

# SynTable: A Synthetic Data Generation Pipeline for Unseen Object Amodal Instance Segmentation of Cluttered Tabletop Scenes

## Supplementary Material

### 7. Overview

This supplementary material offers dataset visualization, qualitative results, and additional technical details to support the main paper. Section 8 provides additional information about each step in our dataset generation process. Section 9 provides a comprehensive elaboration of the evaluation metrics employed. Section 10 illustrates how occlusion order accuracy is calculated and the validity of the metric. Furthermore, Section 11 delineates the process of generating an occlusion order directed acyclic graph from the occlusion order adjacency matrix to classify objects in three distinct order layers. Lastly, Section 12 showcases some qualitative inference results of UOAIS-Net on the OSD-Amodal dataset.

#### 7.1. 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. The video can be found at <https://www.youtube.com/watch?v=zHM8H58Kn3E>.

#### 7.2. Management of the SynTable-Sim Dataset

The source code for our work is available at <https://github.com/ngzhili/SynTable>. 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](https://doi.org/10.5281/zenodo.10565517). The dataset can be accessed at <https://doi.org/10.5281/zenodo.10565517>.

### 8. Additional Details About the Dataset Generation Process

#### 8.1. 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

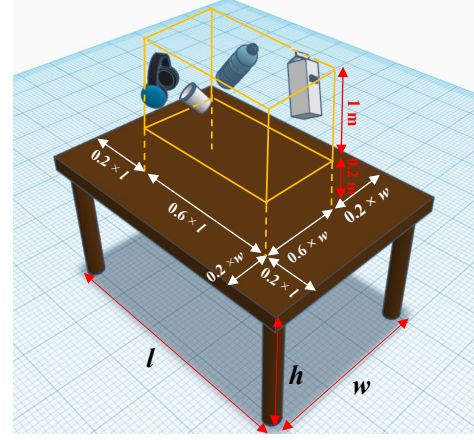


Figure 4. Initialization of objects with randomized coordinates and rotations. The initial position of the objects in the scene is randomized but constrained to be within the dimensions of the 3D orange box. The orange box is 0.2 m above the tabletop. The roll, pitch, and yaw of each object are also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

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.

#### 8.2. 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), halting any further movement of the objects. 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.

#### 8.3. Sampling of Camera Viewpoints

To capture annotations for each scene from multiple viewpoints, we enhance the approach by Gilles *et al.* [18]—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

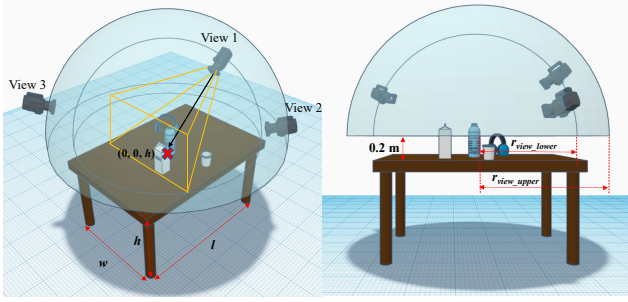


Figure 5. Sampling of camera viewpoints within concentric hemispheres (shown in blue). The two concentric hemispheres’ origins are centered at the tabletop surface’s center coordinate with an offset of 0.2 m in the positive  $z$  direction in the world frame. This allows the camera viewpoints to minimally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

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.

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

$$r_{view\_upper} = 1.7 \times r_{view\_lower} \quad (3)$$

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.

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

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

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

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  $(0, 0, h)$ .

#### 8.4. 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 method discussed in Section 8.3. In contrast to the approach by Back *et al.* [2], we use

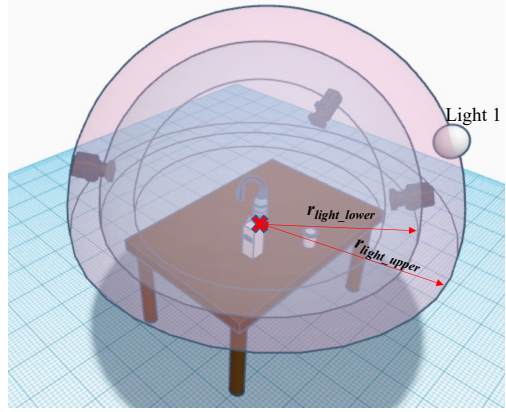


Figure 6. Sampling of lighting within concentric hemispheres (shown in pink). Each spherical light source lies within the constraints of two concentric hemispheres of arbitrary radius between  $r_{light\_lower}$  to  $r_{light\_upper}$ . Note that the radii constraints for the spherical light source concentric hemispheres are larger than those for the camera viewpoints’ and are customizable by the user.

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 to 20,000 lx, and the default light intensity of ceiling lights in the scene is also sampled uniformly between 100 lx to 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:

$$r_{light\_lower} = r_{view\_upper} + 0.1m \quad (7)$$

$$r_{light\_upper} = r_{light\_lower} + 1m \quad (8)$$

#### 8.5. Saving of Ground Truth Annotations

We saved the RGB and depth images as PNG images. The OOAM of the objects in each image is saved as a NumPy file. The amodal, visible, and occlusion masks are saved as Run-length Encoding (RLE) in COCO JSON format to optimize disk space used by the generated datasets. We also recorded each object’s visible bounding box, image ID, and object name in the generated COCO JSON file.

#### 9. Details about Evaluation Metrics

In this paper, we employ the precision/recall/F-measure (P/R/F) metrics, as defined in [5, 19, 29]. This metric favors methods that accurately segment the desired objects while

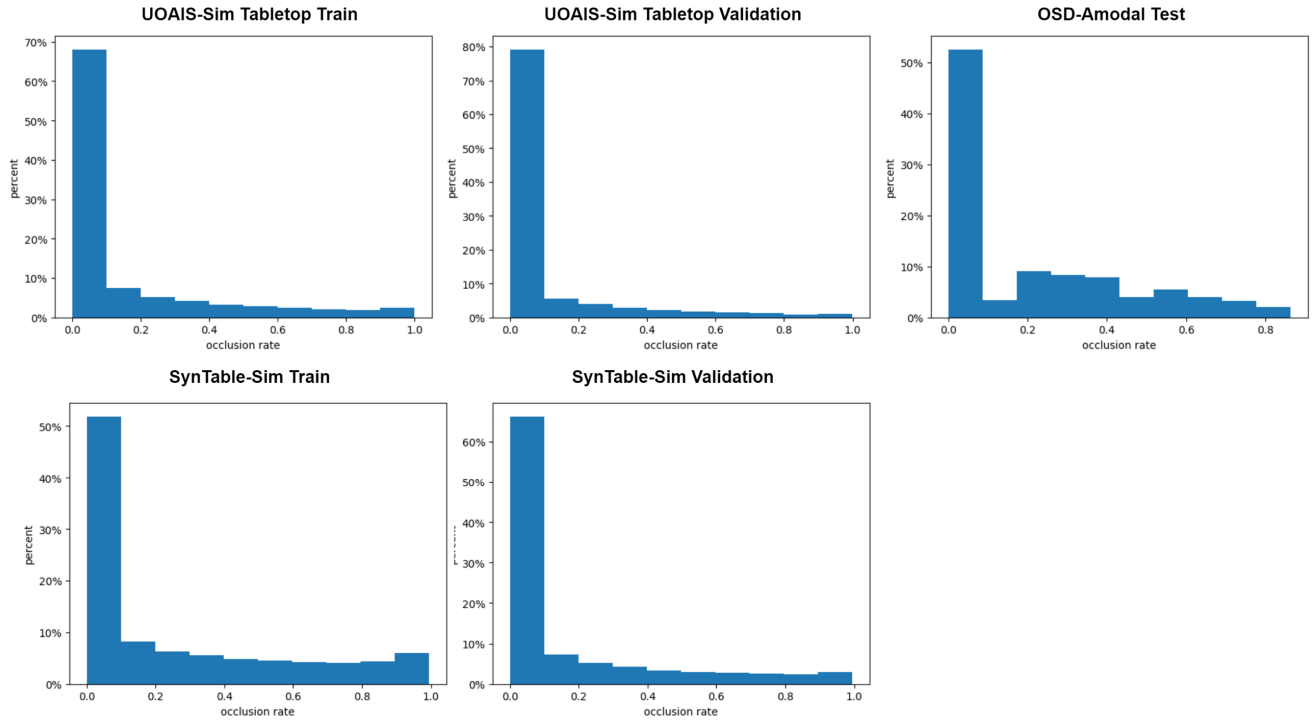


Figure 7. Histogram of occlusion rate for UOAIS-Sim tabletop, SynTable-Sim and OSD-Amodal datasets

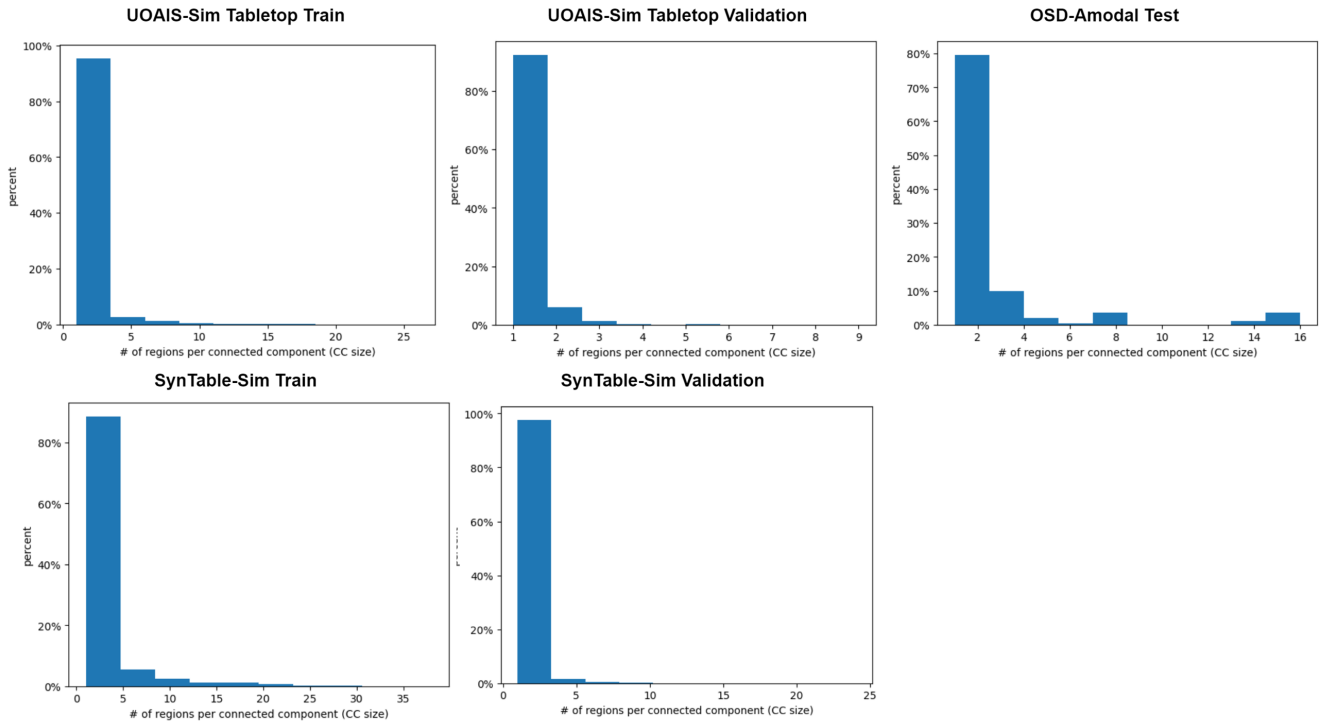


Figure 8. Histogram for number of regions per connected component (connected component size) for UOAIS-Sim tabletop, SynTable-Sim and OSD-Amodal datasets

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:

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

$$F = \frac{2PR}{P+R} \quad (10)$$

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.* [29] 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:

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

$$F = \frac{2PR}{P+R} \quad (12)$$

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 [20] and the Overlap P/R/F as defined in [5].

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 [5] and stated in Equation (14).

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

$$F_{ij} = \frac{2P_{ij}R_{ij}}{P_{ij} + R_{ij}} \quad (14)$$

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.

## 10. 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 9. The predicted masks will then be re-indexed to match the ids 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.

Figure 9 also illustrates the calculation of occlusion order accuracy in an image  $v$ . The similarity matrix (denoted as *similarityMatrix* in Figure 9) 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 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 9).

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 9). Thus, we subtract the number of elements along the

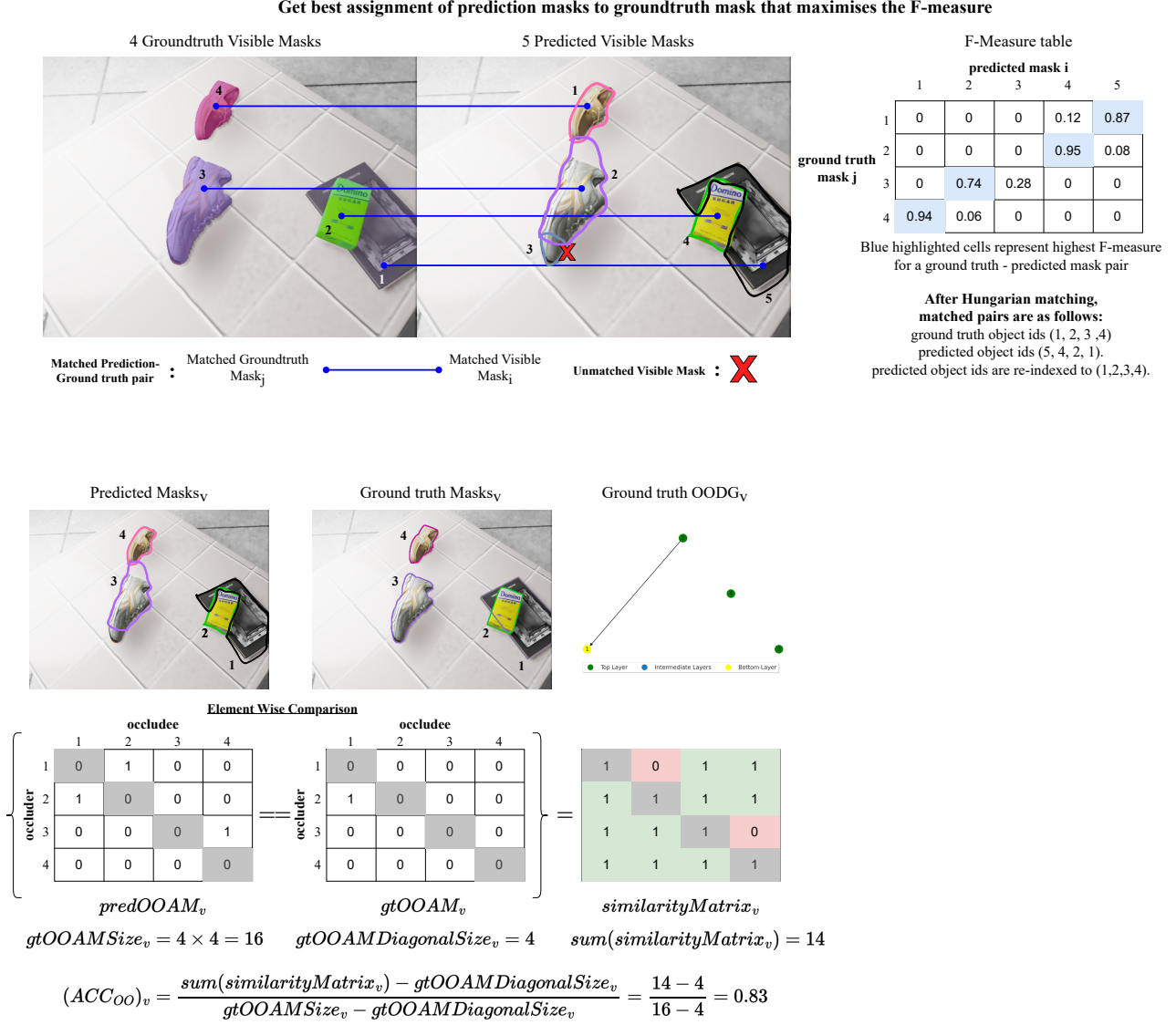


Figure 9. Hungarian Matching and calculating Occlusion Order Accuracy of image v

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<sub>oo</sub>*. This demonstrates the appropriate assign-

ment of penalties by *ACC<sub>oo</sub>* to different error types for measuring object occlusion ordering in a scene.

## 11. Occlusion Order Directed Acyclic Graph (OODAG)

After obtaining the Occlusion Order Adjacency Matrix (OOAM), we can generate the occlusion order directed graph 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 10 shows

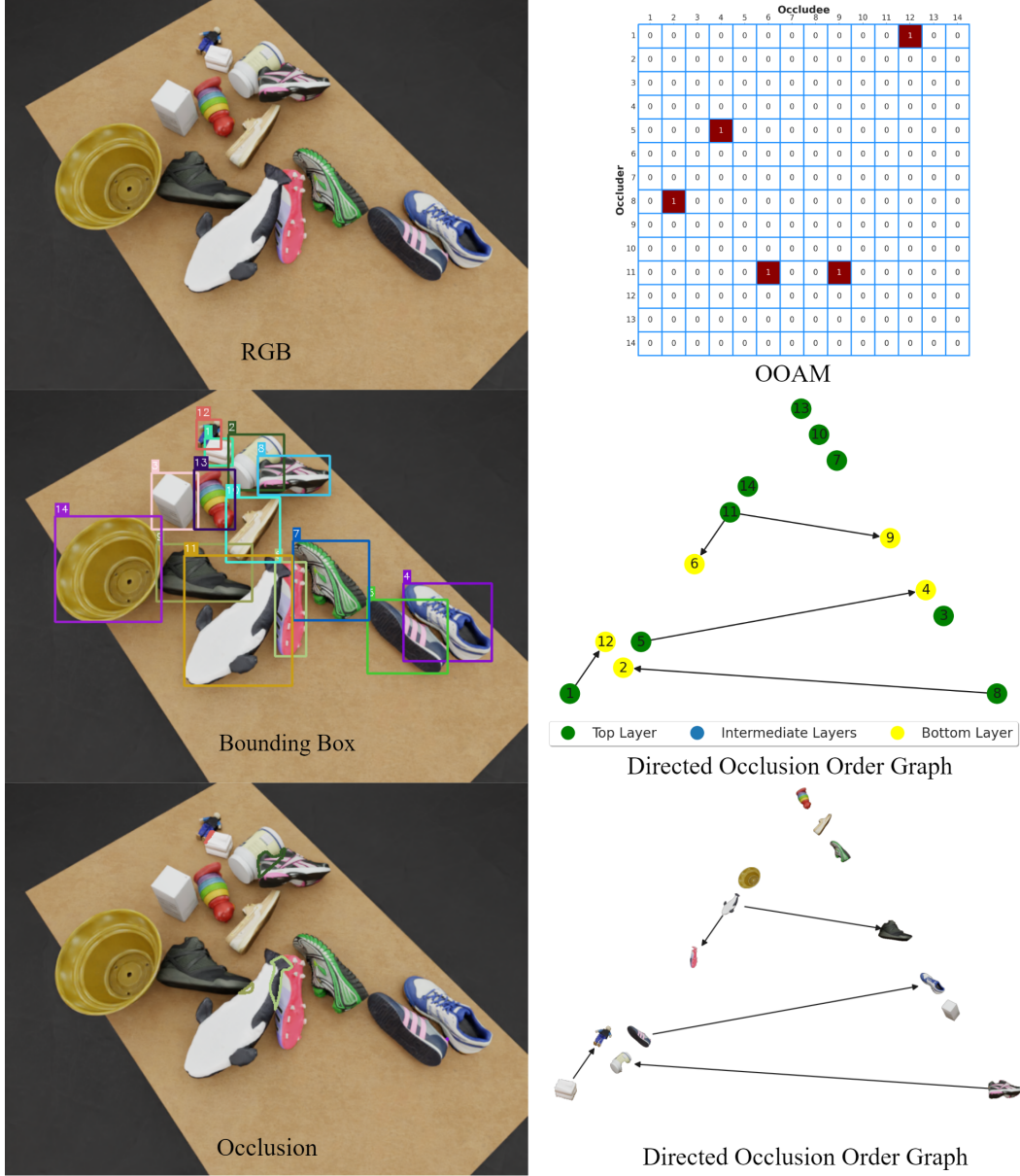


Figure 10. A visualisation of annotations for a cluttered tabletop image generated by SynTable

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.

In the generated Occlusion Order graph, we further classify objects in three different order layers - Top, Intermedi-

ate, and Bottom. Objects at the top layer represent objects that are not occluded by any other object. 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.

## 12. Qualitative Inference Results of UOAIS-Net on the OSD-Amodal Dataset

After training the UOAIS-Net model [2] on both SynTable-Sim and UOAIS-Sim (tabletop) datasets [2], we present some of our qualitative results in Figure 11. 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 11. 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.

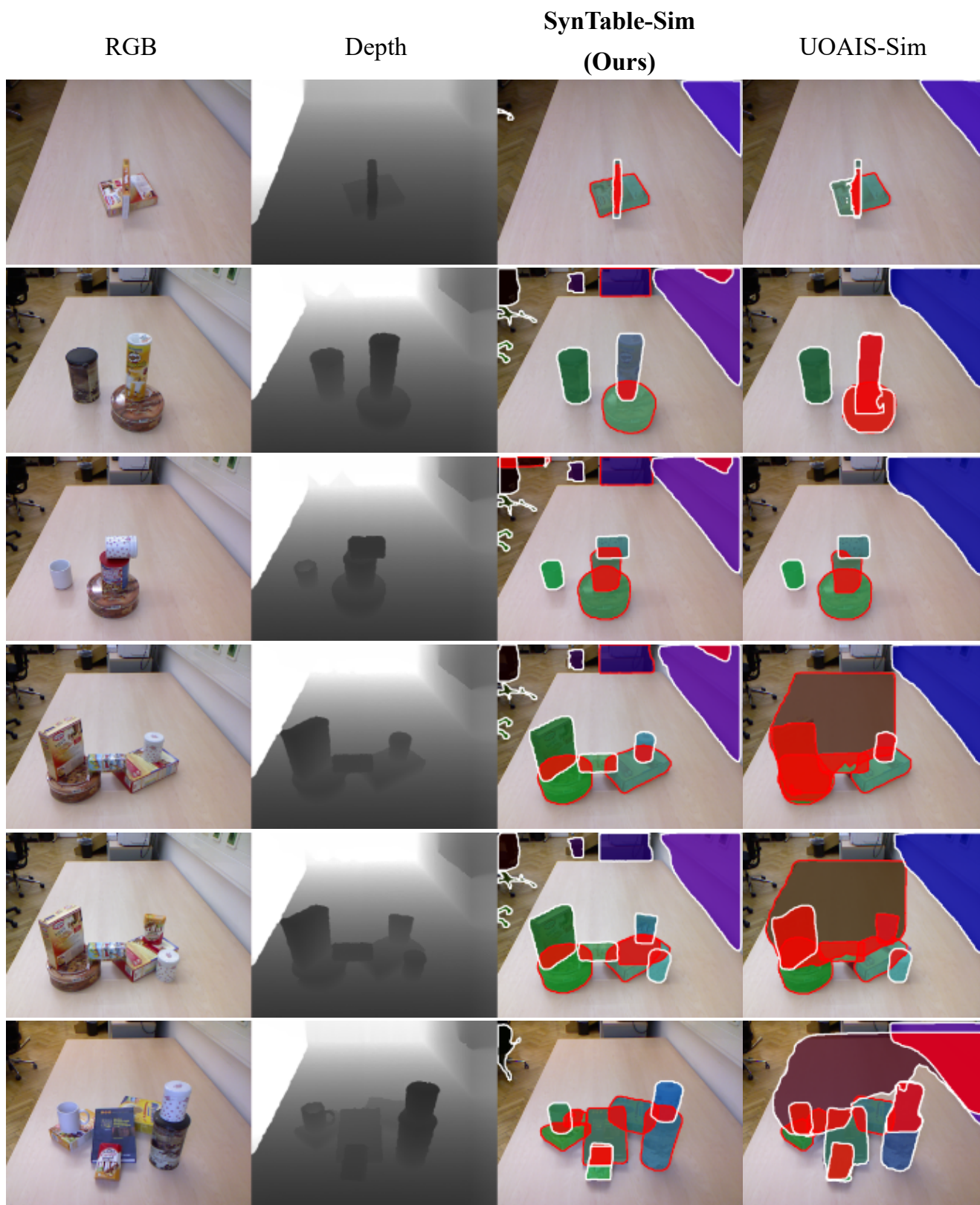


Figure 11. Comparison of the inference results on the OSD-Amodal dataset. **SynTable-Sim (Ours)**: the performance of UOAIS-Net on the OSD-Amodal dataset after training on the SynTable-Sim dataset. **UOAIS-Sim**: the performance of UOAIS-Net on the OSD-Amodal dataset after training on the UOAIS-Sim tabletop dataset.