Contents

A. Depth-based Mapping Details

Action failure checking. For both learnable and heuristic explorers, actions may fail. For map based explorers, an obstacle might be below the field of view and hence not captured as occupied space. For learnable explorers, the output policy may heavily favor a failed action (e.g., MOVEAHEAD), which could be executed repeatedly.

To improve the action success of our CoWs, we employ a simple strategy. We compute the absolute difference between depth channels in the observations $\Delta_{i,i+1} = |D_i - D_{i+1}|$. We then compute the mean $\mu(\Delta_{i,i+1})$ and standard deviation $\sigma(\Delta_{i,i+1})$ as representative statistics. These quantities have interpretable meaning in meters. Since actions should move the agent forward by approximately a fixed distance or rotation, we can then set reasonable thresholds for μ, σ below which we can be confident actions failed. In our studies, we set these to $\mu = 0.1$m, $\sigma = 0.1$m.

Note, that in modern robot navigation systems, on-board pose estimation and bumper sensors for obstacle avoidance are nearly ubiquitous. Hence, it is possible to implement action failure checking beyond visual inputs, even though in this paper *we only consider vision-based failure checking*.

Formalization of registering new depth observations. Recall, a CoW constructs a top-down map based on egocentric depth observation and approximated poses deltas. We provide a formalization of this process.

To create this map, we first estimate the pose of the CoW's coordinate frame. Let C_i be the current local coordinate frame of a CoW at timestep i. Let W denote a world frame. We would like to align observations from each local frame to W to keep a single, consistent map. When a CoW is initialized, we set ${}^{W}\hat{T}_{C_0} = I$, where I is the identity SE(3) transform and ${}^{W}\hat{T}_{C_0}$ is a transform, which aligns frame C_0 to W. Since a CoW knows the intended consequences of its actions (e.g., MOVEFORWARD should result in a translation of 0.25m), each action can be represented as a delta transform, which models an action transition. By concatenating these transforms over time, we get pose estimates. To estimate pose at timestep $t+1$, we compute ${}^{W}\hat{T}_{C_{t+1}} = {}^{W}\hat{T}_{C_t} \cdot {}^{C_t}\hat{T}_{C_{t+1}}$. Due to sensor-noise and action failures, these estimates are sensitive to pose drift. We use the current estimated pose ${}^{W}T_{C_t}$, camera intrinsic matrix K, current depth observation D_t, standard back-projection of D_t, and pose concatenation to register new observations in frame W.

Since navigation is mostly concerned with free v.s. occupied space, we do not keep the whole 3D map. Rather we project the map on the ground-plane using the known agent height and down gravity direction. Points already near the floor are considered free space, while other points are considered occupied as shown in. Additionally, we discretize the map, which additionally helps deal with sensor noise.

B. Exploration Details

Learnable exploration. We use the AllenAct [76] framework to conduct exploration agent training. Our DD-PPO hyperparameters are in Tab. 5. We employ a simple state visitation count based reward. An agent receives a reward of 0.1 for visiting a previously unvisited voxel location (at 0.125m resolution), and a step penalty of -0.01. We train two agents, one on ROBOTHOR and the other on HABITAT MP3D train sets, which are disjoint from the downstream validation sets we use for testing.

Hyperparameter	Value
Discount factor (γ)	0.99
GAE parameter (λ)	0.95
Clipping parameter (ϵ)	0.1
Value loss coefficient	0.5
Entropy loss coefficient	0.01
LR	3e-4
Optimizer	Adam
Training steps	60M
Steps per rollout	500

Table 5. **DD-PPO hyperparameters.** Used for training exploration agents in both HABITAT and ROBOTHOR.

C. Localization Details

Here we provide a review of the CLIP model [61], the concept of prompt ensembling as it relates to CLIP, the gradient saliency method introduced by Chefer *et al.* [13], MDETR [36], and OWL-ViT [49].

CLIP overview. Recent open-vocabulary models for zero-shot image classification include CLIP [61], ALIGN [35], and BASIC [58]. These methods pre-train contrastively on image-caption pairs from the internet. In the case of the original CLIP model, on 400M pairs. The key insight is that the image representation—extracted by a vision tower—and the caption representation—extracted by a text tower—should be similar. Hence, the contrastive objects encourages image and text representations for a positive pairs to be similar and for these representations to be dissimilar from other images and captions.

More formally, CLIP jointly trains two encoders: a visual encoder f_θ and a language encoder f_λ. Given a dataset of image-text pairs $\mathcal{D} = \{..., (I_k, t_k), (I_{k+1}, t_{k+1}), ...\}$, which can be generated at massive scale by leveraging internet data. CLIP employs standard mini-batch style training. Given a batch of size n, with n image-text pairs, the current functions f_θ and f_λ are used to featurize the data, yielding *L-2 normalized* embeddings for the batch: $\{(z_0^I, z_0^t), (z_1^I, z_1^t), ...(z_n^I, z_n^t), \}$. It is then possible to define a symmetric contrastive loss of the following form, for each sample in the batch:

$$\mathcal{L}(z_i^I, z_i^t) = \frac{1}{2}\left(\frac{z_i^I \cdot z_i^t}{\sum_{j \neq i} z_i^I \cdot z_j^t} + \frac{z_i^I \cdot z_i^t}{\sum_{j \neq i} z_j^I \cdot z_i^t}\right). \quad (1)$$

By minimizing this loss, the representation for images and their corresponding captions are pushed closer together, while these representations are pushed farther from other images and text features in the batch, which are assumed to contain dissimilar concepts.

After training, these models can be thought to match images and captions. Given a set of captions identifying concepts (e.g., "a photo of an apple.", "a photo of an orange", etc.) it is possible to construct a classification head by extracting text features, via f_λ, and L-2 normalizing each of them. Now given an image, say of an apple, we can extract a visual feature, via f_θ, and again L-2 normalize. By dotting the visual feature with the text features, we can find the highest similarity score amongst the text captions and assign the image the corresponding label (e.g., "apple" from the caption "a photo of an apple."). Because this downstream classifier has not been trained to specifically classify images (say apples), it is considered a zero-shot classifier.

CLIP prompt ensembling. Radford *et al.* [61] find a simple strategy to boost performance of a CLIP zero-shot classifier. Instead of using one prompt for a class (e.g., "a photo of a apple."), they instead compute many text features for a class (e.g., "a photo of a apple.", "a blurry photo of a apple.", etc.). By simply averaging all these text features, they find performance improves. For CLIP-Ref., CLIP-Patch, and CLIP-Grad. strategies we similarly use prompt ensembling using the set of 80 prompt templates used for ImageNet-1k evaluation.[3] We present a prompt ablation in Appx. G.

Grad-CAM [70] and Chefer *et al.* [13] overview.

We begin by reviewing the Grad-CAM formulation, from which many gradient-based interpretability methods draw inspiration. Given a target class c, an input image x and a model f_θ, Grad-CAM produces a localization map L_c, capturing the importance of each pixel for the image to be classified as the target class. For standard convolutional neural networks, neuron importance weights α_k^c are obtained from average pooling the gradients of the activations A^k:

$$\alpha_k^c = \mathsf{AveragePool}_{i,j}\left(\frac{\partial y^c}{\partial A_{ij}^k}\right). \quad (2)$$

The relevance map is then given by a linear combination of the activations, weighted by the importance from Eq. 2:

$$L_c = \mathsf{ReLU}\left(\sum_k \alpha_k^c A^k\right), \quad (3)$$

where ReLU denotes the rectified linear unit function, $\mathsf{ReLU}(x) = \max(0, x)$. The final relevance map is obtained by resizing L_c to the same size as the original image, using bilinear interpolation.

For ViTs [23], we follow the method of Chefer *et al.* [13]. Specifically, given the attention maps A^k for each transformer block k, and relevance scores R^k [50], the relevance map L_c^{ViT} is given by:

$$L_c^{\text{ViT}} = \prod_k \bar{A}^k, \quad (4)$$

[3] notebook exploring the effects of the 80 prompts on ImageNet-1k

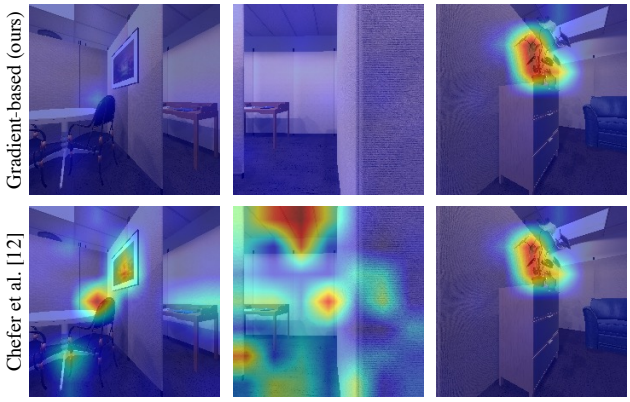

Figure 7. **Gradient-based relevance visualization.** The target object is a plant. For our gradient-based strategy, derived from that of Chefer *et al*. [13], relevance is low when the object is not in the image and high otherwise. In contrast the original method, produces spurious relevance when the target object is not in the frame due to the normalization it employs. Notice when the plant is in the frame, the relevance map is similar.

$$\bar{A}^k = I + \mathbb{E}_h \left[\mathsf{ReLU} \left(\nabla A^k \odot R^k \right) \right], \tag{5}$$

where $\mathbb{E}_h$ computes the mean over the attention heads and $\odot$ represents the element-wise product. In our case, we only look at attention maps from the final transformer block. Hence, $k = 1$.

It is standard to use an interpretability method like those of Selvaraju *et al*. [70] or Chefer *et al*. [13] to query for a class that is known to be in the image a priori. Hence, it is common practice to normalize relevance maps by subtracting the minimum relevance and normalizing by the difference between the maximum and minimum relevance. This results in a max value of 1.0. As mentioned in Sec. 4.3 this is not a suitable strategy for object navigation because in many frames the object is *not* in the image. Hence, in early experiments, we removed the normalization. Fig. 7 makes the differences between these strategies apparent. We notice this simple modification gives signal not only for true positive detections, but—critically—also for true negatives. Notice that when the plant is not in view, relevance is qualitatively low. However, when the plant is in view, relevance spikes in the region of the plant.

MDETR overview. MDETR [36] utilizes an encoder-decoder scheme to associate tokens in an input prompt referring to parts of an input image, with output boxes. MDETR utilizes a vision tower and a pre-trained language tower to project an image, text pair into a joint embedding space (similar to CLIP as discussed above). Image and text features are concatenated and passed to a transformer decoder head, which outputs boxes. The model is trained on a dataset of 200,000 images annotated with captions, boxes, and correspondence between words and boxes.

The model employs three losses during training. (1) DETR bipartite matching loss without class information: Following DETR [8], each ground truth box is matched with its most similar predicted box and an L1 loss is applied. Unlike DETR, MDETR does not use any class information for the matching step. (2) A soft-token classification loss: when classifying a box, the target is a uniform distribution over all the tokens the box refers to and zero for other tokens. In this way the box is assigned to potentially many tokens depending on the ground truth. (3) A contrastive loss in the bottleneck embedding space: this is analogous to the CLIP loss discussed above.

To fine-tune MDETR for segmentation, the original box model is fine-tuned in two stages on the PhraseCut dataset [80]. First the model is fine-tuned for box prediction discussed above on PhraseCut data, which contains referring expressions and corresponding regions in the image (masks and boxes). The weights are frozen and a new segmentation head is trained using Dice/F1 loss and focal loss.

OWL ViT overview. OWL ViT [49] employs a two-stage training procedure to create an open-vocabulary model. During the first stage they train a CLIP-like model. However, while the original CLIP model uses the feature corresponding to a ViT [CLS] token to construct its multi-modal embedding space, OWL ViT instead pools over patch tokens to obtain an image representation. Intuitively, this encourages the image global feature to encode more local information from each patch. During the second stage of training, the pooling layer is removed. The patch tokens are passed to a linear projection head where they are then dotted against text features to determine class probabilities. An MLP box-projection head is introduced that predicts a box for each projected patch token. Note this process and losses for box fine-tuning are similar to that used by DETR [8]. Stage 1 is trained using 3.6 billion image-text pairs from the dataset used in LiT [85], using a standard CLIP loss. Stage 2 is fine-tuned using an agglomeration of existing box datasets with ∼2M images total.

Localization thresholds. Each object localization method discussed in Sec. 4.3, requires a confidence threshold, which is standard when using a detector. To tune this threshold, we render 500 images in ROBOTHOR and 500 images in HABITAT MP3D with box annotations. Critically, we use the training rooms for these datasets, so none of the scenes overlap with those seen during downstream navigation testing. See Tab. 6 for thresholds, which are chosen to maximize an F1-score. Because PASTURE is a test set, we do *no* hyper-tuning on PASTURE. Instead we use the hyperparameters from ROBOTHOR for PASTURE evaluations.

Here we present more information on computing our F1-scores. For each image, there are some object categories $\mathcal{O}^+$ that appear in that image and other categories $\mathcal{O}^-$ that do not. We consider a predicted binary localization mask $M^{o^+}_{\text{pred}}$ a true positive (TP) if

IDs	Localizer	Arch.	HABITAT	ROBOTHOR and PASTURE
	CLIP-Ref.	B/32	–	0.25
	CLIP-Ref.	B/16	–	0.125
	CLIP-Patch	B/32	–	0.875
	CLIP-Patch	B/16	–	0.75
	CLIP-Grad.	B/32	0.375	0.625
	CLIP-Grad.	B/16	–	0.375
	MDETR	B3	–	0.95
	OWL	B/32	0.2	0.125
	OWL	B/16	–	0.125

Table 6. **Object localization hyperparameters.** Hyperparameters returned from a grid-search on object localization performance on HABITAT and ROBOTHOR train scenes. Note, no hyperparameter tuning was conducted on PASTURE, which is strictly a test set. Missing entries indicate that we did not evaluate these models due to lacking performance on other datasets (i.e., on ROBOTHOR).

$\text{score}^+ = \sum(M^{o^+}_{\text{pred}} \odot M^{o^+}_{\text{GT}}) / \sum M^{o^+}_{\text{pred}} > 0.5$, where $o^+ \in \mathcal{O}^+$, $M^{o^+}_{\text{GT}}$ is the ground truth mask, the summation is over all binary pixel values, and $\odot$ is an element-wise product that gives binary mask intersection. This is a more lenient measure than traditional jaccard index; however, also more applicable for our navigation setting where only part of the object needs to be identified correctly to provide a suitable navigation target. False positives (FP) arise when $0 < \text{score}^+ \leq 0.5$ or $\sum M^{o^-}_{\text{pred}} > 0$, where $o^- \in \mathcal{O}^-$. False negatives (FN) arise when $\sum M^{o^+} = 0$. F1 is computed in the standard way: $\text{TP}/(\text{TP} + 0.5(\text{FP} + \text{FN}))$. In each domain, F1 is computed per category and then averaged over the categories (i.e., macro F1). This yields F1^H for HABITAT and F1^R for ROBOTHOR.

GPT-3.5 [55] prompting details. Recall that we experiment with injecting object-level priors into CoW to exploit relationships between an object and a scene. We prompt GPT-3.5, with the following prompt: "`where can one usually find {} in a house \n {} are usually`", where {} is the name of the object and `\n` is the new line character. We set language model temperature to 0.0 to ensure deterministic outputs (i.e., greedy sampling) and set the max token generation length to 256.

D. Additional Visualization

To give a qualitative perspective on our results, we visualize some key aspects of the method and some trajectories. In Fig. 8, we show back-projecting 2D object relevancy to 3D, which is a key part of the CoW pipeline. We provide success and failure sample trajectories in Fig. 9.

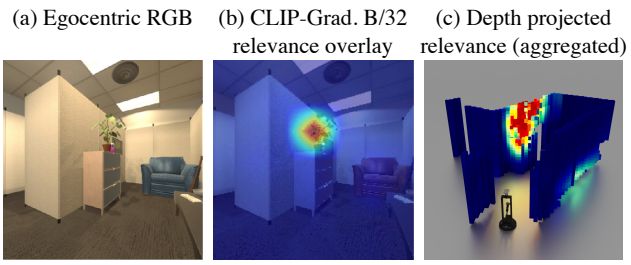

Figure 8. **Projection of object relevance.** (a) Egocentric RGB. Note, a CoW also receives a depth image. (b) Raw CLIP-Grad. B/32 prediction for the image targeting the "plant" class. (c) Back-projection of object relevance, aggregated over time, into a 3D map using agent pose estimates. Areas of high relevence make natural targets for navigation.

E. Dataset Details

ROBOTHOR and HABITAT MP3D dataset details. There are 11 HABITAT MP3D and 15 ROBOTHOR test scenes, in the official validation sets, which we use as our test sets. HABITAT includes 2,195 evaluation episodes, while ROBOTHOR has 1,800. We consider the following 21 categories in HABITAT: `chair`, `table`, `picture`, `cabinet`, `cushion`, `sofa`, `bed`, `chest_of_drawers`, `plant`, `sink`, `toilet`, `stool`, `towel`, `tv_monitor`, `shower`, `bathtub`, `counter`, `fireplace`, `gym_equipment`, `seating`, `clothes`. We consider the following 12 categories in ROBOTHOR: `AlarmClock`, `Apple`, `BaseballBat`, `BasketBall`, `Bowl`, `GarbageCan`, `HousePlant`, `Laptop`, `Mug`, `SprayBottle`, `Television`, `Vase`. Both of these lists include all objects for the HABITAT and ROBOTHOR CVPR 2021 object navigation challenges respectively. While the distribution of navigation episodes in ROBOTHOR is equally split per category, this is not the case in HABITAT MP3D. We provide testing episode counts per category in Fig. 10.

PASTURE additional statistics. In Fig. 11, we show word counts for various color, material, and spatial attributes for the appearance and spatial class remapping. In Fig. 12, we repeat similar analysis for hidden objects, reporting the word frequency for objects that contain the target objects.

PASTURE uncommon object asset licensing. We found the base instances for the PASTURE uncommon object split from CGTrader, an online repository where CAD hobbyists, artists, and professionals host their models. We choosing our objects, we selected 12 instances that had a "Royalty Free License" and were free to download at the time of dataset construction. We acknowledge all artists and provide links to their work:

- "tie-dye surfboard" by *adhil3dartist* and re-textured to be rainbow

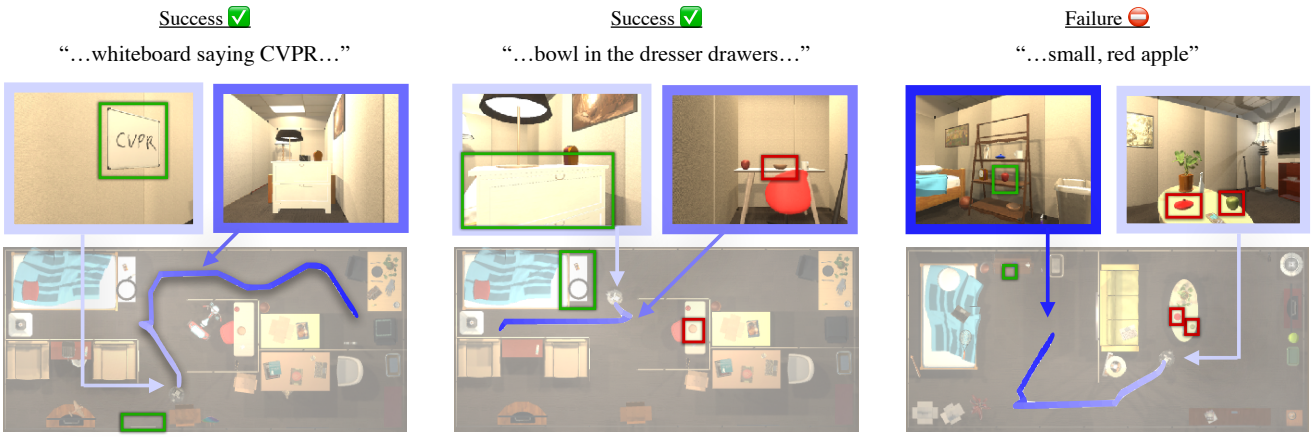

Figure 9. **Trajectory visualization.** Frames are egocentric views. Color indicates trajectory progress, where blue indicating trajectory start and white indicating trajectory end. Target objects are boxed in green, while distractor objects are boxed in red.

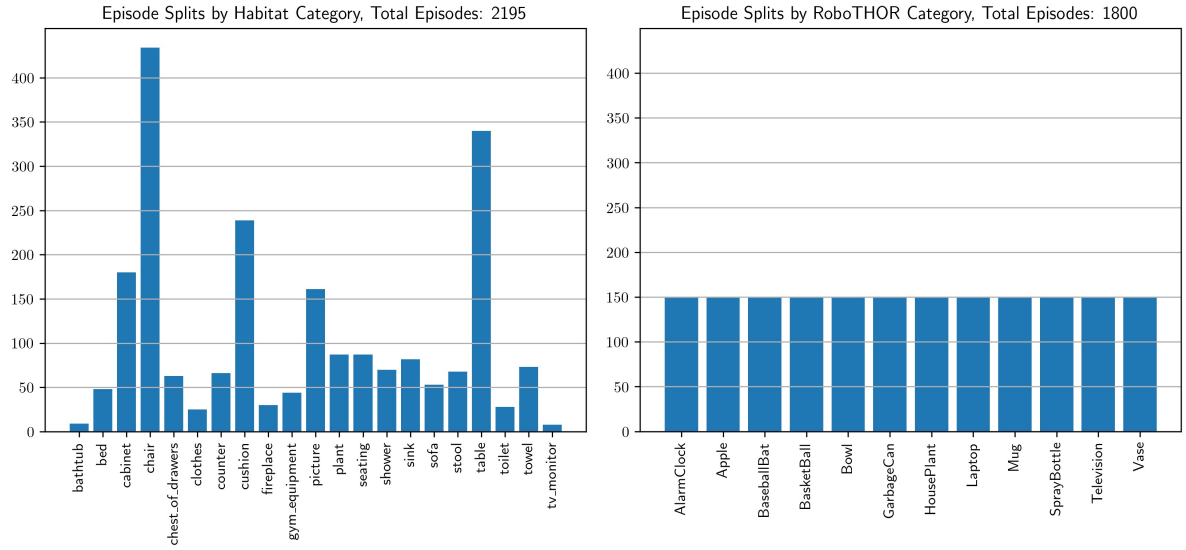

Figure 10. **HABITAT MP3D and ROBOTHOR episode splits.** Distribution of episodes in each CVPR 2021 object navigation challenge validation set that we adopt as our test set.

- "whiteboard saying CVPR" by *w-stone-art* and re-textured to say CVPR
- "llama wicker basket" by *eelh*
- "green plastic crate" by *Snowdrop-2018* and modified to include only the green create
- "rice cooker" by *fleigh* and re-textured
- "mate gourd" by *mcgamescompany*
- "red and blue tricycle" by *POLY1PROPS*
- "white electric guitar" by *demolitions2000*
- "espresso machine" by *WolfgangNikolas*
- "wooden toy airplane" by *fomenos*
- "gingerbread house" by *Empire-Assets*
- "graphics card" by *Biggie-3D*

PASTURE sample prompts. Here we provide some sample prompts for our appearance, spatial, and hidden object instance remapping.

- appearance remapping: from "spray bottle" to "small, green, plastic spray bottle"
- spatial remapping: from "spray bottle" to "spray bottle on a coffee table near a house plant"
- hidden remapping: from "spray bottle" to "spray bottle under the bed"

Retrieval of uncommon objects on LAION-5B [68] We include sample data from CLIP retrieval for both ROBOTHOR and PASTURE uncommon objects in Fig. 13. We included the results to show that CLIP is familiar with the concepts that we chose when creating the uncommon

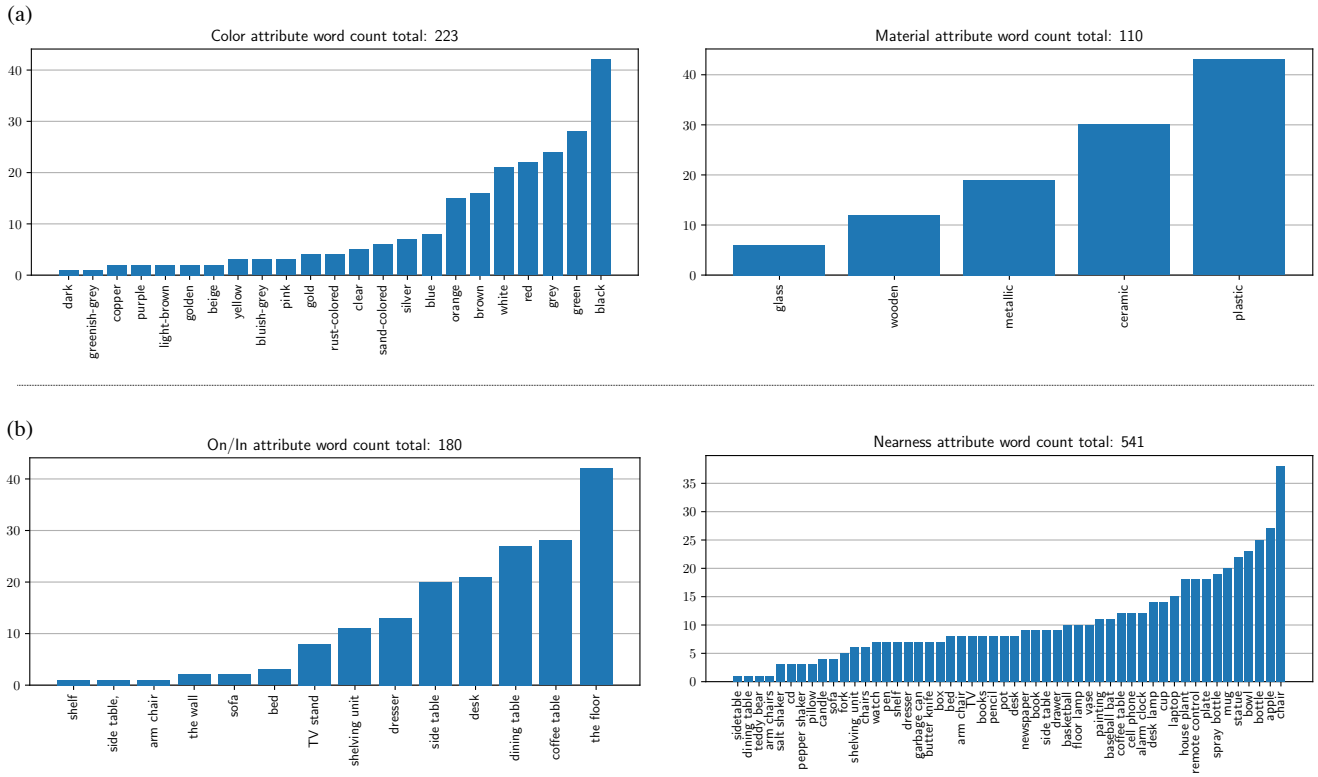

Figure 11. **Attribute distribution for PASTURE.** (a) Distribution of color and material attribute word frequency in the PASTURE appearance captions. (b) Distribution of on/in and near attributes by frequency in the PASTURE spatial captions.

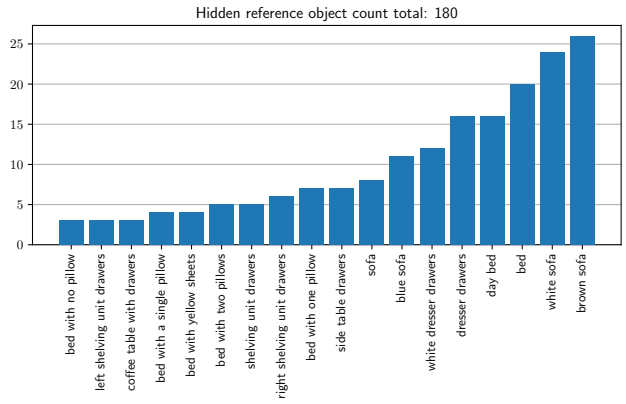

Figure 12. **Hidden reference object distribution for PASTURE.** Word frequency of large objects that target objects are hidden "in" or "under" in the PASTURE hidden object captions. Names are based on minimal descriptions needed to identify the object in the room. For example, "brown sofa" vs. "white sofa".

split of PASTURE and that qualitative image results rival those of the more common ROBOTHOR objects. CoW allows us to take advantage of the broad distribution of concepts that CLIP was fit on, without adding additional overhead associated with navigation fine-tuning.

F. Post-Processing Ablation

Removing post-processing (discussed in Sec. 4.3) decreases performance for the best models as seen in Tab. 7.

G. Prompt Ensemble Ablation

To verify the benefits of using the 80 prompt ensemble created by OpenAI—denoted as *prompt ens.*—, we compare performance for two models on ROBOTHOR. The competing approach uses a single prompt for each class "a photo of a {}."—denoted as *photo*. As we see in Tab. 8, prompt ens. boosts SPL by 0.2 for the CLIP-Grad. model. These, results suggest that the two variations of prompting strategies have marginal influence on downstream performance. For all experiments in the main paper that use CLIP, we use the 80 prompt ensemble in conjunction with the class label specific for a task (e.g., "spray bottle under the bed").

H. Category-level Results

For completeness we also include salient category-level results for an OWL (▲) and CLIP-Grad. (▲) B/32 models with post-processing for HABITAT MP3D, ROBOTHOR, and PASTURE. For a comparison for the appearance (Ap-

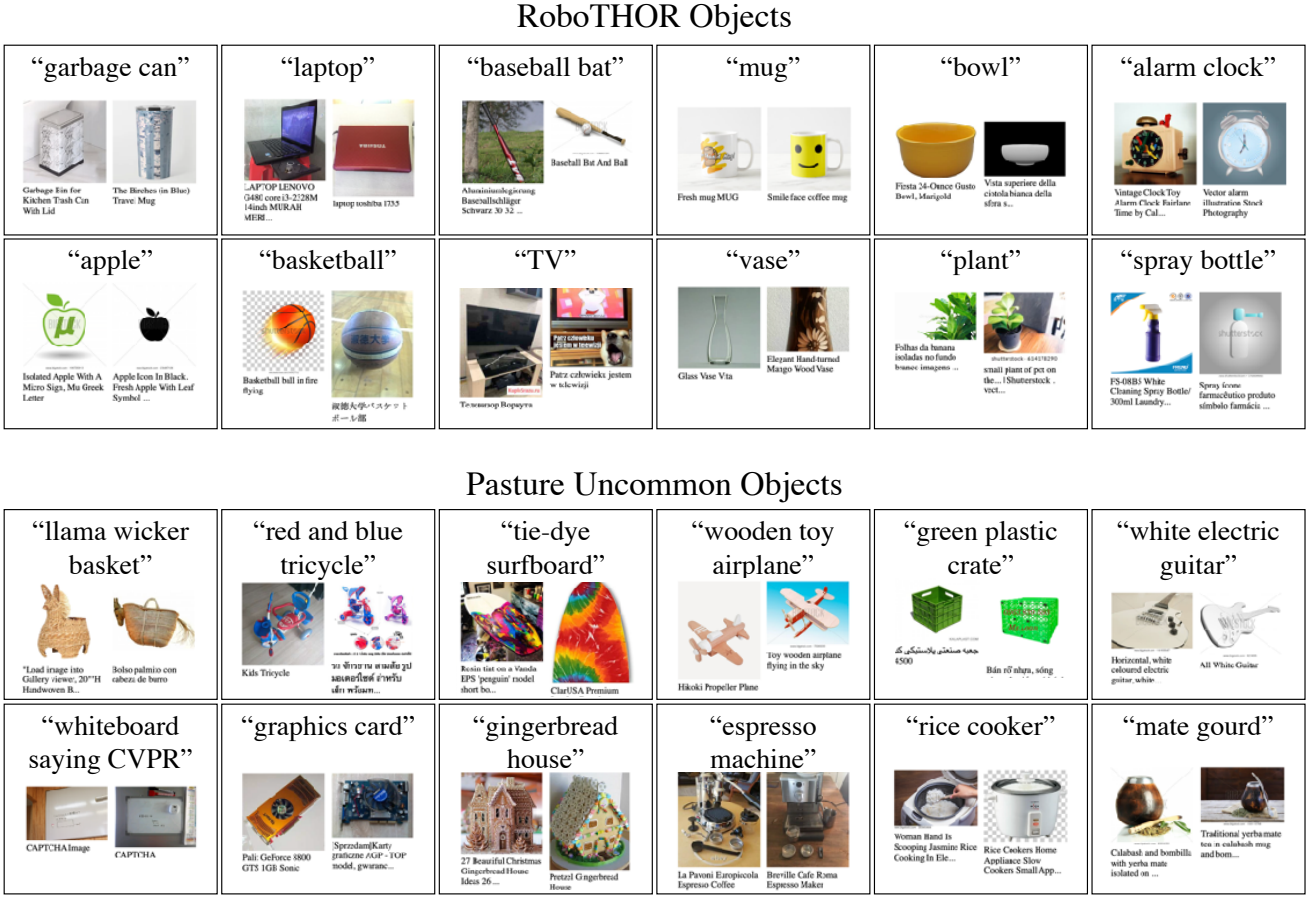

Figure 13. **Qualitative CLIP retrieval.** We conduct CLIP retrieval on LAION-5B [68] using the class names shown in quotes with the interface provided here. We show results for both RoboTHOR and Pasture uncommon objects. While CLIP is not trained on LAION-5B, this figure is included to give an idea of the type of noisy internet image-text data is trained on. Retrieval returns reasonable results for uncommon objects, suggesting CLIP is able to semantically distinguish these objects.

	CoW breeds		Pasture								RoboTHOR	
			Uncom.	Appear.	Space	Appear. distract	Space distract	Hid.	Hid. distract	Avg.		
ID	Localizer	Arch.	SR	SR	SR	SR	SR	SR	SR	SPL	SR	SPL	SR
△	CLIP-Ref.	B/32	2.8 (-0.8)	1.4 (+0.8)	1.4 (-0.3)	0.8 (+0.2)	1.4 (-0.3)	4.7 (+2.5)	5.0 (+2.5)	1.2 (+0.3)	2.5 (+0.7)	1.6 (+0.6)	2.2 (+0.4)
□	CLIP-Ref.	B/16	1.4 (+0.0)	1.7 (-1.1)	1.7 (-1.1)	1.9 (-1.2)	1.9 (-1.4)	2.8 (+1.1)	2.2 (+0.3)	1.7 (+0.0)	1.9 (-0.5)	2.4 (+0.3)	2.6 (-0.1)
△	CLIP-Patch	B/32	10.6 (-7.5)	9.7 (-3.6)	6.7 (-6.6)	6.4 (-2.2)	6.4 (-4.4)	16.7 (-0.8)	16.7 (-1.1)	7.5 (-1.5)	10.4 (-3.8)	9.0 (-1.6)	14.3 (-6.0)
□	CLIP-Patch	B/16	5.6 (-5.0)	7.8 (-3.6)	3.9 (-3.9)	5.0 (-5.8)	3.9 (-4.2)	10.6 (-5.8)	10.8 (-4.8)	5.4 (-2.3)	6.8 (-4.7)	8.2 (-1.5)	10.3 (-5.4)
△	CLIP-Grad.	B/32	13.6 (-2.5)	10.6 (-1.3)	9.2 (-2.5)	7.5 (-2.2)	7.2 (-3.1)	13.9 (-0.5)	12.8 (-3.3)	8.3 (-0.9)	10.7 (-2.2)	9.6 (-0.1)	13.8 (-1.4)
□	CLIP-Grad.	B/16	6.1 (-2.0)	5.8 (-5.0)	5.0 (-3.6)	5.0 (-3.6)	4.7 (-2.0)	8.3 (-2.8)	6.9 (-4.5)	4.9 (-1.8)	6.0 (-3.3)	7.3 (-1.3)	8.8 (-2.8)
◇	MDETR	B3	3.1 (+0.0)	6.9 (-0.3)	4.4 (-0.6)	7.2 (0.3)	4.7 (+0.0)	7.8 (-0.3)	8.9 (+0.0)	5.3 (-0.1)	6.2 (-0.1)	8.3 (-0.1)	9.8 (-0.1)
△	OWL	B/32	23.1 (-9.7)	26.1 (-0.3)	14.4 (-5.0)	18.3 (-1.1)	11.7 (-4.4)	13.9 (-5.3)	13.1 (-1.3)	11.1 (-1.5)	17.2 (-3.9)	16.6 (-0.3)	25.4 (-1.3)
□	OWL	B/16	25.8 (-6.1)	23.6 (-3.3)	15.3 (-3.6)	17.2 (-2.2)	12.5 (-2.2)	13.1 (-5.0)	13.9 (-1.9)	11.4 (-1.2)	17.3 (-3.5)	16.2 (-1.0)	24.8 (-2.7)

Table 7. **Benchmarking CoWs without post-processing on Pasture for L-ZSON.** Compared to the filled (●) IDs presented in Tab. 1, unfilled (○) row IDs without post-processing perform worse. Bracketed numbers show the deltas relative to the corresponding model with post-processing. Hence, using only the center pixel as a representative target for navigation helps in general (see Sec. 4.3 for more details on post-processing).

ID	Loc.	CoW breeds Arch.	Post	Exp. Strategy	RoboTHOR SPL	RoboTHOR SR
▲	CLIP-Grad.	B/32	✓	photo	9.5	**15.2**
▲	CLIP-Grad.	B/32	✓	prompt ens.	**9.7**	**15.2**

Table 8. **Prompt ablation.** For a fixed object localizers (CLIP-Grad. B/32 with post processing), we ablate over different choices of prompts. We find that the 80 prompt ensemble (prompt ens.), introduced by OpenAI, outperforms the prompt: "a photo of a {}." (photo) in most cases. However, deltas are not large, suggesting that this is a less critical design decision in the CoW framework.

	Pasture Appear.				Pasture Space			
category	SR ▲	SPL ▲	SR ▲	SPL ▲	SR ▲	SPL ▲	SR ▲	SPL ▲
AlarmClock	6.7	4.0	**23.3**	**10.7**	3.3	3.3	**10.0**	**3.8**
Apple	6.7	5.7	**36.7**	**17.0**	**10.0**	**8.4**	3.3	1.0
BaseballBat	0.0	0.0	**3.3**	**1.2**	3.3	2.5	**6.7**	**2.7**
BasketBall	6.7	2.8	**36.7**	**24.1**	10.0	5.6	**36.7**	**26.8**
Bowl	3.3	0.5	**13.3**	**5.9**	10.0	5.6	**16.7**	**6.9**
GarbageCan	26.7	20.2	**50.0**	**31.5**	30.0	23.0	**40.0**	**23.2**
HousePlant	20.0	16.9	**30.0**	**20.2**	13.3	10.8	**40.0**	**21.9**
Laptop	13.3	10.6	**20.0**	**11.5**	13.3	9.6	**20.0**	**13.7**
Mug	10.0	7.5	**46.7**	**27.4**	10.0	**7.5**	**13.3**	5.4
SprayBottle	16.7	13.6	**33.3**	**19.2**	**16.7**	**15.8**	**16.7**	6.8
Television	10.0	**10.0**	**13.3**	8.9	6.7	6.4	**20.0**	**9.9**
Vase	**23.3**	**17.5**	10.0	9.1	**13.3**	**9.8**	10.0	5.4

Table 9. **Attribute object navigation.** Appearance-based captions consistently perform better than spatial captions. OWL consistently performs better than CLIP-Grad.

	Pasture Appear. distract				Pasture Space distract			
category	SR ▲	SPL ▲	SR ▲	SPL ▲	SR ▲	SPL ▲	SR ▲	SPL ▲
AlarmClock	3.3	3.0	**13.3**	**6.5**	**6.7**	**6.3**	**6.7**	2.8
Apple	**10.0**	6.4	**10.0**	**7.4**	3.3	3.3	**10.0**	**4.0**
BaseballBat	0.0	0.0	**13.3**	**4.5**	0.0	0.0	**10.0**	**8.1**
BasketBall	6.7	3.3	**20.0**	**12.6**	**16.7**	9.4	**16.7**	**9.7**
Bowl	3.3	3.2	**16.7**	**8.5**	10.0	8.6	**23.3**	**12.6**
GarbageCan	26.7	19.9	**30.0**	**21.6**	13.3	10.5	**26.7**	**18.2**
HousePlant	10.0	6.0	**16.7**	**11.0**	13.3	10.8	**23.3**	**13.7**
Laptop	16.7	**13.6**	**23.3**	11.9	**16.7**	**11.9**	**16.7**	11.8
Mug	6.7	5.1	**26.7**	**17.8**	**10.0**	**7.8**	6.7	2.7
SprayBottle	13.3	12.4	**26.7**	**15.2**	16.7	**15.4**	**20.0**	8.0
Television	6.7	6.6	**26.7**	**15.5**	6.7	3.3	**20.0**	**12.8**
Vase	**13.3**	**10.2**	10.0	8.6	10.0	6.5	**13.3**	**8.4**

Table 10. **Attribute object navigation with distractors.** Distractors consistently hurt performance compared to the no distractor numbers in Tab. 9, suggesting that models cannot make full use of remapped classes with attributes.

pear.) and spatial (Space) Pasture splits, see Tab. 9. For

	Pasture Hidden				Pasture Hidden distract			
category	SR ▲	SPL ▲	SR ▲	SPL ▲	SR ▲	SPL ▲	SR ▲	SPL ▲
AlarmClock	**26.7**	**15.8**	6.7	3.0	20.0	**9.9**	**23.3**	8.3
Apple	10.0	**9.5**	**13.3**	8.9	6.7	4.7	**10.0**	**7.9**
BaseballBat	**13.3**	**9.2**	**13.3**	5.8	**10.0**	**7.2**	3.3	2.9
BasketBall	10.0	5.1	**26.7**	**15.2**	16.7	12.2	**20.0**	**15.0**
Bowl	16.7	9.2	**23.3**	**11.9**	**23.3**	**15.1**	20.0	10.6
GarbageCan	13.3	7.8	**26.7**	**13.8**	13.3	7.0	**16.7**	**10.9**
HousePlant	13.3	9.6	**16.7**	**10.8**	**16.7**	10.9	**16.7**	**11.6**
Laptop	**10.0**	8.1	**10.0**	**8.4**	**26.7**	**20.7**	10.0	8.4
Mug	13.3	7.3	**33.3**	**23.8**	20.0	12.5	**23.3**	**18.4**
SprayBottle	**13.3**	**8.4**	0.0	0.0	**20.0**	**10.3**	0.0	0.0
Television	16.7	8.0	**36.7**	**22.9**	10.0	6.9	**16.7**	**11.4**
Vase	16.7	**11.9**	**23.3**	11.2	10.0	7.0	**13.3**	**7.0**

Table 11. **Hidden object navigation category-level results.**

	Pasture Uncom.			
category	SR ▲	SPL ▲	SR ▲	SPL ▲
GingerbreadHouse	20.0	14.3	**26.7**	**18.6**
EspressoMachine	10.0	7.7	**46.7**	**24.6**
Crate	23.3	18.2	**40.0**	**27.0**
ElectricGuitar	16.7	10.0	**46.7**	**30.8**
RiceCooker	3.3	2.9	**20.0**	**11.6**
LlamaWickerBasket	16.7	12.6	**30.0**	**24.5**
Whiteboard	**63.3**	**43.2**	30.0	18.7
Surfboard	26.7	20.6	**60.0**	**38.9**
Tricycle	10.0	9.0	**53.3**	**31.7**
GraphicsCard	3.3	2.1	**13.3**	**6.0**
Mate	**0.0**	**0.0**	**0.0**	**0.0**
ToyAirplane	0.0	0.0	**26.7**	**13.7**

Table 12. **Uncommon object navigation category-level results.**

Appear. and Space with distractors (distract), see Tab. 10. For hidden object with and without distractors see Tab. 11. For uncommon objects see Tab. 12. For RoboTHOR see Tab. 13. For Habitat see Tab. 14.

I. Additional Failure Analysis

For addition failure analysis on CLIP-Ref., CLIP-Patch, and CLIP-Grad. see Fig. 14. All models have a ViT-B/32 architecture and apply post-processing. We notice that that CLIP-Patch and CLIP-Grad. have a higher fraction of object localization failure when compared to OWL. For CLIP-Ref., where performance is in the very low success regime (e.g., $< 3\%$), we notice less consistent patterns.

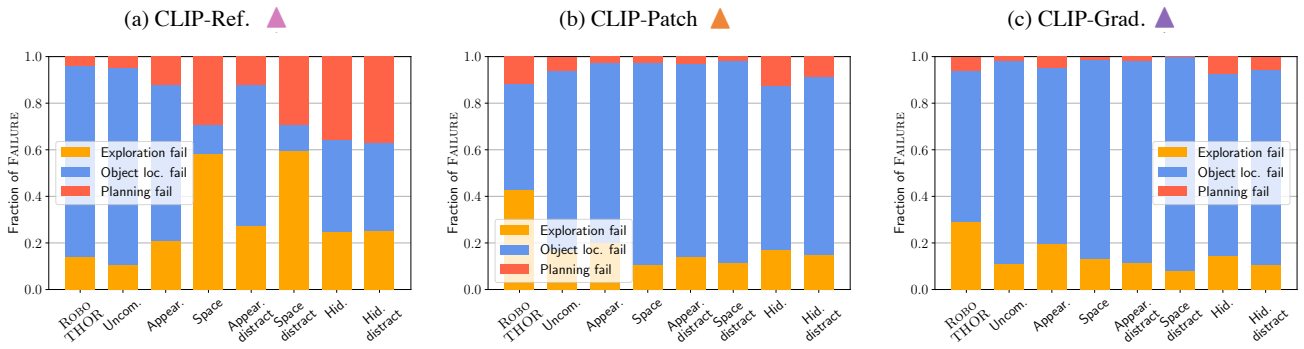

Figure 14. **Failure analysis for more models.** (a, b, c) All show additional error analysis. When comparing CLIP-Patch and CLIP-Grad. to OWL shown in Fig. 6, we notice that the former have a higher percentage of object localization failures and also lower success rate downstream on average.

	ROBOTHOR			
	▲		▲	
category	SR	SPL	SR	SPL
ALARMCLOCK	5.3	3.1	**30.7**	**19.2**
APPLE	15.3	9.7	**34.0**	**19.6**
BASEBALLBAT	**4.0**	**1.5**	2.0	0.5
BASKETBALL	19.3	14.8	**36.0**	**25.3**
BOWL	5.3	4.0	**18.0**	**11.1**
GARBAGECAN	30.0	21.0	**50.0**	**32.2**
HOUSEPLANT	30.7	18.7	**36.7**	**24.8**
LAPTOP	16.0	10.6	**20.0**	**11.3**
MUG	10.7	5.0	**40.7**	**25.5**
SPRAYBOTTLE	8.0	5.2	**23.3**	**13.8**
TELEVISION	**29.3**	**16.9**	23.3	13.4
VASE	**8.0**	**5.8**	6.0	5.7

Table 13. **ROBOTHOR category-level results.**

	PASTURE HABITAT			
	▲		▲	
category	SR	SPL	SR	SPL
CHAIR	5.1	2.2	**7.4**	**3.7**
TABLE	**34.7**	**18.4**	21.2	9.6
PICTURE	1.2	0.5	**1.9**	**1.1**
CABINET	**9.4**	4.7	8.3	**4.9**
CUSHION	5.0	3.0	**12.1**	**5.8**
SOFA	**0.0**	**0.0**	**0.0**	**0.0**
BED	**0.0**	**0.0**	**0.0**	**0.0**
CHEST_OF_DRAWERS	0.0	0.0	**1.6**	**0.4**
PLANT	**5.7**	**3.6**	2.3	1.3
SINK	**2.4**	1.6	**2.4**	**1.8**
TOILET	**0.0**	**0.0**	**0.0**	**0.0**
STOOL	0.0	0.0	**2.9**	**2.4**
TOWEL	0.0	0.0	**1.4**	**0.7**
TV_MONITOR	**0.0**	**0.0**	**0.0**	**0.0**
SHOWER	**7.1**	**4.4**	1.4	0.9
BATHTUB	0.0	0.0	**11.1**	**3.1**
COUNTER	**6.1**	**3.1**	0.0	0.0
FIREPLACE	**10.0**	**5.0**	3.3	1.7
GYM_EQUIPMENT	**0.0**	**0.0**	**0.0**	**0.0**
SEATING	**12.6**	**5.2**	0.0	0.0
CLOTHES	**8.0**	**4.3**	4.0	2.1

Table 14. **HABITAT category-level results.**