Appendix

A. Extended BEHAVIOR-1K Assets

Fig. 1 shows more examples of Extended BEHAVIOR-1K Assets (main paper Sec.3.1): (a-f) examples for different main categories. (g) examples of improved collision mesh quality. (h) examples of articulation objects. (i) examples of different light sources, (j) examples of fillable volumes for containers.

B. Customizable Dataset Generator

We prepare a **video** in the supplementary material to show an example visualization of the Customizable Dataset Generator: specifically, we show the scene instance augmentation by scene object (furniture) randomization and inserting additional everyday objects.

C. Experiments

C.1. Parametric Model Evaluation

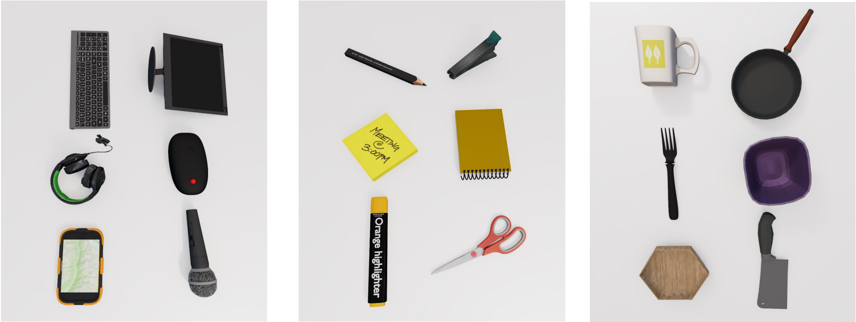
Details of Dataset Generation Process. We synthesized the evaluation videos for each axis (Object articulation, Lighting, Visibility, Zoom, Pitch) according to the following pipeline.

As shown in the main paper Sec. 4.1, each video includes a target object with changes focused on a single parameter under examination. First, we sample one of the scene instances and randomly choose a target object in the scene. For the object articulation axis, we only sample objects with movable parts, such as cabinets, microwaves, refrigerators, etc. Next, we sample a random camera angle and distance with the target object placed in the center. Then, for all except pitch, we keep this camera pose and perform an axis-specific manipulation to generate a video with the desired variation:

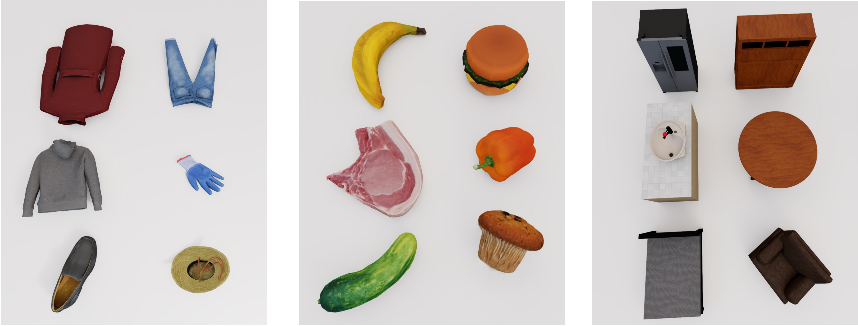
- **Object articulation**: We linearly interpolate the joint angle from being closed to fully open, utilizing the joint maximum range annotations provided in BVS assets. We record the image with the joint in each intermediate state. For objects with multiple movable parts, e.g., a cabinet with three drawers, we randomly sample a subset of joints to manipulate and keep the rest closed.
- **Lighting**: We linearly increase the intensity of all indoor light sources in the scene simultaneously.
- **Visibility**: There are three key components in the visibility (occlusion) setting: camera, target object, and occluding object. We first set the camera centering on the target object, then we place an occluding object (relatively large object, e.g., cabinet) in the line between the camera and the target object, fully occluding the target in view. Then, we fix the distance between the camera and the target object and move the camera around the target object until the target is fully visible. The visibility score (number of visible pixels/number of total pixels) of each frame is calculated by rendering the video again and removing the occluding cabinet. Although the object orientation in camera view might slightly change since the camera is not static, we implemented the following practices to eliminate the effect of this factor. First, we set the camera relatively far from the target but occluding objects close to the camera, allowing minimal camera pose change needed to capture the "fully occluded to fully visible" process. In addition, the initial object pose is randomized, so when we average evaluation performance, the effect of this factor shall largely cancel out.
- **Zoom**: With the camera pose fixed, we change its focal length to model the zooming effect. We strategically changed the focal length such that the resulting video illustrates an approximately linear zooming behavior. We always make the target object at the center of the view, and it remains mostly unaffected by distortion, even under extreme focal length.
- **Pitch**: We linearly change the camera pitch angle while keeping the original camera distance as well as the yaw angle unmodified.

After each video was collected, we filtered out the videos where the target object was not properly visible. The detailed statistics for each axis is shown in main Table 2.

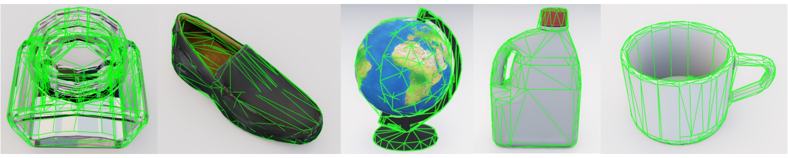
Metric Details. Each generated video contains exactly one target object (main paper Figure 3 magenta). We use different open-vocabulary object detection and segmentation models to detect or segment the target object. These models act as indicators of performance in challenging environments, such as those with limited lighting or long-distance zoom. Therefore, we compute the Average Precision (AP) metric using the target object as the sole ground



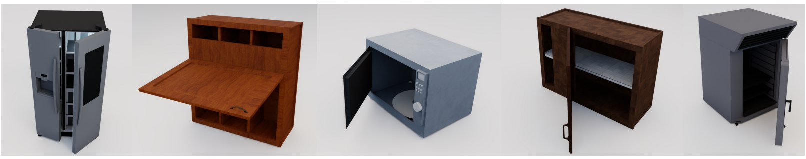
(a) Electronics (b) Office supplies (c) Kitchenware



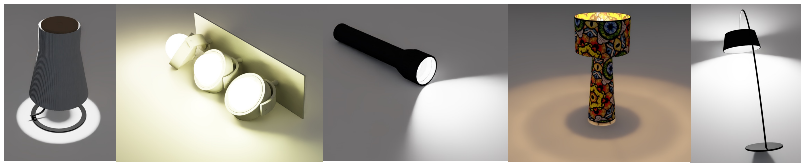
(d) Wearable (e) Food (f) furniture



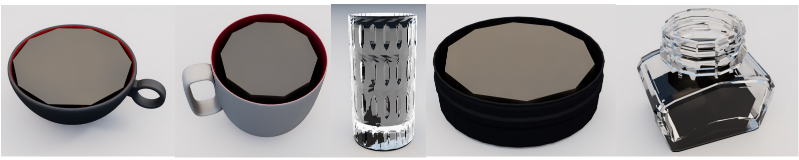
(g) Collision mesh



(h) Articulation objects



(i) Light source



(j) Fillable volumes for containers

Figure 1. More examples of Extended BEHAVIOR-1K Assets: (a-f) examples for different main categories. (g) examples of improved collision mesh quality. (h) examples of articulation objects. (i) examples of different light sources, (j) examples of fillable volumes for containers.

truth, considering only predictions that classify the target object class. However, it is plausible to encounter scenarios where multiple objects of the same category as the target object exist. For instance, in a video, there might be several chairs, and the target object is one of those chairs. Models might have correctly detected all chairs, but since only one is the ground truth, all rest will be marked as incorrect. To counter such an undesired situation, we employ a simple heuristic to filter predictions for the non-target object: For non-target objects sharing the same category, we calculate the IoU with each prediction and exclude those with an IoU exceeding a predefined threshold of 0.3. This means treating these predictions as valid for non-target objects rather than false positives. This threshold is chosen empirically based on a few selected cases where objects of the same category are densely packed together. We believe this choice generalizes well to less crowded scenes and ensures the reliability of our evaluation process.

Failure Case Analysis on Five Evaluation Parameters. In Fig. 2, we present failure case examples of Grounding DINO [7] across five evaluation axes: Object articulation, Lighting, Visibility, Zoom, and Pitch. Each row in our presentation represents one axis and comprises four example groups. In each group, there are two images: the left image illustrates the ground truth, highlighting the target object in magenta, while the right image shows the Grounding DINO's prediction for the target object, as indicated at the top of the first image in each group. The example groups are arranged such that, from left to right, the intensity along the respective axis increases (e.g., progressing from zoomed in to zoomed out), the intensity value (0-1) is shown on top of each prediction. We find and highlight some interesting findings for each parametric evaluation in Fig. 2 and detailed below. For the full axis, we prepare a **video** in supplementary to show qualitative examples about running different detection models on the same video.

- **Articulation.** The model may have limited exposure to open states for articulation objects, which makes it less likely to predict a microwave with its door open correctly.
- **Lighting.** When the environment is dark, the model performance is negatively affected. However, when the lighting exceeds a certain threshold, in this case 0.5, the model becomes robust to increasing illumination.
- **Visibility.** The model's detection performance suffers when most of the target object is occluded. Surprisingly, a correct prediction can be made with only half of the object visible.
- **Zoom.** When the model is zoomed out, nearby objects become distorted, leading the model to identify irrelevant objects as the target mistakenly. This suggests that the model's recognition may rely partly on contours rather than solely on semantic information.
- **Pitch.** We find that, generally, the model can achieve better performance in a look-down angle compared to a look-up angle.

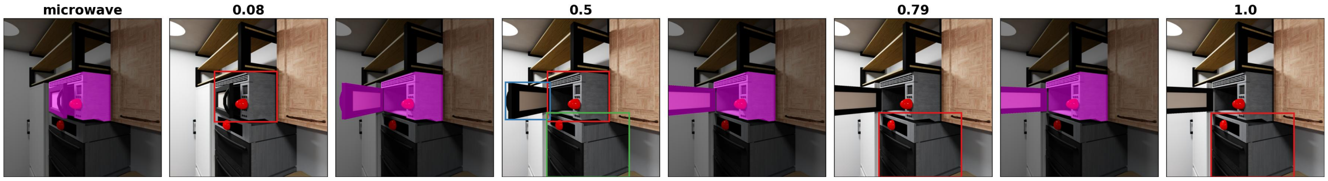
Segmentation Results on Five Axes. Figure 3 of the main paper shows the performance of open-vocabulary detection models on five axes. In Fig. 3, we show the performance of open-vocabulary segmentation models instead. The average performance for each axes corresponds to one angle in the radar plot (main Figure 4). Observations in main section 3.1 also apply to segmentation tasks.

Real Experiment Setup and Results In order to evaluate the sim2real transfer capability of parametric evaluation results, we curated a set of real images to perform the same evaluation. In total, we manually collected 430 images of 15 to 22 objects from various categories. For each axis, we took photo of each target object with 5 levels of the corresponding distribution shift that matches the intensity level 0, 0.25, 0.5, 0.75, 1 in the synthetic data. For example, when collecting data for a microwave object for the articulation axis, we collect 5 images of the microwave being fully closed, 25% open, half open, 75% open, and fully open. An example object from the real dataset (and the comparison to the most similar counterpart in simulation) is shown in . For the zoom axis, due to the limited focal length range of our real cameras, we only covered intensity levels 0 to 0.3.

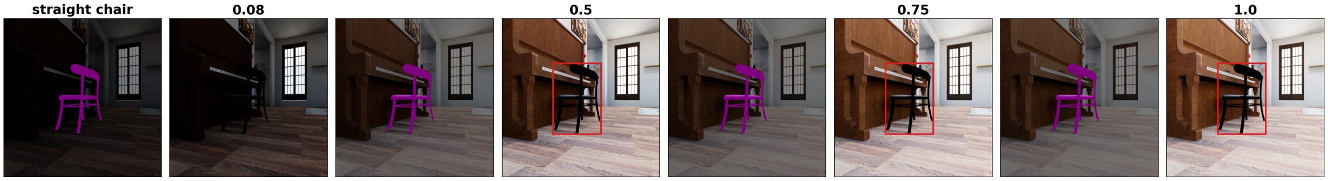
We evaluated the SOTA detection methods with manually labeled bounding boxes. Fig. 4 shows that under different types of distribution shift, the performance of the SOTA methods varies on real data just as it varies in simulation.

C.2. Holistic Scene Understanding (main paper Sec. 4.2)

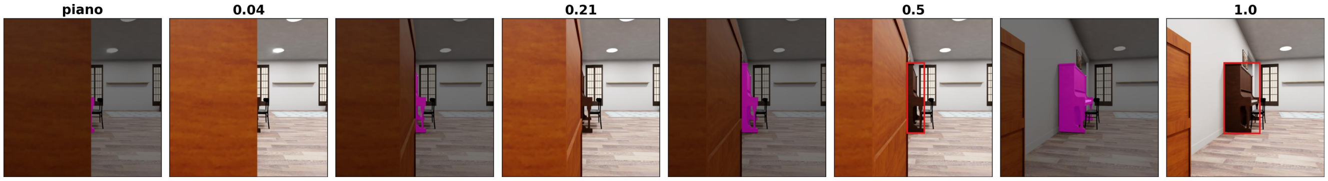
Details of Generation Process. To generate a scene traversal video, we adhere to a standard process. Initially, we sample a scene instance and subsequently define a camera trajectory using the BVS toolkit. Following this, we render the traversal video, incorporating all required labels. This section will detail the specifics of the trajectory sampling procedure. In general, we want the sampled video to provide rich information (good coverage) about the scene, which can be broken down into two aspects. Firstly, we aim for the camera to physically cover the room. That means the sampled camera positions should enable visiting most open spaces in the scene, rather than just focusing on the largest open space. This guides our design for camera position sampling. Second, we want the actual video to capture as many objects in the scene as possible while still being realistic (i.e., facing the direction of movement while moving). This guides our design for camera orientation sampling. Next, we will establish the detailed steps. We will open-source all codes and generate a video dataset.



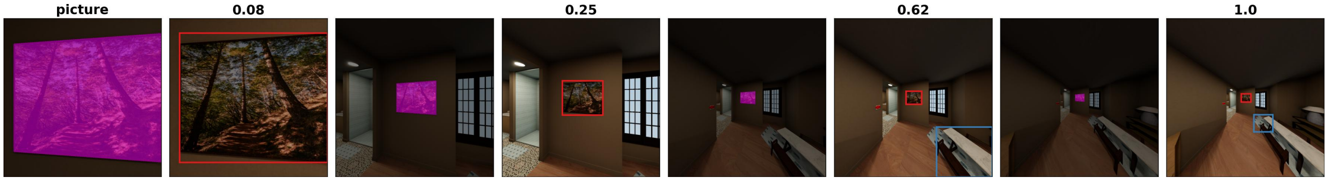
(a) **Articulation.** The model may have limited exposure to open states for articulation objects, which makes it less likely to recognize a microwave with its door open correctly.



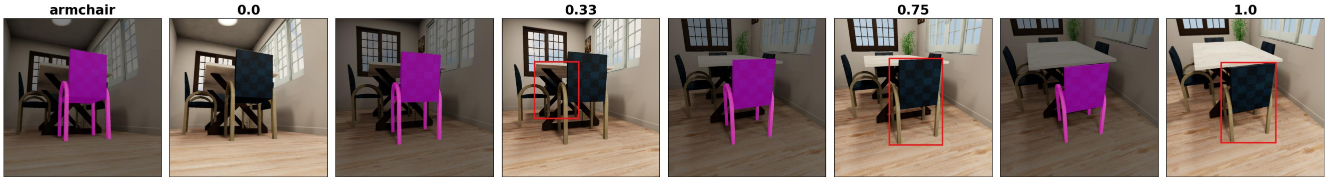
(b) **Lighting.** When the environment is dark, the model performance is negatively affected. However, when the lighting exceeds a certain threshold, in this case 0.5, the model becomes robust to increasing illumination.



(c) **Visibility.** The model's detection performance suffers when most of the target object is occluded. Surprisingly, a correct prediction can be made with only half of the object visible.



(d) **Zoom.** When the model is zoomed out, nearby objects become distorted, leading the model to identify irrelevant objects as the target mistakenly. This suggests that the model's recognition may rely partly on contours rather than solely on semantic information.



(e) **Pitch.** We find that, generally, the model can achieve better performance in a look-down angle compared to a look-up angle.

Figure 2. Error analysis for Grounding DINO [7]. Similar trends are also observed in other detection models. Each row in our presentation represents one axis and comprises four example groups. In each group, there are two images: the left image illustrates the ground truth, highlighting the target object in magenta, while the right image shows the Grounding DINO's predictions (colored differently) for the target object, as indicated at the top of the first image in each group. The example groups are arranged such that, from left to right, the intensity along the respective axis increases (e.g., progressing from zoomed in to zoomed out), and the intensity value (0-1) is shown on top of each prediction.

- To sample camera positions within the trajectory, a basic approach might involve randomly selecting traversable points within the scene. However, this brings the issue that points in larger rooms are more likely to be selected compared to those in smaller rooms. Our objective is to achieve a more uniform coverage across the entire scene, avoiding overconcentration in larger open areas. Thus, we used the **farthest point sampling** method to sample a set of key points that sparsely span the scene. Our focus is on ensuring the trajectory covers the main open spaces, without the necessity of navigating narrow spaces such as the gap between a cabinet and a wall. To achieve this, we perform the sampling on an **eroded** version of the traversal map. This technique effectively highlights the larger open areas in a scene while eliminating smaller gaps and corners.
- Now we have a set of candidate key points sampled in the scene, but a view from many of them may provide simi-

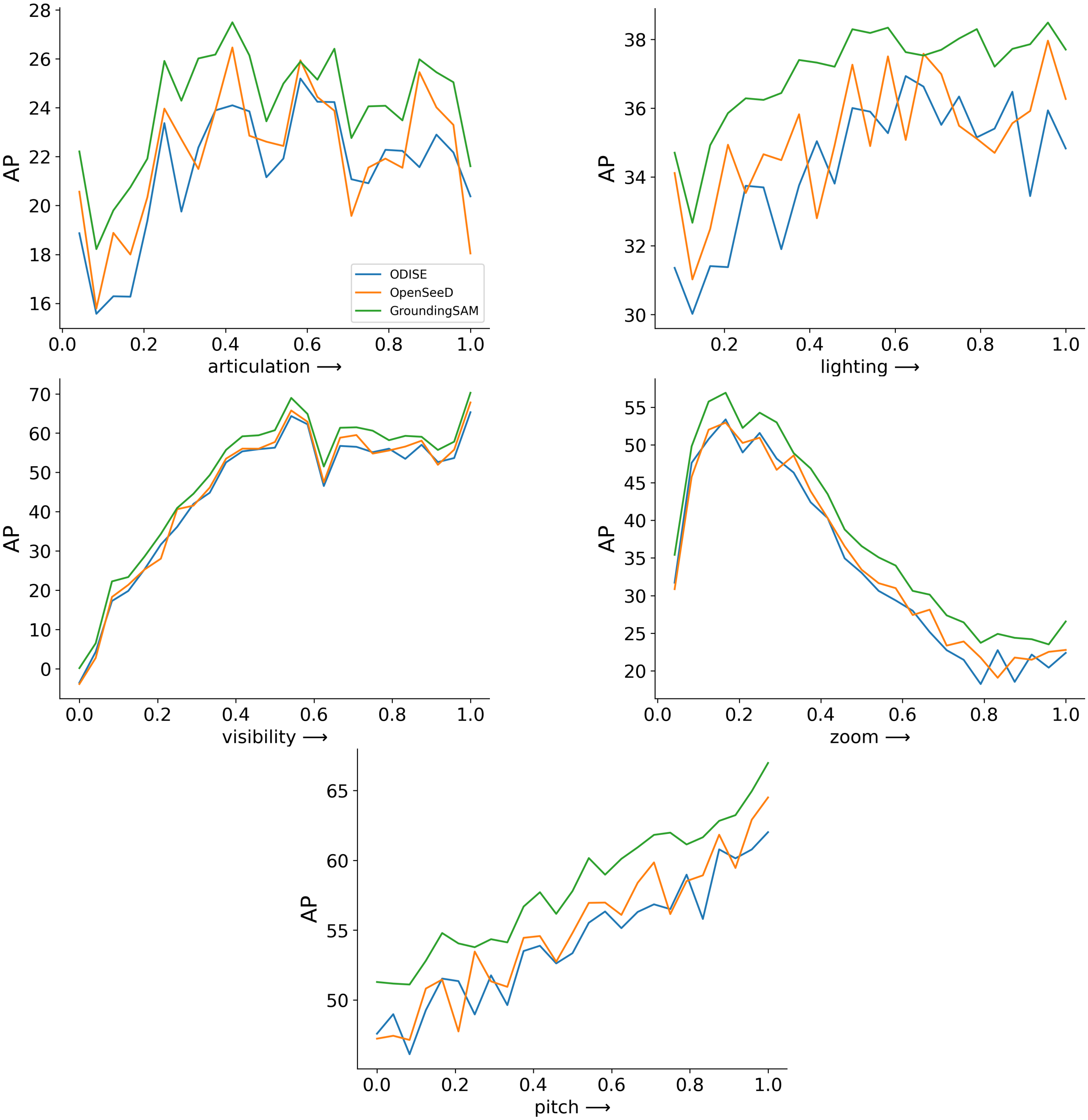


Figure 3. Similar to Figure 3 in the main (Section 4.1), we plot the Segmentation model results for each of the five axes. AP is calculated using target object only.

lar information about the scene (for example, two nearby points in the same room). We don't need to visit both of them in the same trajectory. To ensure efficiency and avoid redundancy, we need to select a subset of these key points while still preserving a comprehensive view of the scene (such as not excluding all points from a specific room). Our selection process begins by assessing the unique information each key point provides, specifically the objects visible from that point. We place a virtual camera at each key point and rotate it 360 degrees, recording the angle at which the maximum number of objects in the scene are visible. We note both the visible objects and

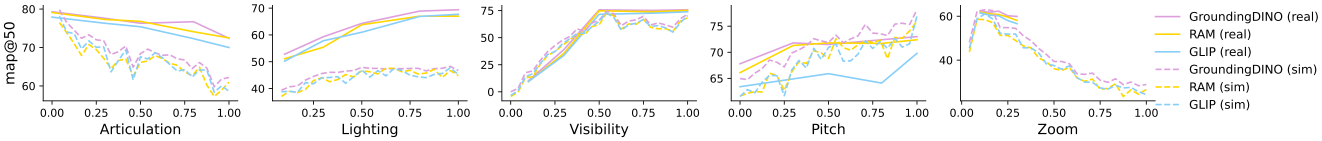


Figure 4. We also observe a comparable pattern in the parametric evaluation conducted on real data as observed in synthetic data (main Figure 3).

their corresponding angles for each key point. Next, we randomize the order of the key points and go over each point sequentially to select the points that offer additional information. If a key point reveals an object not visible from **all** previously selected points, we retain it. Otherwise, we discard it. This method results in a smaller, more efficient subset of key points — referred to as waypoints in the subsequent step — which still allows a comprehensive observation of most objects in the scene.

- Once we have determined the set of waypoints, our next step is to devise a trajectory that connects them in the shortest possible path. To achieve this, we frame the task as a traveling salesman problem, treating the waypoints as nodes to be visited on the scene traversal graph. In this step, to ensure the camera maintains a safe distance from walls and furniture, we slightly erode the scene traversal graph. This adjustment prevents the camera from getting too close to these obstacles, ensuring smoother navigation through the scene.
- After establishing the sequence of positions in the previous step, we focus on determining the camera's orientation at each position. Our goal is to mimic the behavior of a real agent exploring the scene. Therefore, while in motion, the camera faces the direction it is moving towards. Additionally, at each waypoint, the camera 'makes a stop' and turns to an optimal angle. This angle, predetermined in an earlier step, allows the camera to capture the most objects from that viewpoint. This approach serves two purposes: firstly, to ensure that the trajectory includes keyframes or angles for optimal object visibility, and secondly, to simulate the natural behavior of an agent pausing to observe the surroundings.

C.3. Object States and Relations Prediction (main paper Sec. 4.3)

Details of Generation Process. For binary object relationship prediction, we synthesized 12.5k images, each annotated with one or more of the following five labels: `open`, `close`, `ontop`, `inside`, `under`. For instance, an image might depict a toy inside an open cabinet, thus making both "inside" and "open" labels applicable.

The image sampling process is as follows: Firstly, we select a scene and a piece of furniture to serve as the primary object in the relation (e.g., a table for placing items on top). Subsequently, we determine a plausible relationship related to this base object, with annotations provided via BVS. For example, an item might be positioned `ontop` or `under` a table, but not `inside` it. Following this, we select a random object to place in the scene. This object is then integrated into the scene, employing the physical state sampling function from BVS to ensure its placement aligns with the predetermined relationship. For instance, we might sample a cupcake and place it at a random location on top of the table. Lastly, we sample a random camera pose, ensuring the placed object is centered in the frame. We then filter out any instances where the objects of interest are not adequately visible. This procedure is repeated iteratively to compile our final dataset.

For unary object state prediction, we generated 500 images that either consist of a `filled` or empty (not filled) container, similarly 500 images for `folded`. The sampling process for unary states is simpler – we randomly sample a scene, then place a random container/cloth object in the scene, by 50% chance sample a filled or folded state for the target object, then also sample a random camera pose with the target object in the center.

Details of Architecture Design and Hyperparameters. Our model architecture is adapted from [5, 8]. Given an image input with two (or one) bounding boxes, the model predicts the binary spatial (or unary) relationship between objects corresponding to boxes (Fig. 5). First, the model utilizes a Segment Anything image encoder [6] to extract hidden features. Subsequently, RoIAlign [4] is applied to the extracted features using the two (or one) bounding boxes. In the binary case, where the objective is to predict the spatial relationship between the two objects, RoIAlign effectively captures spatial information from the representation

Additional features are incorporated into the aligned representations to enhance semantic information. Unlike [8], which relies on word2vec vectors [9] and category names for the objects (which may not always be readily available in the world), we opt to use the Segment Anything extracted feature, from the cropped image encompassing the union of the two bounding boxes, as the additional feature. This approach preserves both spatial and semantic information.

The concatenated features are then fed into a trainable CNN to predict seven-way logits. To prevent overfitting, we

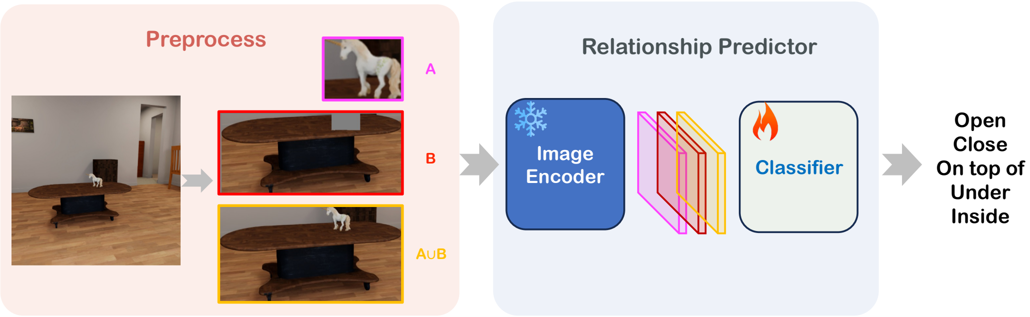


Figure 5. Relationship prediction model architecture used in Sec 3.3.

freeze the Segment Anything image encoder, ensuring that the only learnable parameters are those of the randomly initialized CNN. Under a 0.3 learning rate with linear scheduling, the model is trained on 13.5k synthetic images only but can achieve strong performance in the real test set (Table 4 in the main paper).

Lastly, we discuss the details of **zero-shot CLIP** [10] baseline, which is used to mimic scenarios where synthetic datasets are not accessible. In this scenario, we have to rely on CLIP's zero-shot capacity. Specifically, akin to our architecture, the image is cropped to maximize the semantic information. Then the image embedding of the cropped images is compared with the label text embeddings from all verbalized prompts. A verbalized prompt can take the form of `<A> on top of <B>` where `<A>` and `<B>` are the placeholder for the actual object category name. Empirically we find including the category name in the prompt outperforms using predicate only (e.g., only `on top of`). We emphasize that having prior knowledge of category names in advance renders this task more straightforward and less fair to our approach where we have no assumption on access to any category name.

Shown in main Tables 4 and 5, BVS is capable of generating high-quality synthetic training data by demand [2, 3]. Models trained on such synthetic training data are able to capture the essence of predicate prediction and bridge the sim-to-real gap.

Folded and Filled Prediction BVS also supports nuanced unary object predicate such as `folded` and `filled`. Models training on synthesized photo-realistic images can transfer well to real images. We train two linear probes on top of the EVA02 [1] encoded representation to predict `folded` and `filled`, respectively. We manually collect 50 real test images for each of the two predicates and observe that linear probes can achieve 86% and 93% real test accuracy for `folded` and `filled`, respectively.

D. Demo Videos

We have provided `video-demo.zip`, which consists of demo videos of our generated videos and model prediction. Specifically, `BVS-highlight.mp4` shows all visualization content. `customizable-dataset-generator.mp4` shows scene instance augmentation. `predicate_prediction.mp4` consists of predicate prediction model prediction on our collected real videos (main Sec. 4.3). `compare-detection-{axis}.mp4` consists of three detection model predictions on five parametric evaluation axes (main Sec. 4.1).

References

[1] Yuxin Fang, Quan Sun, Xinggang Wang, Tiejun Huang, Xinlong Wang, and Yue Cao. Eva-02: A visual representation for neon genesis. *arXiv preprint arXiv:2303.11331*, 2023. 7

[2] Yunhao Ge, Harkirat Behl, Jiashu Xu, Suriya Gunasekar, Neel Joshi, Yale Song, Xin Wang, Laurent Itti, and Vibhav Vineet. Neural-sim: Learning to generate training data with nerf. In *European Conference on Computer Vision*, pages 477–493. Springer, 2022. 7

[3] Yunhao Ge, Jiashu Xu, Brian Nlong Zhao, Neel Joshi, Laurent Itti, and Vibhav Vineet. Beyond generation: Harnessing text to image models for object detection and segmentation. *arXiv preprint arXiv:2309.05956*, 2023. 7

[4] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015. 6

[5] Sho Inayoshi, Keita Otani, Antonio Tejero-de Pablos, and Tatsuya Harada. Bounding-box channels for visual relationship detection. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part V 16*, pages 682–697. Springer, 2020. 6

[6] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. *arXiv preprint arXiv:2304.02643*, 2023. 6

[7] Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Chunyuan Li, Jianwei Yang, Hang Su, Jun

Zhu, et al. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. *arXiv preprint arXiv:2303.05499*, 2023. 3, 4

[8] Toki Migimatsu and Jeannette Bohg. Grounding predicates through actions. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 3498–3504, 2022. 6

[9] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013. 6

[10] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021. 7