A. Dataset Details

A.1. Data Generation

In this section, we provide additional details about the data generation pipeline.

(A) Query Image. As mentioned in Sec. 3.1, the complete list of text queries to retrieve raw images from the LAION dataset includes `living room`, `bedroom`, and `kitchen`.

(B) Find Objects of Interest. For each image, we use Detic [19] and SAM [64] to find segmentation masks of the 9 object categories of interest. First, we prompt Detic to find all instances of each of these 9 categories within an image. If no object is detected, the image is discarded. Next, for each detected object instance, we compute the center point $(center_x, center_y)$ of its bounding box $[x_1, y_1, x_2, y_2]$ and use this center point as a prompt for SAM [64] to predict a segmentation mask. Among the 3 masks predicted by SAM [64], we choose the one with the highest confidence for downstream inpainting.

In Fig. 8 we visualize additional qualitative examples from the SP training dataset generated using our automatic data generation pipeline. Additionally, we also visualize examples of failures detected by our Detic filter and failed inpainting examples in Fig. 9.

A.2. HSSD Image Dataset

To finetune our CLIP-UNet model for SP mask prediction on a high-quality image dataset free from inpainting artifacts, we utilize the Habitat [16, 17] simulator along with the HSSD [13] scene dataset. HSSD is a synthetic indoor environment dataset comprising 211 high-quality 3D scenes, containing 18,656 models of real-world objects. We generate the HSSD image dataset using the Habitat simulator, which allows us to manipulate scenes to render images with or without object, thereby avoiding any artifacts that models could exploit. The training dataset consists of $\sim 80k$ images generated using 135 train scenes with 8 object categories. Similarly, we create an evaluation dataset of $\sim 18k$ images using 33 val scenes with 8 object categories. Next, we describe the details of our image sampling process for different objects using the simulator.

Image Sampling. To generate images from diverse viewpoints for each object instance, we first sample a set of candidate camera poses determined from polar coordinates (r, θ) relative to the object centroid, where $r \in \{0.5m, 1.0m, 1.5m, 2m\}$ and $\theta \in \{0°, 10°, ..., 360°\}$. We sample two types of viewpoints:

- **Looking at Object**: For images looking at the objects of interest, we capture images with the camera's principal axis parallel to a ray from the camera's center to the object's centroid. We only keep the frames where the object of interest covers at least 5% of the frame. This step ensures the inclusion of images where the target object and a valid placement is visible.
- **Random Viewpoints**: To add diversity, we also generate images from random viewpoints. Specifically, we run a frontier exploration [69] navigation agent in the environment to achieve $\sim$90% coverage. We then randomly sample N images from the navigation trajectory, with $N = 250$ in our case, and add them to our dataset. We run this navigation agent 3 times from random locations in each scene. We do not apply any frame coverage constraint during this phase to include images where no possible placement for an object exists.

After determining all the viewpoints for each object instance in a scene, we programatically generate images with and without objects, target placement mask, and receptacle masks to add to our dataset.

A.3. Real Evaluation Dataset

For our experiments in Sec. 5, we use a real image dataset comprising 400 images, collected from the LAION dataset [9] and 2 real-world environments from [70, 71]. Specifically, this dataset includes 200 images from the LAION dataset that were not seen during training, and an additional 200 images from the real-world environments from [70, 71].

B. Metric Details

B.1. Receptacle Priors

To compute receptacle precision and recall metrics, we use the receptacles shown in Tab. 5 for each object type from HSSD [13] scenes. To find the receptacle categories, we retrieve a list of receptacles that have an instance of the target object category placed on top, using metadata from the simulator. It is important to note that since all `Trash Can` instances are usually found on the floor of an environment, there is no designated receptacle category for the `Trash Can` category. Similarly, while some instances of the `Potted Plant` category are also found on the floor, we do not include `Floor` as a receptacle category. This exclusion is due to the fact that the annotations for the `Floor` category cover the entire scene, making it challenging to quantify which part of the `Floor` annotation is a good or bad for object placement.

B.2. Human Evaluation

To assess the performance of various methods on the Semantic Placemen (SP) task, we conduct a human evaluation study using Amazon Mechanical Turk. Specifically, we conduct a user preference experiment in which human annotators are asked to compare SP mask predictions from 5 models (baselines from Tab. 2) and rank them from most to least preferred. We conduct two types of the user study: one with

Object Category	Receptacles
Cushion	Couch, Bed, Sofa, Armchair
Potted Plant	Coffee Table, Table, Chest of Drawers, Shelve, Kitchen Counter
Book	Coffee Table, Table, Shelves, Couch, Sofa
Vase	Coffee Table, Table, Chest of Drawers, Shelf, Kitchen Counter
Alarm Clock	Bedside Table, Table, Chest of Drawers
Laptop	Bed, Desk, Coffee Table, Table
Table Lamp	Bedside Table, Chest of Drawers
Toaster	Kitchen Counter
Trash Can	-

Table 5. Mapping of receptacles for each object category.

Object Category	Receptacles
Cushion	Couch, Bed, Sofa, Armchair, Bench
Potted Plant	Window Sill, Table, Chest of Drawers, Shelve, Balcony
Book	Coffee Table, Table, Bookshelf, Desk, Nightstand, Bed
Vase	Coffee Table, Table, Shelf, Mantle, Window Sill
Alarm Clock	Bedside Table, Nightstand, Desk, Shelf
Laptop	Desk, Table, Workstation
Table Lamp	Desk, Nightstand, End Table, Shelf
Toaster	Kitchen Counter, Shelf, Pantry
Trash Can	Kitchen, Bathroom, Bedroom, Office

Table 6. Prior + Detector Baseline. Mapping of receptacles from a LLM for each object category.

the real image dataset and another with images from the HSSD [13] scene dataset used in our experiments. For each study, we randomly select 400 images from the evaluation split of the respective datasets. Each Amazon Mechanical Turk worker is assigned 20 images to evaluate preferences, and each worker is allowed to participate in the study only once. We report percentage of times annotators rank each model's SP predictions as the best (*i.e.* ranked above all other SP predictions) in Tab. 2 of the main paper.

C. Baseline Details

Prior + Detector. For this baseline we leverage commonsense priors available in LLMs to find target receptacles for a particular object and use a open-vocabulary detector, Detic [19], to localize the receptacle in the image. For each of the 9 object categories in the dataset we prompt an LLM for common receptacle categories on which each object is found in indoor environment, shown in Tab. 6. Next, during evaluation we use object detector to localize the segmentation mask of all valid receptacles for a object category in an image.

LLaVA. VLMs like LLaVa [14] connect vision encoders to LLMs, enabling general purpose vision-and-language understanding. To evaluate LLaVA on the SP task, given an input image, we prompt it to output normalized bounding box coordinates for localizing a placement area. The prompt we use is as follows:

"You are a smart assistive robot tasked with cleaning this house. Localize the area in image as a bounding box in normalized coordinates to place the <object_category>".

Subsequently, we convert the predicted normalized bounding box into a binary segmentation mask, which is then used as the SP mask predictions for downstream applications. Refer Fig. 10 and Fig. 11 for qualitative examples.

GPT4V [65]. Similar to LLaVA [14], GPT4V is a multimodal LLM renowned for its vision-and-language understanding capabilities. To evaluate GPT4V for the SP task, we feed it an input image and prompt it to output normalized bounding box coordinates. These coordinates are then localized to a placement area and converted into a binary segmentation mask for use as SP mask predictions. We use the following prompt:

"Here is an image of an indoor living environment. We would like to determine all places in the image where one could potentially place an object of type <object_type> so that environment remains tidy. For example, you should not place a blender on the floor as blenders are not typically found on the floor.

Please respond, in text, with a list of bounding box coordinates of potential locations. These bounding box coordinates should be of the form

[min x, min y, max x, max y]

where x and y are 0-1 valued and correspond to the fraction of the image along the width and height of the image with the top left of the image as the origin. Each set of coordinates should be on a new line. If there are no locations in the image where a <object_type> could be placed, respond only with 'NONE'. Respond ONLY with these coordinates or NONE, do not include any other text in your response."

Subsequently, the predicted normalized bounding boxes are converted to binary segmentation masks as SP mask predictions for downstream evaluation. Refer Fig. 10 and Fig. 11 for qualitative examples.

C.1. Open-Vocab Object Detector Ablations

Tab. 7 presents results for when varying the open vocabulary object detectors used in our LLM+Detector baseline. We compare performance on the HSSD validation split using TrP, RSP, and RSR metrics and consider three open vocabulary detectors: Detic [19], OwlViT [72], and GroundedSAM [20, 73]. Overall, we find Detic achieves the highest RSP, RSR, and comparable or better TrP compared to OwlVit and GroundedSAM.

	Method	HSSD VAL		
		TrP (↑)	RSP (↑)	RSR (↑)
1)	LLM + Detic	10.1	**41.0**	**38.2**
2)	LLM + OwlVit	**11.4**	26.2	26.2
3)	LLM + GroundedSAM	8.9	35.1	32.1

Table 7. Ablations of object detectors for prior based baselines.

D. Qualitative Results

In Fig. 10 and Fig. 11, we visualize qualitative examples from the CLIP-UNet, Prior + Detector (Detic), LLaVA, and GPT4V baselines. These examples are images in the SP real evaluation split, which were used for human evaluation.

E. Embodied Evaluation Setup

In this section, we detail the Embodied Semantic Placement Policy used in Sec. 5.2 for evaluating the eSP task of building a tidying robot. Our experiments employ Hello Robot's Stretch robot [1] with the full action space as defined in [66]. Specifically, the observation space, shown in the Fig. 7 Observations, includes RGB+Depth images from the robot's head camera, the camera pose, the robot's joint and gripper states, and the robot's pose relative to the starting pose of an episode. The robot's action space comprises discrete navigation actions: MOVE_FORWARD ($0.25m$), TURN_LEFT ($30°$), TURN_RIGHT ($30°$), LOOK_UP ($30°$), and LOOK_DOWN ($30°$). For manipulation, we use a continuous action space for fine-grained control of the gripper, arm extension and arm lift. The head tilt, pan and gripper's yaw, roll and pitch can be changed by a maximum of $0.02 - 0.1$ radians in a single step, while the arm's extension and lift can be changed by up to $2 - 10cm$ per step. To perform the task with only the robot's observations and SP mask predictions from the CLIP-UNet at each frame, we build a two-stage modular policy, comprising "navigation" and "place" policies, illustrated in Fig 7 (a.) SP Guided Navigation Policy and (b.) SP Guided Place Policy, respectively. The details for both policies are as follows.

SP Guided Navigation Policy. Building upon the navigation policy from [68], we replace the semantic map module with our semantic placement (SP) map module. To construct the SP affordance map, we predict the SP mask using egocentric observations at each timestep. This mask is then backprojected into a point cloud using preceived depth. We bin the point cloud into a 3D SP voxel map and sum it over height to derive the 2D SP map. Similar to [68], our navigation policy employs frontier exploration [69], using the 2D SP map. We first build a SP map by running the policy with the goal of maximizing coverage of the environment for 250 steps. On average, we achieve about 60% coverage of an environment within these 250 steps. Subsequently, the agent uses the SP map to navigate towards the SP mask instance that occupies the largest area on the 2D map.

SP Guided Place Policy. We build upon heuristic place policy from [66]. This policy assumes that the robot is within interactable distance (within $0.2m$) of the target receptacle where the object is to be placed. First, the agent takes a panoramic turn until a valid SP prediction is found (*i.e.* not on the floor). This involves projecting the depth and SP prediction onto a point cloud, transforming it into the agent's base coordinates, and applying a height filter. Once a valid SP prediction is identified, we estimate a placement point at the center of the largest slab (point cloud) for object placement on a flat surface. To identify the largest flat surface slab, we score each point based on the number of surrounding points in the X/Y plane (with Z being up) within a $3cm$ height threshold, similar to [66]. After determining the placement point, we rotate the robot to facing the point. This is required because the Stretch robot's arm is not aligned with the camera by default. If the robot is at least $38.5cm$ away from the placement point, we move the robot forward, and re-estimate the placement point as described in [66]. Finally, when the robot is sufficiently close, we use inverse kinematics to compute a sequence of actions to move the arm $15cm$ above the sampled voxel (to avoid collisions) to place (or drop) the object.

F. Failure Modes

In this section, we describe various failure modes of our CLIP-UNet model observed during its evaluation on the SP task and in the downstream embodied evaluation of the eSP task.

F.1. Semantic Placement

Refer to Fig. 14 for examples of failure modes in SP mask predictions by our CLIP-UNet model. The common failure modes include:

Surface Grounding. Predictions that are not properly grounded to a surface of the receptacle in the image.

Incorrect Receptacle. Predictions with a 0 Intersection over Prediction (IoP), indicating no overlap with any of the visible receptacles in the image.

Geometry Unaware. Our method, by design, is not capable of predicting SP masks that are object shape aware (as our model's only knowledge about the object is the object's category). Consequently, we sometimes observe placements predicted by the model that are not geometry-aware, meaning the SP masks highlight areas where there is insufficient space to place a new object.

Misc. This category contains all other failure cases, including predictions from the model that are noisy, placed on the floor/ceiling, or involve closed receptacles, etc.

F.2. Embodied Semantic Placement

The majority of eSP evaluation failures come from the navigation and place planner, which include:

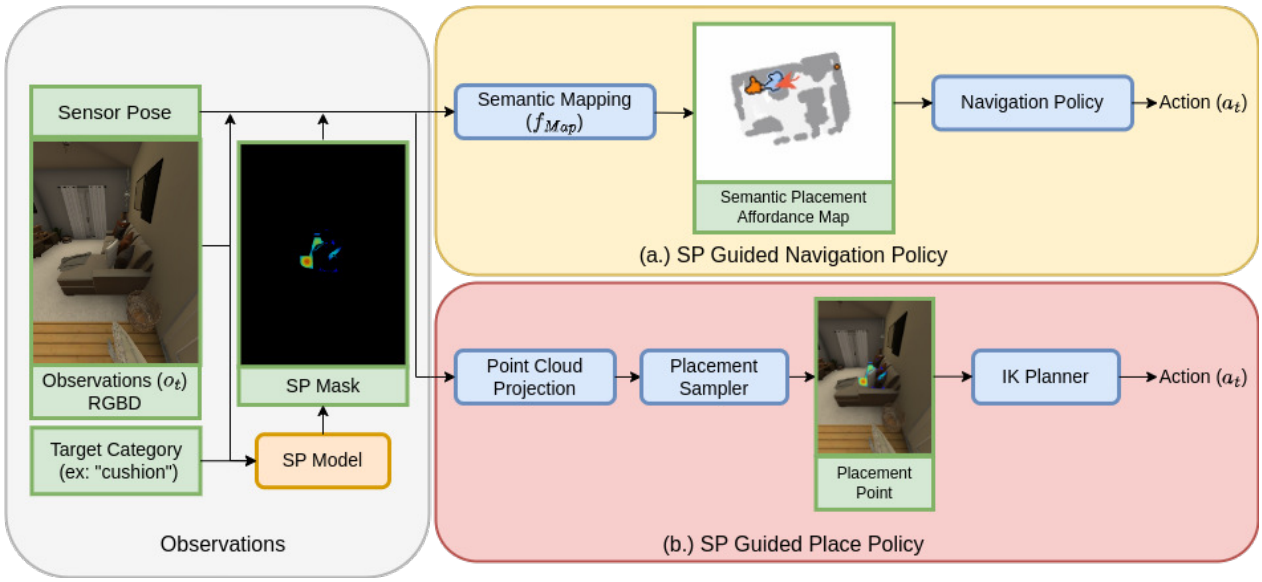

Figure 7. Embodied Evaluation Pipeline. We build a two-stage modular policy consisting of: 1.) SP Guided Navigation Policy: Uses frontier exploration and semantic placement affordance 2D map to navigate to placement area, 2.) SP Guided Place Policy: Uses predicted SP mask, projects it onto a pointcloud to sample placement point and uses IK planner to place the object.

Navigation Failure. In 53.5% of cases, the navigation policy fails to reach within $0.2m$ of the predicted SP mask. This is often due to the requirement for precise navigation around clutter.

Place Failure. The place policy fails 31.0% of the time to execute fine-grained control to realize the highlighted SP prediction. Occasionally, realizing SP predictions is not feasible with the Stretch embodiment. For example, if a SP mask indicates a placement at the center of a dining table, the robot might be unable to reach it due to the table's size and the maximum arm extension of the Stretch robot. This highlights the need for future work in learning SP in an embodiment-aware manner to improve downstream performance.

Incorrect SP Masks. In 15.5% of cases, the placement predicted by the SP mask is incorrect, such as when the SP mask is placed on an incorrect receptacle.

Refer to the attached videos in the supplementary material for examples of these failure modes.

G. Limitations

Our approach is fundamentally constrained by the limitations of open-vocabulary object detectors, segmentation models, and inpainting models. Since we employ these advanced "foundation" models off-the-shelf for automatic data generation, the quality of our generated data is heavily dependent on the performance of these models. Moreover, the occasional poor performance of these models can introduce biases into the training dataset, which downstream models might exploit. For example, false positive detections from open-vocabulary detectors (*e.g.* a ceiling light detected as a lamp) may lead to biases in predicting SP masks for lamps on the ceiling. Similarly, imperfect inpainting models can produce artifacts like partially inpainted generations that bypass our detector-based validations, resulting in training data that may instill unrealistic biases in our model. While finetuning on simulated data from HSSD can mitigate some of these biases, it might also introduce a domain gap for sim-to-real transfer. Collecting high-quality real-world data for finetuning could help to alleviate this limitation. Another challenge is that deploying the SP prediction model zero-shot for applications like eSP might yield SP predictions that are not realizable given the robot's physical capabilities. A potential solution could involve finetuning the SP model with the downstream task in an end-to-end manner. This aspect, however, remains as part of future work.

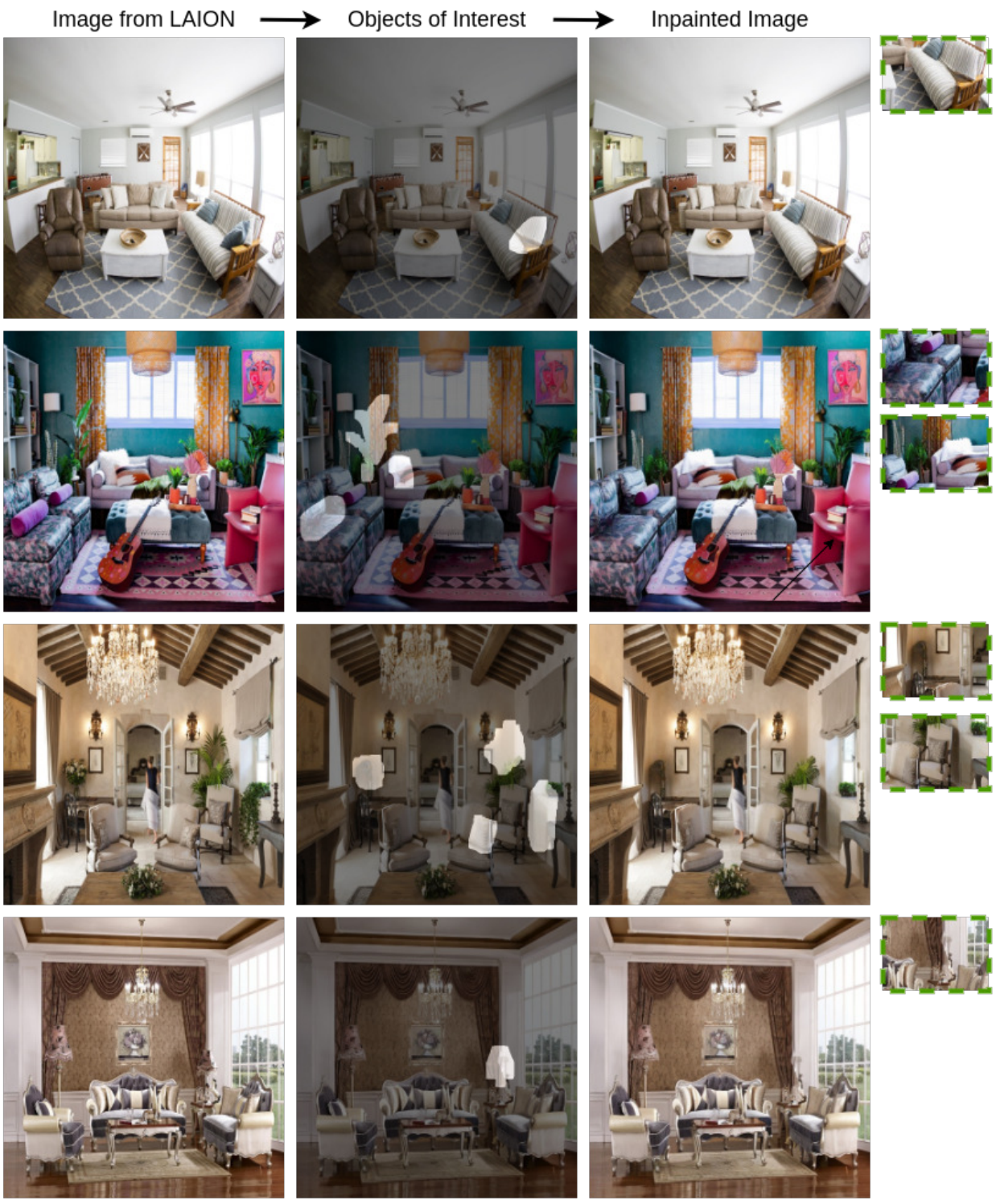

Figure 8. Qualitative examples from SP train data generated using our proposed automatic data generation pipeline.

Figure 9. Qualitative examples of inpainting failure during SP data generation using our proposed automatic data generation pipeline.

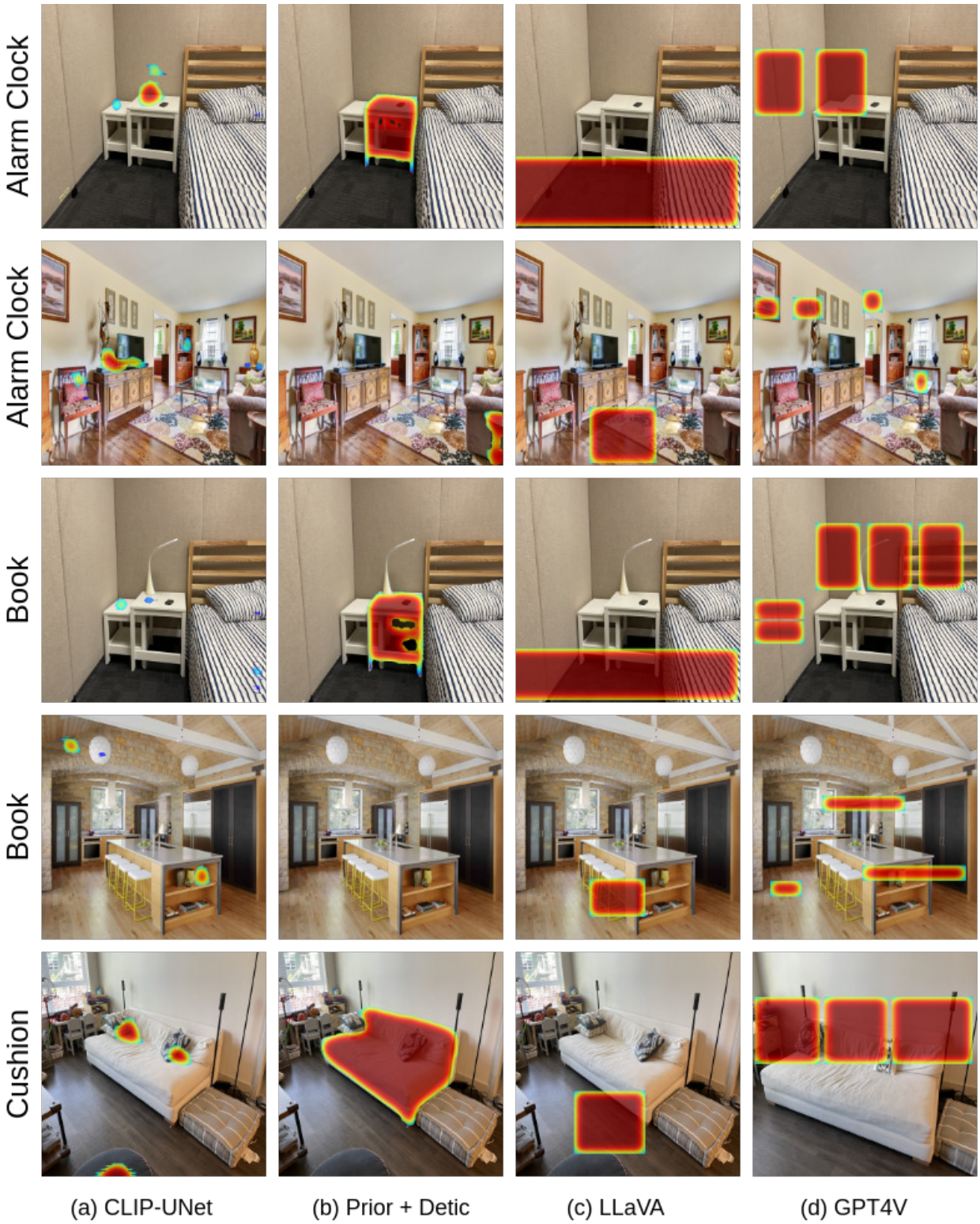

Figure 10. Qualitative examples of SP masks predicted by all the baselines on SP Real val dataset

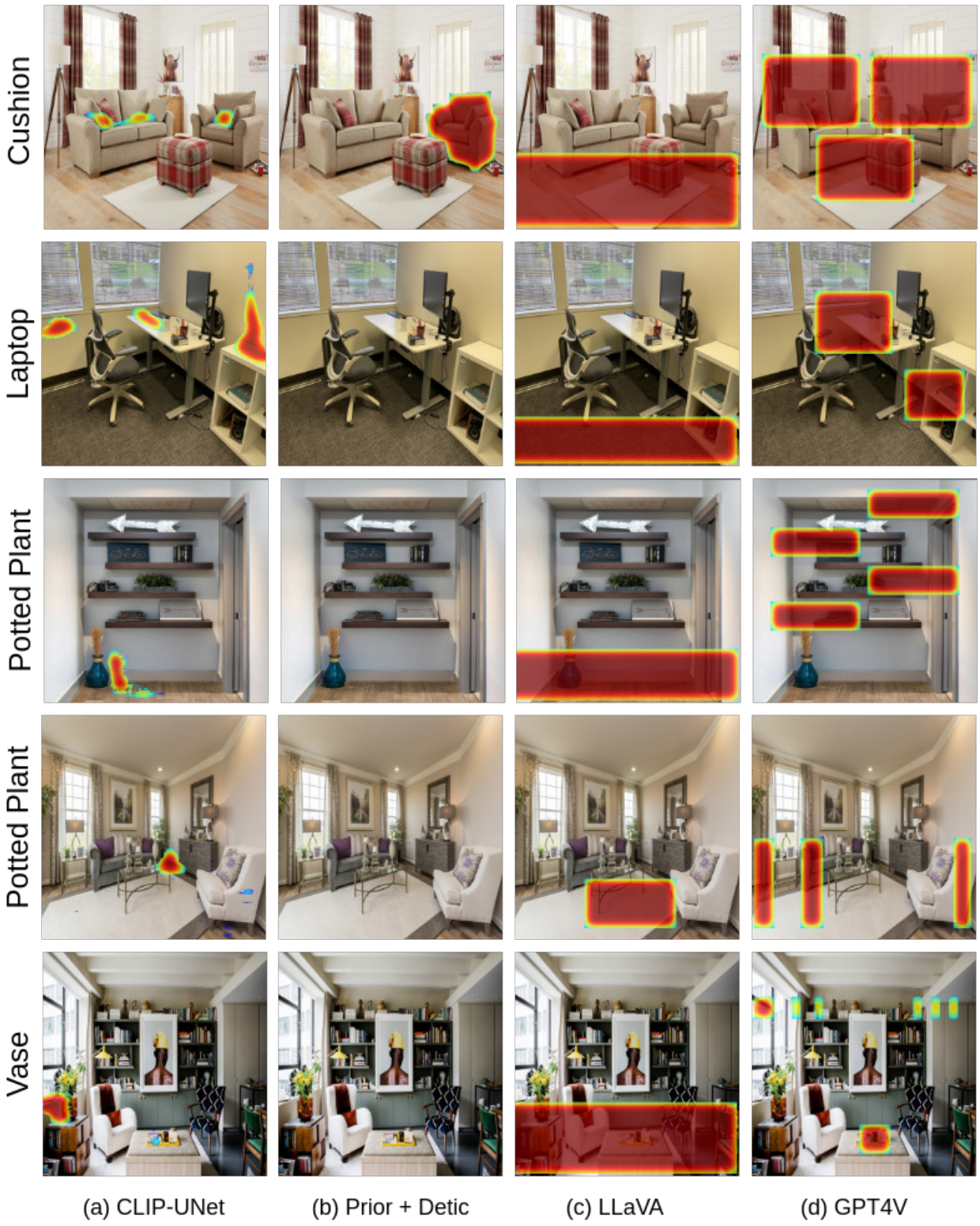

Figure 11. Qualitative examples of SP masks predicted by all the baselines on SP Real val dataset

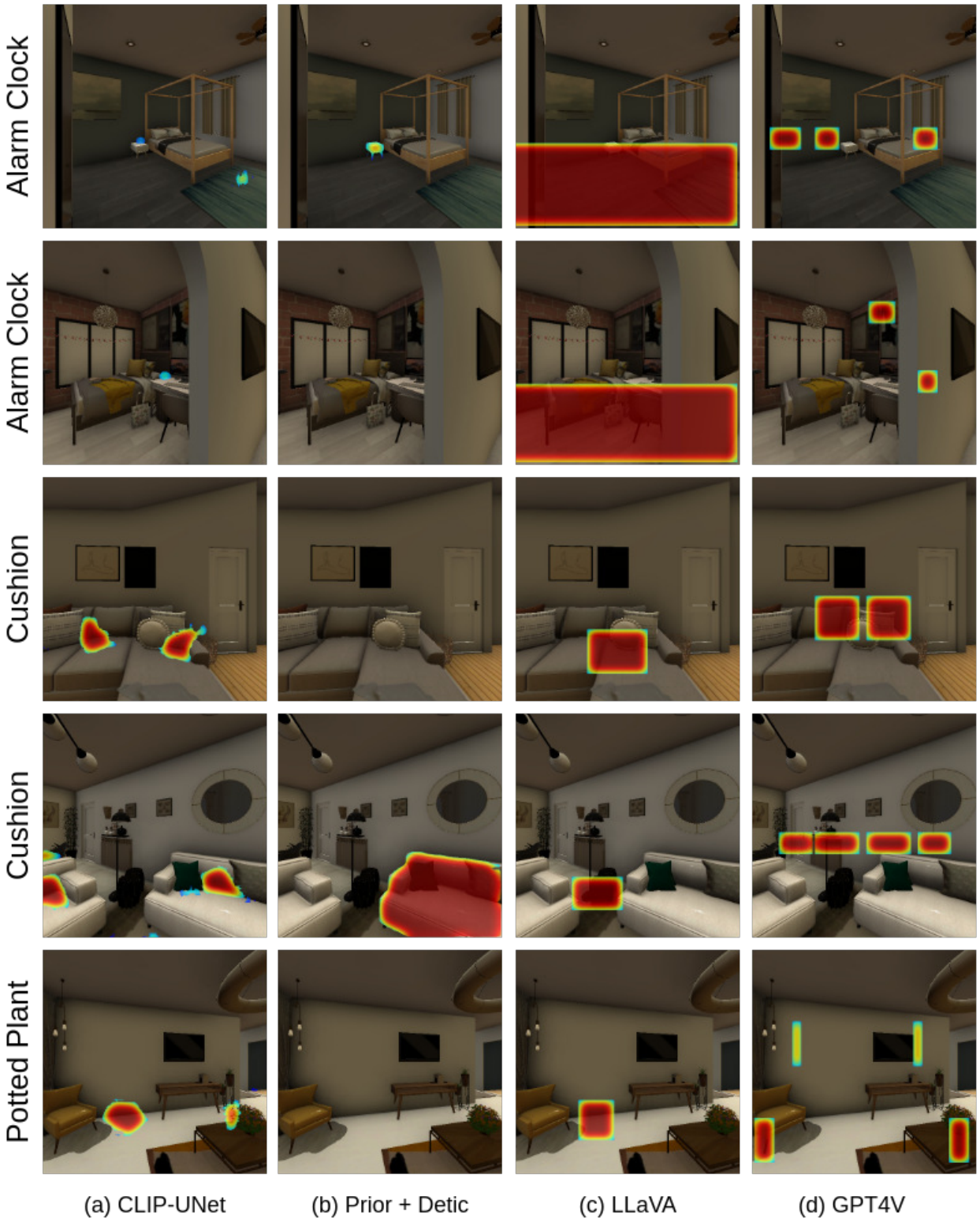

Figure 12. Qualitative examples of SP masks predicted by all the baselines on SP HSSD val dataset

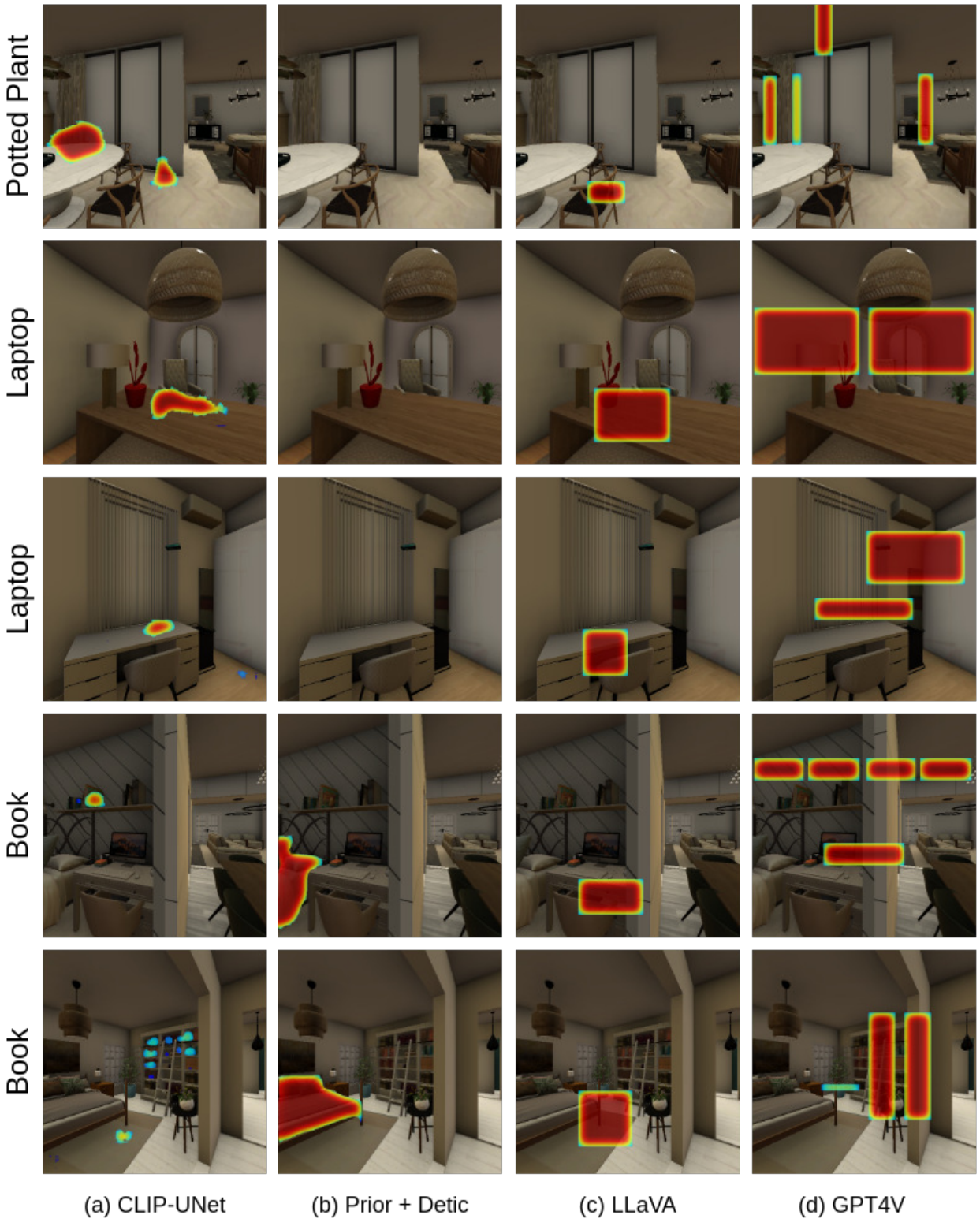

Figure 13. Qualitative examples of SP masks predicted by all the baselines on SP HSSD val dataset

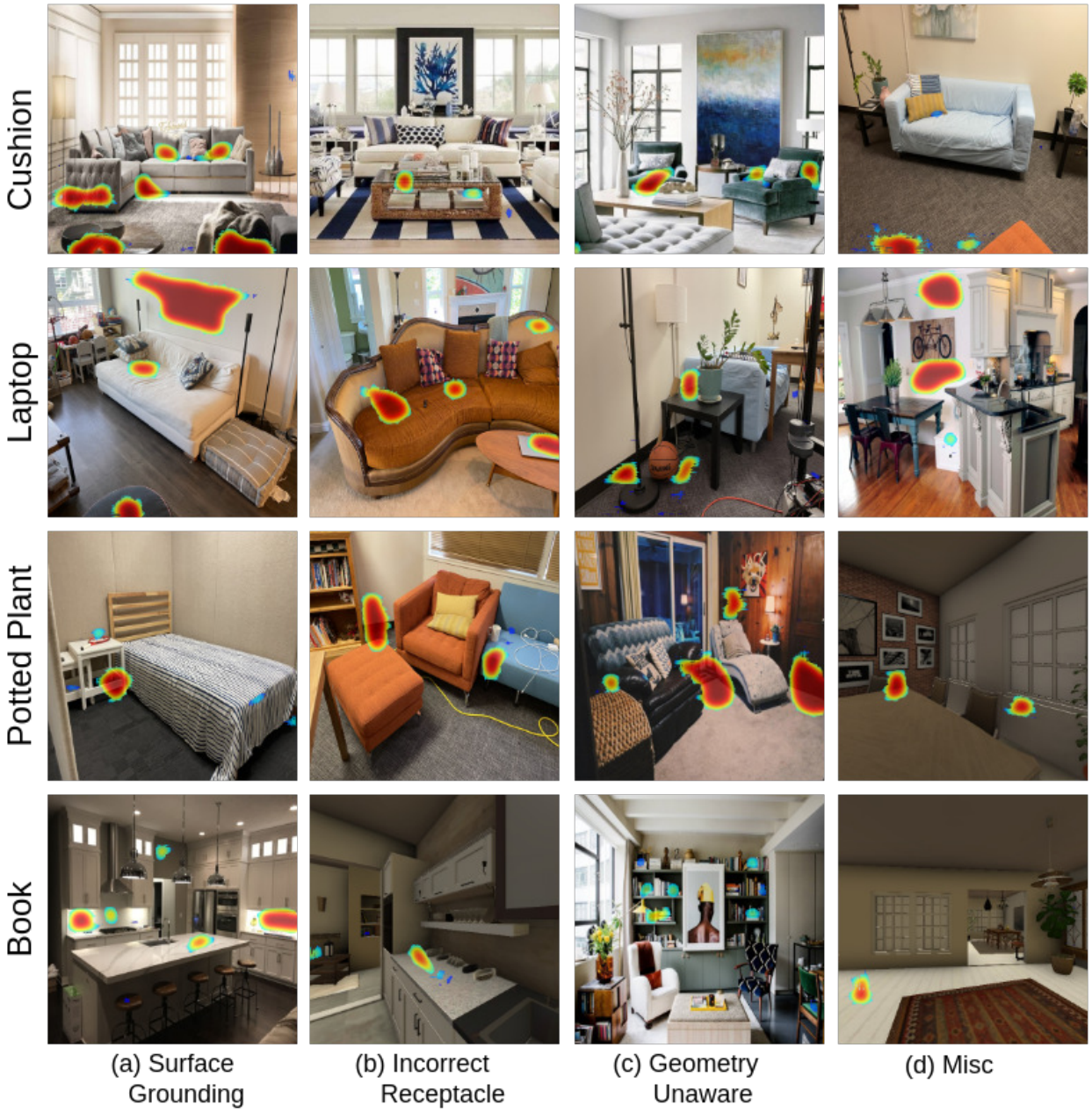

Figure 14. Qualitative examples of failure modes of SP mask prediction by our approach