Semantic Visual Navigation by Watching YouTube Videos

Matthew Chang **Arjun Gupta** **Saurabh Gupta**
University of Illinois at Urbana-Champaign
{mc48, arjung2, saurabhg}@illinois.edu

Abstract

Semantic cues and statistical regularities in real-world environment layouts can improve efficiency for navigation in novel environments. This paper learns and leverages such semantic cues for navigating to objects of interest in novel environments, by simply watching YouTube videos. This is challenging because YouTube videos don't come with labels for actions or goals, and may not even showcase optimal behavior. Our method tackles these challenges through the use of Q-learning on pseudo-labeled transition quadruples (image, action, next image, reward). We show that such off-policy Q-learning from passive data is able to learn meaningful semantic cues for navigation. These cues, when used in a hierarchical navigation policy, lead to improved efficiency at the ObjectGoal task in visually realistic simulations. We observe a relative improvement of $15 - 83\%$ over end-to-end RL, behavior cloning, and classical methods, while using minimal direct interaction.

1 Introduction

Consider the task of finding your way to the bathroom while at a new restaurant. As humans, we can efficiently solve such tasks in novel environments in a zero-shot manner. We leverage common sense patterns in the layout of environments, which we have built from our past experience of similar environments. For finding a bathroom, such cues will be that they are typically towards the back of the restaurant, away from the main seating area, behind a corner, and might have signs pointing to their locations (see Figure 1). Building computational systems that can similarly leverage such semantic regularities for navigation has been a long-standing goal.

Hand-specifying what these semantic cues are, and how they should be used by a navigation policy is challenging. Thus, the dominant paradigm is to directly learn what these cues are, and how to use them for navigation tasks, in an end-to-end manner via reinforcement learning. While this is a promising approach to this problem, it is sample inefficient, and requires many million interaction samples with dense reward signals to learn reasonable policies.

But, is this the most direct and efficient way of learning about such semantic cues? At the end of the day, these semantic cues are just based upon spatial consistency in co-occurrence of visual patterns next to one another. That is, if there is always a bathroom around the corner towards the back of the restaurant, then we can learn to find this bathroom, by simply finding corners towards the back of the restaurant. This observation motivates our work, where we pursue an alternate paradigm to learn semantic cues for navigation: learning about this spatial co-occurrence in indoor environments through video tours of indoor spaces. People upload such videos to YouTube (see project video) to showcase real estate for renting and selling. We develop techniques that leverage such YouTube videos to learn semantic cues for effective navigation to semantic targets in indoor home environments (such as finding a *bed* or a *toilet*).

Project website with code, models, and videos: https://matthewchang.github.io/value-learning-from-videos/.

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

Figure 1: Semantic Cues for Navigation. Even though you don't see a restroom, or a sign pointing to one in either of these images, going straight ahead in the left image is more likely to lead to a restroom than going straight in the right image. This paper seeks to learn and levarage such semantic cues for finding objects in novel environments, by watching egocentric YouTube videos.

Such use of videos presents three unique and novel challenges, that don't arise in standard learning from demonstration. Unlike robotic demonstrations, videos on the Internet don't come with any action labels. This precludes learning from demonstration or imitation learning. Furthermore, goals and intents depicted in videos are not known, *i.e.*, we don't apriori know what each trajectory is a demonstration for. Even if we were to label this somehow, the depicted trajectories may not be optimal, a critical assumption in learning from demonstration [54] or inverse reinforcement learning [44].

Our formulation, *Value Learning from Videos* or *VLV*, tackles these problems by **a)** using pseudo action labels obtained by running an inverse model, and **b)** employing Q-learning to learn from video sequences that have been pseudo-labeled with actions. We follow work from Kumar *et al.* [38] and use a small number of interaction samples ($40K$) to acquire an inverse model. This inverse model is used to pseudo-label consecutive video frames with the action the robot would have taken to induce a similar view change. This tackles the problem of missing actions. Next, we obtain goal labels by classifying video frames based on whether or not they contain the desired target objects. Such labeling can be done using off-the shelf object detectors. Use of Q-learning [65] with consecutive frames, intervening actions (from inverse model), and rewards (from object category labels), leads to learning *optimal* Q-functions for reaching goals [60, 65]. We take the maximum Q-value over all actions, to obtain value functions. These value functions are exactly γ^s, where s is the number of steps to the nearest view location of the object of interest (γ is the Q-learning discount factor). These value functions implicitly learn semantic cues. An image looking at the corner towards the back of the restaurant will have a higher value (for *bathroom* as the semantic target) than an image looking at the entrance of the restaurant. These learned value functions when used with a hierarchical navigation policy, efficiently guide locomotion controllers to desired semantic targets in the environment.

Learning from such videos can have many advantages, some of which address limitations of learning from direct interaction (such as via RL). Learning from direct interaction suffers from high sample complexity (the policy needs to discover high-reward trajectories which may be hard to find in sparse reward scenarios) and poor generalization (limited number of instrumented physical environments available for reward-based learning, or sim2real gap). Learning from videos side-steps both these issues. We observe a $47 - 83\%$ relative improvement in performance over RL and imitation learning methods, while also improving upon strong classical methods.

2 Related Work

This paper tackles semantic visual navigation in novel environments. Our proposed solution is a hierarchical policy that employs value functions learned from videos. We survey different navigation tasks, the different representations used to tackle them, and the different training methodologies employed to build those representations.

Navigation Tasks. Navigation tasks take many forms [3], but can largely be grouped into two categories based on whether they require exploration or not. Finding paths in known environments [71], or going to a known relative offset in a previously unknown environment [27], do not require very much exploration. On the other hand, tasks such as finding an object [27] (or a given image target [11]) in

a novel environment, or exhaustively mapping one [10, 12], require exploration and are thus more challenging. Our down-stream task of finding objects in previously unseen novel environments falls into this second category. Most current work [16, 27, 45, 70] on this task employ end-to-end, interaction-heavy learning to get at necessary semantic cues. Our work instead seeks to mine them from videos with minimal active interaction.

Representations. Solving navigation tasks, requires building and maintaining representations for space. These range from explicit metric maps [20, 61, 68] or topological representations [11, 35, 52], to more abstract learned implicit representations [41]. Such learned representations can effectively learn about semantic cues. Research has also focused on making classical metric and topological representations more semantic: explicitly by storing object detector or scene classifier outputs [8, 25, 30, 36, 42, 47, 67], or implicitly by storing abstract learned feature vectors useful for the end-task [27]. In our work, we use a hybrid topological and metric representation that incorporates implicit semantic information. Our focus is on investigating alternate ways of learning such semantic information.

Hierarchical Policies. Researchers have pursued many different hierarchical policies [7] for navigation: no hierarchy [41], macro-actions [27, 71], low-level controllers [6, 33], and sub-policies [10, 15, 25]. In particular, Chaplot *et al.* [10, 11] decompose exploration policies into a global policy, for high-level semantic reasoning, and a local policy, for low-level execution to achieve short-term goals produced by the global policy. We follow a similar decomposition, but tackle a different task (object goal), and investigate learning from unlabeled passive data *vs.* active interaction or strong supervision.

Training Methodology. Different papers pursue different strategies for training navigation policies: no training [61], supervised learning for collision avoidance [22, 24], behavior cloning, DAgger [27, 37, 49], reinforcement learning with sparse and dense rewards [10, 41, 50, 51, 66, 71], and combinations of imitation and RL [12, 14, 48]. In contrast, this paper designs a technique to derive navigation policies by watching YouTube videos. This is most similar to work from Kumar *et al.* [38] that studies how to learn low-level locomotion sub-routines from *synthetic* videos. In contrast, we learn high-level semantic cues from actual YouTube videos.

Learning for Acting from Videos. Learning about affordances [21], state-transitions [2, 31], and task-solving procedures [13], with the goal of aiding learning for robots, is a long-standing goal in computer vision. Our work is also a step in this direction, although our output is directly useful for building navigation policies, and our experiments demonstrate this.

Learning without Action Labels. A number of recent papers focus on learning from observation-only (or state-only) demonstrations (*i.e.* demonstrations without action labels). Some works focus on directly learning policies from such data [19, 23, 46, 55, 62, 63], while others focus on extracting a reward function for subsequent policy learning through RL [5, 17, 18, 40, 57, 58]. All of these works focus on learning a policy for the *same* task in the *same* environment that is depicted in the observation-only demonstrations (with the exception of Gangwani *et al.* [23] who show results in MDPs with different transition dynamics). Our work relaxes both these assumptions, and we are able to use video sequences to derive cues that aid solving novel tasks in novel environments.

3 Proposed Approach

The final task we tackle is that of reaching semantic goals in a novel environment, *i.e.*, at test time we will place the agent in a novel environment and measure how efficiently it can find common house-hold objects (bed, chair, sofa, table and toilets).

Overview. We design a 2-level hierarchical policy. The *high-level policy* incrementally builds a topological graph and uses *semantic reasoning* to pick promising directions of exploration. It generates a short-term goal (within $2m$) for the *low-level policy*, that achieves it or returns that the short-term goal is infeasible. This process is repeated till the agent reaches its goal. We describe the details of this hierarchical policy in Supplementary Section S1. Our central contribution is the procedure for learning the semantic reasoning function, which we call a *value* function (following RL terminology [60]), for the high-level policy from videos, and we describe this next.

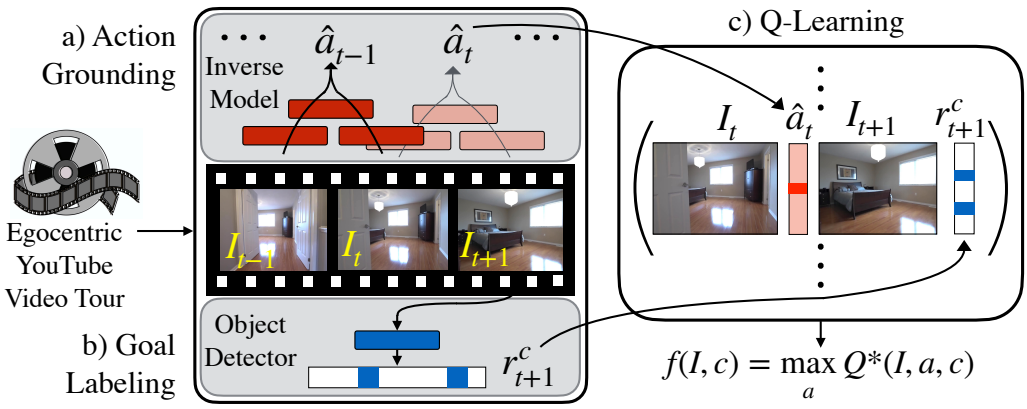


Figure 2: Learning Values Functions from Videos. Egocentric videos tours of indoor spaces are **a)** grounded in actions (by labeling via an inverse model), **b)** labeled with goals (using an object detector). This prepares them for **c)** Q-learning, which can extract out optimal Q-functions for reaching goals purely by watching egocentric videos. See Section 3.1 for more details.

3.1 Value Learning from Videos

Given an image I and a set of object categories $\mathcal{C}$, we seek to learn a function $f(I, c)$ that can predict the *value* for image I for reaching an object of category $c \in \mathcal{C}$. Images that show physical space close to instances of category c should have a higher value than images that show regions far away from it.

Let's say we have $\mathcal{V}$, a set of egocentric video tours of indoor spaces. We seek to learn this function from such videos. We follow a three step procedure: **a)** imagining robot actions that convey the robot between intervening frames, **b)** labeling video frames of images containing instances of the desired object category, and **c)** Q-learning on the resulting reward-labeled image-action sequence trajectories. Figure 2 shows an overview of this process, we describe it in more detail below.

Action grounding. Such videos don't come with any information for how one image is related to another. We follow the pseudo-labeling approach from [38, 62], to imagine the actions the robotic agent would have taken to induce the depicted transformation. We collect a small amount of interaction data, where a robot executes random actions in a handful of environments. This data is in the form of image action sequences, $\ldots, I_t, a_t, I_{t+1}, \ldots$, and importantly, has information of the action that was executed to go from I_t to I_{t+1}. We use this interaction dataset to train a *one-step inverse model* ψ [1, 32] that uses I_t and I_{t+1} to predict $\hat{a}_t = \psi(I_t, I_{t+1})$. ψ is trained via a cross-entropy loss between its prediction $\hat{a}_t$ and ground truth a_t. We use this inverse model ψ to *pseudo-label* the video dataset $\mathcal{V}$ with action labels to obtain $\hat{\mathcal{V}}$.

Labeling Video Frames with Goals. Our next step involves labeling video frames with the presence of object instances from categories in $\mathcal{C}$. This can simply be done by using an off-the-shelf object detector D (such as Mask RCNN [28]) trained on the MS-COCO dataset [39]. We assign a binary reward value $r^c(I)$ for each category c for each video frame I: $+1$ if object detected, and 0 otherwise.

Value Learning via Off-policy Q-Learning. Our next step is to derive value function $f(I, c)$ for the different categories. The above two steps, generate reward-labeled, image-action trajectories for traversals in indoor environments. For each category $c \in \mathcal{C}$, these are in the form of quadruples $\left(I_t, \hat{a}_t, I_{t+1}, r^c_{t+1}\right)$, where I_t and I_{t+1} are consecutive frames, $\hat{a}_t$ is the pseudo-label as predicted from the inverse model ψ, and r^c_{t+1} is the label for category c for image I_{t+1}. These quadruples can be thought of as transitions from a Markov Decision Process (MDP) [60], where the agent gets $+1$ reward for entering into a location close to the desired target object, and 0 reward otherwise.

Thus, even though we don't have access to the physical environment, a simple video traversal of an indoor space can be pseudo-labeled to obtain transition samples from the underlying MDP operating in this environment. Under mild conditions, such samples are all that are necessary for learning *optimal* value functions via Q-learning [65]. Thus, instead of directly learning the value function $f(I, c)$, we learn a Q-function $Q(I, c, a)$ that predicts the Q value of executing action a when at image I and seeking to find object from category c.

Q-learning takes the following form, where we seek to learn the fixed point, Q^* of the following Bellman equation (for each category c): $Q^*(I_t, a_t, c) = \max_{a'} \left(r^c_{t+1} + \gamma Q^* \left(I_{t+1}, a', c\right)\right)$. This is done by finding the Q that minimizes the following objective function, over transition quadruples from $\hat{\mathcal{V}}$ (we parameterize Q as a convolutional neural network (more details in Section 4)):

$$\sum_{\hat{\mathcal{V}}} \left[Q(I_t, a_t, c) - \left(r^c_{t+1} + \gamma \max_{a'} Q\left(I_{t+1}, a', c\right)\right)\right]^2. \tag{1}$$

Value function $f(I, c)$ can be obtained by simply taking a maximum of the Q-values over all actions, *i.e.*, $f(I, c) = \max_a Q(I, a, c)$. This gives us our desired value function.

Note, Q-learning can learn optimal Q-functions independent of where transition quadruples come from (as long as they cover the space), and in particular, can learn from off-policy data. This allows us to learn *optimal* value functions even though the video dataset may not follow optimal paths to any targets. This also leads us to favor Q-learning over the simpler alternative of employing Monte Carlo or TD(0) *policy evaluation* [60]. Policy evaluation is simpler as it does not involve reasoning about intervening actions, but consequently only learns the value of the underlying policy depicted in the video, rather than the optimal policy. Our experiments demonstrate this contrast between these two design choices, in scenarios where videos don't show the optimal goal reaching behavior.

The learned Q-function, and the associated value function $f(I, c)$, implicitly learn semantic cues for navigation. They can learn what images lead to the desired category, and what don't. Relative magnitude of their prediction can be used to pick directions for exploration. It is worth noting, this obtained value function is the optimal value function under the dynamics of the agent recording the video. We are implicitly assuming that optimal value function under the robot's action space or dynamics would be similar enough. This assumption may not always be true (specially at fine temporal scales), but is true in a number of situations at coarser time scales.

4 Experiments

We show results on the ObjectGoal task in novel environments [3]. Our experiments test the extent to which we are able to learn semantic cues for navigation by watching videos, and how this compares to alternate techniques for learning such cues via direct interaction. We also compare against alternate ways of learning from passive video data, and show visualizations of our learned value functions.

Video Dataset. We mined for real estate tours from YouTube. This ***YouTube House Tours Dataset*** consists of 1387 videos with a total run length of 119 hours. A sample video is shown in supplementary video. We sample a frame every 1.5 seconds resulting in 550K transitions tuples I_t, I_{t+1} for Q-learning (after removing outdoor scenes and people). We denote this dataset as $\mathcal{V}_{\text{yt}}$.

Experimental Setup. We work with a simulated robot in visually realistic simulation environments. We use the Habitat simulator [53] with the Gibson environments [69] (100 training environments from the medium split, and the 5 validation environments from the tiny split). These environments are derived from scans of real world environments, and thus retain the visual and layout complexity of the real world, but at the same time allow for systematic experimentation.

We split the 105 environments into three sets: $\mathcal{E}_{\text{train}}$, $\mathcal{E}_{\text{test}}$, and $\mathcal{E}_{\text{video}}$ with 15, 5, and 85 environments respectively. The robot has access to, and can directly interact with environments in $\mathcal{E}_{\text{train}}$. $\mathcal{E}_{\text{test}}$ is same as the official Gibson tiny validation set that comes with human verified semantic class labels [4]. It is used to setup downstream semantic navigation tasks for evaluation. $\mathcal{E}_{\text{train}}$ and $\mathcal{V}_{\text{yt}}$ are used for learning via our proposed formulation. Learned policies are evaluated on $\mathcal{E}_{\text{test}}$. For some additional control experiments, we also create a dataset of synthetic videos $\mathcal{V}_{\text{syn}}$ using the 85 environments in $\mathcal{E}_{\text{video}}$ (generation procedure described in supplementary). Our splitting procedure ensures: **a)** final testing happens in novel, previously unseen environments, and **b)** the robot does not have direct access to environments in which videos were shot (neither the $\mathcal{E}_{\text{video}}$ used to generate $\mathcal{V}_{\text{syn}}$, nor the real estate shown in YouTube House Tours Dataset $\mathcal{V}_{\text{yt}}$).

Robot Model. We use a simplified robot action space with four actions: move forward by $25cm$, rotate left $30°$, rotate right $30°$ and stop. We assume perfect localization, that is, the robot exactly knows where it is relative to its previous location. This can be achieved by running a SLAM system,

or using additional sensors such as an IMU. The robot is a $1.25m$ long cylinder of radius $10cm$, and has a RGB-D camera with 90° field of view, mounted at a height of $1.25m$.

Semantic Visual Navigation Task. We set up the ObjectGoal task [3] in $\mathcal{E}_{\text{test}}$ for testing different models. Note that $\mathcal{E}_{\text{test}}$ is same as the Gibson tiny validation set (and does not overlap with environments in $\mathcal{E}_{\text{train}}$ or $\mathcal{E}_{\text{video}}$), and comes with human-verified annotations for semantic classes. We use these semantic annotations to set up the ObjectGoal task for 5 categories: bed, chair, couch, dining table, and toilet. We sample 1075 test episodes, equally distributed among these 5 classes. For each episode, the agent is initialized at the starting location, and asked to go to the chosen object category. An episode is considered successfully solved if the agent reaches within $1m$ of *any* instance of the target category. We report both the success rate and SPL [3]. Minimum geodesic distance to *any* instance of the target category, is used as the reference path length for computing SPL. We consider two settings: *Oracle Stop* (episode is automatically terminated and deemed successful when the agent is within $1m$ of the target category), and *Policy Stop* (agent needs to indicate that it has reached the goal). We report results along with a 90% bootstrap confidence interval.

4.1 Implementation Details

Action Grounding. Inverse model ψ processes RGB images I_t and I_{t+1} using a ResNet-18 model [29], stacks the resulting convolutional feature maps, and further processes using 2 convolutional layers, and 2 fully connected layers to obtain the final prediction for the intervening action. We train ψ on 40K interaction frames gathered by randomly executing actions in $\mathcal{E}_{\text{train}}$. This is an easy learning task, we obtain close to 96% classification accuracy on a held-out validation set. We use this inverse model to pseudo-label video dataset $\mathcal{V}_{\text{yt}}$ and $\mathcal{V}_{\text{syn}}$ to obtain $\hat{\mathcal{V}}_{\text{yt}}$ and $\hat{\mathcal{V}}_{\text{syn}}$.

Object Detectors. We use Mask RCNN [28] trained on MS-COCO dataset [39] as our detector D_{coco}. Frames with detections with score in the top 10% are labeled as +1 reward frames. D_{coco} also predicts a foreground mask for each detection. We use it to evaluate a stopping criterion at test time.

Q-Learning. We represent our Q-function with ResNet 18 models, followed by 1 convolutional layer, and 2 fully connected layers with ReLU non-linearities. We use Double DQN (to prevent chronic over-estimation [64]) with Adam [34] for training the Q-networks, and set $\gamma = 0.99$. As our reward is bounded between 0 and 1, clipping target value between 0 and 1 led to more stable training.

Semantic Navigation Policy. High-level policy stores 12 images for each node in the topological graph (obtained by rotating 12 times by 30° each). It uses the learned value function, $f(I, c)$, to score these $12n$ images (for a n node topological graph), and samples the most promising direction for seeking objects of category c. The sampled direction is converted into a short-term goal by sampling a location at an offset of $1.5m$ from the chosen node's location, in the chosen view's direction. Low-level policy [26] uses occupancy maps (built using depth images) [20] with fast marching planning [59] to execute robot actions to reach the short-term goal. It returns control on success / failure / timeout. The High-level policy also factors in the distance to the sampled direction, and score from D_{coco} while sampling directions. Stopping criterion: The agent chooses to stop if D_{coco} fires with confidence $\geq \tau_c$ and median depth value in the predicted mask is $\leq d_c$ distance. More details are provided in Supplementary Section S1.

4.2 Results

Table 1 reports performance on the ObjectGoal task for our method and compares it to other methods for solving this task. An important aspect to consider is the amount and type of supervision being used by different methods. We explicitly note the scale (number of frames, environments) and type (reward signals) of active interaction used by the different methods. For *Policy Stop* setting, for all methods, we found our stopping criterion to work much better than using the method's own stop signal. We use it for all methods. Using only 40K reward-less interaction samples from $\mathcal{E}_{\text{train}}$, along with in-the-wild YouTube videos our proposed method is able to achieve an OS-SPL (Oracle Stop SPL) of 0.53 and PS-SPL (Policy Stop SPL) of 0.22 respectively in the Oracle and Policy stop settings. We put this in context of results from other methods.

Topological Exploration exhaustively explores the environment. It uses our hierarchical policy but replaces $f(I, c)$ with a random function, and ignores scores from D_{coco} to score different directions. As the topological map grows, this baselines systematically and exhaustively explores the environment.

Table 1: Results: Performance for ObjectGoal in novel environments $\mathcal{E}_{\text{test}}$. Details in Section 4.2.

Method	Training Supervision			Oracle Stop		Policy Stop (using D_{coco})	
	# Active Frames	Reward	Other	SPL	Success (SR)	SPL	Success (SR)
Topological Exploration	-	-	-	0.30 ± 0.02	0.67 ± 0.02	0.13 ± 0.01	0.29 ± 0.02
Detection Seeker	-	-	-	0.46 ± 0.02	0.75 ± 0.02	0.19 ± 0.02	0.37 ± 0.02
RL (RGB-D ResNet+3CNN)	100K ($\mathcal{E}_{\text{train}}$)	Sparse	-	0.17 ± 0.01	0.37 ± 0.02		
RL (RGB-D ResNet+3CNN)	10M ($\mathcal{E}_{\text{train}} \cup \mathcal{E}_{\text{video}}$)	Dense	-	0.26 ± 0.02	0.54 ± 0.02		
RL (RGB-D 3CNN)	38M ($\mathcal{E}_{\text{train}} \cup \mathcal{E}_{\text{video}}$)	Dense	-	0.28 ± 0.02	0.57 ± 0.03		
RL (RGB ResNet)	20M ($\mathcal{E}_{\text{train}}$)	Dense	-	0.29 ± 0.02	0.56 ± 0.03	0.08 ± 0.01	0.21 ± 0.02
RL (Depth 3CNN)	38M ($\mathcal{E}_{\text{train}}$)	Dense	-	0.25 ± 0.02	0.52 ± 0.02		
Behavior Cloning	40K ($\mathcal{E}_{\text{train}}$)	-	$\hat{\mathcal{V}}_{\text{yt}}$	0.25 ± 0.02	0.53 ± 0.03	0.08 ± 0.01	0.20 ± 0.02
Behavior Cloning + RL	12M ($\mathcal{E}_{\text{train}}$)	Dense	$\hat{\mathcal{V}}_{\text{yt}}$	0.24 ± 0.02	0.58 ± 0.02		
Our (Value Learning from Videos)	40K ($\mathcal{E}_{\text{train}}$)	-	$\hat{\mathcal{V}}_{\text{yt}}$	$\mathbf{0.53} \pm 0.02$	$\mathbf{0.79} \pm 0.02$	$\mathbf{0.22} \pm 0.02$	$\mathbf{0.39} \pm 0.03$
Behavior Cloning	40K ($\mathcal{E}_{\text{train}}$)	-	$\hat{\mathcal{V}}_{\text{syn}}$	0.36 ± 0.02	0.71 ± 0.02	0.10 ± 0.01	0.26 ± 0.02
Behavior Cloning + RL	12M ($\mathcal{E}_{\text{train}}$)	Dense	$\hat{\mathcal{V}}_{\text{syn}}$	0.24 ± 0.02	0.55 ± 0.03		
Our (Value Learning from Videos)	40K ($\mathcal{E}_{\text{train}}$)	-	$\hat{\mathcal{V}}_{\text{syn}}$	0.48 ± 0.02	0.75 ± 0.02	0.21 ± 0.02	0.38 ± 0.03
Strong Supervision Values	Labeled Maps ($\mathcal{E}_{\text{video}}$)			0.55 ± 0.02	0.81 ± 0.02	0.24 ± 0.02	0.43 ± 0.02
Strong Supervision + VLV (Ours)	Labeled Maps ($\mathcal{E}_{\text{video}}$) + $\hat{\mathcal{V}}_{\text{yt}}$			0.57 ± 0.02	0.82 ± 0.02	0.23 ± 0.02	0.41 ± 0.02

Thus, this is quite a bit stronger than executing random actions (OS-SPL of 0.15). It is able to find objects often (67%), though is inefficient with OS-SPL of 0.30.

Detection Seeker also does topological exploration, but additionally also uses scores from D_{coco} to seek the object once it has been detected. This performs quite a bit better at 0.46 SPL. This indicates that object detectors provide a non-trivial signal for object goal navigation. Even lower confidence detection scores for more distant but partially visible objects will guide the agent in the right direction. Our method captures more out of view context, and consequently does better across all settings.

End-to-end RL. We also compare against many variants of end-to-end RL policies trained via direct interaction. We use the PPO [56] implementation for CNN+GRU policies that are implemented in Habitat [53]. We modify them to work with ObjectGoal tasks (feeding in one-hot vector for target class, modifying rewards), and most importantly adapt them to use ImageNet initialized ResNet-18 models [29] for RGB (given no standard initialization for Depth image, it is still processed using the original 3-layer CNN in Habitat code-base). The fairest comparison is to train using sparse rewards (dense rewards will require environment instrumentation not needed for our method) in $\mathcal{E}_{\text{train}}$ for 40K interaction samples with RGB-D sensors. This unsurprisingly did not work (OS-SPL: 0.17 and OS-SR: 37%). Thus, we aided this baseline by providing it combinations of more environments ($\mathcal{E}_{\text{train}} \cup \mathcal{E}_{\text{video}}$), many times more samples, and dense rewards. Even in these more favorable settings, end-to-end RL didn't perform well. The best model had a OS-SPL of 0.29 and OS-SR of 56% (*vs.* 0.50 and 75% for our method), even when given interaction access to $6\times$ more environments, $250\times$ more interaction, and dense rewards (*vs.* no rewards). This demonstrates the power of our proposed formulation that leverages YouTube videos for learning about spatial layout of environments. Policy stop evaluation is computationally expensive so, we report the score only for the strongest model.

Behavior Cloning (BC) on Pseudo-Labeled Videos $\hat{\mathcal{V}}$. We pre-process the videos to find trajectories that lead to objects of interest (as determined by D_{coco}). We train CNN+GRU models to predict the pseudo-labeled action labels on these trajectories. As this is passive data that has already been collected, we are limited to using behavior cloning wth RGB input as opposed to richer inputs or the more sophisticated DAgger [49]. This is effectively the BCO(0) [62] algorithm. This performs fairly similarly to RL methods and with negligible sample complexity, though still lags far behind our proposed method that utilizes the exact same supervision. Perhaps this is because our proposed method uses pseudo-labeled action indirectly and is more tolerant to mismatch in action space. In contrast, behavior cloning is critically reliant on action space similarity. This is brought out when we use $\hat{\mathcal{V}}_{\text{syn}}$ instead of $\hat{\mathcal{V}}_{\text{yt}}$ where the action space is more closely matched. Behavior cloning performs much better at 0.36 OS-SPL, though our method still performs better than all the baselines even when trained on videos in $\hat{\mathcal{V}}_{\text{syn}}$.

Behavior Cloning+RL. We also experimented with combining behavior cloning and RL. We use the behavior cloning policies obtained above, and finetune them with RL. For the same reasons as above, this policy is limited to use of RGB inputs. When finetuning from behavior cloning policy trained on

$\hat{\mathcal{V}}_{\text{yt}}$ we found performance to remain about the same (OS-SPL 0.24). When starting off from a policy trained on $\hat{\mathcal{V}}_{\text{syn}}$, we found the performance to drop to OS-SPL of 0.24. We believe that the dense reward shaped RL may be learning a qualitatively different policy than one obtained from behavior cloning. Furthermore, use of dense rewards for RL, may limit the benefit of a good initialization.

Strong Supervision Value Function. While our focus is on learning purely from passive data, our semantic navigation policy can also be trained using strong supervision obtained using semantically labeled maps. We train $f(I, c)$ to predict 'ground-truth' Q-values computed using the number of steps to the nearest instance of category c on the meshes from environments in $\mathcal{E}_{\text{video}}$. This model is strong at OS-SPL of 0.55. This serves as a very competitive ObjectNav policy in the regime where we allow such strong supervision. Our proposed method that uses significantly less supervision (in-the-wild videos from YouTube *vs.* environment scans) is still close to the performance of this strongly supervised method (OS-SPL 0.53). When we combine the two by training the strongly supervised objective jointly with our Q-learning based objective, performance is even stronger at OS-SPL of 0.57 (significant at a p-value of 0.025).

Thus, in conclusion, value functions learned via our approach from YouTube video tours of indoor spaces are effective and efficient for semantic navigation to objects of interest in novel environments. They compare favorably to competing reinforcement learning based methods, behavior cloning approaches, and strong exploration baselines, across all metrics.

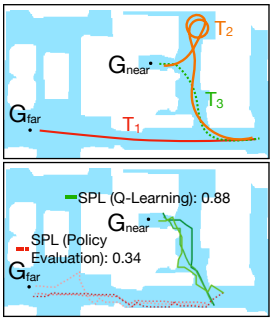
4.3 Ablations

We present ablations when testing policies on $\mathcal{E}_{\text{train}}$ in Oracle Stop setting. Note $\mathcal{E}_{\text{train}}$ was only used to train the inverse model, and not the Q-learning models that we seek to compare. The *base setting* from which we ablate corresponds to training $f(I, c)$ on $\hat{\mathcal{V}}_{\text{syn}}$ with pseudo-labeled actions, D_{coco} based reward labels, and the use of $f(I, c)$ and spatial consistency for sampling short-term goals. This achieves an OS-SPL of 0.40 ± 0.02. We summarize results below, table in supplementary.

We notice only a minor impact in performance when **a)** using true actions as opposed to actions from inverse model ψ (0.41 ± 0.03), **b)** using true detections as opposed to detections from D_{coco} (0.40 ± 0.03), **c)** using true reward locations as opposed to frames from which object is visible as per D_{coco} (0.41 ± 0.03) (the proposed scheme treats frames with high-scoring detections as reward frames as opposed to true object locations), and **d)** using optimal trajectories as opposed to noisy trajectories (0.43 ± 0.03). Albeit on simulated data, this analysis suggests that there is only a minor degradation in performance when using inferred estimates in place of ground truth values.

Perhaps, a more interesting observation is that there is a solid improvement when we additionally use D_{coco} score to sample short-term goals (0.46 ± 0.03). We believe use of D_{coco} produces a more peak-y directional signal when the object is in direct sight, where as differences in $f(I, c)$ are more useful at long-range. Secondly, we found that use of 360° images at training time also leads to a strong improvement (0.47 ± 0.02). We believe use of 360° images at training time prevents *perceptual aliasing* during Q-learning. In the base setting, Q-values can erroneously propagate via an image that looks directly at a wall. Presence of 360° context prevents this. While this is useful for future research, we stick with the base setting as we are limited by what videos we could find on YouTube.

Is action pseudo-labeling necessary? As discussed in Section 3.1, we favored use of Q-learning over action agnostic methods, such as policy evaluation, as this allows us to learn optimal value functions as opposed to value of the policy depicted in the video. To test this, we train different methods in the *branching environment* as shown in the figure on the right (top). Desired goal locations are labeled by G_{near} and G_{far}. We investigate the learned behavior at the branch point B, by initializing the agent at random locations in the circle S. Desired behavior is for agent to reach G_{near}. In departure from all other experiments, here we train and test in the same branching environment. This is a deliberate choice as we seek to understand how different methods interpret the training data.



Videos in this branching environment are a $50 - 49.5 - 0.5\%$ mix of trajectories T_1, T_2, and T_3. T_1 and T_2 are sub-optimal trajectories to reach G_{near} and G_{far} respectively, while T_3 is the optimal trajectory to reach G_{near}. The policy evaluation method doesn't use any action labels, and correctly infers the values for the policy from which videos are sampled. As expected, this causes it to pursue

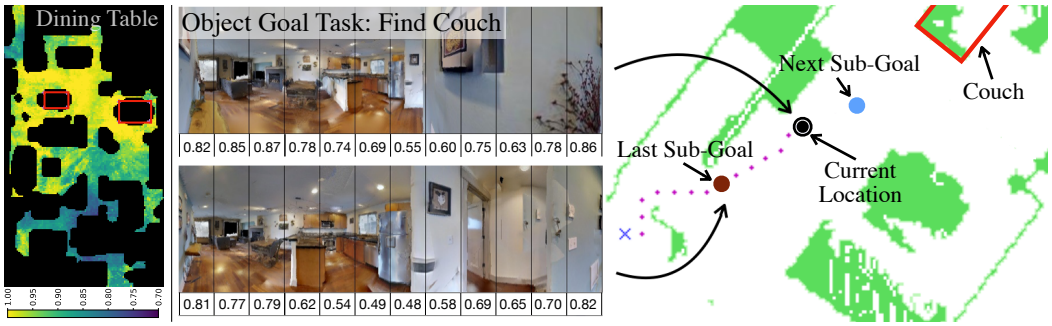


Figure 3: Left figure shows predicted values for reaching a *dining table* at different points on the top-view map in a novel environment. Values are high near the dining tables (denoted by the red boxes), and smoothly bleed out to farther away regions. Right shows a sample execution of our navigation policy finding a *couch* in a novel environment. More in Supplementary.

the sub-optimal goal (red paths in bottom figure). In contrast, Q-learning with pseudo-labeled actions, estimates the *optimal* value function, and consistently reaches G_{near} (green paths).

5 Discussion

We presented a technique to enable learning of semantic cues for finding objects in novel environments from in-the-wild YouTube videos. Our proposed technique employs Q-learning on pseudo-labeled transition quadruples. This allows learning of effective semantic cues even in the absence of action grounding and goal-directed optimal behavior. When coupled with a hierarchical navigation policy, these cues convey the agent to desired objects more effectively than competitive exploration baselines and RL methods at a fraction of interaction cost. In the future, we will test our policies on real robots and extend to other navigation tasks.

Broader Impact

Our specific research in this paper lowers barriers for the training of navigation policies. Instead of needing fully instrumented environments, or large-scale 3D scans, we can now train using video tours of indoor spaces. This significantly expands the environments that such methods can be trained on. Existing datasets [9, 69] used for training current systems have a bias towards expensive houses. This is because sensors and services involved in constructing such scans are expensive. While our current YouTube Walks dataset also has some of this bias, a video tour can be collected merely by using a phone with a camera. This will allow training of navigation policies that will work well in more typical environments, and will democratize the use of learning-based policies for navigation. We also acknowledge that the use of publicly available data from the Internet (in our case YouTube videos) raises questions about privacy and consent. These issues require a broader discussion.

Our broader research aims to improve policies for navigation in unstructured environments. This by itself has numerous desirable applications (such as automated delivery, search and monitoring in hazardous environments, automated crop inspection and mechanical weeding via under-canopy robots). Such applications can save lives, prevent food shortage (by preventing herbicide resistance), and enable development of other automation technologies.

While there are a number of critical applications that our research can potentially enable, we acknowledge that our research falls under automation, and as with all other research in this area, in the future it could replace jobs currently performed by humans. However, this must be viewed in context of the critical applications described above. Resolving or even fully understanding this trade-off will need a much broader discussion.

Acknowledgement: We thank Sanjeev Venkatesan for help with data collection. We also thank Rishabh Goyal, Ashish Kumar, and Tanmay Gupta for feedback on the paper. This material is based upon work supported by NSF under Grant No. IIS-2007035, and DARPA Machine Common Sense.

Gibson dataset license: http://svl.stanford.edu/gibson2/assets/GDS_agreement.pdf

References

[1] Pulkit Agrawal, Ashvin V Nair, Pieter Abbeel, Jitendra Malik, and Sergey Levine. Learning to poke by poking: Experiential learning of intuitive physics. In *NIPS*, 2016. 4

[2] Jean-Baptiste Alayrac, Ivan Laptev, Josef Sivic, and Simon Lacoste-Julien. Joint discovery of object states and manipulation actions. In *ICCV*, 2017. 3

[3] Peter Anderson, Angel Chang, Devendra Singh Chaplot, Alexey Dosovitskiy, Saurabh Gupta, Vladlen Koltun, Jana Kosecka, Jitendra Malik, Roozbeh Mottaghi, Manolis Savva, and Amir Zamir. On evaluation of embodied navigation agents. *CoRR*, 2018. 2, 5, 6

[4] Iro Armeni, Zhi-Yang He, JunYoung Gwak, Amir R Zamir, Martin Fischer, Jitendra Malik, and Silvio Savarese. 3D scene graph: A structure for unified semantics, 3D space, and camera. In *ICCV*, 2019. 5

[5] Yusuf Aytar, Tobias Pfaff, David Budden, Thomas Paine, Ziyu Wang, and Nando de Freitas. Playing hard exploration games by watching youtube. In *NIPS*, 2018. 3

[6] Somil Bansal, Varun Tolani, Saurabh Gupta, Jitendra Malik, and Claire Tomlin. Combining optimal control and learning for visual navigation in novel environments. In *CoRL*, 2019. 3

[7] Andrew G Barto and Sridhar Mahadevan. Recent advances in hierarchical reinforcement learning. *Discrete event dynamic systems*, 2003. 3

[8] Sean L Bowman, Nikolay Atanasov, Kostas Daniilidis, and George J Pappas. Probabilistic data association for semantic slam. In *ICRA*, 2017. 3

[9] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3D: Learning from RGB-D data in indoor environments. *CoRR*, 2017. 9

[10] Devendra Singh Chaplot, Dhiraj Gandhi, Saurabh Gupta, Abhinav Gupta, and Ruslan Salakhutdinov. Learning to explore using active neural mapping. In *ICLR*, 2020. 3, 14

[11] Devendra Singh Chaplot, Ruslan Salakhutdinov, Abhinav Gupta, and Saurabh Gupta. Neural topological slam for visual navigation. In *CVPR*, 2020. 2, 3, 13

[12] Tao Chen, Saurabh Gupta, and Abhinav Gupta. Learning exploration policies for navigation. In *ICLR*, 2019. 3

[13] Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Sanja Fidler, Antonino Furnari, Evangelos Kazakos, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, and Michael Wray. Scaling egocentric vision: The epic-kitchens dataset. In *ECCV*, 2018. 3

[14] Abhishek Das, Samyak Datta, Georgia Gkioxari, Stefan Lee, Devi Parikh, and Dhruv Batra. Embodied question answering. In *CVPR*, 2018. 3

[15] Abhishek Das, Georgia Gkioxari, Stefan Lee, Devi Parikh, and Dhruv Batra. Neural modular control for embodied question answering. In *CoRL*, 2018. 3

[16] Raphael Druon, Yusuke Yoshiyasu, Asako Kanezaki, and Alassane Watt. Visual object search by learning spatial context. *IEEE Robotics and Automation Letters*, 5(2):1279–1286, 2020. 3

[17] Debidatta Dwibedi, Jonathan Tompson, Corey Lynch, and Pierre Sermanet. Learning actionable representations from visual observations. *CoRR*, abs/1808.00928, 2018. 3

[18] Ashley D Edwards and Charles L Isbell. Perceptual values from observation. *CoRR*, 2019. 3

[19] Ashley D Edwards, Himanshu Sahni, Yannick Schroecker, and Charles L Isbell. Imitating latent policies from observation. In *ICML*, 2019. 3

[20] Alberto Elfes. Using occupancy grids for mobile robot perception and navigation. *Computer*, 1989. 3, 6, 14

[21] David F. Fouhey, Vincent Delaitre, Abhinav Gupta, Alexei A. Efros, Ivan Laptev, and Josef Sivic. People watching: Human actions as a cue for single-view geometry. In *ECCV*, 2012. 3

[22] Dhiraj Gandhi, Lerrel Pinto, and Abhinav Gupta. Learning to fly by crashing. In *IROS*, 2017. 3

[23] Tanmay Gangwani and Jian Peng. State-only imitation with transition dynamics mismatch. In *ICLR*, 2020. 3

[24] Alessandro Giusti, Jérôme Guzzi, Dan C Cireşan, Fang-Lin He, Juan P Rodríguez, Flavio Fontana, Matthias Faessler, Christian Forster, Jürgen Schmidhuber, Gianni Di Caro, et al. A machine learning approach to visual perception of forest trails for mobile robots. *IEEE Robotics and Automation Letters*, 2015. 3

[25] Daniel Gordon, Aniruddha Kembhavi, Mohammad Rastegari, Joseph Redmon, Dieter Fox, and Ali Farhadi. IQA: Visual question answering in interactive environments. In *CVPR*, 2018. 3

[26] Saurabh Gupta. Classical Mapping and Planning Baseline for Habitat Challenge. https://github.com/s-gupta/map-plan-baseline. Accessed: 28 May 2020. 6, 14

[27] Saurabh Gupta, Varun Tolani, James Davidson, Sergey Levine, Rahul Sukthankar, and Jitendra Malik. Cognitive mapping and planning for visual navigation. *IJCV*, 2019. 2, 3

[28] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *ICCV*, 2017. 4, 6

[29] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 6, 7

[30] Joao F Henriques and Andrea Vedaldi. Mapnet: An allocentric spatial memory for mapping environments. In *CVPR*, 2018. 3

[31] Phillip Isola, Joseph J Lim, and Edward H Adelson. Discovering states and transformations in image collections. In *CVPR*, 2015. 3

[32] Michael I Jordan and David E Rumelhart. Forward models: Supervised learning with a distal teacher. *Cognitive science*, 16(3):307–354, 1992. 4

[33] Elia Kaufmann, Mathias Gehrig, Philipp Foehn, René Ranftl, Alexey Dosovitskiy, Vladlen Koltun, and Davide Scaramuzza. Beauty and the beast: Optimal methods meet learning for drone racing. In *ICRA*, 2019. 3

[34] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *ICLR*, 2015. 6

[35] Kurt Konolige, James Bowman, JD Chen, Patrick Mihelich, Michael Calonder, Vincent Lepetit, and Pascal Fua. View-based maps. *IJRR*, 2010. 3

[36] Benjamin Kuipers and Yung-Tai Byun. A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations. *Journal of robotics and autonomous systems*, 1993. 3

[37] Ashish Kumar, Saurabh Gupta, David Fouhey, Sergey Levine, and Jitendra Malik. Visual memory for robust path following. In *NIPS*, 2018. 3

[38] Ashish Kumar, Saurabh Gupta, and Jitendra Malik. Learning navigation subroutines by watching videos. In *CoRL*, 2019. 2, 3, 4

[39] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft COCO: Common objects in context. In *ECCV*, 2014. 4, 6

[40] Yuxuan Liu, Abhishek Gupta, Pieter Abbeel, and Sergey Levine. Imitation from observation: Learning to imitate behaviors from raw video via context translation. *CoRR*, abs/1707.03374, 2017. 3

[41] Piotr Mirowski, Razvan Pascanu, Fabio Viola, Hubert Soyer, Andrew J Ballard, Andrea Banino, Misha Denil, Ross Goroshin, Laurent Sifre, Koray Kavukcuoglu, et al. Learning to navigate in complex environments. In *ICLR*, 2017. 3

[42] Arsalan Mousavian, Alexander Toshev, Marek Fišer, Jana Košecká, Ayzaan Wahid, and James Davidson. Visual representations for semantic target driven navigation. In *ICRA*, 2019. 3

[43] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. Orb-slam: a versatile and accurate monocular slam system. *IEEE transactions on robotics*, 2015. 14

[44] Andrew Y Ng, Stuart J Russell, et al. Algorithms for inverse reinforcement learning. In *ICML*, 2000. 2

[45] Tai-Long Nguyen, Do-Van Nguyen, and Thanh-Ha Le. Reinforcement learning based navigation with semantic knowledge of indoor environments. In *International Conference on Knowledge and Systems Engineering (KSE)*, 2019. 3

[46] Brahma S. Pavse, Faraz Torabi, Josiah P. Hanna, Garrett Warnell, and Peter Stone. RIDM: reinforced inverse dynamics modeling for learning from a single observed demonstration. *CoRR*, abs/1906.07372, 2019. 3

[47] Andrzej Pronobis. *Semantic mapping with mobile robots*. PhD thesis, KTH Royal Institute of Technology, 2011. 3

[48] Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *CoRR*, abs/1709.10087, 2017. 3

[49] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *AISTATS*, 2011. 3, 7

[50] Fereshteh Sadeghi. DIViS: Domain invariant visual servoing for collision-free goal reaching. In *RSS*, 2019. 3

[51] Fereshteh Sadeghi and Sergey Levine. CAD^{2}RL: Real single-image flight without a single real image. In *RSS*, 2017. 3

[52] Nikolay Savinov, Alexey Dosovitskiy, and Vladlen Koltun. Semi-parametric topological memory for navigation. In *ICLR*, 2018. 3

[53] Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, Devi Parikh, and Dhruv Batra. Habitat: A platform for embodied AI research. In *ICCV*, 2019. 5, 7, 15

[54] Stefan Schaal. Learning from demonstration. In *NIPS*, 1997. 2

[55] Karl Schmeckpeper, Annie Xie, Oleh Rybkin, Stephen Tian, Kostas Daniilidis, Sergey Levine, and Chelsea Finn. Learning predictive models from observation and interaction. *CoRR*, abs/1912.12773, 2019. 3

[56] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017. 7

[57] Pierre Sermanet, Corey Lynch, Yevgen Chebotar, Jasmine Hsu, Eric Jang, Stefan Schaal, and Sergey Levine. Time-contrastive networks: Self-supervised learning from video. In *ICRA*, 2018. 3

[58] Pierre Sermanet, Kelvin Xu, and Sergey Levine. Unsupervised perceptual rewards for imitation learning. In *RSS*, 2017. 3

[59] James A Sethian. A fast marching level set method for monotonically advancing fronts. *Proceedings of the National Academy of Sciences*, 93(4):1591–1595, 1996. 6, 14

[60] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998. 2, 3, 4, 5

[61] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic robotics*, volume 1. MIT press Cambridge, 2000. 3

[62] Faraz Torabi, Garrett Warnell, and Peter Stone. Behavioral cloning from observation. In *IJCAI*, 2018. 3, 4, 7

[63] Faraz Torabi, Garrett Warnell, and Peter Stone. Generative adversarial imitation from observation. In *ICML*, 2019. 3

[64] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double Q-learning. In *Thirtieth AAAI conference on artificial intelligence*, 2016. 6

[65] Christopher John Cornish Hellaby Watkins. Learning from delayed rewards. 1989. 2, 4

[66] Erik Wijmans, Abhishek Kadian, Ari Morcos, Stefan Lee, Irfan Essa, Devi Parikh, Manolis Savva, and Dhruv Batra. DD-PPO: Learning near-perfect pointgoal navigators from 2.5 billion frames. In *ICLR*, 2020. 3

[67] Yi Wu, Yuxin Wu, Aviv Tamar, Stuart Russell, Georgia Gkioxari, and Yuandong Tian. Bayesian relational memory for semantic visual navigation. In *ICCV*, 2019. 3

[68] Kai M Wurm, Armin Hornung, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. Octomap: A probabilistic, flexible, and compact 3d map representation for robotic systems. In *Proc. of the ICRA 2010 workshop on best practice in 3D perception and modeling for mobile manipulation*, volume 2, 2010. 3

[69] Fei Xia, Amir R. Zamir, Zhiyang He, Alexander Sax, Jitendra Malik, and Silvio Savarese. Gibson Env: real-world perception for embodied agents. In *CVPR*, 2018. 5, 9

[70] Wei Yang, Xiaolong Wang, Ali Farhadi, Abhinav Gupta, and Roozbeh Mottaghi. Visual semantic navigation using scene priors. In *ICLR*, 2019. 3

[71] Yuke Zhu, Roozbeh Mottaghi, Eric Kolve, J Lim, and Abhinav Gupta. Target-driven visual navigation in indoor scenes using deep reinforcement learning. In *ICRA*, 2017. 2, 3

S1 Hierarchical Policies for Semantic Navigation

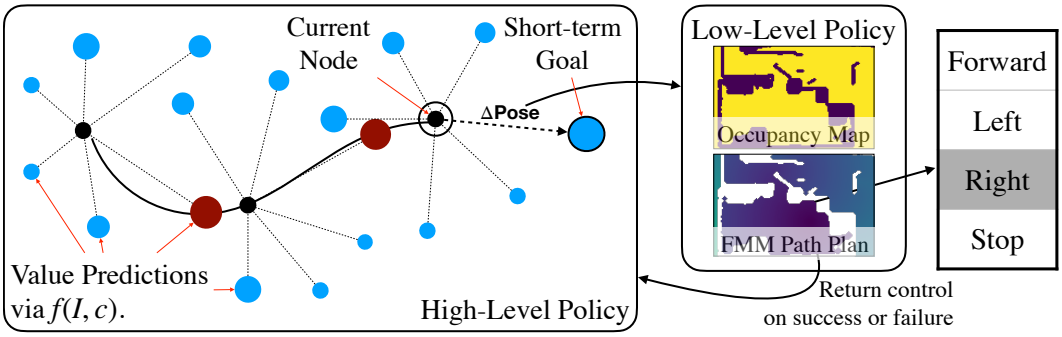


Figure S1: Hierarchical navigation policy. High-level policy does semantic reasoning (using the learned value functions) over images in different directions and outputs short-term goals, that are consumed by the low-level policy. The low-level policy employs classical mapping and planning to achieve the short-term goal, and returns control to the high-level policy if it achieves the short-term goal, or determines it to be infeasible. Black nodes depict nodes stored by the high-level policy in the topological graph, and blue nodes show the value predictions in different directions from each of the black nodes (size indicates predicted value, we use 12 uniformly sampled directions but only show few for clarity). Current location is indicated by the hollow circle. High-level policy outputs the most promising direction to pursue as the short-term goal. Relative offset of this location from the current location (ΔPose) is passed to the low-level policy. Low-level policy incrementally builds occupany map. It uses the fast-marching method to plan a path to the desired short-term goal, and outputs low-level robot actions. Low-level policy returns control on success (reaching the short-term goal), infeasible goal (short-term goal determined to be in occupied space), or timeout.

We use the learned value function $f(I, c)$ from Section 3.1 in a hierarchical navigation policy for semantic navigation. Our hierarchical policy is motivated by Chaplot *et al.* [11], and consists of a *high-level* policy and a *low-level* policy. The high-level policy outputs short-term goals that are achieved by the low-level policy. The high-level policy uses value predictions on images seen so far (at short-term goal locations), to sample a short-term goal in the most promising direction. This short-term goal is expressed as a relative offset from the agent's current location. The low-level policy emits low-level robot actions to navigate to this short-term goal, or returns that the short-term goal is infeasible. This process is repeated, *i.e.*, the high-level policy takes feedback from the low-level policy, along with the image at the agent's new location to sample the next short-term goal. We describe these two policies in more detail below. Figure S1 shows an overview of this navigation policy.

S1.1 High-level policy

The high-level policy, Π builds a hybrid spatial and topological representation. It stores 360° images along with their locations at each short-term goal location. 360° images are obtained by incrementally rotating the agent 12 times by 30° each. High-level policy also stores the value prediction from $f(I, c)$ on these 12 images, for the category of interest c. These 12 values denote the promise of exploring in the different directions for reaching the objects of the desired class. These predicted values are combined with object detector output and a spatial consistency term to give the final score:

$$f_{comb}(I, c) = \lambda_1 f(I, c) + \lambda_2 \underbrace{\mathbb{1}_{\geq 0.5}\left[D_{coco}(I, c)\right] \cdot \left(1 + D_{coco}(I, c)\right)}_{\text{Object Detector}} + 0.05\lambda_3 \underbrace{\max\left(10 - d, 0\right)}_{\text{Spatial Consistency}} \tag{S1}$$

where $f(I, c)$ is the semantic score for the object class of interest c on the image I, $D_{coco}(I, c)$ is the maximum confidence for Mask-RCNN detections of class c in I, d is the *estimated* geodesic distance (based on the current map) of the proposed short-term goal from the current agent position in meters, and $\mathbb{1}_{\geq 0.5}$ is an indicator function that outputs 1 if $D_{coco}(I, c) \geq 0.5$, and 0 otherwise. We set $\lambda_1 = \lambda_2 = \lambda_3 = 1$.

As it is expensive to get these images (it costs 12 steps), we only store these at locations where the short-term policy returns control to the high-level policy (we call these locations as *semantic reasoning locations*, and these are marked in Figure S1 with black dots).

The high-level policy maintains a priority heap of all of these $12N$ values (along with their location and associated direction vectors in the agent's coordinate frame), where N is the number of semantic reasoning nodes currently stored in the topological graph. At each time step, the high-level policy pops the highest of these $12N$ values[1] from the priority heap, and samples k ($= 100$) short-term goals in this direction ($\pm 7^\circ$) that are between $1m$ and $2m$ from the parent node. These k short-term goals are passed onto the low-level policy, which pursues the first of these k goals that is not known to be infeasible, and returns control to the high-level policy if it succeeds, or determines that the sampled short-term goal is infeasible or too far away.

S1.2 Low-level policy

The low-level policy uses metric occupancy maps [20] along with fast-marching method (FMM) path planners [59] to incrementally plan paths to provided short-term goals. The low-level policy filters the provided k goals for feasibility (using the current occupancy map). It takes the first one of these filtered short-term goals, plans a path to it, and outputs planned robot actions. Low-level policy continues to re-plan when the occupancy map updates. Low-level policy executes actions output from the FMM planner. It stops and returns control when **a)** it has reached the goal, **b)** it has already executed enough steps (based on estimate from original FMM computation), or **c)** the short-term goal turns out to be infeasible or much further than originally anticipated (as more of the map becomes visible). We assume access to depth images, and adapt code from the map and plan implementation from [26], to implement the low-level policy.

As our focus is on high-level semantic cues, for simplicity we assume access to perfect agent pose for this hierarchical policy. This can be achieved using additional sensors on the robot (depth cameras, and IMU units), or using a SLAM system [43], or just with RGB images by using learned pose estimators and free space estimators [10].

S1.3 Stopping Criteria

We elaborate on the stopping criteria used for Policy Stop setting. At every semantic reasoning step, we compute a proxy measure for whether we are close to an object of the desired category or not by using the depth image and D_{coco}. For all high-scoring detections for class c from D_{coco} (detection score more than $\tau_c = 0.75$), we approximate the distance to the detected object instance by the median depth value within the predicted instance segmentation mask. If any detected instance is within a distance d_c, the agent emits a stop signal. d_c is a per-category hyper-parameter (as object sizes vary drastically across categories). We set it using 100 episodes sampled in $\mathcal{E}_{\text{train}}$.

As noted in Section 4.2, we found that this hand-crafted stopping criteria also led to best performance for all methods that we compare to (as opposed to using the method's own stopping method). Threshold τ_c was fixed to 0.75 for all methods, while d_c was optimized for each category *for each method* on the same 100 episodes from $\mathcal{E}_{\text{train}}$ using the exact same procedure. For behavior cloning and RL methods, stopping criteria is evaluated at *all* times steps, where as for our method and baselines based on our method, it is evaluated at every semantic reasoning step.

[1] As we keep popping values from the priority heap, there are $11N + 1$ (and not $12N$) entries in the heap at the popping time.

S2 Experimental Details

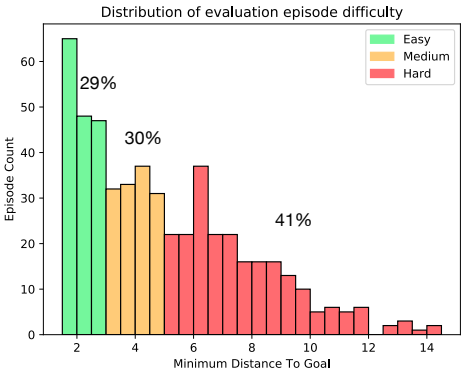
S2.1 Environment Splits

Table S1: List of Gibson environments in different splits. See Section 4 for details.

Split	Environments
$\mathcal{E}_{\text{train}}$	Andover, Annona, Adairsville, Brown, Castor, Eagan, Goodfield, Goodwine, Kemblesville, Maugansville, Nuevo, Springerville, Stilwell, Sussex
$\mathcal{E}_{\text{test}}$	Collierville, Corozal, Darden, Markleeville, Wiconisco
$\mathcal{E}_{\text{video}}$	Airport, Albertville, Allensville, Anaheim, Ancor, Arkansaw, Athens, Bautista, Beechwood, Benevolence, Bohemia, Bonesteel, Bonnie, Broseley, Browntown, Byers, Chilhowie, Churchton, Clairton, Coffeen, Cosmos, Cottonport, Duarte, Emmaus, Forkland, Frankfort, Globe, Goffs, Goodyear, Hainesburg, Hanson, Highspire, Hildebran, Hillsdale, Hiteman, Hominy, Irvine, Klickitat, Lakeville, Leonardo, Lindenwood, Lynchburg, Maida, Marland, Marstons, Martinville, Merom, Micanopy, Mifflinburg, Musicks, Neibert, Neshkoro, Newcomb, Newfields, Onaga, Oyens, Pamelia, Parole, Pinesdale, Pomaria, Potterville, Ranchester, Readsboro, Rogue, Rosser, Shelbiana, Shelbyville, Silas, Soldier, Stockman, Sugarville, Sunshine, Sweatman, Thrall, Tilghmanton, Timberon, Tokeland, Tolstoy, Tyler, Victorville, Wainscott, Willow, Wilseyville, Winooski, Woodbine

S2.2 Difficulty Distribution of Test Episodes

We plot the distribution of difficulty (distance to nearest object of interest) of the evaluation episodes in $\mathcal{E}_{\text{test}}$ in figure on right. We group these episodes into 3 difficulty levels, based on distance to the nearest instance of the target category: *easy* ($\leq 3m$, green), *medium* ($3m$ to $5m$, orange), and *hard* ($5m$ to $15m$, red). In total there were 313 easy, 324 medium and 438 hard episodes. There were 200, 250, 200, 125, 300 episodes each for object categories *Bed*, *Chair*, *Couch*, *Dining Table*, *Toilet* respectively.



S2.3 Generation of $\mathcal{V}_{\text{syn}}$

We use environments in $\mathcal{E}_{\text{video}}$ to render out egocentric navigation tours. We employ a path planner to compute shortest path between *random* pairs of points in each environment. We render out panorama images (4 images: straight facing, left facing, back facing, and right facing, relative to the direction of motion) along these shortest paths and throw out the sequence of actions that were executed, to arrive at the dataset of videos $\mathcal{V}_{\text{syn}}$. To make these tours more realistic, we execute a random action with 20% probability at each time step (and replan accordingly). We sample 300 trajectories in each of the 85 environments. Average trajectory length is 40 steps.

S2.4 More Implementation Details

We note further implementation details for our method and baselines.

1. Topological Exploration and Detection Seeker are implemented by setting $(\lambda_1, \lambda_2, \lambda_3)$ to be $(0, 0, 1)$, and $(0, 1, 1)$ respectively in Eq. S1. This assures a fair comparison between the three methods, and tests the effectiveness of our learned function $f(I, c)$.
2. For End-to-End RL, we experimented with different architectures as noted in Table 1 in the main paper. Baselines as part of Habitat [53] use a 3 layered CNN (denoted by 3CNN and SimpleCNN interchangeably in the main paper) to represent RGB, Depth or RGB-D input. We report performance with this default network (RL (RGB-D 3CNN, RL Depth 3CNN)) in Table 1 in main paper. We found that using a ResNet-18 model (initialized by pre-training on ImageNet) worked better than using this SimpleCNN to represent RGB images. Thus we

additionally also reported performance with ResNet-18 models (RGB-D ResNet-18+3CNN, RGB ResNet-18). For RGB-D models, we could only use ResNet-18 for the RGB part. Depth is still processed through the same 3-layer CNN (as there is no standard initialization for Depth models that is commonly used). Output from ResNet-18 for RGB and 3CNN for Depth were concatenated before feeding into the LSTM model.

3. Our Q-learning models were optimized using Adam with a learning rate of 10^{-4}, $\beta_1 = 0.9$ and $\beta_2 = 0.999$. Model was trained for $300K$ mini-batches of size 16 and the model after the last update was used for experiments.
4. **Architecture of Q-network:** The architecture of the Q-network was based off of ResNet-18. We used a ResNet-18 pretrained on ImageNet removing the last convolution layer and all later layers. We add to the pre-trained head, an additional convolution layer with kernel size 3×3 and 64 channels. After this convolution layer there are 3 fully-connected layers of size $[512, 256, 15]$ respectively. The output of the final layer is reshaped to 3×5 to represent the value of taking each of the 3 possible actions with respect to the 5 possible classes.
5. **Compute Infrastructure:** All experiments were conducted on a single GPU server with 8 GPUs (NVidia 2080 Ti). Model training for our method was done on a single GPU and took 22 hours.

S3 Detailed Results

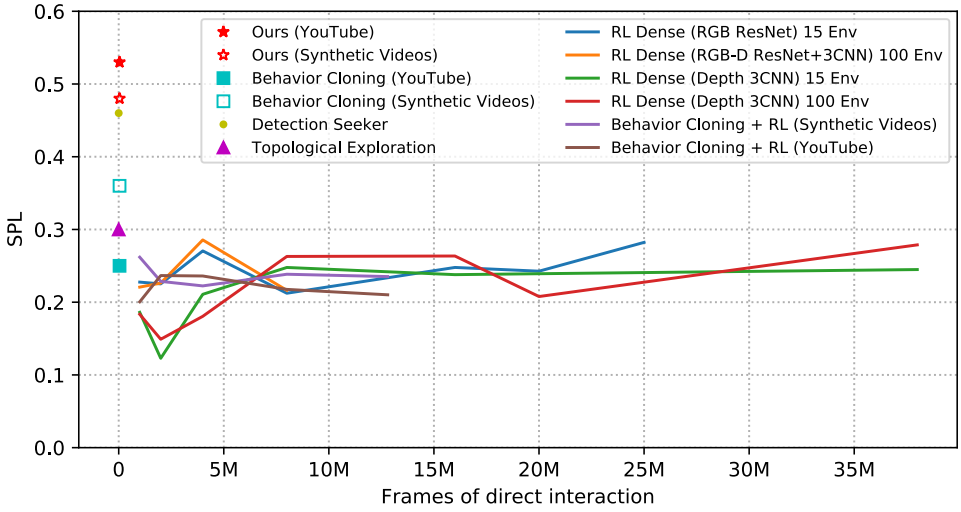


Figure S2: Oracle Stop SPL for various methods against the number of direct interaction samples used.

Table S2: Results: SPL and Success Rate for ObjectGoal wth **Oracle Stop** in novel environments $\mathcal{E}_{\text{test}}$ by episode difficulty. Details in Section 4.2.

Method	Training Supervision			SPL			
	# Active Frames	Reward	Other	Easy	Medium	Hard	Overall
Topological Exploration	-	-	-	0.47 ± 0.03	0.31 ± 0.03	0.16 ± 0.02	0.30 ± 0.02
Detection Seeker	-	-	-	0.73 ± 0.03	0.53 ± 0.03	0.22 ± 0.02	0.46 ± 0.02
RL (RGB-D ResNet+3CNN)	100K ($\mathcal{E}_{\text{train}}$)	Sparse	-	0.30 ± 0.03	0.19 ± 0.03	0.07 ± 0.02	0.17 ± 0.02
RL (RGB-D ResNet+3CNN)	10M ($\mathcal{E}_{\text{train}} \cup \mathcal{E}_{\text{video}}$)	Dense	-	0.42 ± 0.03	0.29 ± 0.03	0.11 ± 0.02	0.26 ± 0.02
RL (RGB-D 3CNN)	38M ($\mathcal{E}_{\text{train}} \cup \mathcal{E}_{\text{video}}$)	Dense	-	0.42 ± 0.03	0.32 ± 0.03	0.15 ± 0.02	0.28 ± 0.02
RL (RGB ResNet)	20M ($\mathcal{E}_{\text{train}}$)	Dense	-	0.40 ± 0.04	0.30 ± 0.03	0.21 ± 0.02	0.29 ± 0.02
RL (Depth 3CNN)	38M ($\mathcal{E}_{\text{train}}$)	Dense	-	0.40 ± 0.04	0.24 ± 0.03	0.15 ± 0.02	0.25 ± 0.02
Behavior Cloning	40K ($\mathcal{E}_{\text{train}}$)	-	$\hat{\mathcal{V}}_{\text{yt}}$	0.44 ± 0.04	0.29 ± 0.03	0.07 ± 0.01	0.25 ± 0.02
Behavior Cloning + RL	12M ($\mathcal{E}_{\text{train}}$)	Dense	$\hat{\mathcal{V}}_{\text{yt}}$	0.41 ± 0.03	0.26 ± 0.03	0.09 ± 0.01	0.24 ± 0.02
Our (Value Learning from Videos)	40K ($\mathcal{E}_{\text{train}}$)	-	$\hat{\mathcal{V}}_{\text{yt}}$	$\mathbf{0.75} \pm 0.03$	$\mathbf{0.63} \pm 0.03$	$\mathbf{0.30} \pm 0.02$	$\mathbf{0.53} \pm 0.02$
Behavior Cloning	40K ($\mathcal{E}_{\text{train}}$)	-	$\hat{\mathcal{V}}_{\text{syn}}$	0.48 ± 0.04	0.34 ± 0.03	0.24 ± 0.02	0.34 ± 0.02
Behavior Cloning + RL	12M ($\mathcal{E}_{\text{train}}$)	Dense	$\hat{\mathcal{V}}_{\text{syn}}$	0.42 ± 0.03	0.22 ± 0.03	0.11 ± 0.02	0.24 ± 0.02
Our (Value Learning from Videos)	40K ($\mathcal{E}_{\text{train}}$)	-	$\hat{\mathcal{V}}_{\text{syn}}$	0.71 ± 0.03	0.55 ± 0.03	0.26 ± 0.02	0.48 ± 0.02
Strong Supervision Values	Labeled Maps ($\mathcal{E}_{\text{video}}$)			0.73 ± 0.03	0.60 ± 0.03	0.33 ± 0.02	0.53 ± 0.02
Strong Supervision + VLV (Ours)	Labeled Maps ($\mathcal{E}_{\text{video}}$) + $\hat{\mathcal{V}}_{\text{yt}}$			0.80 ± 0.03	0.65 ± 0.03	0.35 ± 0.03	0.57 ± 0.02

Method	Training Supervision			Success Rate			
	# Active Frames	Reward	Other	Easy	Medium	Hard	Overall
Topological Exploration	-	-	-	0.89 ± 0.03	0.80 ± 0.04	0.41 ± 0.04	0.67 ± 0.02
Detection Seeker	-	-	-	0.95 ± 0.02	$\mathbf{0.90} \pm 0.03$	0.50 ± 0.04	0.75 ± 0.02
RL (RGB-D ResNet+3CNN)	100K ($\mathcal{E}_{\text{train}}$)	Sparse	-	0.62 ± 0.05	0.41 ± 0.05	0.15 ± 0.03	0.37 ± 0.02
RL (RGB-D ResNet+3CNN)	10M ($\mathcal{E}_{\text{train}} \cup \mathcal{E}_{\text{video}}$)	Dense	-	0.81 ± 0.04	0.65 ± 0.05	0.26 ± 0.03	0.54 ± 0.03
RL (RGB-D 3CNN)	38M ($\mathcal{E}_{\text{train}} \cup \mathcal{E}_{\text{video}}$)	Dense	-	0.79 ± 0.04	0.65 ± 0.04	0.35 ± 0.04	0.57 ± 0.03
RL (RGB ResNet)	20M ($\mathcal{E}_{\text{train}}$)	Dense	-	0.75 ± 0.04	0.59 ± 0.05	0.40 ± 0.04	0.56 ± 0.03
RL (Depth 3CNN)	38M ($\mathcal{E}_{\text{train}}$)	Dense	-	0.73 ± 0.04	0.58 ± 0.05	0.32 ± 0.04	0.52 ± 0.02
Behavior Cloning	40K ($\mathcal{E}_{\text{train}}$)	-	$\hat{\mathcal{V}}_{\text{yt}}$	0.81 ± 0.04	0.68 ± 0.04	0.21 ± 0.03	0.53 ± 0.02
Behavior Cloning + RL	12M ($\mathcal{E}_{\text{train}}$)	Dense	$\hat{\mathcal{V}}_{\text{yt}}$	0.86 ± 0.03	0.69 ± 0.04	0.29 ± 0.03	0.58 ± 0.02
Our (Value Learning from Videos)	40K ($\mathcal{E}_{\text{train}}$)	-	$\hat{\mathcal{V}}_{\text{yt}}$	0.95 ± 0.02	$\mathbf{0.90} \pm 0.03$	$\mathbf{0.58} \pm 0.04$	$\mathbf{0.79} \pm 0.02$
Behavior Cloning	40K ($\mathcal{E}_{\text{train}}$)	-	$\hat{\mathcal{V}}_{\text{syn}}$	0.84 ± 0.03	0.74 ± 0.04	0.55 ± 0.04	0.69 ± 0.02
Behavior Cloning + RL	12M ($\mathcal{E}_{\text{train}}$)	Dense	$\hat{\mathcal{V}}_{\text{syn}}$	0.83 ± 0.03	0.62 ± 0.04	0.31 ± 0.04	0.55 ± 0.02
Our (Value Learning from Videos)	40K ($\mathcal{E}_{\text{train}}$)	-	$\hat{\mathcal{V}}_{\text{syn}}$	$\mathbf{0.96} \pm 0.02$	0.88 ± 0.03	0.51 ± 0.04	0.75 ± 0.02
Strong Supervision Values	Labeled Maps ($\mathcal{E}_{\text{video}}$)			0.95 ± 0.02	0.92 ± 0.03	0.64 ± 0.04	0.81 ± 0.02
Strong Supervision + VLV (Ours)	Labeled Maps ($\mathcal{E}_{\text{video}}$) + $\hat{\mathcal{V}}_{\text{yt}}$			0.98 ± 0.01	0.94 ± 0.02	0.62 ± 0.04	0.82 ± 0.02

Table S3: Results: SPL and Success Rate for ObjectGoal wth **Oracle Stop** in novel environments $\mathcal{E}_{\text{test}}$ by object class. Details in Section 4.2.

Method	Training Supervision			SPL				
	# Active Frames	Reward	Other	Bed	Chair	Couch	Dining Table	Toilet
Topological Exploration	-	-	-	0.35 ± 0.04	0.39 ± 0.03	0.29 ± 0.03	0.27 ± 0.04	0.19 ± 0.03
Detection Seeker	-	-	-	$\mathbf{0.49} \pm 0.05$	0.64 ± 0.04	0.48 ± 0.04	0.53 ± 0.06	0.26 ± 0.03
RL (RGB-D ResNet+3CNN)	100K ($\mathcal{E}_{\text{train}}$)	Sparse	-	0.12 ± 0.03	0.29 ± 0.04	0.16 ± 0.03	0.20 ± 0.05	0.11 ± 0.03
RL (RGB-D ResNet+3CNN)	10M ($\mathcal{E}_{\text{train}} \cup \mathcal{E}_{\text{video}}$)	Dense	-	0.24 ± 0.04	0.37 ± 0.04	0.29 ± 0.04	0.38 ± 0.05	0.09 ± 0.02
RL (RGB-D 3CNN)	38M ($\mathcal{E}_{\text{train}} \cup \mathcal{E}_{\text{video}}$)	Dense	-	0.30 ± 0.04	0.39 ± 0.04	0.26 ± 0.04	0.36 ± 0.05	0.14 ± 0.03
RL (RGB ResNet)	20M ($\mathcal{E}_{\text{train}}$)	Dense	-	0.30 ± 0.04	0.44 ± 0.04	0.24 ± 0.04	0.27 ± 0.05	0.21 ± 0.03
RL (Depth 3CNN)	38M ($\mathcal{E}_{\text{train}}$)	Dense	-	0.29 ± 0.04	0.32 ± 0.04	0.26 ± 0.04	0.32 ± 0.05	0.13 ± 0.02
Behavior Cloning	40K ($\mathcal{E}_{\text{train}}$)	-	$\hat{\mathcal{V}}_{\text{yt}}$	0.24 ± 0.04	0.34 ± 0.04	0.28 ± 0.04	0.36 ± 0.05	0.10 ± 0.02
Behavior Cloning + RL	12M ($\mathcal{E}_{\text{train}}$)	Dense	$\hat{\mathcal{V}}_{\text{yt}}$	0.23 ± 0.04	0.33 ± 0.03	0.25 ± 0.03	0.28 ± 0.04	0.13 ± 0.02
Our (Value Learning from Videos)	40K ($\mathcal{E}_{\text{train}}$)	-	$\hat{\mathcal{V}}_{\text{yt}}$	$\mathbf{0.49} \pm 0.04$	$\mathbf{0.68} \pm 0.03$	$\mathbf{0.60} \pm 0.04$	$\mathbf{0.71} \pm 0.05$	0.32 ± 0.03
Behavior Cloning	40K ($\mathcal{E}_{\text{train}}$)	-	$\hat{\mathcal{V}}_{\text{syn}}$	0.36 ± 0.05	0.45 ± 0.04	0.37 ± 0.04	0.35 ± 0.05	0.20 ± 0.03
Behavior Cloning + RL	12M ($\mathcal{E}_{\text{train}}$)	Dense	$\hat{\mathcal{V}}_{\text{syn}}$	0.21 ± 0.03	0.31 ± 0.03	0.23 ± 0.03	0.35 ± 0.05	0.14 ± 0.03
Our (Value Learning from Videos)	40K ($\mathcal{E}_{\text{train}}$)	-	$\hat{\mathcal{V}}_{\text{syn}}$	0.44 ± 0.04	0.60 ± 0.04	0.48 ± 0.04	0.56 ± 0.06	$\mathbf{0.37} \pm 0.04$
Strong Supervision Values	Labeled Maps ($\mathcal{E}_{\text{video}}$)			0.46 ± 0.04	0.59 ± 0.03	0.57 ± 0.04	0.68 ± 0.05	0.43 ± 0.03
Strong Supervision + VLV (Ours)	Labeled Maps ($\mathcal{E}_{\text{video}}$) + $\hat{\mathcal{V}}_{\text{yt}}$			0.50 ± 0.04	0.69 ± 0.03	0.58 ± 0.04	0.77 ± 0.04	0.43 ± 0.04

Method	Training Supervision			Success rate				
	# Active Frames	Reward	Other	Bed	Chair	Couch	Dining Table	Toilet
Topological Exploration	-	-	-	0.67 ± 0.05	0.85 ± 0.04	0.71 ± 0.05	0.68 ± 0.07	0.48 ± 0.05
Detection Seeker	-	-	-	0.74 ± 0.05	0.92 ± 0.03	0.80 ± 0.05	0.80 ± 0.06	0.57 ± 0.05
RL (RGB-D ResNet+3CNN)	100K ($\mathcal{E}_{\text{train}}$)	Sparse	-	0.36 ± 0.06	0.54 ± 0.05	0.34 ± 0.05	0.43 ± 0.07	0.21 ± 0.04
RL (RGB-D ResNet+3CNN)	10M ($\mathcal{E}_{\text{train}} \cup \mathcal{E}_{\text{video}}$)	Dense	-	0.48 ± 0.06	0.77 ± 0.04	0.62 ± 0.05	0.86 ± 0.05	0.19 ± 0.04
RL (RGB-D 3CNN)	38M ($\mathcal{E}_{\text{train}} \cup \mathcal{E}_{\text{video}}$)	Dense	-	0.66 ± 0.05	0.74 ± 0.05	0.57 ± 0.06	0.74 ± 0.07	0.30 ± 0.04
RL (RGB ResNet)	20M ($\mathcal{E}_{\text{train}}$)	Dense	-	0.64 ± 0.05	0.74 ± 0.05	0.55 ± 0.06	0.50 ± 0.07	0.38 ± 0.04
RL (Depth 3CNN)	38M ($\mathcal{E}_{\text{train}}$)	Dense	-	0.54 ± 0.06	0.71 ± 0.05	0.54 ± 0.06	0.63 ± 0.07	0.30 ± 0.04
Behavior Cloning	40K ($\mathcal{E}_{\text{train}}$)	-	$\hat{\mathcal{V}}_{\text{yt}}$	0.46 ± 0.06	0.74 ± 0.05	0.63 ± 0.06	0.72 ± 0.07	0.24 ± 0.04
Behavior Cloning + RL	12M ($\mathcal{E}_{\text{train}}$)	Dense	$\hat{\mathcal{V}}_{\text{yt}}$	0.59 ± 0.06	0.79 ± 0.04	0.66 ± 0.06	0.67 ± 0.07	0.30 ± 0.04
Our (Value Learning from Videos)	40K ($\mathcal{E}_{\text{train}}$)	-	$\hat{\mathcal{V}}_{\text{yt}}$	$\mathbf{0.76} \pm 0.05$	$\mathbf{0.94} \pm 0.03$	$\mathbf{0.86} \pm 0.04$	$\mathbf{0.91} \pm 0.04$	0.57 ± 0.05
Behavior Cloning	40K ($\mathcal{E}_{\text{train}}$)	-	$\hat{\mathcal{V}}_{\text{syn}}$	0.70 ± 0.05	0.84 ± 0.04	0.70 ± 0.05	0.82 ± 0.06	0.51 ± 0.05
Behavior Cloning + RL	12M ($\mathcal{E}_{\text{train}}$)	Dense	$\hat{\mathcal{V}}_{\text{syn}}$	0.59 ± 0.06	0.77 ± 0.04	0.60 ± 0.06	0.68 ± 0.07	0.25 ± 0.04
Our (Value Learning from Videos)	40K ($\mathcal{E}_{\text{train}}$)	-	$\hat{\mathcal{V}}_{\text{syn}}$	0.70 ± 0.05	0.90 ± 0.03	0.77 ± 0.05	0.82 ± 0.06	$\mathbf{0.62} \pm 0.05$
Strong Supervision Values	Labeled Maps ($\mathcal{E}_{\text{video}}$)			0.72 ± 0.05	0.91 ± 0.03	0.86 ± 0.04	0.94 ± 0.03	0.71 ± 0.04
Strong Supervision + VLV (Ours)	Labeled Maps ($\mathcal{E}_{\text{video}}$) + $\hat{\mathcal{V}}_{\text{yt}}$			0.81 ± 0.05	0.94 ± 0.02	0.82 ± 0.04	0.94 ± 0.03	0.69 ± 0.04

Table S4: Results: SPL and Success Rate for ObjectGoal wth **Policy Stop** in novel environments $\mathcal{E}_{\text{test}}$ by episode difficulty. Details in Section 4.2.

Method	Training Supervision			SPL			
	# Active Frames	Reward	Other	Easy	Medium	Hard	Overall
Topological Exploration	-	-	-	0.22 ± 0.03	0.12 ± 0.02	0.06 ± 0.01	0.13 ± 0.01
Detection Seeker	-	-	-	0.31 ± 0.04	0.22 ± 0.03	0.09 ± 0.01	0.19 ± 0.02
RL (RGB ResNet)	20M ($\mathcal{E}_{\text{train}}$)	Dense	-	0.10 ± 0.02	0.09 ± 0.02	0.05 ± 0.01	0.08 ± 0.01
Behavior Cloning	40K ($\mathcal{E}_{\text{train}}$)	-	$\hat{\mathcal{V}}_{\text{yt}}$	0.16 ± 0.03	0.10 ± 0.02	0.02 ± 0.01	0.08 ± 0.01
Our (Value Learning from Videos)	40K ($\mathcal{E}_{\text{train}}$)	-	$\hat{\mathcal{V}}_{\text{yt}}$	$\mathbf{0.32} \pm 0.04$	$\mathbf{0.29} \pm 0.03$	0.11 ± 0.02	$\mathbf{0.22} \pm 0.02$
Behavior Cloning	40K ($\mathcal{E}_{\text{train}}$)	-	$\hat{\mathcal{V}}_{\text{syn}}$	0.13 ± 0.02	0.12 ± 0.02	0.07 ± 0.01	0.10 ± 0.01
Our (Value Learning from Videos)	40K ($\mathcal{E}_{\text{train}}$)	-	$\hat{\mathcal{V}}_{\text{syn}}$	0.29 ± 0.04	0.23 ± 0.03	$\mathbf{0.13} \pm 0.02$	0.21 ± 0.02
Strong Supervision Values	Labeled Maps ($\mathcal{E}_{\text{video}}$)			0.34 ± 0.04	0.25 ± 0.03	0.15 ± 0.02	0.24 ± 0.02
Strong Supervision + VLV (Ours)	Labeled Maps ($\mathcal{E}_{\text{video}}$) + $\hat{\mathcal{V}}_{\text{yt}}$			0.31 ± 0.04	0.29 ± 0.03	0.13 ± 0.02	0.23 ± 0.02

Method	Training Supervision			Success Rate			
	# Active Frames	Reward	Other	Easy	Medium	Hard	Overall
Topological Exploration	-	-	-	0.43 ± 0.04	0.31 ± 0.04	0.17 ± 0.03	0.29 ± 0.02
Detection Seeker	-	-	-	0.52 ± 0.05	0.43 ± 0.05	0.21 ± 0.03	0.37 ± 0.02
RL (RGB ResNet)	20M ($\mathcal{E}_{\text{train}}$)	Dense	-	0.27 ± 0.04	0.26 ± 0.04	0.12 ± 0.03	0.21 ± 0.02
Behavior Cloning	40K ($\mathcal{E}_{\text{train}}$)	-	$\hat{\mathcal{V}}_{\text{yt}}$	0.36 ± 0.04	0.25 ± 0.04	0.05 ± 0.02	0.20 ± 0.02
Our (Value Learning from Videos)	40K ($\mathcal{E}_{\text{train}}$)	-	$\hat{\mathcal{V}}_{\text{yt}}$	$\mathbf{0.53} \pm 0.04$	$\mathbf{0.48} \pm 0.05$	0.22 ± 0.03	$\mathbf{0.39} \pm 0.02$
Behavior Cloning	40K ($\mathcal{E}_{\text{train}}$)	-	$\hat{\mathcal{V}}_{\text{syn}}$	0.35 ± 0.04	0.28 ± 0.04	0.18 ± 0.03	0.26 ± 0.02
Our (Value Learning from Videos)	40K ($\mathcal{E}_{\text{train}}$)	-	$\hat{\mathcal{V}}_{\text{syn}}$	0.50 ± 0.05	0.42 ± 0.05	$\mathbf{0.26} \pm 0.03$	0.38 ± 0.02
Strong Supervision Values	Labeled Maps ($\mathcal{E}_{\text{video}}$)			0.58 ± 0.05	0.45 ± 0.05	0.30 ± 0.04	0.43 ± 0.02
Strong Supervision + VLV (Ours)	Labeled Maps ($\mathcal{E}_{\text{video}}$) + $\hat{\mathcal{V}}_{\text{yt}}$			0.50 ± 0.05	0.50 ± 0.05	0.27 ± 0.04	0.41 ± 0.02

Table S5: Results: SPL and Success Rate for ObjectGoal wth **Policy Stop** in novel environments $\mathcal{E}_{\text{test}}$ by object class. Details in Section 4.2.

Method	Training Supervision			SPL				
	# Active Frames	Reward	Other	Bed	Chair	Couch	Dining Table	Toilet
Topological Exploration	-	-	-	0.18 ± 0.04	0.17 ± 0.03	0.09 ± 0.02	0.11 ± 0.03	0.09 ± 0.02
Detection Seeker	-	-	-	$\mathbf{0.25} \pm 0.04$	0.25 ± 0.04	0.13 ± 0.03	$\mathbf{0.23} \pm 0.05$	0.14 ± 0.02
RL (RGB ResNet)	20M ($\mathcal{E}_{\text{train}}$)	Dense	-	0.02 ± 0.01	0.15 ± 0.03	0.05 ± 0.02	0.07 ± 0.03	0.08 ± 0.02
Behavior Cloning	40K ($\mathcal{E}_{\text{train}}$)	-	$\hat{\mathcal{V}}_{\text{yt}}$	0.13 ± 0.03	0.12 ± 0.03	0.02 ± 0.01	0.12 ± 0.04	0.05 ± 0.02
Our (Value Learning from Videos)	40K ($\mathcal{E}_{\text{train}}$)	-	$\hat{\mathcal{V}}_{\text{yt}}$	$\mathbf{0.25} \pm 0.04$	$\mathbf{0.28} \pm 0.04$	$\mathbf{0.22} \pm 0.04$	0.20 ± 0.05	0.17 ± 0.03
Behavior Cloning	40K ($\mathcal{E}_{\text{train}}$)	-	$\hat{\mathcal{V}}_{\text{syn}}$	0.06 ± 0.02	0.16 ± 0.03	0.09 ± 0.02	0.14 ± 0.04	0.08 ± 0.02
Our (Value Learning from Videos)	40K ($\mathcal{E}_{\text{train}}$)	-	$\hat{\mathcal{V}}_{\text{syn}}$	0.21 ± 0.04	0.25 ± 0.03	0.13 ± 0.03	0.17 ± 0.04	$\mathbf{0.24} \pm 0.03$
Strong Supervision Values	Labeled Maps ($\mathcal{E}_{\text{video}}$)			0.24 ± 0.04	0.24 ± 0.03	0.22 ± 0.04	0.28 ± 0.05	0.22 ± 0.03
Strong Supervision + VLV (Ours)	Labeled Maps ($\mathcal{E}_{\text{video}}$) + $\hat{\mathcal{V}}_{\text{yt}}$			0.14 ± 0.04	0.31 ± 0.04	0.18 ± 0.04	0.32 ± 0.05	0.24 ± 0.03

Method	Training Supervision			Success Rate				
	# Active Frames	Reward	Other	Bed	Chair	Couch	Dining Table	Toilet
Topological Exploration	-	-	-	0.27 ± 0.05	0.35 ± 0.05	0.26 ± 0.05	0.29 ± 0.07	0.27 ± 0.04
Detection Seeker	-	-	-	0.38 ± 0.06	0.48 ± 0.05	0.23 ± 0.05	$\mathbf{0.37} \pm 0.07$	0.36 ± 0.05
RL (RGB ResNet)	20M ($\mathcal{E}_{\text{train}}$)	Dense	-	0.07 ± 0.03	0.38 ± 0.05	0.12 ± 0.04	0.15 ± 0.05	0.23 ± 0.04
Behavior Cloning	40K ($\mathcal{E}_{\text{train}}$)	-	$\hat{\mathcal{V}}_{\text{yt}}$	0.23 ± 0.05	0.30 ± 0.05	0.06 ± 0.03	0.33 ± 0.07	0.14 ± 0.03
Our (Value Learning from Videos)	40K ($\mathcal{E}_{\text{train}}$)	-	$\hat{\mathcal{V}}_{\text{yt}}$	$\mathbf{0.40} \pm 0.06$	$\mathbf{0.52} \pm 0.05$	$\mathbf{0.36} \pm 0.06$	0.31 ± 0.07	0.32 ± 0.04
Behavior Cloning	40K ($\mathcal{E}_{\text{train}}$)	-	$\hat{\mathcal{V}}_{\text{syn}}$	0.11 ± 0.04	0.43 ± 0.05	0.24 ± 0.05	0.34 ± 0.07	0.20 ± 0.04
Our (Value Learning from Videos)	40K ($\mathcal{E}_{\text{train}}$)	-	$\hat{\mathcal{V}}_{\text{syn}}$	0.30 ± 0.05	0.49 ± 0.05	0.26 ± 0.05	0.34 ± 0.07	$\mathbf{0.43} \pm 0.05$
Strong Supervision Values	Labeled Maps ($\mathcal{E}_{\text{video}}$)			0.34 ± 0.06	0.50 ± 0.05	0.37 ± 0.05	0.54 ± 0.07	0.42 ± 0.05
Strong Supervision + VLV (Ours)	Labeled Maps ($\mathcal{E}_{\text{video}}$) + $\hat{\mathcal{V}}_{\text{yt}}$			0.20 ± 0.05	0.53 ± 0.05	0.30 ± 0.05	0.52 ± 0.08	0.47 ± 0.05

Table S6: We report various ablations of our method, when using automatic stopping behavior, evaluated on $\mathcal{E}_{\text{train}}$. Base setting uses noisy trajectores, action labels from inverse models and panorama images. We ablate these settings. See Section 4.3 for details.

	SPL				Success Rate			
Method	Easy	Medium	Hard	Overall	Easy	Medium	Hard	Overall
Base setting	0.62 ± 0.04	0.42 ± 0.04	0.23 ± 0.03	0.40 ± 0.02	0.95 ± 0.03	0.86 ± 0.05	0.56 ± 0.05	0.75 ± 0.03
True actions	0.61 ± 0.05	0.45 ± 0.05	0.25 ± 0.03	0.41 ± 0.03	0.94 ± 0.03	0.86 ± 0.05	0.51 ± 0.05	0.73 ± 0.03
True detections	0.62 ± 0.05	0.45 ± 0.05	0.22 ± 0.03	0.40 ± 0.03	0.95 ± 0.03	0.86 ± 0.05	0.48 ± 0.05	0.72 ± 0.03
True rewards	0.64 ± 0.05	0.46 ± 0.05	0.21 ± 0.03	0.41 ± 0.03	0.95 ± 0.03	0.86 ± 0.05	0.48 ± 0.05	0.72 ± 0.03
No noise in videos	0.65 ± 0.05	0.46 ± 0.04	0.25 ± 0.03	0.43 ± 0.03	0.95 ± 0.03	0.92 ± 0.04	0.59 ± 0.05	0.78 ± 0.03
D_{coco} score	0.73 ± 0.04	0.48 ± 0.05	0.26 ± 0.03	0.46 ± 0.03	0.98 ± 0.02	0.88 ± 0.05	0.58 ± 0.06	0.78 ± 0.03
Train on 360° videos	0.66 ± 0.04	0.51 ± 0.05	0.32 ± 0.03	0.47 ± 0.02	0.98 ± 0.02	0.92 ± 0.04	0.66 ± 0.05	0.82 ± 0.03

S4 Visualizations

S4.1 Value Predictions on Panorama

Figure S3: Example panoramas from novel environments with scores from our value network. Scores for each object class (Bed, Chair, Couch, Dining Table, and Toilet) are reported. We can see that value is high in the likely direction of objects even if the object is not directly visible.

Figure S4: Example panoramas from novel environments with scores from our value network. Scores for each object class (Bed, Chair, Couch, Dining Table, and Toilet) are reported. We can see that value is high in the likely direction of objects even if the object is not directly visible.

S4.2 Executed Trajectories

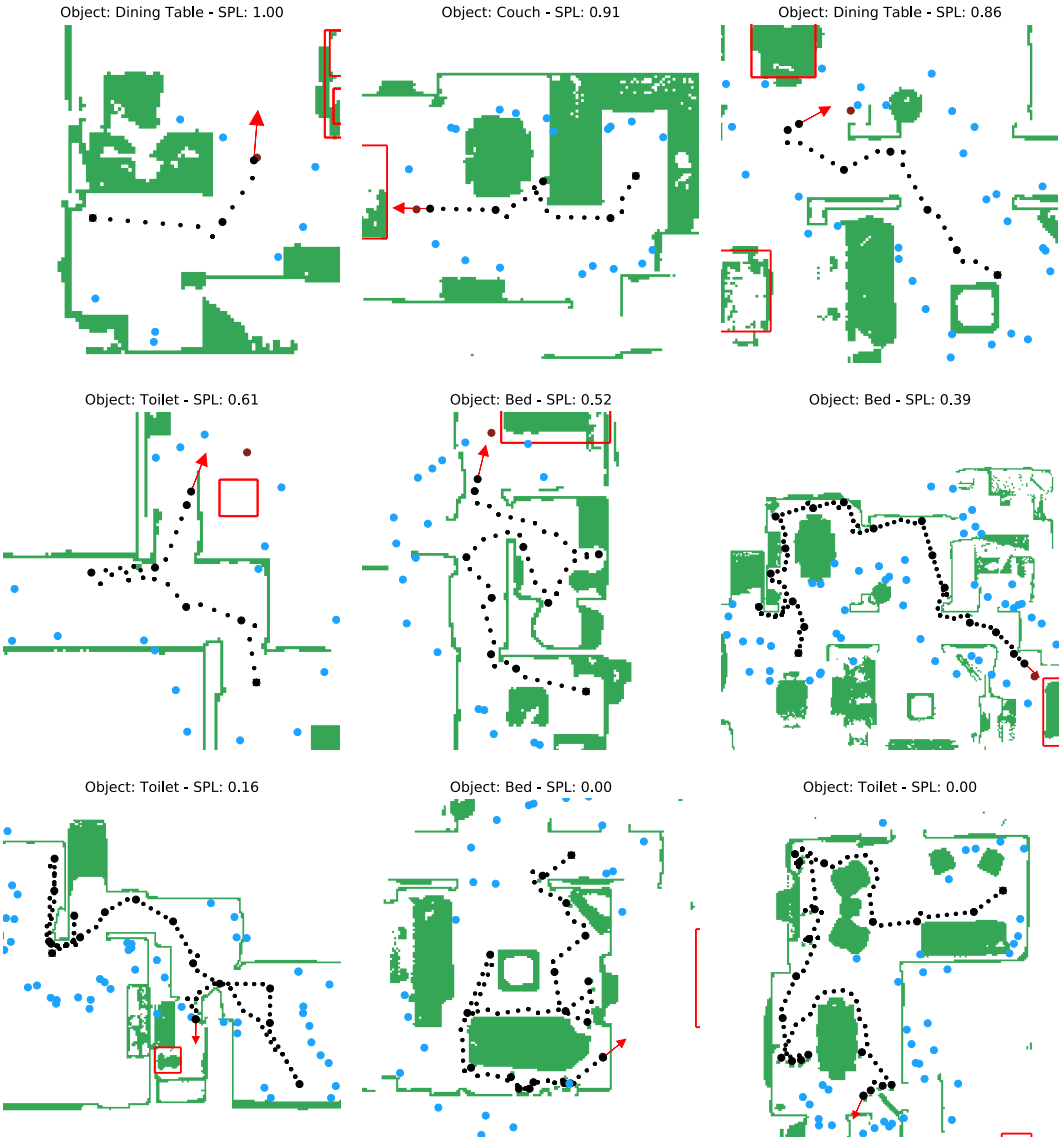


Figure S5: Example trajectores from our method navigating in novel environments, sorted by SPL (first few show successes, last few show failures). The black path indicates the trajectory taken by the agent. A blue circle indicates potential short-term goal, and a red rectangle indicates the object goal.

S4.3 $\mathcal{E}_{\text{test}}$ Problem Setup Visualization

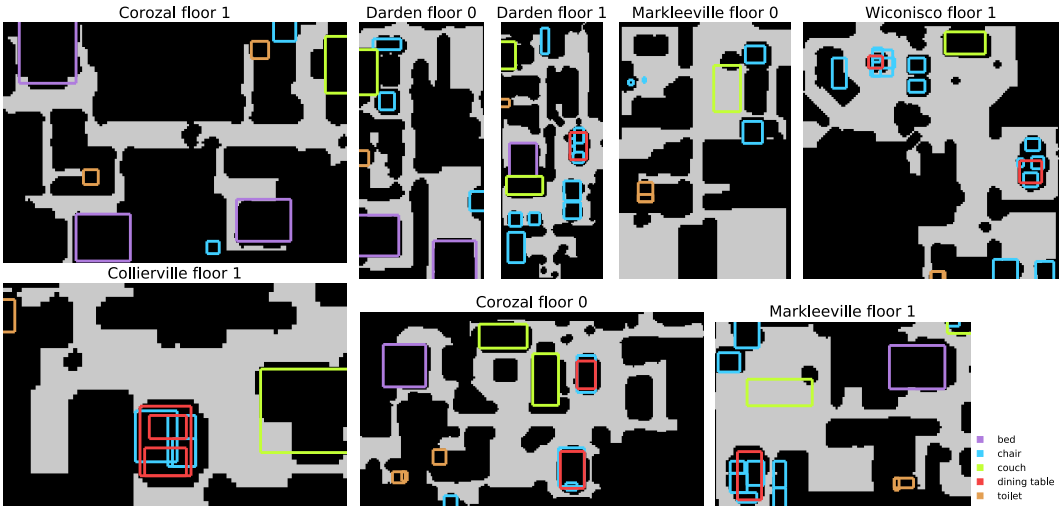


Figure S6: Top-down maps of selected floors from the $\mathcal{E}_{\text{test}}$ environments. We also show ground truth object locations. Agent does not have access to any of these maps or ground truth object locations. Visualizations here are provided only to show the difficulty and realism of our problem setup.

S4.4 Predicted Values on Held-out Environments

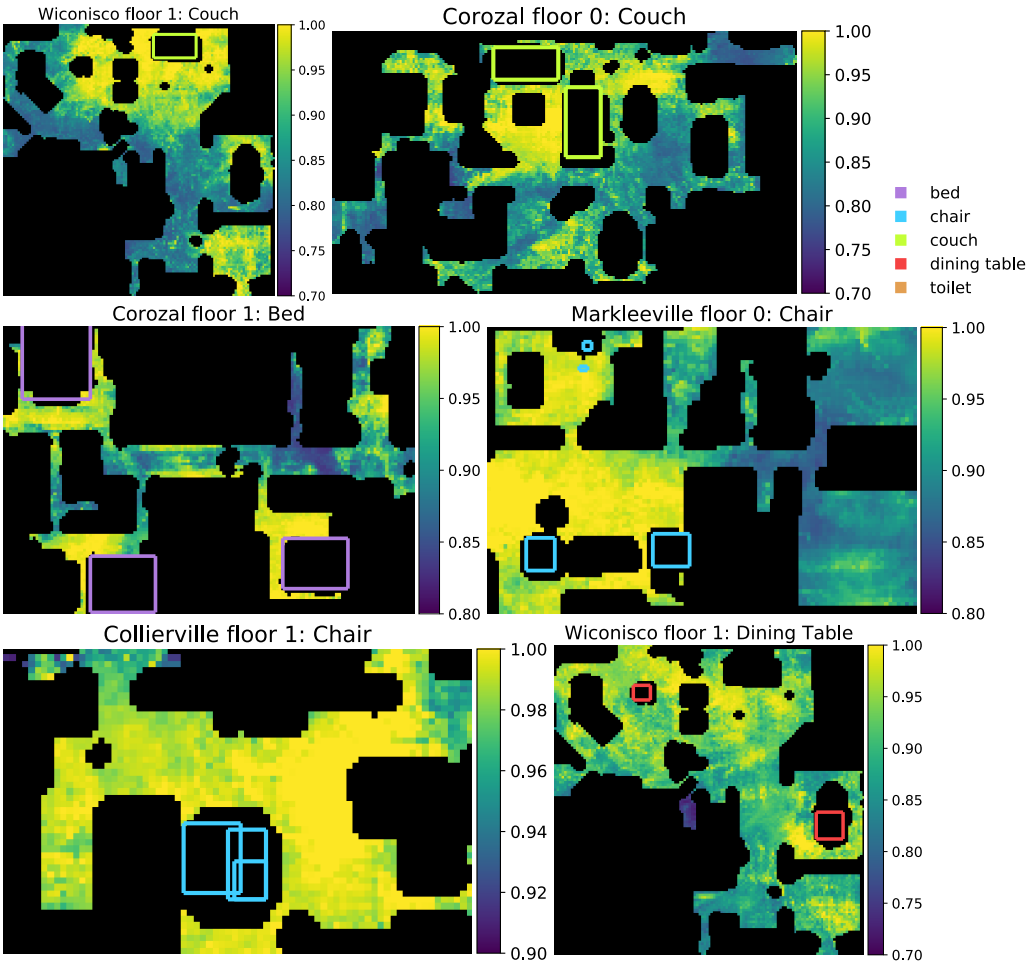


Figure S7: Maps representing the value of different locations in novel environments as predicted by our method trained on $\hat{\mathcal{V}}_{\text{syn}}$. We can see that high value regions fall off smoothly as the distance from object goals increases.

S4.5 Value in Branching Environment

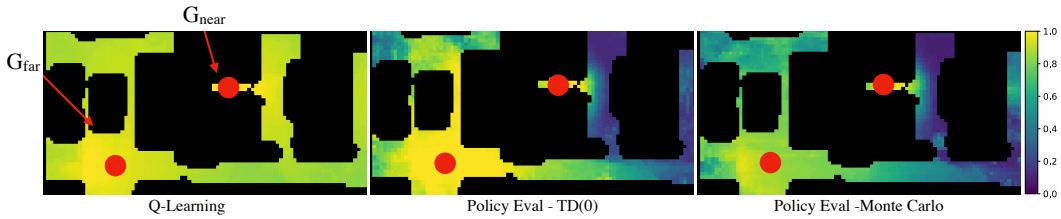


Figure S8: The predicted value in the branching environment using models trained with Q-learning, and policy evaluation via TD(0) and Monte Carlo. We see that the policy evaluation methods drastically under estimate the value in the optimal direction at the branch point. This leads to sub-optimal policies for those methods while the Q-learning based value function finds the optimal trajectory. See Section 4.3 for details.