(1-v))\sin(2\pi u) \tag{5}$$

$$z = r_{view} \cos(\arccos(1-v)) \tag{6}$$

Once the camera coordinates are set, the orientation of each camera is set such that each viewpoint looks directly at the center of the tabletop surface (0, 0, h).

B.4. Sampling of Lighting Conditions

To simulate different indoor lighting conditions, we resample L spherical light sources between L_{lower} to L_{upper} for each viewpoint (Figure 6). By default, we set L_{lower} and L_{upper} to be 0 and 2, respectively. To position L spherical light sources for a viewpoint, we adopt a similar approach to the camera viewpoint sampling

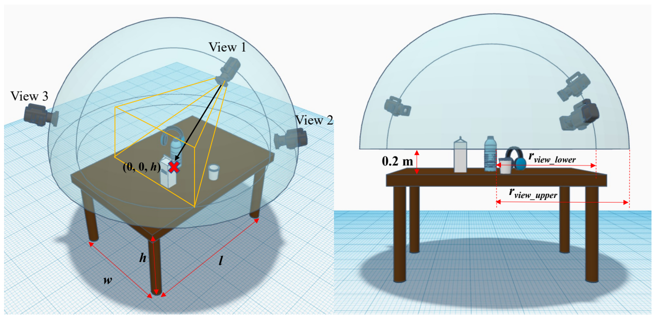


Fig. 5: Sampling of camera viewpoints within concentric hemispheres (shown in blue). The two concentric hemispheres' origins are centered at the tabletop surface's center coordinate with an offset of 0.2 m in the positive z direction in the world frame. This allows the camera viewpoints to at least have a direct line of sight to the tabletop surface to capture part of the tabletop plane. This figure is best viewed zoomed in.

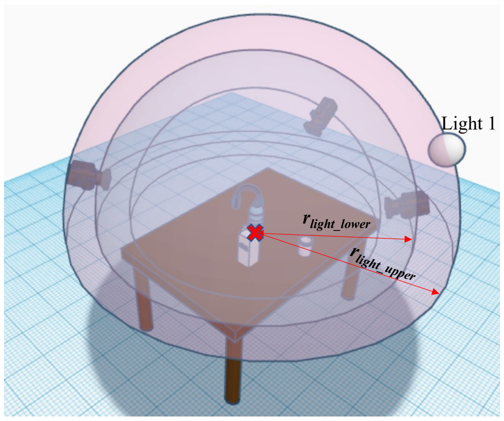


Fig. 6: Sampling of lighting within concentric hemispheres (shown in pink). Each spherical light source lies within the constraints of two concentric hemispheres of arbitrary radius between r_{light_lower} to r_{light_upper}. Note that the radii constraints for the spherical light source concentric hemispheres are larger than those for the camera viewpoints' and are customizable by the user.

method discussed in Section 3.3 in the main paper. In contrast to the approach by Back *et al.* [7], we use spherical light sources that emit light in all directions. Furthermore, we uniformly sample light source temperatures between 2,000 K to 6,500 K. The default light intensity of each light source is uniformly sampled between 100 lx and 20,000 lx, and the default light intensity of ceiling lights in the scene is also sampled uniformly between 100 lx and 2,000 lx. To achieve diverse indoor lighting conditions for tabletop scenes, users have the flexibility to adjust the number of spherical light sources, as well as their intensities and temperatures.

Similar to the sampling method for the camera viewpoint coordinates, we have designed a feature that samples the lower and upper radii bounds for the light sources based on the camera hemisphere's upper bound radius, r_{view_upper}. The sampled lower and upper bound radii constraints for the lighting hemisphere r_{light_lower} and r_{light_upper} are as follows:

$$r_{light\_lower} = r_{view\_upper} + 0.1m \tag{7}$$

$$r_{light\_upper} = r_{light\_lower} + 1m \tag{8}$$

B.5. Capturing of Ground Truth Annotations

The process of capturing the ground truth annotations for a scene is illustrated in Figure 3 in the main paper. At each view, the RGB and depth images of the tabletop scene will be captured (Figure 3(a)). The built-in instance segmentation function in Isaac Sim Replicator Composer is employed to capture the instance segmentation mask of the entire scene from a viewpoint (Figure 3(b)). Subsequently, each object's visible mask is cropped from the instance segmentation mask of the scene. To obtain the amodal mask of each object on the simulated tabletop scene, we have developed the subsequent steps.

Initially, all objects' visibility is disabled. For each object o within the scene, its visibility is enabled, and the instance segmentation function is utilized to capture its amodal mask (Figure 3(c)). Following this, we compute the object's occlusion mask and occlusion rate, as presented in (Figure 3(d)). The occlusion mask of an object o can be acquired by subtracting its visible mask from its amodal mask.

The occlusion rate of the object o can be computed by dividing the number of pixels in the occlusion mask by the number of pixels in the amodal mask. If the occlusion rate of the object o is equal to 1, it implies that object o is completely obscured from the viewpoint, thus we do not save the object o's annotation for this view. The visibility of object o is then disabled to capture the masks of the next object. Following the preservation of all objects' masks, we use Algorithm 1 to generate the Occlusion Order Adjacency Matrix (OOAM) for this viewpoint (Figure 3(e)). For a scene with M objects, the OOAM contains $M \times M$ elements, where the element (i, j) is a binary value in the matrix which indicates whether object i occludes object j. Given the OOAM, we can easily construct the Occlusion Order Directed Graph (OODG) to visualize the occlusion order in the viewpoint (Figure 3(e)). We provide a detailed explanation of the OODG in our supplementary materials. After that, the visibility of all objects is enabled to prepare for the capturing of annotations from the next viewpoint of the scene.

C. DETAILS ABOUT EVALUATION METRICS

In this paper, we employ the precision/recall/F-measure (P/R/F) metrics, as defined in [10, 27, 28]. This metric favors methods that accurately segment the desired objects while penalizing those that produce false positives. Specifically, the precision, recall, and F-measure are calculated between all pairs of predicted and ground truth objects. The Hungarian method, employing pairwise F-measure, is utilized to establish a match between predicted objects and ground truth. Given this matching, the Overlap P/R/F is computed by:

$$P = \frac{\sum_i |c_i \cap g(c_i)|}{\sum_i |c_i|}, \quad R = \frac{\sum_i |c_i \cap g(c_i)|}{\sum_j |g_j|} \tag{9}$$

$$F = \frac{2PR}{P+R} \tag{10}$$

where c_i denotes the set of pixels belonging to predicted object i, $g(c_i)$ is the set of pixels of the matched ground truth object of c_i after Hungarian matching, and g_j is the set of pixels for ground truth object j.

Although the aforementioned metric provides valuable information, it fails to consider the boundaries of the objects. Therefore, Xie *et al.* [10] proposed the Boundary P/R/F measure to supplement the Overlap P/R/F. The calculation of Boundary P/R/F involves the same Hungarian matching as used in the computation of Overlap P/R/F. Given these matchings, the Boundary P/R/F is computed by:

$$P = \frac{\sum_i |c_i \cap D[g(c_i)]|}{\sum_i |c_i|}, \quad R = \frac{\sum_i |D[c_i] \cap g(c_i)|}{\sum_j |g_j|} \tag{11}$$

$$F = \frac{2PR}{P+R} \tag{12}$$

Here, overloaded notations are used to represent the sets of pixels belonging to the boundaries of the predicted object i and the ground truth object j as c_i and g_j, respectively. The dilation operation is denoted by $D[\cdot]$, which allows for some tolerance in the prediction. The metrics we use are a combination of the F-measure described in [29] and the Overlap P/R/F as defined in [27].

In our work, we use the Overlap and Boundary P/R/F evaluation metrics to evaluate the accuracy of the predicted visible, invisible, and amodal masks. In the context of the Overlap P/R/F metrics, c_i denotes the set of pixels belonging to the predicted visible, invisible, and amodal masks, $g(c_i)$ denotes the set of pixels belonging to the matched ground-truth visible, invisible and amodal masks annotations, and g_j is the ground-truth visible, invisible and amodal mask. The meaning of c_i, $g(c_i)$, and g_j are similar in the context of the Boundary P/R/F metrics.

An additional vital evaluation metric used in our paper is the *F@.75*. This metric represents the proportion of segmented objects with an Overlap F-measure greater than 0.75. It is important not to confuse this metric with the F-measure computed for the Overlap and Boundary P/R/F. The F-measure for Overlap and Boundary is a harmonic mean of a model's average precision and average recall, while *F@.75* indicates the percentage of objects from a dataset that can be segmented with high accuracy. The F in F@.75 refers to the F-measure computed for a ground truth object after the Hungarian matching of the ground truth mask j with the predicted mask i as defined in [27] and stated in Equation (14).

$$P_{ij} = \frac{|c_i \cap g_j|}{|c_i|}, \quad R_{ij} = \frac{|c_i \cap g_j|}{|g_j|} \tag{13}$$

$$F_{ij} = \frac{2P_{ij}R_{ij}}{P_{ij}+R_{ij}} \tag{14}$$

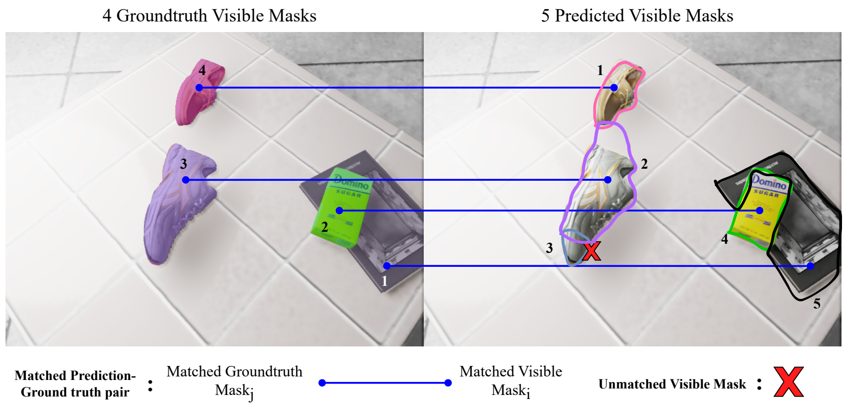
The notation c_i denotes the set of pixels that belong to a predicted region i, while g_j represents all the pixels that belong to a non-background ground truth region j. In addition, P_{ij} represents the precision score, R_{ij} represents the recall score, and F_{ij} represents the F-measure score that corresponds to this particular pair of predicted and ground truth regions.

D. OCCLUSION ORDER ACCURACY ACC_{OO} METRIC

Given an image v that depicts a typical cluttered tabletop scene, we get the ground truth-prediction assignment pairs after Hungarian matching as illustrated in Figure 7. The predicted masks will then be re-indexed to match the indices of the ground truth masks. Following that, the *predVisible* and *predOcclusion* masks that belong to the assigned pairs will be extracted. After that, the ground truth OOAM (*gtOOAM*) and the predicted OOAM (*predOOAM*) will be obtained using Algorithm 1 in the main paper.

Figure 7 also illustrates the calculation of occlusion order accuracy in an image v. The similarity matrix (denoted as *similarityMatrix* in Figure 7) is obtained by conducting an element-wise equality comparison between the *gtOOAM* and *predOOAM*. After that, ACC_{oo} can be calculated using Equation 1 in the main paper.

Get best assignment of prediction masks to groundtruth mask that maximises the F-measure



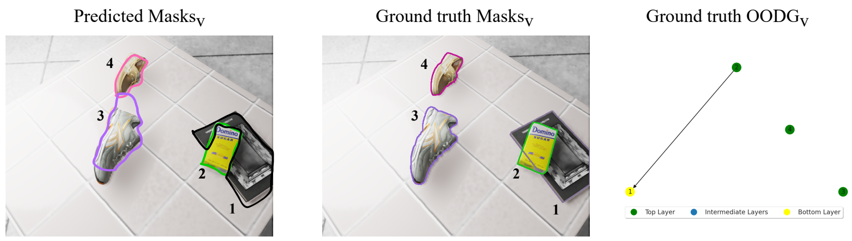
4 Groundtruth Visible Masks
5 Predicted Visible Masks
1
2
3
4
1
2
3
4
5
Matched Prediction-Ground truth pair :
Matched Groundtruth Mask$_j$
Matched Visible Mask$_i$
Unmatched Visible Mask :

F-Measure table

ground truth mask j \ predicted mask i	1	2	3	4	5
1	0	0	0	0.12	0.87
2	0	0	0	0.95	0.08
3	0	0.74	0.28	0	0
4	0.94	0.06	0	0	0

Blue highlighted cells represent highest F-measure for a ground truth - predicted mask pair

After Hungarian matching, matched pairs are as follows:
ground truth object ids (1, 2, 3 ,4)
predicted object ids (5, 4, 2, 1).
predicted object ids are re-indexed to (1,2,3,4).



Element Wise Comparison

$$\left\{ \begin{array}{c|cccc} \text{occluder} \backslash \text{occludee} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 1 & 0 & 0 \\ 2 & 1 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 & 1 \\ 4 & 0 & 0 & 0 & 0 \end{array} == \begin{array}{c|cccc} \text{occluder} \backslash \text{occludee} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 & 0 \\ 4 & 0 & 0 & 0 & 0 \end{array} \right\} = \begin{array}{cccc} 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{array}$$

$$predOOAM_v \qquad gtOOAM_v \qquad similarityMatrix_v$$

$$gtOOAMSize_v = 4 \times 4 = 16 \qquad gtOOAMDiagonalSize_v = 4 \qquad sum(similarityMatrix_v) = 14$$

$$(ACC_{OO})_v = \frac{sum(similarityMatrix_v) - gtOOAMDiagonalSize_v}{gtOOAMSize_v - gtOOAMDiagonalSize_v} = \frac{14-4}{16-4} = 0.83$$

Fig. 7: Hungarian Matching and calculating Occlusion Order Accuracy of image *v*

In Equation [1], the ACC_{oo} represents the ratio of the number of correct predicted occlusion nodes over the number of ground truth occlusion nodes. Let *#correctPredictedOcclusionNodes* denote the number of correct occluder and occludee predictions for all objects in a viewpoint (represented by green highlighted cells in *similarityMatrix* in Figure [7]).

A summation of all the elements in the similarity matrix is carried out to obtain *#correctPredictedOcclusionNodes*. Let *#groundtruthOcclusionNodes* denote the number of ground truth occluder and occlude nodes in a viewpoint. To obtain *#groundtruthOcclusionNodes*, we count the number of elements (*gtOOAMSize*) in the ground truth OOAM. As an object cannot occlude itself, the diagonal of any OOAM is always 0, and the diagonal of any similarity matrix is always 1 (depicted as grey highlighted cells in Figure [7]). Thus, we subtract the number of elements along the diagonal of the gtOOAM (denoted by *gtOOAMDiagonalSize*) from the calculation of *#correctPredictedOcclusionNodes* and *#groundtruthOcclusionNodes*.

Correct occlusion order predictions occur when the predicted occlusion relationship for each object matches the ground truth. Incorrect occlusion order predictions can result from erroneous predictions or missing visible mask predictions of object instances. When there are missing predictions, setting the corresponding row and column of the missing object instance in the similarity matrix to 0 penalizes the model for the missing object predictions. The smaller element-wise sum of the similarity matrix leads to a smaller ACC_{oo}. This demonstrates the appropriate assignment of penalties by ACC_{oo} to different error types for measuring object occlusion ordering in a scene, highlighting its significance in the context of scene understanding for robotic grasp planning.

E. OCCLUSION ORDER DIRECTED GRAPH (OODG)

After obtaining the Occlusion Order Adjacency Matrix (OOAM), we can generate the OODG from it. For each non-zero entry (i, j) in the OOAM, we draw a directed edge from node i to node j. If the entry is zero, we do not draw an edge. A non-zero entry at (i,j) represents that object i is occluding object j.

For example, the OOAM generated in Figure 8 shows that $(i, j) = (1, 12)$ where i and j are the object indices (the bounding box labels) in the image. This means that object 1 occludes object 12, and a directed edge will point from object 1 to 12. From the generated Directed Occlusion Graph, we can also check if the graph is cyclic or acyclic using graph cyclic detection methods such as Depth First Search (DFS) and Breadth First Search (BFS). Only if the graph has no directed cycles (Directed Acyclic Occlusion Graph) can topological sorting be implemented to find the picking sequence to grasp all objects in the scene safely.

In the generated Occlusion Order graph, we further classify objects in three different order layers - Top, Intermediate, and Bottom. Objects at the top layer represent objects that are not occluded by any other object and can be grasped directly. Objects in the intermediate layers mean that they are occluded but they also occlude other objects. For objects in the bottom layer, they are occluded but they do not occlude other objects.

F. QUALITATIVE INFERENCE RESULTS OF UOAIS-NET ON THE OSD-AMODAL DATASET

After training the UOAIS-Net model [7] on both SynTable-Sim and UOAIS-Sim (tabletop) datasets [7], we present some of our qualitative results in Figure 9. As discussed in the main text of our paper, the UOAIS-Net trained on the SynTable-Sim dataset exhibits superior performance in contrast to the UOAIS-Net trained on the UOAIS-Sim tabletop dataset. This observation is further supported by the inference results presented in Figure 9. Furthermore, as the scene becomes more and more cluttered, the UOAIS-Net model trained on the SynTable-Sim dataset evidently outperforms that of the UOAIS-Net trained on the UOAIS-Sim tabletop dataset. In the context of robotic grasping on cluttered tabletops, foreground masking algorithms can be utilized to filter out the background and out-of-the-table predicted object instances.

G. ADDITIONAL QUANTITATIVE INFERENCE RESULTS ON THE OSD-AMODAL DATASET

We also evaluated the effectiveness of SynTable-Sim across different UOAIS models comprising distinct architectures. Table 5 compares the performance of UOAIS models— Amodal MRCNN, ORCNN, ASN, and UOAIS-Net—on the OSD-Amodal dataset after training on the UOAIS-Sim tabletop dataset and our SynTable-Sim sample dataset. For each model result in our experiments, we used seed 7 for training. Generally, across most metrics, the UOAIS models trained on SynTable-Sim outperform the same models trained on the UOAIS-Sim tabletop dataset. There is also a significant improvement in the results of ACC_{oo} for Amodal MRCNN, ORCNN, and ASN when trained on our SynTable-Sim as compared to the UOAIS-Sim tabletop dataset. This is consistent with the performance trend observed for UOAIS-Net and, therefore, demonstrates that SynTable is an effective tool for generating high-quality datasets that can improve the performance of UOAIS models. A detailed breakdown of the precision P, recall R, and F-measure F, and *F@.75* scores for the amodal, invisible, and visible masks are shown in Table 6.

As shown in Table 7, the UOAIS models trained on the SynTable-Sim dataset outperform the same models trained on the UOAIS-Sim tabletop dataset in all metrics when they are benchmarked on the SynTable-Sim validation dataset.

Table 5: The performance of Amodal MRCNN, ORCNN, ASN, and UOAIS-Net on the **OSD-Amodal dataset** after training on the UOAIS-Sim and SynTable-Sim datasets. UOAIS-Net is trained with RGB-D images. **OV**: Overlap F-measure, **BO**: Boundary F-measure, **F@.75**: Percentage of segmented objects with an Overlap F-measure greater than 0.75, $\mathbf{ACC_{OO}}$: Occlusion Order Accuracy

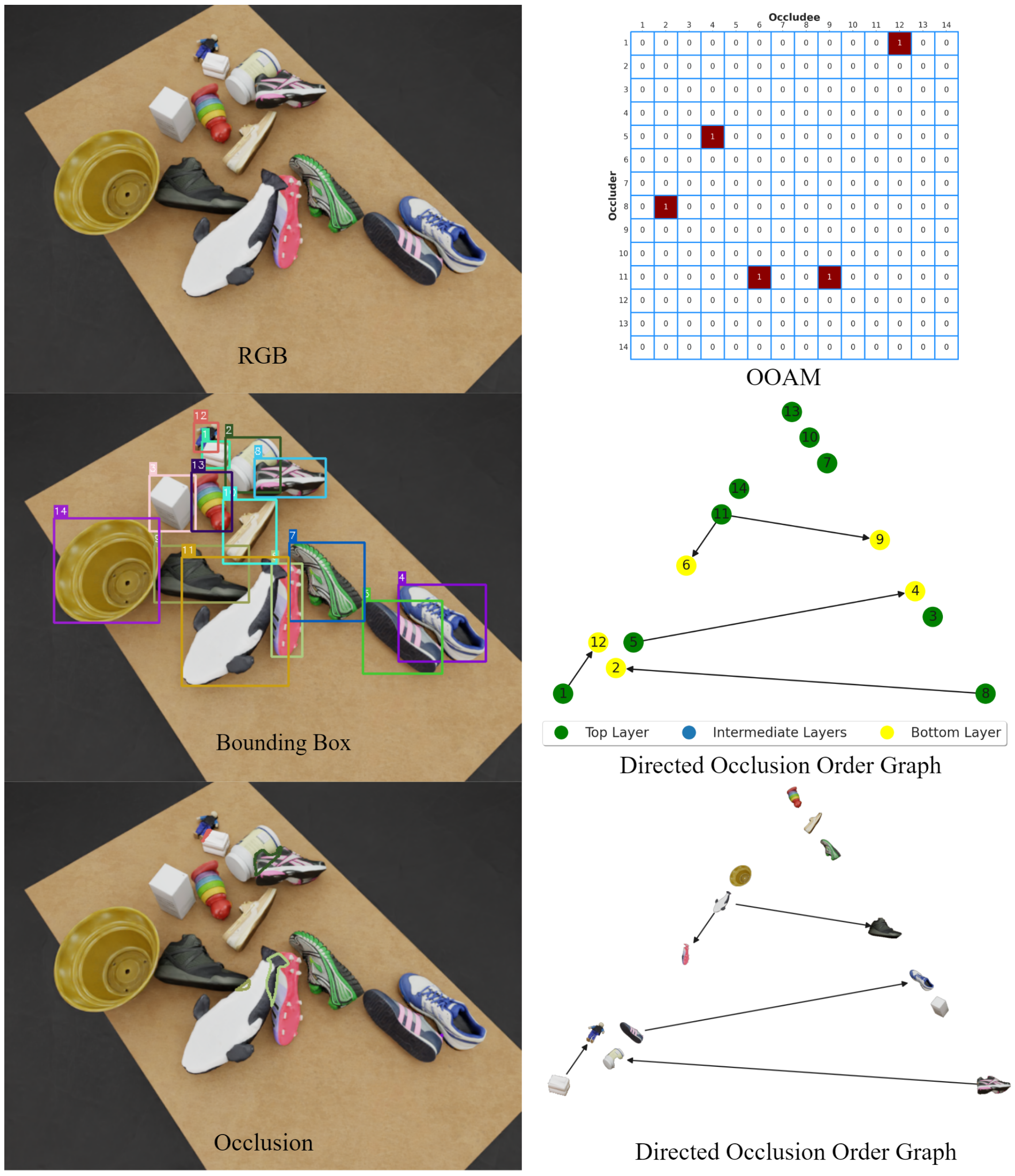
Training Set	Method	Amodal Mask			Invisible Mask			Occlusion		Visible Mask			ACC_{OO}
		OV	BO	*F@.75*	OV	BO	*F@.75*	$F_{\mathcal{O}}$	$ACC_{\mathcal{O}}$	OV	BO	*F@.75*	
UOAIS-Sim (Tabletop)	Amodal MRCNN	36.7	26.9	45.7	8.8	4.8	7.7	39.2	54.8	38.7	26.3	32.2	15.6
	ORCNN	36.3	25.4	47.0	12.2	6.7	9.0	43.8	59.2	30.5	21.8	29.6	21.5
	ASN	40.5	33.6	49.8	17.4	12.1	15.0	47.0	63.2	39.3	31.6	36.8	17.8
	UOAIS-Net	49.0	50.3	82.7	42.3	23.9	40.3	68.9	84.0	47.3	50.0	70.6	80.4
SynTable-Sim (Ours)	Amodal MRCNN	74.5	57.5	77.2	41.3	23.5	37.6	69.3	79.4	73.8	57.7	66.1	79.2
	ORCNN	74.2	58.2	77.1	44.7	24.3	33.8	72.9	82.2	72.0	58.3	67.7	79.1
	ASN	78.2	60.2	75.3	46.4	27.7	35.8	72.6	83.0	78.1	61.8	68.9	80.2
	UOAIS-Net	64.4	51.5	84.3	47.3	24.2	47.4	60.0	91.9	65.3	53.7	78.2	87.0

Table 6: A breakdown of the precision, recall, and F-measure of the amodal, invisible, and visible mask predictions by Amodal MRCNN, ORCNN, ASN, and UOAIS-Net on the **OSD-Amodal dataset** after training on the UOAIS-Sim and SynTable-Sim dataset. **P**: Precision, **R**: Recall, **F**: F-measure

Training Set	Method	Amodal Mask							Invisible Mask							Visible Mask						
		Overlap			Boundary			*F@.75*	Overlap			Boundary			*F@.75*	Overlap			Boundary			*F@.75*
		P	R	F	P	R	F		P	R	F	P	R	F		P	R	F	P	R	F	
UOAIS-Sim (Tabletop)	Amodal MRCNN	27.9	66.7	36.7	22.5	39.8	26.9	45.7	20.2	24.9	8.8	16.4	19.9	4.8	7.7	30.1	60.5	38.7	22.0	37.8	26.3	32.2
	ORCNN	26.3	71.1	36.3	19.8	42.4	25.4	47.0	41.5	22.7	12.2	33.9	17.9	6.7	9.0	21.4	63.4	30.5	16.6	38.2	21.8	29.6
	ASN	31.7	67.8	40.5	28.6	45.4	33.6	49.8	47.6	23.4	17.4	38.8	20.2	12.1	15.0	32.0	63.5	39.3	28.2	41.5	31.6	36.8
	UOAIS-Net	37.2	85.5	49.0	41.1	71.3	50.3	82.7	50.9	54.0	42.3	24.8	41.1	23.9	40.3	35.4	81.6	47.3	41.3	69.3	50.0	70.6
SynTable-Sim (Ours)	Amodal MRCNN	72.3	81.6	74.5	54.6	64.5	57.5	77.2	54.9	48.0	41.3	30.3	38.8	23.5	37.6	72.1	78.4	73.8	55.1	63.7	57.7	66.1
	ORCNN	73.7	80.8	74.2	55.6	64.5	58.2	77.1	61.0	47.1	44.7	31.1	38.6	24.3	33.8	69.8	79.1	72.0	55.7	64.3	58.3	67.7
	ASN	78.2	80.3	78.2	57.8	64.9	60.2	75.3	65.2	46.2	46.4	32.9	38.7	27.7	35.8	77.5	80.1	78.1	60.4	65.4	61.8	68.9
	UOAIS-Net	53.9	86.3	64.4	40.9	74.6	51.5	84.3	53.0	60.0	47.3	20.5	48.3	24.2	47.4	55.0	86.2	65.3	43.2	75.3	53.7	78.2

Table 7: The performance of Amodal MRCNN, ORCNN, ASN, and UOAIS-Net on the **SynTable-Sim validation dataset** after training on the UOAIS-Sim and SynTable-Sim datasets. UOAIS-Net is trained with RGB-D images. **OV**: Overlap F-measure, **BO**: Boundary F-measure, **F@.75**: Percentage of segmented objects with an Overlap F-measure greater than 0.75, $\mathbf{ACC_{OO}}$: Occlusion Order Accuracy

Training Set	Method	Amodal Mask			Invisible Mask			Occlusion		Visible Mask			ACC_{OO}
		OV	BO	*F@.75*	OV	BO	*F@.75*	$F_{\mathcal{O}}$	$ACC_{\mathcal{O}}$	OV	BO	*F@.75*	
UOAIS-Sim (Tabletop)	Amodal MRCNN	27.1	25.2	23.8	6.7	6.2	3.5	35.8	66.0	29.1	26.2	23.5	19.0
	ORCNN	30.9	29.0	28.1	12.5	11.4	8.0	39.9	68.4	31.8	30.2	27.3	23.1
	ASN	33.3	34.4	35.3	10.3	9.1	5.0	47.6	72.3	35.0	36.0	34.1	31.6
	UOAIS-Net	39.9	40.5	38.6	17.0	15.5	9.6	49.6	74.9	41.6	40.7	35.9	31.6
SynTable-Sim (Ours)	Amodal MRCNN	83.5	76.2	72.5	35.4	31.8	16.4	73.2	80.3	85.7	79.1	71.1	72.8
	ORCNN	83.4	76.0	72.2	34.4	29.3	15.3	67.2	73.7	85.3	78.9	70.9	73.0
	ASN	83.6	76.9	73.9	38.5	35.1	18.5	74.8	81.5	86.1	80.0	72.8	75.8
	UOAIS-Net	83.7	77.5	75.1	40.3	36.7	20.2	75.5	82.0	86.2	80.1	73.3	77.4



Occluder \ Occludee	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	0	0	0	0	0	0	0	0	0	0	0	1	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	1	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	1	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	1	0	0	1	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fig. 8: A visualisation of annotations for a cluttered tabletop image generated by SynTable

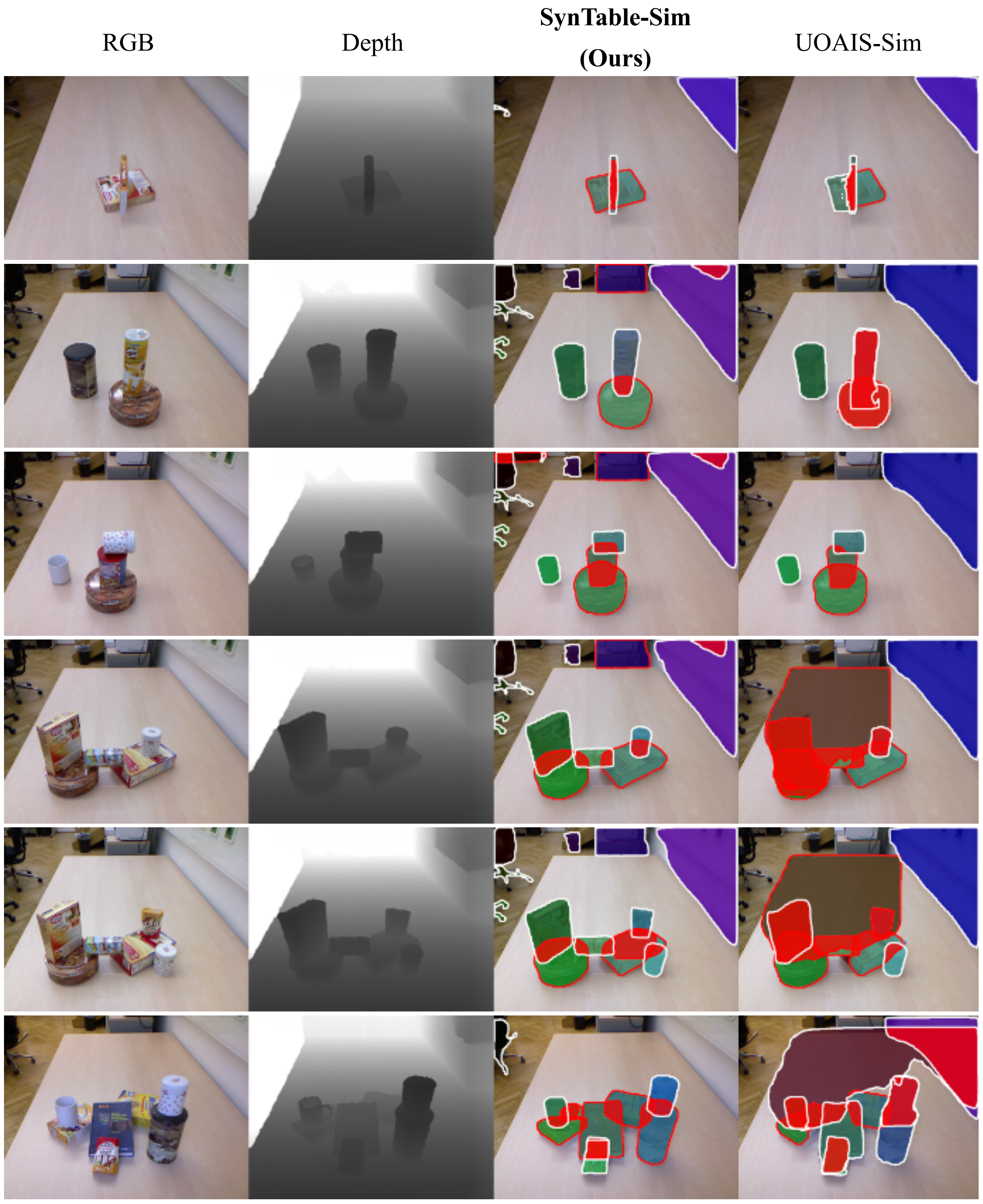


Fig. 9: Comparison of the inference results on the OSD-Amodal dataset. **SynTable-Sim (Ours):** the performance of UOAIS-Net on the OSD-Amodal dataset after training on the SynTable-Sim dataset. **UOAIS-Sim**: the performance of UOAIS-Net on the OSD-Amodal dataset after training on the UOAIS-Sim tabletop dataset.