# Seeing the Unseen: Visual Common Sense for Semantic Placement

Ram Ramrakhya[1*] Aniruddha Kembhavi[2] Dhruv Batra[1] Zsolt Kira[1] Kuo-Hao Zeng[2†] Luca Weihs[2†]

[1]Georgia Institute of Technology [2]PRIOR @ Allen Institute of AI

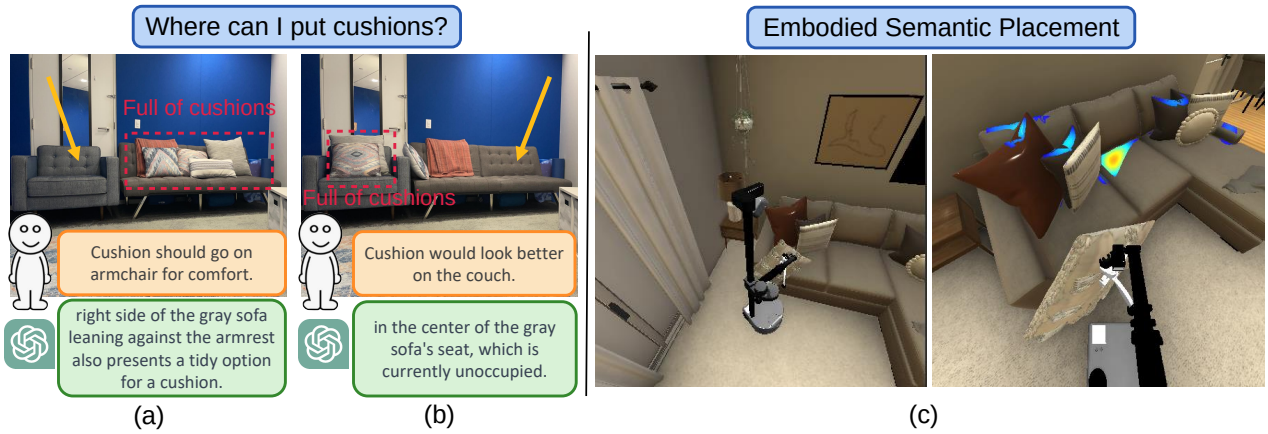


**Fig. 1. Semantic Placement**. Consider asking an agent to place cushions in a living room. In (a), the couch on the right is already full with cushions, and a natural human preference would be to place the cushion against the backrest of the armchair. In (b), a natural placement preference would be center of the couch. We propose the problem of Semantic Placement (SP) – given an image and a name of an object, a vision system must predict a semantic mask indicating a valid placement for the object in the image. For both (a) and (b) GPT4V gives meaningful natural language responses but, as we show, struggles to localize regions precisely in pixel space. (c) Our SP predictions enable a Stretch robot [1] from Hello Robot to perform Embodied Semantic Placement (eSP) task within a photorealistic simulated environment.

***Abstract*— Computer vision tasks typically involve describing what is present in an image (*e.g.* classification, detection, segmentation, and captioning). We study a visual common sense task that requires understanding 'what is not present'. Specifically, given an image (*e.g.* of a living room) and a name of an object ("cushion"), a vision system is asked to predict semantically-meaningful regions (masks or bounding boxes) in the image where that object *could be placed* or is likely be placed by humans (*e.g.* on the sofa). We call this task: Semantic Placement (SP) and believe that such common-sense visual understanding is critical for assitive robots (tidying a house), AR devices (automatically rendering an object in the user's space), and visually-grounded chatbots with common sense. Studying the invisible is hard. Datasets for image description are typically constructed by curating relevant images (e.g. via image search with object names) and asking humans to annotate the contents of the image; neither of those two steps are straightforward for objects *not* present in the image. We overcome this challenge by operating in the opposite direction: we start with an image of an object in context, which is easy to find online, and then *remove* that object from the image via inpainting. This automated pipeline converts unstructured web data into a dataset comprising pairs of images with/without the object. With this proposed data generation pipeline, we collect a novel dataset, containing $\sim$1.3M images across 9 object categories. We then train a SP prediction model, called CLIP-UNet, on our dataset. The CLIP-UNet outperforms existing VLMs and baselines that combine semantic priors with object detectors, generalizes well to real-world and simulated images and exhibits semantics-aware reasoning for object placement. In our user studies, we find that the SP masks predicted by CLIP-UNet are favored 43.7% and 31.3% times when comparing against the 4 SP baselines on real and simulated images. In addition, leveraging SP mask predictions from CLIP-UNet enables downstream applications like building tidying robots in indoor environments.***

*Work done as part of the internship at PRIOR @ AI2

†Equal advising

## I. Introduction

When tasked with putting away a cushion in a home, humans quickly bring to bear extensive priors about how cushions are used and where they are most frequently placed. For instance, cushions are generally put on or near seating areas (*e.g.*, on a couch). However, these priors themselves are not enough: consider an example living room shown in Fig. 1(a). As shown in the figure, the couch already has cushions on both armrests so, to avoid redundancy, one might place the cushion against the back of the armchair for the comfort of anyone who might later sit upon it. On the other hand, given the same task with the image from Fig. 1(b), the answer might change to placing the cushion in center of the couch to give the room a more aesthetically pleasant feel as the armchair already has a cushion on it. Notice that the answer from humans about object placement changes based on changes in the visual context. We call this task Semantic Placement (SP), and believe that such common-sense visual understanding is critical for assistive robots (tidying a house), AR devices (automatically rendering an object in the user's space), and visually-grounded chatbots with common sense.

How can we build vision systems with SP prediction abilities? Modern computer vision tasks have focused on classifying, localizing, and describing what is visible in an image (*e.g.* classification, object detection, segmentation, and captioning). Most visual representation learning approaches, *e.g.* CLIP [2], [3], [4], [5], [6], use losses that encourage the learned representations to capture what is shown in the image

but are not designed to be used to answer queries about the invisible in the image *zero-shot*; the visual context generated by these models is, however, extremely valueable and we use CLIP as the visual backbone in this work. Recent advances in vision-and-language (VLM) foundation models has made some progress in this direction. We can ask VLMs questions that require reasoning about the invisible, conditioned on visual context to infer the answers to a question. However, existing VLMs are still in early stages and struggle to answer queries that require precise localization in pixel space as shown in our experiments (see Sec. III).

In this paper, we study the problem of Semantic Placement (SP) of objects in images. In particular, given an image (*e.g.* showing a living room) and name of an object ("cushion"), a vision system is tasked to predict a pixel-level mask highlighting semantically-meaningful regions (referred as SP masks) in an image where that object could be placed or is likely to be placed by humans (*e.g.* a couch). Learning to predict SP masks is hard, since the target object is typically not visible in the given image. Datasets for image description are typically constructed by curating relevant images (e.g. via image search with object names) and asking humans to annotate the contents of the image; neither of those two steps are straightforward for objects *not* present in the image.

To overcome this challenge, we propose to operate in the opposite direction – specifically, we start with an image of an object in context (which is easy to find online) and *remove* that object from the image via inpainting [7], [8]. This automated pipline converts unstructured web data into a a dataset comprising pairs of images with/without the object at scale *without expensive human annotation*. However, inpainting models are not perfect. We find that SP prediction models, when trained on inpainted images, tend to latch onto inpainting artifacts. This leads to high performance on inpainted images, but lower performance on real images. To remedy this, we propose a novel data augmentation method, combining results from multiple inpainting models, diffusion based augmentations, and common data augmentations (refer Sec. II-A section for more details). Using this automated pipeline, we generate a large SP dataset using real world images from LAION [9], including $\sim$1.3 million images across 9 object categories.

We propose a simple method for SP mask prediction by using a frozen CLIP [2] backbone with a language conditioned UNet [10] decoder inspired by LingUNet [11] and CLIPort [12], in Sec. II-C. First, we pretrain the CLIP-UNet model on images from our SP dataset and then finetune on a small high-quality image dataset of $\sim 80k$ synthetic images collected from synthetic HSSD [13] scenes, where inpainting is unnecessary as objects can be removed programmatically from the underlying 3D scenes. We find finetuning on this small but high-quality dataset with ground truth object placement annotations improves performance of our CLIP-UNet baseline and enables better generalization to both real and synthetic images.

For evaluation we use 400 real world images from LAION [9] and $\sim 18k$ from HSSD [13] scenes. We find that CLIP-UNet outperforms strong baselines leveraging VLMs, including LLaVa-1.5 [14] and GPT4V [15], and methods using open-vocabulary object detection and segmentation models with placement priors coming from LLMs. In user studies, we find that the SP mask predicted by our method are favored 43.7% times against the baselines on real images and by 31.3% times on images from HSSD scenes.

SP mask predictions hold potential for a variety of downstream applications, including assistive agents, real-time AR rendering, and visually-grounded chatbots. In this paper, we demonstrate that SP masks predicted by CLIP-UNet enable embodied agents to perform Embodied Semantic Placement (eSP) task in a photorealistic, physics-enabled simulated environment, Habitat [16], [17], [18] using Hello Robot's Stretch robot [1]. In eSP, an agent is spawned at a random location in an indoor environment and is tasked with placing an instance of a target object category at a semantically meaningful location with access to robot observations (RGB, Depth, and pose) and SP masks from a SP model. Using SP masks predicted by our CLIP-UNet model, agent achieves a 12.5% success rate on 8 categories when evaluated in 10 unique indoor scenes over 106 episodes. While the absolute success is indeed low, we note that majority of failures $\sim$80% for downstream eSP task are due to imperfect control policy for object placement and fine-grained navigation, which is orthogonal to the focus of this work. We show a qualitative example of a placement prediction by our agent for object category 'cushion' while performing the task in Fig. 1 (c).

In summary, our contributions include: (1) a novel task called Semantic Placement (SP), (2) an automated data curation pipeline leveraging inpainting and object detection models to supervise an end-to-end SP prediction model using real-world data, (3) a novel data augmentation method to alleviate overfitting to inpainting artifacts, and (4) an approach that generates SP predictions which generalize well to the real-world and enable downstream robot execution.

## II. Approach

### A. Dataset Generation

To collect paired data for training (referred as LAION-SP) the SP model, we propose leveraging recent advances in open-vocabulary object detectors, segmentation models, and image inpainting models. With these powerful off-the-shelf "foundation" models, we can generate paired training data at scale using images in the wild. Fig. 2 shows our automated data generation pipeline. At the end of the pipeline, each output image is paired with object categories and includes masks showing where such categories can be placed. Details follow below.

**(A) *Query Image.*** First, we gather 1M indoor images from the LAION dataset by using text queries such as 'living room', 'bedroom', and 'kitchen' to filter out irrelevant images *i.e.* images not from houses.

**(B) *Find Objects of Interest.*** Next, for each image we use Detic [19], an open vocabulary object detector, to detect objects of interest for our task. We use 9 target object categories in this paper, specifically `Plotted Plant`, `Lamp`, `Cushion`, `Vase`, `Trash Can`, `Toaster`, `Table Lamp`, `Alarm Clock`,

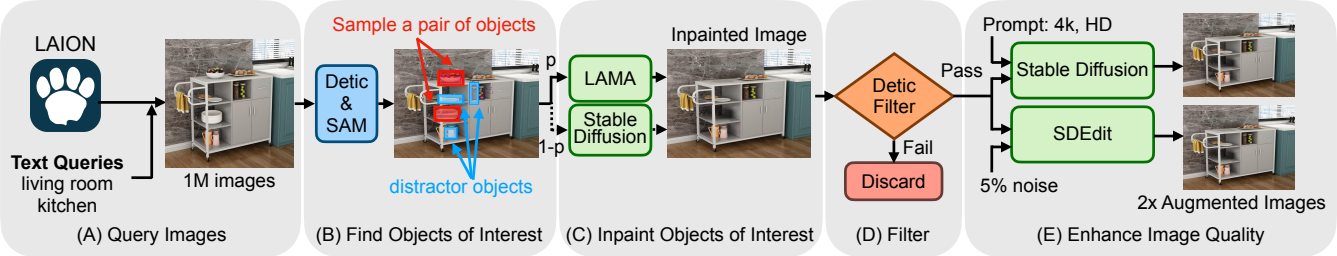


**Fig. 2. Automatic Training Dataset Generation Pipeline Utilizing Foundation Models and Web Data.** Our pipeline consists of five steps: (A) *Query Images*, (B) *Find Objects of Interest*, (C) *Inpaint Objects of Interest*, (D) *Filter*, (E) *Enhance Image Quality*

and `Laptop`. For each detected instance, we generate a segmentation mask using SAM [20]. We use SAM masks instead of Detic masks as they are fine-grained and result in better inpainting performance. For details on prompts used for SAM and Detic see Appendix A.

**(C) *Inpaint Objects of Interest.*** Using the detection results, we pass the segmentation masks of instances of a sampled object category and original image to one of the two inpainting models (each sampled with 50% probability), LAMA [7] or Stable Diffusion [21], to generate an inpainted sample. Specifically, we randomly sample a few instances of a target object category and 1-4 distractor objects of different category for inpainting. We add distractor instances to make the task of SP prediction more challenging as the model cannot simply predict the, possibly only, free space. This also helps prevent the model from overfitting to inpainting artifacts.

**(D) *Filter.*** Inpainting models are imperfect and we need strict validation mechanisms to check if inpainting was successful or not. To do so, we use 2D instance matching between original and inpainted images using the detections from Detic [19]. Specifically, if we find an object instance post-inpainting with IOU greater than $90\%$ with an instance from original image, the inpainting model failed and we discard the generated result. Samples that pass the check are used in training dataset.

**(E) *Enhance Image Quality.*** In our initial experiments, we found that training the SP model directly using the dataset generated by the *Filter* step leads to overfitting.[1] The model quickly latches onto the artifacts introduced from the inpainting models. To mitigate this issue, we generate two augmented versions of each inpainted image with the help of diffusion models. To create the first augmented variant, we add $5\%$ Gaussian noise to the image and use SDEdit [22] to denoise the image similar to Affordance Diffusion [23]. To create the second variant, we feed the inpainted image to Stable Diffusion img2img [8] model and prompt it with 'high resolution, 4k' which, in practice, results in small object texture changes. We find this acts as regularization and helps avoid overfitting on inpainting artifacts during training.

In total, we generate 1,329,186 images with an object category and its corresponding SAM mask from 48,728 unique images queried from the LAION dataset. In addition to LAION-SP training dataset of $\sim 1.3M$ images we also create a dataset of 400 unseen original images from LAION for our evaluation referred as LAION-SP Val dataset.

[1]The model trained on the inpainted images without quality enhancement (*i.e.*, Step E) yields $\sim$0 TP zero-shot evaluating on HSSD dataset.

### B. Synthetic Images

For finetuning, we collect a small high-quality image dataset without needing inpainting from synthetic HSSD [13] scenes, a synthetic indoor environment dataset comprising 211 high-quality 3D scenes, containing 18,656 models of real-world objects. Using 135 training scenes we generate $\sim 80k$ training images across 8 object categories. Similarly, using 33 unseen evaluation scenes we generate a dataset of $\sim 18k$ images for evaluation with the same 8 object categories. Additional details on data generation is in App. B.

### C. Learning Semantic Placement

To learn an SP mask prediction model, we pretrain our model using the LAION-SP dataset generated in Sec. II-A, then finetune it using small high-quality synthetic dataset mentioned in Sec. II-B. The inputs to the SP model include an RGB image $I$ in $H \times W \times 3$ size and a target object category $q$ in text. The model outputs an affordance mask $M$, size $H \times W \times 1$, conditioned on the target object. Fig. 5 shows the architecture of our proposed CLIP-UNet model. Inspired by CLIPort [12], we use a frozen ResNet50 [24], pre-trained by CLIP [2], to encode the input image $I$ into a feature tensor $f$ up until the penulitmate layer $\mathbb{R}^{7\times7\times2048}$. The decoder then upsamples the feature tensor $f$ to $f^{(\ell)} \in \mathbb{R}^{H_\ell \times W_\ell \times C_\ell}$ at each layer $\ell$ and, at the end, produces a mask $M \in \mathbb{R}^{H\times W\times 1}$, where $0 \leq M[i,j] \leq 1$. To encode the target object category $q$, we use CLIP pretrained transformer based sentence encoder to construct a target embedding $e \in \mathbb{R}^{1024}$. To condition the decoding process with the target embedding, we first downsample it to $\bar{e} \in \mathbb{R}^{C_\ell}$ and then tile it to match the dimension of feature tensor $f^{(\ell)}$ at layer $\ell$ in the decoder: $\bar{e} \rightarrow \bar{e}^{(\ell)} \in \mathbb{R}^{H_\ell \times W_\ell \times C_\ell}$, where $C_\ell = \{1024, 512, 256\}$ and $\ell \in \{1, 2, 3\}$. Then, we use the tiled target embedding to condition the visual decoder layers through an element-wise product. As CLIP utilizes contrastive loss on the dot-product aligned features from pooled image features and language embeddings, the element-wise product allows us to leverage this learned alignment while the tile operation preserves the original dimensions of visual features. Inspired by LingUNet [11], we apply this language conditioned operation to the first three upsampling layers right after the feature tensor $f$ produced by the frozen ResNet. Following UNet [10], we also add skip connections to decoder layers from the corresponding layers in ResNet encoder. This way the model preserves different levels of semantic information from image. Refer Appendix C for training details.

## III. Experiments

### A. Semantic Placement Evaluation

**Fig. 3.** Qualitative examples of SP masks predicted by our CLIP-UNet model pretrained on LAION-SP dataset and finetuned on HSSD images. (a) shows evaluation results on real image dataset from LAION [9], (b) shows results on images from HSSD dataset [13], and (c) shows results of placement predicted while evaluating tidying robot on Embodied Semantic Placement (eSP).

Here we present evaluation results on two image datasets: 1.) LAION-SP Val: 400 real images collected from LAION [9], 2.) HSSD Val: $18k$ images from unseen HSSD scenes [13]. First, we describe baselines used for evaluation:

**LLM + Detector**. In this baseline we leverage common-sense priors from LLMs to find target receptacles for a particular object and use a open-vocabulary detector, Detic [19], to localize the receptacle in the image. First, 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. Next, during evaluation we use Detic to localize the segmentation mask of all valid receptacles for a object category in an image.

**LLaVA [14] and GPT4V [25]**. To evaluate LLaVA and GPT4V on SP, given the input image we prompt it to output normalized bounding box coordinates to localize a placement area. Next, we convert the predicted normalized bounding box to a binary segmentation mask to use as the SP mask for downstream applications. Refer Appendix I for GPT4V and LLaVA prompts and sample predictions.

**Ours (HSSD)**. Variant of our CLIP-UNet model described in Sec. II-C trained only on data collected from HSSD scenes *i.e.* no pretraining on the LAION-SP dataset from Sec. II-A.

**Ours (LAION-SP→HSSD)**. Our CLIP-UNet model from Sec. II-C: first pretrained on the LAION-SP dataset and then finetuned on small image dataset from HSSD.

| | Method | LAION-SP Val | HSSD Val | | | |
|---|---|---|---|---|---|---|
| | | HP (↑) | HP (↑) | TrP (↑) | RSP (↑) | RSR (↑) |
| 1) | LLM + Detector | 21.5 | 29.8 | 10.1 | **41.0** | 38.2 |
| 2) | LLaVA | 4.9 | 6.8 | 0.0 | 26.3 | **43.4** |
| 3) | GPT4V | 9.4 | 8.3 | – | – | – |
| 4) | Ours (HSSD) | 20.1 | 23.0 | 16.2 | 26.6 | 36.5 |
| 5) | Ours (SP → HSSD) | **43.7** | **31.3** | **18.5** | 24.9 | 35.3 |

**TABLE I. SP evaluation on LAION-SP and HSSD validation splits.** We show evaluation results of our model (rows 4-5), Prior + Detector, and VLM baselines. HP denotes Human Preference, TrP denotes Target Precision, RSP denotes Receptacle Surface Precision, and RSR denotes Receptacle Surface Recall. We use ↑ to indicate that larger values are preferred.

**Results**. Tab. I reports results of evaluating methods on the LAION-SP and HSSD evaluation datasets. In our human preference study, our method (row 5) is favored the most by a large margin on real world images, and modestly in simulated images, when asked to rank predictions from all 5 baselines from Tab. I. This demonstrates the effectiveness of using web data for pretraining our CLIP-UNet model. In addition to human preferences, we also conduct quantitative evaluation using metrics from Appendix D. Our method outperforms a strong baseline that uses an LLM prior and object detector Detic (row 1) on target precision (TrP) by $8.4\%$, is comparable in the RSR metric, and performs worse on RSP metrics. CLIP-UNet (row 5) also outperforms both VLM baselines, *i.e.* LLaVA (row 2), significantly on TrP and achieves comparable performance on the RSP metric. After some preliminary analysis of results of the VLMs we find that, when tasked to output the placement location as language in addition to bounding box coordinates, these VLMs do a good job at giving reasonable responses but fail to precisely localize the output in the image space. More details in App. J. These results demonstrate the difficulty of SP prediction and highlight that there's still scope for improvements in general-purpose VLMs.

### B. Embodied Semantic Placement

In this section, we present the results of using SP models for the downstream application of building a tidying robot. Specifically, in this task, an agent is spawned at a random location in an indoor environment and is tasked with placing an instance of a target object category at a semantically meaningful location. We call this task Embodied Semantic Placement (eSP). For our experiments, we use Hello Robot's Stretch robot [1] with the full action space as defined in [26]. To perform the task with only robot observations and SP mask predictions from a SP model at each frame, we use a two-stage modular policy consisting of "navigation" and "place" policies described in Appendix G. Tab. VI in Appendix G presents the results of evaluating the eSP policy using SP mask predictions from LLM+Detector, LLaVa and our CLIP-UNet (LAION-SP→HSSD) model on HSSD val split. We do not evaluate GPT4V on eSP task due to API limitations, eSP policy evaluation requires running inference using GPT4V after each robot action which amounts to a total of $\sim 53k$ frames for full evaluation. We find our CLIP-UNet eSP policy achieves a $12.5\%$ success on the eSP task across 10 indoor environments, outperforming LLaVa and LLM+Detector eSP baselines by $2-3.5\%$ on task success. Refer Appendix G for more details.

## IV. Conclusion

We propose Semantic Placement (SP), a novel task where, given an image and object type, a vision system must predict a binary mask highlighting semantically-meaningful regions in an image where that object could be placed. Learning to predict the invisible is hard. We address this challenge by making visible objects invisible: we start with an image of an object in context and remove that object from the image via inpainting. This automated data curation pipeline, leveraging inpainting and object detection models, enables us to supervise an end-to-end SP prediction model, CLIP-UNet, using real-world data. Our CLIP-UNet produces SP predictions which generalize well to the real-world, are favored more by humans, and enable downstream robot execution.

## REFERENCES


[1] C. C. Kemp, A. Edsinger, H. M. Clever, and B. Matulevich, "The Design of Stretch: A Compact, Lightweight Mobile Manipulator for Indoor Human Environments," in *2022 International Conference on Robotics and Automation, ICRA 2022, Philadelphia, PA, USA, May 23-27, 2022*, pp. 3150–3157, IEEE, 2022.

[2] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, *et al.*, "Learning transferable visual models from natural language supervision," in *ICML*, 2021.

[3] G. Ilharco, M. Wortsman, R. Wightman, C. Gordon, N. Carlini, R. Taori, A. Dave, V. Shankar, H. Namkoong, J. Miller, H. Hajishirzi, A. Farhadi, and L. Schmidt, "Openclip," 2021.

[4] X. Zhai, B. Mustafa, A. Kolesnikov, and L. Beyer, "Sigmoid loss for language image pre-training," in *ICCV*, 2023.

[5] Q. Sun, Y. Fang, L. Y. Wu, X. Wang, and Y. Cao, "Eva-clip: Improved training techniques for clip at scale," *ArXiv*, vol. abs/2303.15389, 2023.

[6] X. Li, Z. Wang, and C. Xie, "An inverse scaling law for clip training," *ArXiv*, vol. abs/2305.07017, 2023.

[7] R. Suvorov, E. Logacheva, A. Mashikhin, A. Remizova, A. Ashukha, A. Silvestrov, N. Kong, H. Goka, K. Park, and V. Lempitsky, "Resolution-robust large mask inpainting with fourier convolutions," in *CVPR*, 2021.

[8] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," 2021.

[9] C. Schuhmann, R. Beaumont, R. Vencu, C. Gordon, R. Wightman, M. Cherti, T. Coombes, A. Katta, C. Mullis, M. Wortsman, *et al.*, "Laion-5b: An open large-scale dataset for training next generation image-text models," in *NeurIPS*, 2022.

[10] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *MICCAI*, 2015.

[11] D. K. Misra, A. Bennett, V. Blukis, E. Niklasson, M. Shatkhin, and Y. Artzi, "Mapping instructions to actions in 3d environments with visual goal prediction," in *CoRL*, 2018.

[12] M. Shridhar, L. Manuelli, and D. Fox, "Cliport: What and where pathways for robotic manipulation," in *Proceedings of the 5th Conference on Robot Learning (CoRL)*, 2021.

[13] M. Khanna*, Y. Mao*, H. Jiang, S. Haresh, B. Shacklett, D. Batra, A. Clegg, E. Undersander, A. X. Chang, and M. Savva, "Habitat Synthetic Scenes Dataset (HSSD-200): An Analysis of 3D Scene Scale and Realism Tradeoffs for ObjectGoal Navigation," *arXiv preprint*, 2023.

[14] H. Liu, C. Li, Y. Li, and Y. J. Lee, "Improved baselines with visual instruction tuning," 2023.

[15] OpenAI, "Gpt-4 technical report," *ArXiv*, 2023.

[16] M. Savva, A. Kadian, O. Maksymets, Y. Zhao, E. Wijmans, B. Jain, J. Straub, J. Liu, V. Koltun, J. Malik, *et al.*, "Habitat: A platform for embodied AI research," in *ICCV*, 2019.

[17] A. Szot, A. Clegg, E. Undersander, E. Wijmans, Y. Zhao, J. Turner, N. Maestre, M. Mukadam, D. S. Chaplot, O. Maksymets, A. Gokaslan, V. Vondrus, S. Dharur, F. Meier, W. Galuba, A. Chang, Z. Kira, V. Koltun, J. Malik, M. Savva, and D. Batra, "Habitat 2.0: Training home assistants to rearrange their habitat," in *NeurIPS*, 2021.

[18] X. Puig, E. Undersander, A. Szot, M. D. Cote, R. Partsey, J. Yang, R. Desai, A. W. Clegg, M. Hlavac, T. Min, T. Gervet, V. Vondruš, V.-P. Berges, J. Turner, O. Maksymets, Z. Kira, M. Kalakrishnan, J. Malik, D. S. Chaplot, U. Jain, D. Batra, A. Rai, and R. Mottaghi, "Habitat 3.0: A co-habitat for humans, avatars and robots," 2023.

[19] X. Zhou, R. Girdhar, A. Joulin, P. Krähenbühl, and I. Misra, "Detecting twenty-thousand classes using image-level supervision," in *ECCV*, 2022.

[20] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, P. Dollár, and R. Girshick, "Segment anything," in *ICCV*, 2023.

[21] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *CVPR*, 2022.

[22] C. Meng, Y. He, Y. Song, J. Song, J. Wu, J.-Y. Zhu, and S. Ermon, "SDEdit: Guided image synthesis and editing with stochastic differential equations," in *ICLR*, 2022.

[23] Y. Ye, X. Li, A. Gupta, S. De Mello, S. Birchfield, J. Song, S. Tulsiani, and S. Liu, "Affordance diffusion: Synthesizing hand-object interactions," in *CVPR*, 2023.

[24] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *CVPR*, 2016.

[25] OpenAI, "Gpt-4 technical report," 2023.

[26] S. Yenamandra, A. Ramachandran, K. Yadav, A. Wang, M. Khanna, T. Gervet, T.-Y. Yang, V. Jain, A. W. Clegg, J. Turner, *et al.*, "Homerobot: Open-vocabulary mobile manipulation," in *arXiv preprint arXiv:2306.11565*, 2023.

[27] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, *et al.*, "Segment anything," in *ICCV*, 2023.

[28] B. Yamauchi, "A frontier-based approach for autonomous exploration," in *Proceedings 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA'97. 'Towards New Computational Principles for Robotics and Automation'*, 1997.

[29] M. Deitke, W. Han, A. Herrasti, A. Kembhavi, E. Kolve, R. Mottaghi, J. Salvador, D. Schwenk, E. VanderBilt, M. Wallingford, L. Weihs, M. Yatskar, and A. Farhadi, "RoboTHOR: An Open Simulation-to-Real Embodied AI Platform," in *CVPR*, 2020.

[30] M. Deitke, R. Hendrix, A. Farhadi, K. Ehsani, and A. Kembhavi, "Phone2proc: Bringing robust robots into our chaotic world," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9665–9675, 2023.

[31] C. H. Sudre, W. Li, T. Vercauteren, S. Ourselin, and M. Jorge Cardoso, "Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations," in *MICCAI Workshop*, 2017.

[32] R. Girshick, I. Radosavovic, G. Gkioxari, P. Dollár, and K. He, "Detectron." https://github.com/facebookresearch/detectron, 2018.

[33] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *ECCV*, 2014.

[34] G. Sarch, Z. Fang, A. W. Harley, P. Schydlo, M. J. Tarr, S. Gupta, and K. Fragkiadaki, "Tidee: Tidying up novel rooms using visuo-semantic commonsense priors," in *European Conference on Computer Vision*, 2022.

[35] Y. Kant, A. Ramachandran, S. Yenamandra, I. Gilitschenski, D. Batra, A. Szot, and H. Agrawal, "Housekeep: Tidying virtual households using commonsense reasoning," in *ECCV*, 2022.

[36] J. Wu, R. Antonova, A. Kan, M. Lepert, A. Zeng, S. Song, J. Bohg, S. Rusinkiewicz, and T. Funkhouser, "Tidybot: Personalized robot assistance with large language models," *Autonomous Robots*, 2023.

[37] M. Minderer, A. Gritsenko, A. Stone, M. Neumann, D. Weissenborn, A. Dosovitskiy, A. Mahendran, A. Arnab, M. Dehghani, Z. Shen, *et al.*, "Simple open-vocabulary object detection," in *ECCV*, 2022.

[38] S. Liu, Z. Zeng, T. Ren, F. Li, H. Zhang, J. Yang, C. Li, J. Yang, H. Su, J. Zhu, *et al.*, "Grounding dino: Marrying dino with grounded pre-training for open-set object detection," *arXiv preprint arXiv:2303.05499*, 2023.

[39] D. S. Chaplot, D. Gandhi, A. Gupta, and R. Salakhutdinov, "Object goal navigation using goal-oriented semantic exploration," in *NeurIPS*, 2020.

**Fig. 4. Qualitative Examples of Generated Images**. We present three examples of `Cushion`, `Laptop`, and `Potted Plants`, which include raw images queried from LAION (left), identified objects of interest and their segmentation masks obtained from SAM (middle), and the result images after *Inpainting*, *Flitering*, and *Quality Enhancement* steps (right). For clarity, we have magnified the inpainted regions, highlighted in green dotted boxes.

## APPENDIX

### A. Data Generation

| Category | Potted Plant | Lamp | Cushion | Vase | Trash Can |
|---|---|---|---|---|---|
| # Images | 207,366 | 320,922 | 323,541 | 417,591 | 13,353 |

| Category | Toaster | Table Lamp | Alarm Clock | Laptop |
|---|---|---|---|---|
| # Images | 23,928 | 5,559 | 14,496 | 2,430 |

**TABLE II.** Number of Images per Category in LAION-SP Dataset.

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

**(A) Query Image**. As mentioned in Sec. II-A, 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 [27] 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 [27] to predict a segmentation mask. Among the 3 masks predicted by SAM [27], we choose the one with the highest confidence for downstream inpainting.

In Tab. II, we show the number of generated images per object category. Fig. 4 showcases three qualitative examples generated by our dataset generation pipeline, including `Cushion`, `Laptop`, and `Plotted Plant`.

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.

### B. 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 $(r, \theta)$ 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 [28] 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.

### C. Real Evaluation Dataset

For our experiments in Sec. III, we use a real image dataset comprising 400 images, collected from the LAION dataset [9] and 2 real-world environments from [29], [30]. 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 [29], [30].

We train our CLIP-UNet model in two stages. First, we pretrain our model using the LAION-SP dataset generated in Sec. II-A, containing 1.3M images across 9 categories for 10 epochs using *dice loss* [31]. During pretraining, in addition to diffusion model augmented images, we also use common data augmentations, such as gaussian blurring, additive gaussian noise, horizontal flipping, and color jitter to mitigate inpainting artifacts. Next, we finetune the LAION-SP pretrained model using a small, high-quality, dataset generated

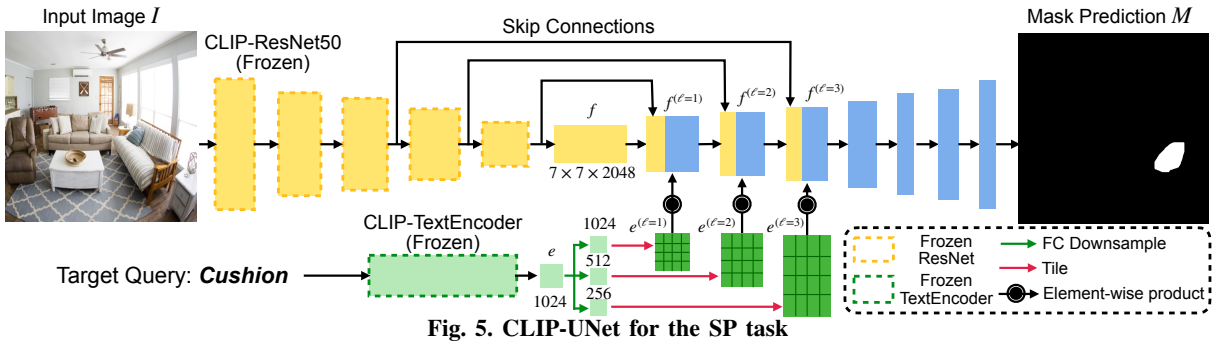


**Fig. 5. CLIP-UNet for the SP task**

**Fig. 6. IoU *v.s.* IoP.** Top left: a hypothetical ground-truth (GT) SP region for objects of type "book". Top right & bottom left: two possible SP predictions. Both predicted regions are high-quality and should be considered true-positives. The IoU for these predictions is, however, $< 0.5$ as the IoU normalizes by the large GT region. The IoP, however, only normalizes by the predicted mask's size and thus is equal to 1 for both predicted regions.

using synthetic HSSD scenes [13], [16], [17] mentioned in Sec. II-B. As the HSSD image dataset is generated using a simulator we can manipulate the scene to render images with/without object images without introducing any artifacts that models can latch on to. This two-stage training improves performance of our CLIP-UNet model and enables better generalization to both real and synthetic images as shown in Sec. III.

### *D. Evaluation Metrics*

In this section, we propose metrics for evaluating SP prediction performance. Before defining these metrics, we will begin by defining what we mean by true/false positive (TP/FP) and true/false negative (TN/FN) SP predictions.

**Preliminaries.** Consider an image $I$, an object type query $q$, and an (exhaustive) set of $\{0,1\}$-valued ground-truth disjoint regions $r_1, ..., r_K \in \{0,1\}^{H \times W}$ describing the locations where objects of type $q$ can be placed in the image $I$. Let the model produced region predictions be denoted by $\widehat{r}_1, ..., \widehat{r}_L \in \{0,1\}^{H \times W}$. Intuitively, we would like a predicted region $\widehat{r}_j$ to be considered a true positive (TP), if it "overlaps sufficiently" with some GT region $r_i$. Measuring region overlap is commonly achieved, *e.g.* in the semantic segmentation and object detection literature [32], [33], [27], using the intersection-over-union (IoU) metric, $\text{IoU}(r, r') = r \cdot r' / (r \cdot r + r' \cdot r' - r \cdot r')$ where $\cdot$ denotes the usual dot product. The IoU works well when one wishes to enforce that two regions overlap *exactly*; for SP prediction, however, requiring exact overlap is too restrictive as it normalizes by too large of a region, see Fig. 6. Instead we use the intersection-over-prediction $\text{IoP}(r, r') = r \cdot r' / (r' \cdot r')$ which normalizes only by the size of the predicted region $r'$. That is, we say that $\widehat{r}_j$ is a TP if there exists some $r_i$ such that $\text{IoP}(\widehat{r}_j, r_i) \geq T$ where $T \in [0,1]$ is some threshold value (for us, $T = 0.5$). We say that $\widehat{r}_j$ is a FP if there is no $r_i$ with $\text{IoP}(\widehat{r}_j, r_i) \geq T$. Importantly: TPs are counted with respect to the ground truth region $r_i$ while FPs are counted with respect to the predicted region $\widehat{r}_j$. This means that that if the model predicts multiple regions $\widehat{r}_j$ which all correspond to a single $r_i$, then these multiple regions will be counted as only a single TP. Additionally, number of FN equal to number of GT regions $r_i$ not covered by any predicted region $\widehat{r}_j$.

**Precision and recall.** Given the above, we can now define the usual recall and precision metrics for an image $I$ as $\text{Precision}(I) = \frac{\#TP}{\#TP + \#FP}$ and $\text{Recall}(I) = \frac{\#TP}{\#TP + \#FN}$. When reporting metrics on our evaluation sets, we report the average precision and recall over all images. If an image $I$ has no GT masks, then $\text{Recall}(I)$ is not well-defined and so we do not include such images when computing the average. We compute these metrics only on HSSD dataset as these require access to accurate GT region annotations.

**Receptacle priors.** One important facet of SP prediction is an understanding of the relationship between receptacle types and the objects that are typically placed upon them. For instance, you will almost always find a plunger on the floor and not on a dining table. Indeed, it is exactly these types of receptacle relationships that some previous work, *e.g.* [34], [35], [36], have focused upon. In order to measure the model's ability to encode such priors, we introduce the receptacle surface precision (RSP) and receptacle surface recall (RSR) metrics. To compute these metrics, we first, for each object type query $q$, curate a collection of receptacle types that such an object is commonly found upon (see Sec Appendix E for more details). We then, for each image $I$ and object type query $q$, assume we have access to segmentation masks $s_1, ..., s_K$ of receptacles upon which $q$ is commonly found. Moreover, as large parts of each receptacle mask will correspond to unplaceable areas (*e.g.* the legs of a couch) we further assume that each $s_i$ corresponds only to the areas of the receptacle that are "placeable", *i.e.* have a surface normal that is pointing (approximately) upward. In practice, computing

| 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 III.** Mapping of receptacles for each object category.

the receptacle masks can often be done automatically by leveraging simulated environments in which object categories and geometry are known (e.g. HSSD), or by using open-vocabulary object detectors and depth maps for real world images. As the results from open-vocabulary detectors and depth maps are noisy we only report these metrics on HSSD image dataset where we have access to ground truth. We can then compute the RSR and RSP just as above by replacing the GT regions $r_i$ with the surface grounded receptacle segmentation masks $s_i$.

**Target Precision (TrP)**. To quantify precision of SP models at localizing possible ground truth placements we compute the Target Precision (TrP) metric. To compute TrP, we programatically compute the GT placement masks for an object category from HSSD scenes and use these as GT regions for computing the precision metrics.

**Human preference (HP).** To understand how humans judge our baselines outputs, we require human annotators to rank each model's SP predictions from most preferred to least preferred when shown predictions from 5 models described in Sec. III. We then report the % of time that these annotators rank each model's predictions as the best, *i.e.* ranked above all others, among 5 SP predictions. Further details in App. E.

### *E. Receptacle Priors*

To compute receptacle precision and recall metrics, we use the receptacles shown in Tab. III 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.

### *F. Additional Semantic Placement Experiments*

| | Method | HSSD VAL | | |
|---|---|---|---|---|
| | | TrP (↑) | RSP (↑) | RSR (↑) |
| 1) | Ours (LAION-SP) | 10.1 | 23.7 | 26.3 |
| 2) | Ours (HSSD) | 16.2 | **26.6** | **36.5** |
| 3) | Ours (LAION-SP → HSSD) | **18.5** | 24.9 | 35.3 |

**TABLE IV. LAION-SP pretraining ablations.** We show the evaluation results by training our CLIP-UNet on different datasets.

| | Baseline | Success (↑) |
|---|---|---|
| 1) | LLM + Detector | 10.5% |
| 2) | LLaVA | 9.0% |
| 3) | Ours (LAION-SP → HSSD) | **12.5%** |

**TABLE VI. Embodied Semantic Place (eSP) evaluation performance on HSSD VAL split**. We evaluate each SP model from Sec. III-B using a modular eSP policy with same hyperparameters.

**Effectiveness of Pretraining on the SP Dataset**. Tab. IV shows results varying the CLIP-UNet training dataset. First, evaluating the model trained on the LAION-SP dataset zero-shot on HSSD (row 1) results in 10.1% TrP, 23.7% RSP, and 26.3% RSR. This suggests the LAION-SP pretrained model is, in general, good at identifying correct receptacle surfaces in HSSD but does not, as shown by low TrP numbers, perform very well in precisely localizing one of the ground truth object placements. In contrast to training on the LAION-SP dataset, if we just train from scratch on HSSD images (row 2 vs 1) we achieve a +6.1 absolute improvement on TrP, +2.9% on RSP, and +10.2% on RSR. However, with small amounts of finetuning of the LAION-SP pretrained model on HSSD dataset (row 3 vs 2), we obtain our best performing model which obtains a further absolute improvement of +2.3% on TrP with comparable performance on RSP and RSR. In addition, as shown in human preference numbers in Tab. I (row 4 vs 5), pretraining on LAION-SP and finetuning on HSSD leads to overall better generalization to both sim and real images.These results effectively demonstrate that pretraining on the LAION-SP dataset enables better generalization. See Fig. 3 for qualitative examples of our HSSD-finetuned model's predictions.

**Open-Vocab Object Detector Ablation.** Tab. V 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 [37], and GroundedSAM [20], [38]. 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 V. Ablations of object detectors for prior based baselines.**

### *G. Embodied Evaluation*

In this section, we present the results of using our CLIP-UNet (LAION-SP → HSSD) model for the downstream application of building a tidying robot. Specifically, in this task, an agent is spawned at a random location in an indoor environment and is tasked with placing an instance of a target object category at a semantically meaningful location. We call this task Embodied Semantic Placement (eSP). For our experiments, we use Hello Robot's Stretch robot [1] with the full action space as defined in [26]. Specifically, the observation space, shown in the Fig. 7, includes RGB+Depth images from the

robot's head camera, camera pose, arm joint and gripper states, and 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.

*1) Evaluation Dataset:* For eSP evaluation, we create a dataset consiting of 106 episodes using HSSD scenes [13], each specified by an agent's starting pose and a target object category. These episodes span 8 object categories across 10 indoor environments. An episode is successful if the agent successfully places the object on one of the semantically valid receptacle (*e.g.* cushion on a bed or couch).

*2) Embodied Semantic Placement Policy:* In this section, we detail the Embodied Semantic Placement Policy used in Appendix G 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 [26]. 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 [39], 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 [39], our navigation policy employs frontier exploration [28], 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 [26]. 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 [26]. 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 [26]. 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.

*3) Results:* Tab. VI presents the results of evaluating the eSP policy using SP mask predictions LLM+Detector, LLaVa and our CLIP-UNet (LAION-SP→HSSD) model on HSSD val split. We do not evaluate GPT4V on eSP task due to API limitations, eSP policy evaluation requires running inference using GPT4V after each robot action which amounts to a total of $\sim 53k$ frames for full evaluation. We find our CLIP-UNet eSP policy achieves a 12.5% success on the eSP task across 10 indoor environments, outperforming LLaVa and LLM+Detector eSP baselines by $2 - 3.5\%$ on task success. We observe that our CLIP-UNet eSP agent can effectively reason about appropriate object placements in these settings. For example, in a living room scenario near a couch, the agent determines that a book should be placed on the coffee table, as shown in Fig. 1(c). For qualitative videos and additional examples, please refer the supplementary.

### *H. 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. I) and rank them from most to least preferred. We conduct two types of the user study: one with 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. I of the main paper.

### *I. 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,

| 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 VII. Prior + Detector Baseline.** Mapping of receptacles from a LLM for each object category.

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. VII. 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 [25]**. 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.

### J. 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.

### K. 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.

*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.

*2) Embodied Semantic Placement:* The majority of eSP evaluation failures come from the navigation and place planner, which include:

**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.

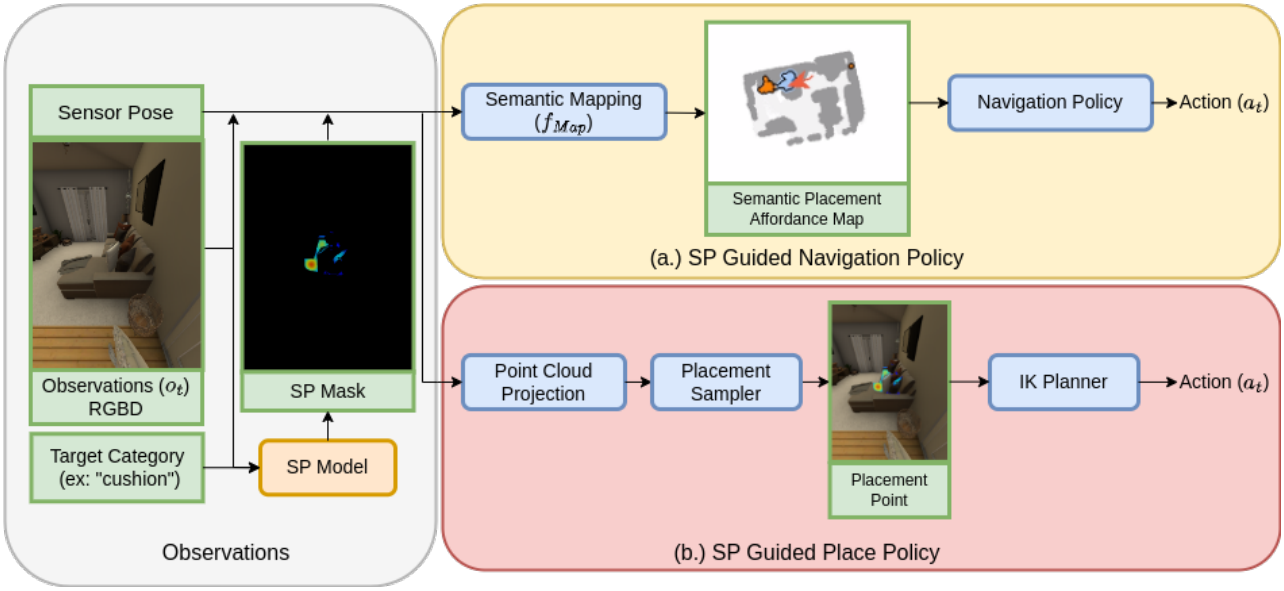


**Fig. 7. Embodied Evaluation Pipeline.** We build a two-stage modular policy consisting of: 1.) SP Guided Navigation Policy: Uses frontier exploration and semantic placement affordance 2D map to navigate to placement area, 2.) SP Guided Place Policy: Uses predicted SP mask, projects it onto a pointcloud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

**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.

### *L. 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.

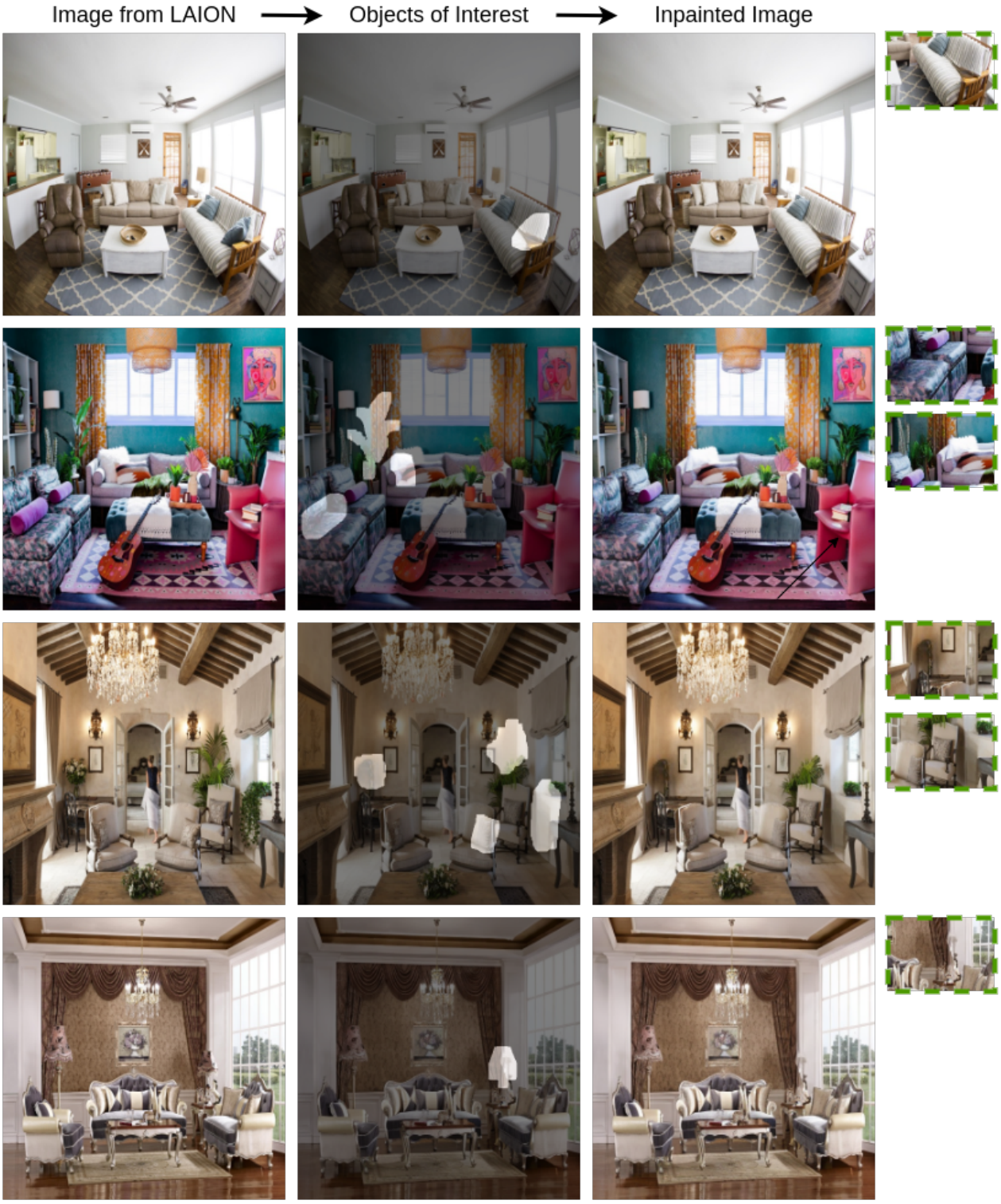


**Fig. 8.** Qualitative examples from SP train data generated using our proposed automatic data generation pipeline.

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

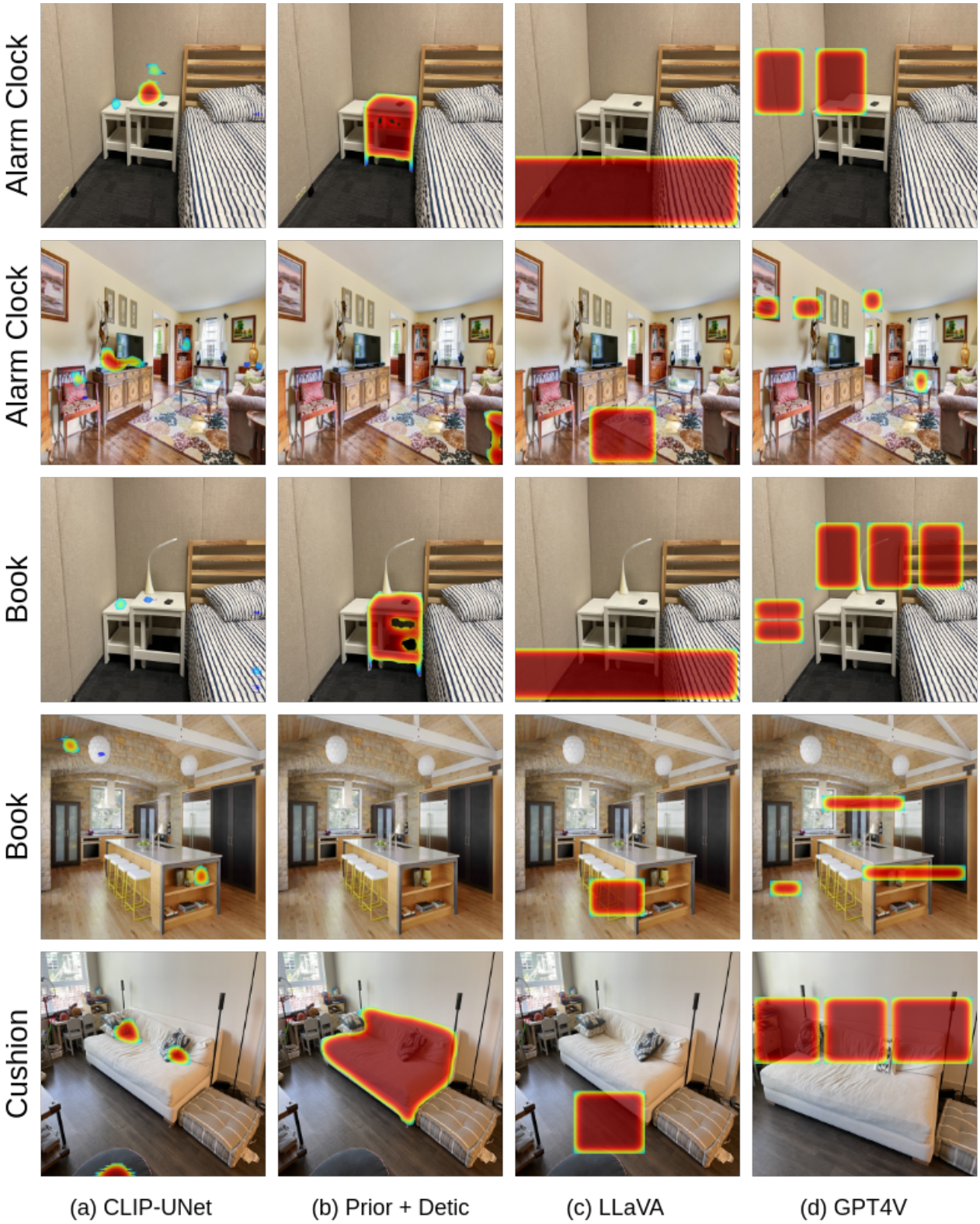


**Fig. 10.** Qualitative examples of SP masks predicted by all the baselines on SP Real val dataset

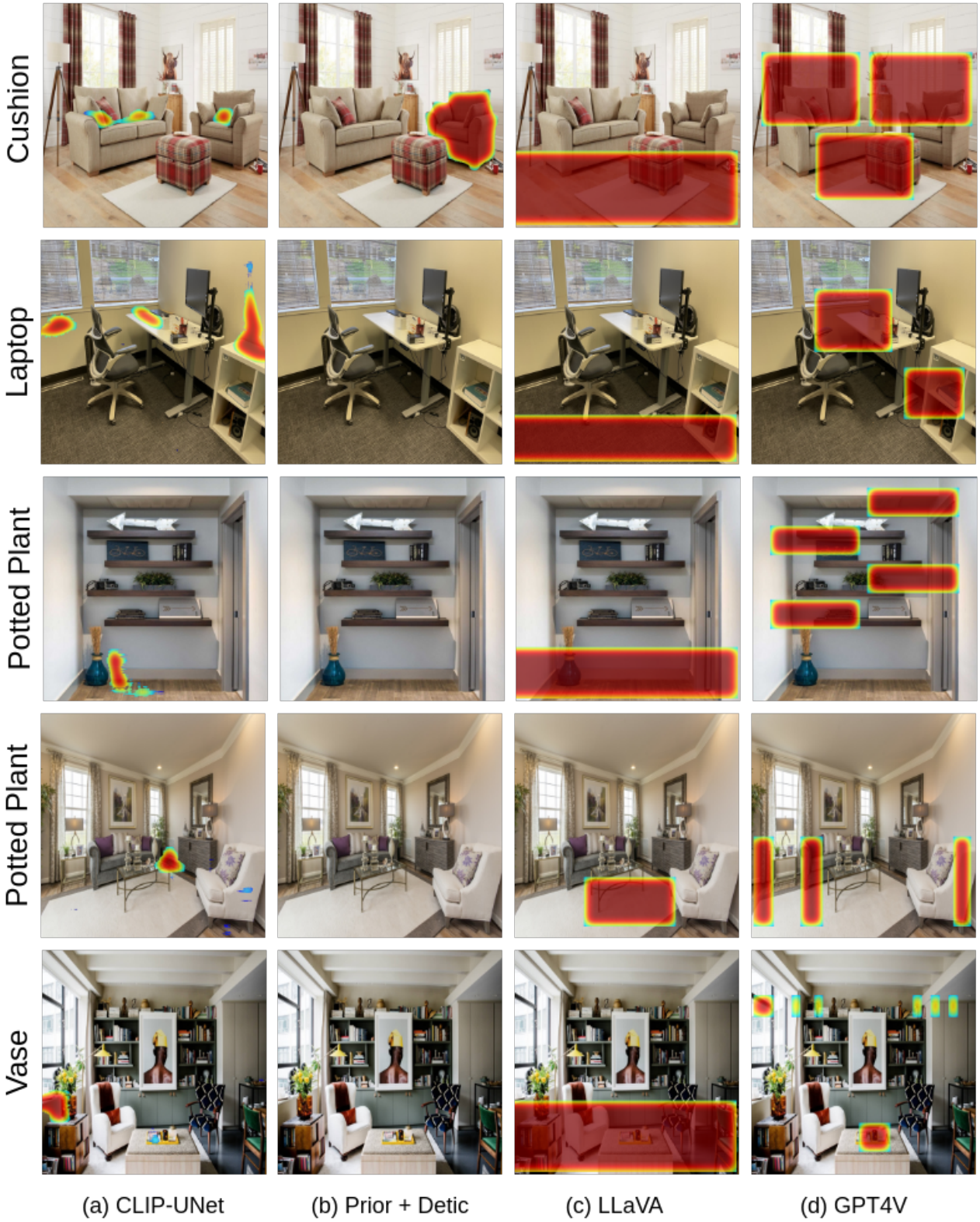


**Fig. 11.** Qualitative examples of SP masks predicted by all the baselines on SP Real val dataset

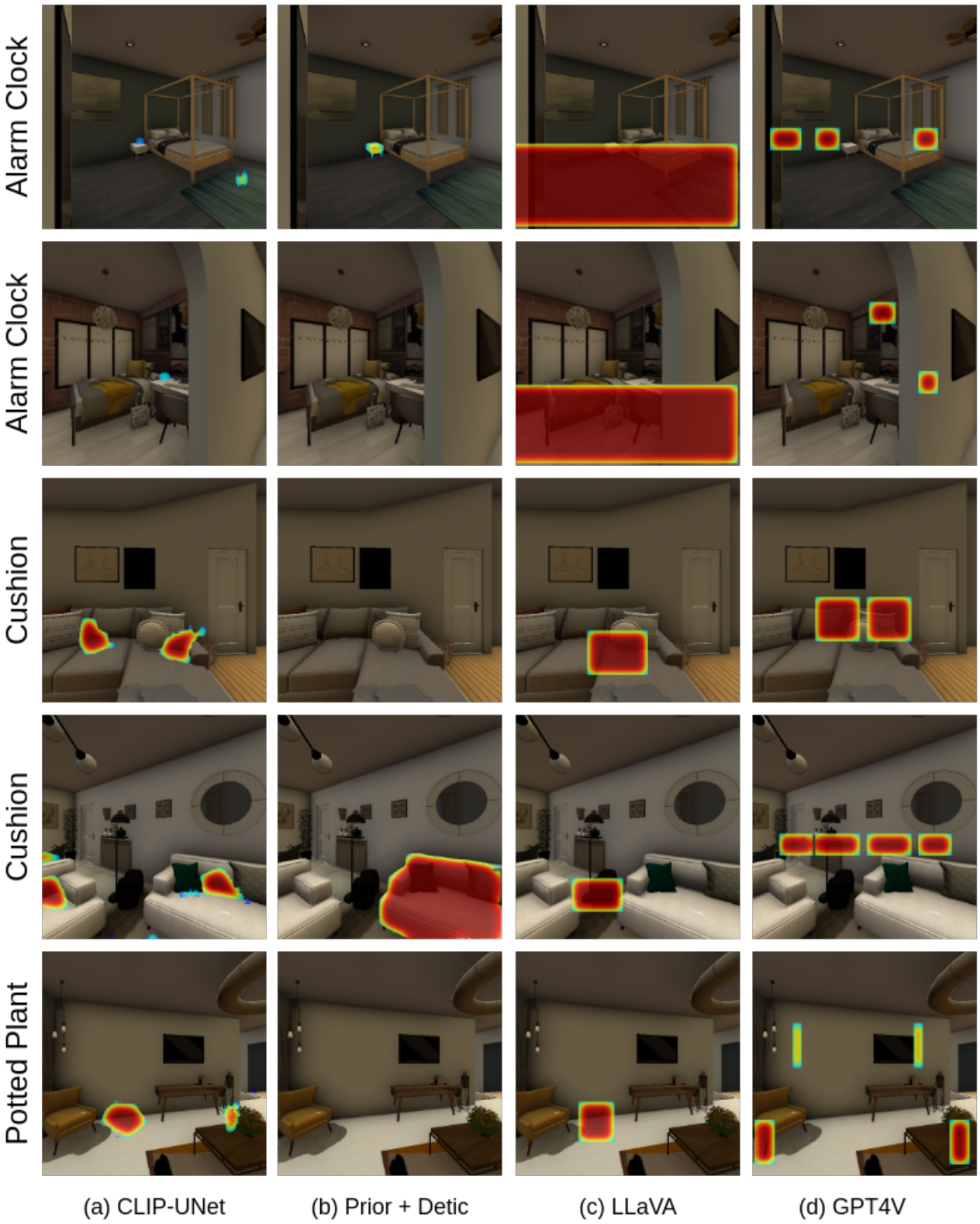


**Fig. 12.** Qualitative examples of SP masks predicted by all the baselines on SP HSSD val dataset

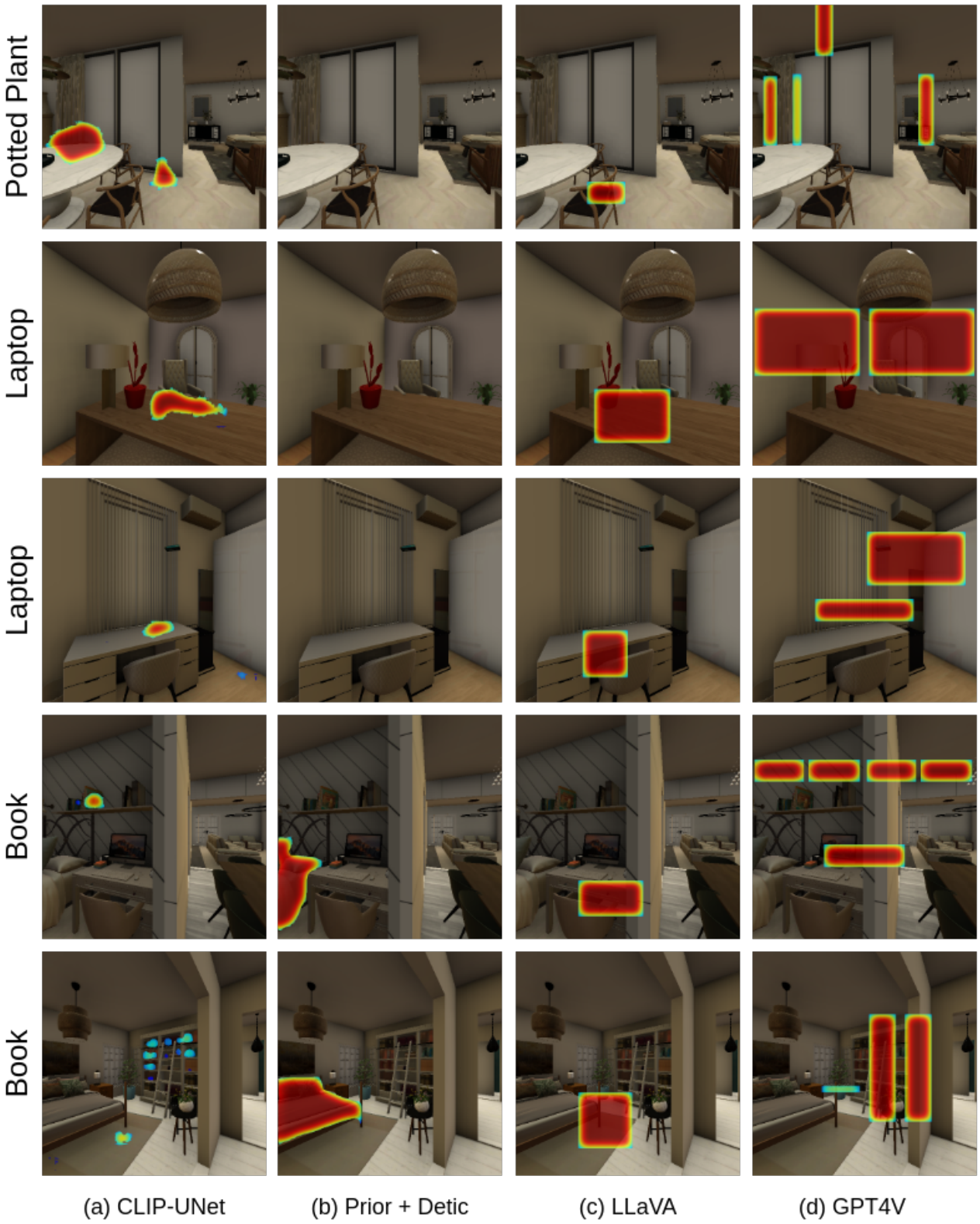


**Fig. 13.** Qualitative examples of SP masks predicted by all the baselines on SP HSSD val dataset

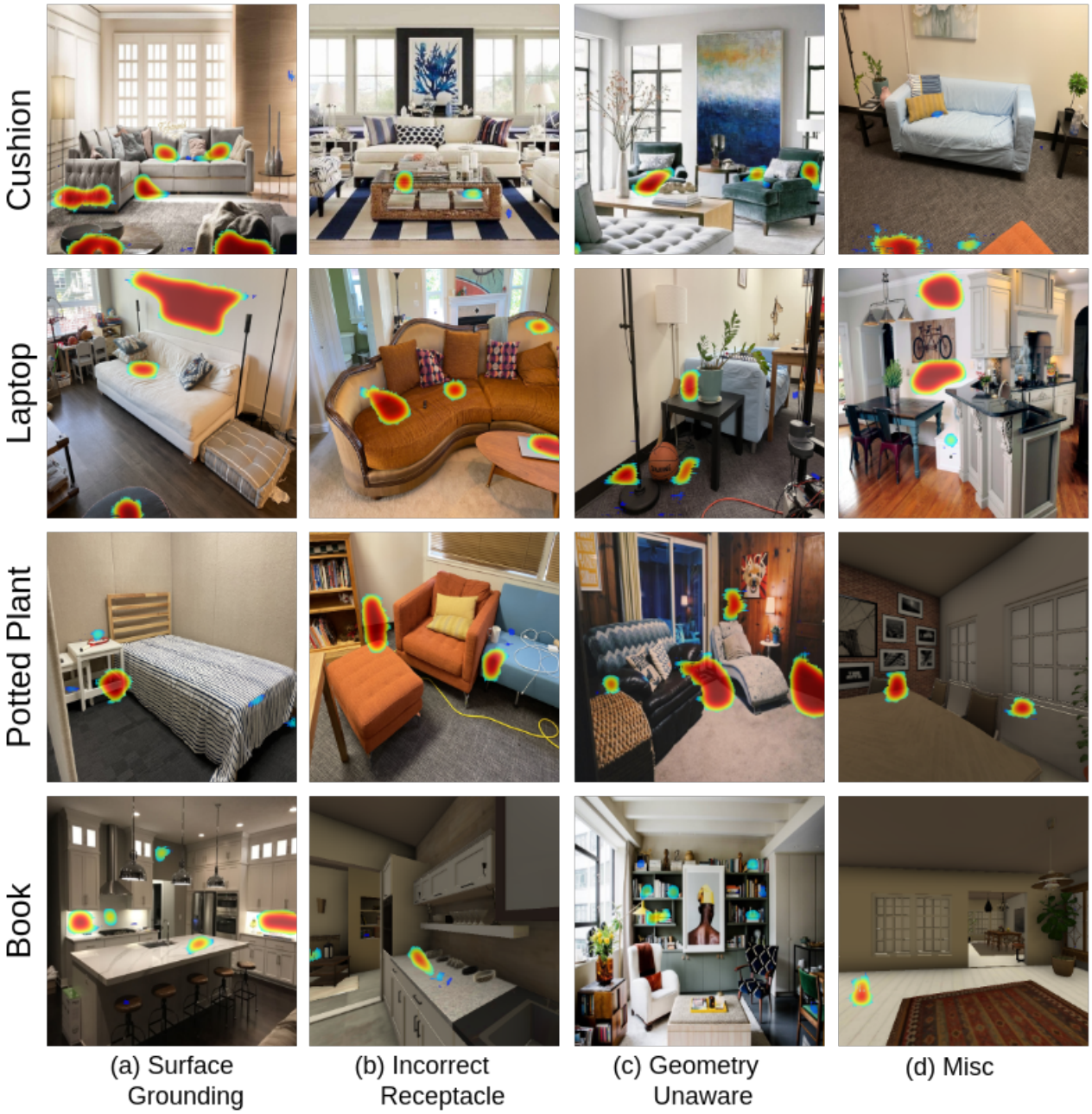


**Fig. 14.** Qualitative examples of failure modes of SP mask prediction by our approach