# Learning Navigation Subroutines from Egocentric Videos

**Ashish Kumar**[1]   **Saurabh Gupta**[3]   **Jitendra Malik**[1,2]
[1]UC Berkeley   [2]Facebook AI Research   [3]UIUC
ashish_kumar@berkeley.edu, saurabhg@illinois.edu, malik@eecs.berkeley.edu

**Abstract:** Planning at a higher level of abstraction instead of low level torques improves the sample efficiency in reinforcement learning, and computational efficiency in classical planning. We propose a method to learn such hierarchical abstractions, or subroutines from egocentric video data of experts performing tasks. We learn a self-supervised inverse model on small amounts of random interaction data to pseudo-label the expert egocentric videos with agent actions. Visuomotor subroutines are acquired from these pseudo-labeled videos by learning a latent intent-conditioned policy that predicts the inferred pseudo-actions from the corresponding image observations. We demonstrate our proposed approach in context of navigation, and show that we can successfully learn consistent and diverse visuomotor subroutines from passive egocentric videos. We demonstrate the utility of our acquired visuomotor subroutines by using them as is for exploration, and as sub-policies in a hierarchical RL framework for reaching point goals and semantic goals. We also demonstrate behavior of our subroutines in the real world, by deploying them on a real robotic platform. Project website: https://ashishkumar1993.github.io/subroutines/.

**Keywords:** Subroutines, Passive Data, Hierarchical Reinforcement Learning

## 1  Introduction

Every morning, when you decide to get a cup of coffee from the kitchen, you go down the hallway, turn left into the corridor and then enter the room on the right. Instead of deciding the exact muscle torques, you reason at this higher level of abstraction by composing these reusable lower level *visuomotor subroutines* to reach your goal. These visuomotor subroutines are classically known as operators in STRIPS planning [1], or more recently as options in RL [2]. Once these subroutines are learned, they can be composed to solve novel tasks, e.g. exiting the building, finding an object, etc, enabling an agent to quickly learn new tasks by simply learning how to compose them together.

These subroutines, or short-horizon policies with consistent behavior, can be manually designed as done in classical robotics or STRIPS, or can be learned through interaction by training a hierarchical agent through reward-based reinforcement learning. Learning through environment interactions is extremely slow, making it prohibitively expensive to operationalize in the real world. We propose a third way of learning these subroutines by using imitation learning on egocentric videos of experts performing tasks. We expect that these videos contain subroutines that have been appropriately combined to solve some tasks, and an appropriate clustering algorithm can be used to isolate and extract these subroutines. For example, in indoor navigation, such clusters could be exiting doors, walking down hallways; and for driving, they could be following a lane, changing lanes, etc. Once isolated, these subroutines just need to be fine tuned through reward-based RL for downstream tasks, which is very sample efficient, as we show in our experiments.

To imitate an expert at a visuomotor task, we need to know both the perceptual input to the expert and the action taken. One way to do this is by instrumenting the agent to collect the perceptual input as well as the action executed, as done in autonomous driving [3]. However, this limits the scalability of the data collection procedure. To scale it up, we could instead learn from videos of people performing tasks uploaded on websites such as YouTube. Such videos fall in two categories:
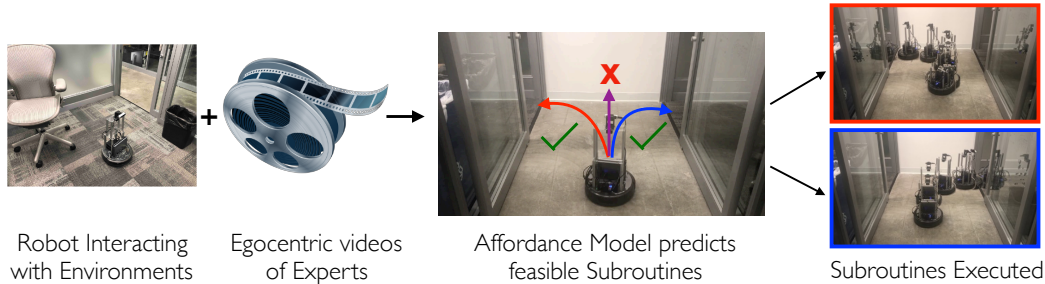
**Figure 1: Approach Overview:** We propose an approach that combines learning from direct environmental interaction, with learning from first-person videos collected over the Internet. Inverse models built using a small number of environmental interactions are used to interpret videos, and learn *affordances* (what can I do) and *subroutines* (how can I do it). Affordances predict which subset of subroutines are feasible given the current image. These subroutines can then be executed in novel environments.

first person (egocentric) or third person video. Third person videos have the benefit of having action information but don't have the perceptual input. Skills learned from such videos don't depend on the perceptual input [4, 5]. But our focus is on navigation tasks for which the perceptual input is important. We thus face the opposite challenge when using egocentric video of a person performing a task (*e.g.* biking with a head mounted GoPro camera). The perceptual input to the agent is available, but the action information is typically missing. In this paper, we will address this case and demonstrate our technique in the navigation domain.

We start with egocentric videos of experts navigating to achieve some task unknown to us. Given these videos, we want our robot to learn meaningful and useful subroutines.

We obtain action labels for the egocentric videos by training a self-supervised inverse model on random interaction data. Egocentric videos can then be pseudo labeled by running the inverse model on consecutive image pairs. Note that the action space of the experts might be different from the action space of the robot (for example if we were to download egocentric navigation videos available online). Hence, these pseudo-action labels are not the actual action taken, but an action imagined by the robot to make the transition between the observations as closely as possible. (Section 3).

Once we have the pseudo action labels, we need to label the subroutines in the videos and learn a controller which can be used in downstream tasks. For this, we slice up a trajectory into smaller sub-trajectories. The slicing length is a hyper parameter, that controls the complexity of the subroutines learned (longer sub-trajectories will lead to more complex subroutines). We then encode each sub-trajectory into a discrete latent variable which should be predictive of the action given the video frame, for every frame in the sub-trajectory. (Section 4).

To effectively use the learned subroutines in downstream tasks, we must also infer which subroutines can be applied where. For this, we additionally train an affordance model to predict which subroutines *can* be invoked for a given input image from our repertoire of learned subroutines. We do this by predicting the inferred one-hot latent encoding of the trajectory from the first image. (Section 4).

We evaluate our learned subroutines and the affordance model on downstream navigation tasks, which are unknown to our method during the subroutine learning phase. We show that our learned subroutines can be composed together for zero-shot exploration in novel environments with a 50% improvement in exploration over several learning and non-learning baselines. We also evaluate our learned subroutines on downstream point navigation and area goal tasks. We fine-tune our affordance model and subroutines through reward-based RL and observe a $4\times$ improvement in learning sample complexity over alternate initializations. (Section 6).

## 2 Related Work

**Classical Navigation.** Classical approaches to navigation employ geometric reasoning to solve the task [6, 7]. While most works optimize in the base action space of the agent, few works employ hand-crafted motion primitives to speed up planning [8]. Dynamic Motion Primitives (DMPs) propose a

framework for specifying macro actions [9], which can be learned from a demonstration [10] for a specific macro action. However, the set of macro actions are still manually specified. In contrast, these behaviors automatically emerge as a consequence of our algorithm.

**Learned Navigation.** Recent learning based-efforts use reinforcement learning or imitation learning to learn policies for solving specific locomotion, navigation or manipulation tasks [11, 12, 13, 14, 15, 16, 17]. While these works learn to leverage high-level semantics, they still directly operate in the base action space of the robot. Learned skills are task and environment specific, and it takes a large number of interaction samples to even solve the same task in a new environment (in navigation [11, 12] as well as in manipulation [16, 17]). To address this, some works [18] use intrinsic rewards such as prediction error. However, these approaches don't distill out composable skills to solve novel tasks. Works like [19] distill out composable skills but do not scale to realistic setups as we show in our experiments. Moreover, as all training signal is derived from interaction with the environment, skill acquisition is extremely expensive. In fact, our experiments show that our use of passive videos for learning skills is more sample efficient and results in better performance than such purely interactive approaches.

**Learning from State-Action Trajectories.** Several works use learning from demonstration in scenarios where they have access to both the observations and the ground truth actions to solve the task [20, 21]. Works like [22] extend these formulations to work with trajectory collections that have multiple modes. However, this line of work relies on ground truth action labels. In contrast, we only assume observation data (without paired actions), and evaluate on novel tasks in novel environments.

**Learning from State Only Trajectories.** Contemporary works [23, 24, 25] study the problem of learning from state only trajectories, similar to our work here. However, all of these works only study the scenario where the agent solves the task in *exactly* the same environment that they have state-only demonstrations for. In contrast, we do not assume access to environments for which we have videos for, making learning more challenging and rendering these past techniques ineffective. Additionally, our goal is to learn subroutines that work in previously unseen environments, which goes beyond the focus of these works. Works like [26, 27, 28] focus on imitation from visual data and learn a monolithic policy from expert data for the task at hand. We focus on learning composable subroutines which can then be used in several downstream tasks.

**Sub-policies and Options in Hierarchical RL.** Hierarchical RL is an active area of research [29, 30, 2], with a number of recent papers (such as [31, 32]). These works acquire sub-policies in a top-down manner while interacting with the environment to solve a reward based task. Our approach on the other hand investigates a bottom-up development of subroutines, and can learn from relatively inexpensive unlabelled passive data. Our learned subroutines are complementary to these frameworks and can be used to initialize any of these top-down HRL methods to accelerate learning.

**Affordance Learning from Videos.** Researchers have studied affordance learning from Internet videos [33] by leveraging YouTube videos to learn about affordances. While this is a great first step, it does not learn a controller that can be used in downstream tasks. In contrast, we learn a controller for each subroutine for the the specific robot at hand, allowing immediate deployment.

## 3 Pseudo-Labeling Egocentric Expert Videos

We need action labels on expert videos to learn a controller for downstream tasks. We build a self-supervised inverse model which takes two consecutive image observations, $o_t$ and $o_{t+1}$, and predicts the action $\hat{a}$ which the agent took to transition from $o_t$ to $o_{t+1}$ . This inverse model is then used to pseudo-label egocentric videos of experts. However, since the videos may come from diverse sources (e.g. internet videos), the expert uploading the video may have a different action space than $S$. To handle this mismatch, we expect the inverse model to predict the action which $S$ *should have taken* to go from $o_t^e$ to $o_{t+1}^e$ and not the action actually taken by the expert.
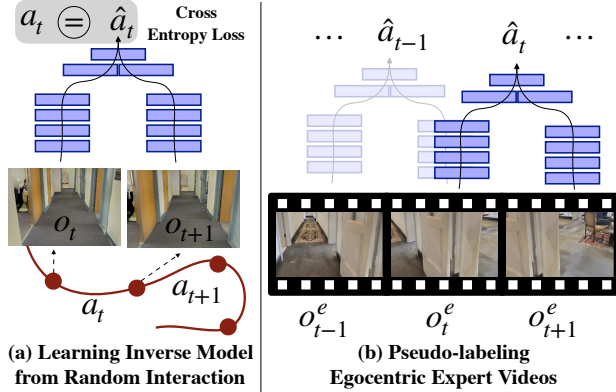
### 3.1 Inverse Model

We first build a self-supervised one-step inverse model $\psi$ [34, 35] for the agent from random interaction data in an environment (simulation environment or the real world). More concretely, given a pair of consecutive image observations, $o_t$ and $o_{t+1}$, we train the model $\psi$ to predict action $a_t$ that was executed to transition from image $o_t$ to $o_{t+1}$ as follows: $\hat{a}_t = \psi(o_t, o_{t+1})$. The agent $S$

collects data $\{\ldots, o_t, a_t, o_{t+1}, a_{t+1} \ldots\}$ to train $\psi$ by sampling $a_t$ uniformly from $\{\texttt{left}, \texttt{right}, \texttt{forward}\}$ and executing it in the environment conveying it from $o_t$ to $o_{t+1}$.

## 3.2 Pseudo-Labeling Video Data

We then use this learned inverse model $\psi$ to pseudo-label the dataset $\mathcal{D}$ of egocentric expert videos. Given a sequence of images $\{o_1^e, o_2^e, \ldots\}$ from a video, we evaluate $\psi$ on consecutive pairs of images to obtain $\hat{a} = \psi(o_t^e, o_{t+1}^e)$. We use observations as a means of implicitly mapping equivalent actions between agents. This generates a pseudo labeled dataset $\hat{\mathcal{D}}$, that contains image action sequences, $\{o_1^e, \hat{a}_1, o_2^e, \hat{a}_2, \ldots, o_T^e\}$.



(a) Learning Inverse Model from Random Interaction

(b) Pseudo-labeling Egocentric Expert Videos

# 4 Visuomotor Subroutines and Affordances

We formally define subroutines and the affordance model for an agent $S$ as $\langle \alpha, \{\pi_i\}_{i=1..N} \rangle$, where $N$ is the

**Figure 2: Pseudo-labeling:** **(a)** We execute random actions in an environment to obtain image-action sequences $\{\ldots, o_t, a_t, o_{t+1} \ldots\}$. We use triplets $(o_t, a_t, o_{t+1})$ to train an inverse model to predict action $a_t$ given consecutive images $o_t$, $o_{t+1}$. **(b)** We use this inverse model to pseudo-label egocentric videos of navigating agents. Grey circles with $=$ sign represent cross-entropy loss.

number of subroutines available to the $S$, $\pi_i$ is the $i^{th}$ subroutine and $\alpha$ is the affordance model. The affordance model predicts the probability distribution $\tilde{p}_t$ given an input observation $o_t$, where $(\tilde{p}_t)_i$ is the probability that $\pi_i$ is applicable given the observation $o_t$. Each subroutine $\pi_i$ is a closed loop policy which takes the current observation $o_t$ and predicts a distribution over actions $\tilde{a}_t$. Thus, $\tilde{p}_t = \alpha(o_t)$ and $\tilde{a}_t = \pi_i(o_t)$.

We isolate these subroutines from $\hat{\mathcal{D}}$ by clustering them to improve the action prediction accuracy of the visuomotor trajectories in $\hat{\mathcal{D}}$. Intuitively, if the observation contains a T-junction with a possible left and right turn, they need to be clustered separately to unambiguously predict the future given the observation and the cluster id.

To effectively use these subroutines, we learn another model to infer which subroutines are applicable in what scenario. For example, given an image of a hallway with two doors, one on the left and other on the right, the model should learn to assign high probabilities to both *go into left door* and *go into right door* subroutines, whereas when there are no doors, it should simply peak on the subroutine *go down a hallway*.

## 4.1 Learning Formulation

We slice up each trajectory (from the pseudo-labeled dataset $\hat{\mathcal{D}}$) into smaller overlapping trajectories of fixed length $T$, where $T$ is a hyper-parameter which determines the complexity of the subroutines.[1] We encode actions in each sub-trajectory into a discrete latent vector $z$. This $z$ is then used to predict actions corresponding to different frames in the video. We implement the trajectory encoder as network $f$ and the subroutines as a network $\pi$ parametrized by the subroutine-id $z$:

$$e = f(\hat{a}_1, \hat{a}_2, \ldots, \hat{a}_T) \tag{1}$$

$$z \sim \text{softmax}(e) \tag{2}$$

$$\tilde{a}_t, h_{t+1} = \pi(o_t, z, h_t) \quad \forall t \in \{1 \ldots T-1\} \tag{3}$$

where state $h_t$ and $h_{t+1}$ are the current and updated hidden states respectively, $o_t$ is the current observation, and $z$ is a discrete latent vector which specifies the subroutine to invoke. We train the affordance model $\alpha$ to predict the subroutine id given the *first* image of the video sequence:

$$\tilde{z} = \alpha(o_1). \tag{4}$$

---

[1] A smaller $T$ leads to simpler subroutines. We show ablations over $T$ in supplementary.

At test time, $\alpha$ takes the current observation as input and predicts which subroutines are applicable, and $\pi$ then executes the selected subroutine in a closed loop, executing the predicted action $\tilde{a}_t$ and receiving the next observation $o_{t+1}$ as it proceeds.

**Training**: Both the networks, $\pi$ and $f$, are jointly optimized to maximize the likelihood of the pseudo-labeled action sequence. $\alpha$ is optimized to maximize the likelihood of $z$ given first observation of the video sequence. The subroutine id $z$ is sampled from the trajectory encoding $e$ through a Gumbel-Softmax distribution [36]. This allows estimating gradients for parameters of $f$ despite the sampling. Figure 3 shows the network diagram.
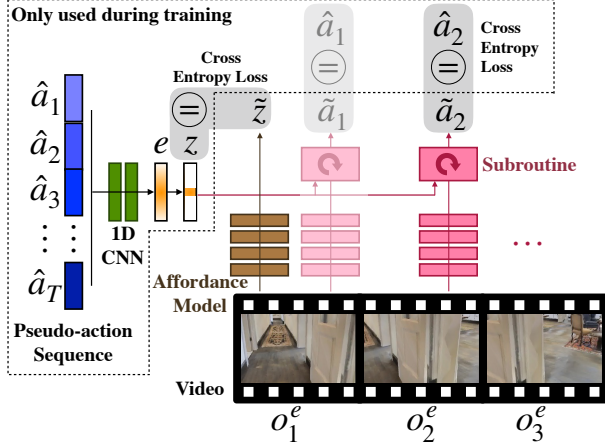


**Figure 3: Learning Visuomotor Routines and Affordances:** We want to mine visuomotor routines with the ability to explicitly invoke them. We implement this as recurrent network that takes in the current observation and a one-hot vector $z$ that specifies the subroutine to invoke. Since we don't have the labels for subroutine id $z$, we obtain it by jointly training two networks: one that looks at the entire future sequence of actions to predict the subroutine id, and other which takes the subroutine id $z$ and the image as input to predict the action to take. Both the networks are jointly trained to minimize cross-entropy loss. Finally, we also train an affordance model that predicts the inferred subroutine id $z$ from the first image.

# 5 Experiments (Learning Subroutines and Affordances)

## 5.1 Experimental Setup

Our experiments involve use of *environments* (where the *agent* can actively interact with the environment) $\mathcal{E}_{train}$ and $\mathcal{E}_{test}$, and a *dataset* of first-person videos $\mathcal{D}$. We describe choices for the environment, agent and this video dataset:

**Environments**: We model environments using a visually realistic simulator derived from scans of real world indoor environments from the Stanford Building Parser Dataset [37] (SBPD) and the Matterport 3D Dataset [38] (MP3D). These scans have been used to study navigation tasks in [13, 39, 40], and we adapt publicly available simulation code from [13]. We split these environments into four disjoint sets: $\mathcal{E}_{train}$, $\mathcal{E}_{video}$, $\mathcal{E}_{val}$ and $\mathcal{E}_{test}$. $\mathcal{E}_{train}$ is used to train the inverse model, $\mathcal{E}_{val}$ is used for development of policies for down-stream tasks, and $\mathcal{E}_{test}$ is used for evaluating performance of our policies on down-stream tasks. $\mathcal{E}_{video}$ is used to create a dataset of egocentric videos.

**Agent Model:** Our agent is modeled as a cylinder with 4 actions: a) stay in place, b,c) rotate left or right by $\theta$ ($= 30°$), and d) move forward by $x$ ($= 40$cm). The robot is equipped with a RGB camera mounted at a height $h$ ($= 120$cm) from the ground with an elevation $\phi$ ($= -5°$) from the horizontal.

**Dataset $\mathcal{D}$:** We create *MP3D Walks Dataset* of egocentric videos. *MP3D Walks Dataset* is auto-generated using the $\mathcal{E}_{video}$ environments, by rendering out images along the path taken by an expert navigator to navigate between given pairs of random points. We implement this expert as an analytical path planner which has access to the ground truth free space map. We additionally ensure that experts have a different action space than our agent (see supplementary for specifics). *MP3D Walks Dataset* consists of around $217K$ clips of $40$ steps each, without any action labels.

## 5.2 Training Details

**Inverse Model Training and Pseudo-labeling.** The agent starts at $1.5K$ different locations spread over 4 environments ($\mathcal{E}_{train}$) and executes random actions for 30 steps. The collected data ($45K$ interaction samples) is used to train the inverse model. See supplementary for ablations over the number of interaction samples and the generalization performance of the inverse model over different camera heights of the test images.

5

(a) Robustness and Diversity of Subroutines      (b) Consistency of Subroutines

**Figure 4: (a) Robustness and Diversity of Subroutines**: Subroutines SubR2 (top row) and SubR1 (bottom row) when deployed on a real robot demonstrate robustness and consistency over perturbations to starting location. **(b) Consistency of Subroutines**: Learned Subroutines when deployed on a real robot demonstrate consistent behavior over different starting locations, as demonstrated for SubR2.



**Figure 5: Affordance Model Visualization:** The images in the first 3 columns were assigned high probability for SubR1 by the affordance model (that goes rightwards), while images in the next three columns were assigned high probability for SubR2 (that goes leftward). These images, indeed afford the predicted subroutines.

This model is then used to pseudo-label videos in $\mathcal{D}$ to obtain dataset $\hat{\mathcal{D}}$ as described in Section 3.2. $\hat{\mathcal{D}}$ is used to learn subroutines $\pi(., z)$ and the affordance model. (Section 4.1).

**Subroutine Training**: We slice each of the $217K$ videos into clips of length 10 steps with a sliding window of 5. This gives us a total of $2.2M$ clips to train our subroutines. We experiment with using 4 subroutines (*i.e.* the $z$ vector is 4-dimensional. We show ablations over the number of subroutines as well as the length of each subroutine in supplementary.). This model is trained by minimizing the cross-entropy loss between the actions output by the policy ($\tilde{a}$) and the pseudo-labels ($\hat{a}$) obtained from the inverse model.

**Affordance Training**: We train the affordance model to predict the inferred subroutine id $z$ given the first image of the length 10 trajectory by minimizing cross-entropy loss over the inferred $z$ label.

## 5.3 Results

**Behavior of Subroutines**: We deployed our subroutines (learned in simulation) in the real world on a real robot (iCreate2 platform equipped with a RGB camera). Figure 4a shows the diversity between two of our learned subroutines SubR1 and SubR2. We observe that SubR1 prefers turning rightward into doors and corridors, and SubR2 prefers turning leftward. The subroutines show robustness to perturbations in the starting location and consistently enter the door. Figure 4b shows that SubR2 consistently turns left into doors and corridors across different starting locations. See supplementary for simulation results showing diversity and consistency of our learned subroutines.

**Affordance Prediction**: We show observations from $\mathcal{E}_{test}$ for which the affordance model prediction is high for SubR1 and SubR2 in Figure 5. Top row shows observations that cause a high prediction for SubR1, while bottom row shows images that excite SubR2.

# 6 Experiments (Using Affordances and Subroutines)

## 6.1 Exploration via VMSR

The exploration task requires the agent to explore a novel environment efficiently.

**Table 1: Exploration Metrics:** VMSR beats 3 hand-crafted baselines and two state-of-the-art learning based techniques [19, 18]. See text for details.

| Method | # Samples ↓ | ADT ↓ | Maximum Distance ↑ | Collision Rate (%) ↓ |
|---|---|---|---|---|
| Random | 0 | 18.09 | 7.5 | 65 |
| Forward Bias Policy | 0 | 15.25 | 13.11 | 82 |
| Always Forward, Rotate on Collision | 0 | 14.89 | 13.31 | 72 |
| Skills from Diversity [19] | $10M$ | 17.63 | 7.85 | 67 |
| Skills from Curiosity [18] | $10M$ | 17.68 | 7.87 | 64 |
| VMSR (Ours) | $45K$ | **7.73** | **27.78** | **12** |

**Task Setup**: We randomly initialize the agent at $100$ different locations in the novel test environment $\mathcal{E}_{test}$. For each location, we do $5$ random executions (each from a randomly chosen initial orientation) of length $408$ each.

**Metrics**: We measure different aspects of exploration via the following metrics: **# Samples** ↓: Number of environment interactions used for training. **Average Distance to Trajectory (ADT)** ↓: Given executed trajectories from a given starting location, we compute the mean geodesic distance of points in the environment to the closest point on the trajectory. If we wanted to visit a point in the environment, this metric measures how much we will need to go off the trajectory to get to this point, in expectation. We report the average over all starting locations. **Maximum Distance** ↑: Measures how far the executed trajectories convey the agent. For each trajectory, we measure the maximum geodesic distance from the starting location to all points on the trajectory. We report the average maximum geodesic distance over trials. **Collision Rate** ↓: Fraction of forward actions that result in collisions. We emphasize that VMSR is not trained to optimize for any of these metrics.

**Exploration via Subroutines**: Given the visual observation from the current location, we repeat the following two steps: a) we use the affordance model $\alpha$ to sample the subroutine $z$ to execute, b) we execute the sampled subroutine for 10 steps.

**Baselines.** We compare with three hand-crafted baselines: a) Random policy (randomly execute one of the 4 actions), b) Forward bias policy (biased to more frequently execute forward action), and c) Always forward but rotate on collision policy. We also compare to the state-of-the-art unsupervised RL-based skill learning methods d) DIAYN [19], and e) Curiosity [18]. These learning based techniques were trained with comparable networks (ResNet 18 models pre-trained on ImageNet) for over $10M$ samples. More details about these baselines are in supplementary.

**Results.** Table 1 shows that VMSR outperforms all the baselines on all three metrics. VMSR successfully learned to bias towards forward action in navigation and the notion of obstacle avoidance, leading to a high maximum distance, and a low collision rate. It outperforms hand-crafted baselines that were designed with these insights in mind. Furthermore, it outperforms state-of-the-art learning based techniques for learning skills [19, 18], that were trained on $200\times$ more interaction samples ($45K$ *vs.* 10M). These past works have only been shown to perform well for low-dimensional state-spaces and simple game environments, and in our experiments they either learn a trivial solution, or suffer with high-dimensional inputs such as real world images (further discussed in supplementary).

## 6.2 PointGoal and AreaGoal via VMSR Initialization in HRL

We next investigate how we can use VMSR to solve goal-driven tasks. We do this by setting up hierarchical RL policies based of our learned subroutines and affordance models.

**Task Setup:** We setup two goal driven navigation tasks, PointGoal and AreaGoal as defined in [41]. For PointGoal task, the agent is required to reach a given goal location (specified as a relative offset from robot's current location). For the AreaGoal task, the agent is required to go to the washroom. We study both tasks in sparse and dense reward settings. RL and HRL policies are developed on the validation environment $\mathcal{E}_{val}$, and finally trained on the test environment $\mathcal{E}_{test}$ with 3 random seeds to assess sample efficiency for learning.
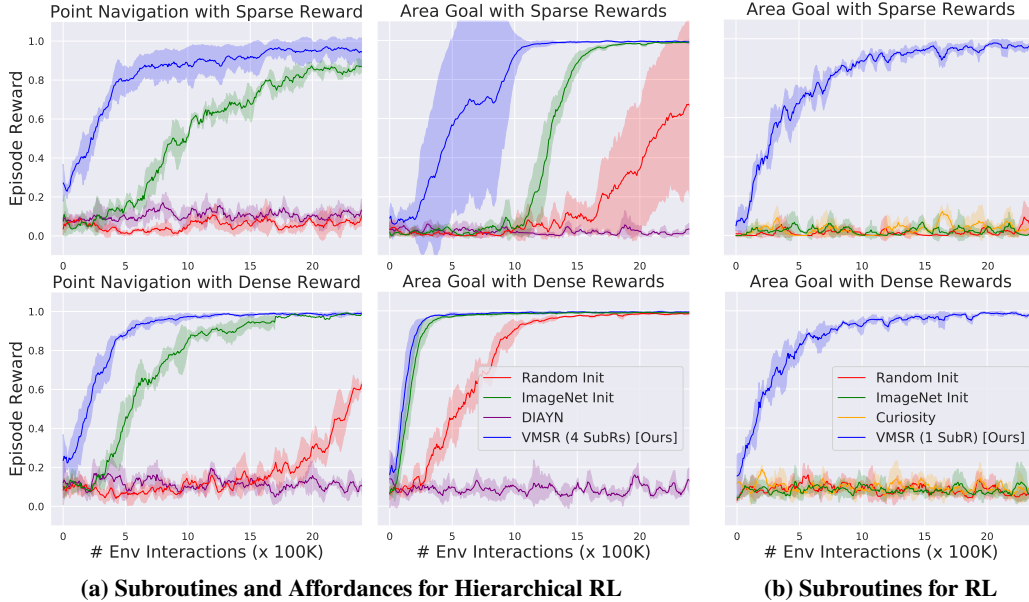
Figure 6: **(a) Subroutines and Affordances for Hierarchical RL**: Initializing from VMSR leads to upto 4x more sample efficient learning in downstream navigation tasks. First Column shows results for PointGoal (go to $(x, y)$ coordinate), second column shows results for AreaGoal (go to washroom). We see improvements across these tasks for both sparse and dense reward scenarios, with larger gains in the harder case of sparser rewards. **(b) Subroutines for RL**: Initializing a flat RL policy with VMSR (with only 1 SubR) leads to improved sample complexity for AreaGoal navigation (go to washroom), compared to alternate initializations.

**HRL via Subroutines and Affordance Model:** We follow the framework in [30] and initialize the sub policies with our subroutines and meta-controller with our affordance model. We then fine tune the sub policies and the meta-controller via reinforcement learning.

**Comparisons:** We compare with the following alternates for initializing the meta-controller and the sub-policies: a) *Random Initialization*, b) *ImageNet Initialization*, and c) *Initialization from skills via DIAYN [19] pre-training*. (c) doesn't provide an affordance model, so we initialize the meta-controller image CNN with the sub-policy CNN. We can also compare VMSR initialization to initialization obtained from Curiosity [18]. Pathak *et al.* [18] use a monolithic policy (*i.e.* without any handle to control what they do), and study an AreaGoal task. Thus, for a fair comparison, we limit the comparison to the AreaGoal task and use a monolithic RL policy instead of the hierarchical policy. We report three training curves: a) *Random Initialization*, b) *Initialization from Curiosity [18]*, and c) *VMSR* with 1 subroutine (to obtain a monolithic policy).

**Results:** Training rewards are plotted in Figure 6a and Figure 6b. Figure 6a shows the comparison among hierarchical policies. We observe upto $4\times$ faster training when initialized with VMSR and affordance models as compared to the next best baseline (which is ImageNet initialization). Improvements are generally larger for the harder case of sparser rewards. DIAYN [19] based initialization entirely fails, as it collapses to a trivial policy (more details in supplementary). Even among non hierarchical policies (Figure 6b), initializing with VMSR (VMSR (1 SubR)) performs the best in AreaGoal Tasks, outperforming random initialization and initialization from curiosity policy [18], which also learns a trivial policy (see supplementary).

## 7 Discussion

In this paper, we developed a technique that combined learning from interaction with learning from videos, to extract meaningful and useful subroutines from egocentric videos of experts performing different tasks. We showed how these extracted subroutines can be used as is for exploration, or can be specialized using hierarchical RL for solving other downstream navigation tasks. We believe advances made in this paper will enable scaling up of policy learning in robotics.

# References

[1] R. E. Fikes and N. J. Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial intelligence*, 1971.

[2] R. S. Sutton, D. Precup, and S. Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 1999.

[3] H. Xu, Y. Gao, F. Yu, and T. Darrell. End-to-end learning of driving models from large-scale video datasets. In *CVPR*, pages 2174–2182, 2017.

[4] X. B. Peng, A. Kanazawa, J. Malik, P. Abbeel, and S. Levine. Sfv: Reinforcement learning of physical skills from videos. In *SIGGRAPH Asia 2018*, page 178. ACM, 2018.

[5] X. B. Peng, P. Abbeel, S. Levine, and M. van de Panne. Deepmimic: Example-guided deep reinforcement learning of physics-based character skills. *ACM Transactions on Graphics (TOG)*, 37(4):143, 2018.

[6] S. M. LaValle. *Planning Algorithms*. Cambridge University Press, Cambridge, U.K., 2006. Available at http://planning.cs.uiuc.edu/.

[7] S. Thrun, W. Burgard, and D. Fox. *Probabilistic robotics*. MIT press, 2005.

[8] K. Hauser, T. Bretl, K. Harada, and J.-C. Latombe. Using motion primitives in probabilistic sample-based planning for humanoid robots. In *Algorithmic Foundation of Robotics*. 2008.

[9] S. Schaal. Dynamic movement primitives-a framework for motor control in humans and humanoid robotics. In *Adaptive motion of animals and machines*, pages 261–280. Springer, 2006.

[10] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal. Dynamical movement primitives: learning attractor models for motor behaviors. *Neural computation*, 25(2):328–373, 2013.

[11] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi. Target-driven visual navigation in indoor scenes using deep reinforcement learning. In *ICRA*, 2017.

[12] P. Mirowski, R. Pascanu, F. Viola, H. Soyer, A. Ballard, A. Banino, M. Denil, R. Goroshin, L. Sifre, K. Kavukcuoglu, et al. Learning to navigate in complex environments. In *ICLR*, 2017.

[13] S. Gupta, J. Davidson, S. Levine, R. Sukthankar, and J. Malik. Cognitive mapping and planning for visual navigation. In *CVPR*, 2017.

[14] F. Sadeghi. Divis: Domain invariant visual servoing for collision-free goal reaching. *RSS*, 2019.

[15] F. Sadeghi and S. Levine. Cad2rl: Real single-image flight without a single real image. *arXiv preprint arXiv:1611.04201*, 2016.

[16] S. Levine, C. Finn, T. Darrell, and P. Abbeel. End-to-end training of deep visuomotor policies. *JMLR*, 2016.

[17] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The International Journal of Robotics Research*, 37(4-5):421–436, 2018.

[18] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell. Curiosity-driven exploration by self-supervised prediction. In *ICML*, 2017.

[19] B. Eysenbach, A. Gupta, J. Ibarz, and S. Levine. Diversity is all you need: Learning skills without a reward function. In *ICLR*, 2019.

[20] B. D. Argall, S. Chernova, M. Veloso, and B. Browning. A survey of robot learning from demonstration. *Robotics and Autonomous systems*, 2009.

[21] A. Billard, S. Calinon, R. Dillmann, and S. Schaal. Robot programming by demonstration. In *Springer Handbook of robotics*. 2008.

[22] K. Hausman, Y. Chebotar, S. Schaal, G. Sukhatme, and J. J. Lim. Multi-modal imitation learning from unstructured demonstrations using generative adversarial nets. In *NIPS*, 2017.

[23] Y. Aytar, T. Pfaff, D. Budden, T. L. Paine, Z. Wang, and N. de Freitas. Playing hard exploration games by watching youtube. *arXiv preprint arXiv:1805.11592*, 2018.

[24] F. Torabi, G. Warnell, and P. Stone. Behavioral cloning from observation. In *IJCAI*, 2018.

[25] A. D. Edwards, H. Sahni, Y. Schroeker, and C. L. Isbell. Imitating latent policies from observation. *arXiv preprint arXiv:1805.07914*, 2018.

[26] D. Pathak, P. Mahmoudieh, G. Luo, P. Agrawal, D. Chen, Y. Shentu, E. Shelhamer, J. Malik, A. A. Efros, and T. Darrell. Zero-shot visual imitation. In *ICLR*, 2018.

[27] C. Finn, T. Yu, T. Zhang, P. Abbeel, and S. Levine. One-shot visual imitation learning via meta-learning. *arXiv preprint arXiv:1709.04905*, 2017.

[28] T. Yu, C. Finn, A. Xie, S. Dasari, T. Zhang, P. Abbeel, and S. Levine. One-shot imitation from observing humans via domain-adaptive meta-learning. *arXiv preprint arXiv:1802.01557*, 2018.

[29] P. Dayan and G. E. Hinton. Feudal reinforcement learning. In *NIPS*, 1993.

[30] A. G. Barto and S. Mahadevan. Recent advances in hierarchical reinforcement learning. *Discrete event dynamic systems*, 2003.

[31] A. S. Vezhnevets, S. Osindero, T. Schaul, N. Heess, M. Jaderberg, D. Silver, and K. Kavukcuoglu. Feudal networks for hierarchical reinforcement learning. *arXiv preprint arXiv:1703.01161*, 2017.

[32] A. Levy, R. Platt, and K. Saenko. Hierarchical actor-critic. *arXiv preprint arXiv:1712.00948*, 2017.

[33] D. F. Fouhey, V. Delaitre, A. Gupta, A. A. Efros, I. Laptev, and J. Sivic. People watching: Human actions as a cue for single view geometry. *IJCV*, 2014.

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

[35] P. Agrawal, A. V. Nair, P. Abbeel, J. Malik, and S. Levine. Learning to poke by poking: Experiential learning of intuitive physics. In *NIPS*, pages 5074–5082, 2016.

[36] E. Jang, S. Gu, and B. Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

[37] I. Armeni, O. Sener, A. R. Zamir, H. Jiang, I. Brilakis, M. Fischer, and S. Savarese. 3D semantic parsing of large-scale indoor spaces. In *CVPR*, 2016.

[38] A. Chang, A. Dai, T. Funkhouser, M. Halber, M. Niessner, M. Savva, S. Song, A. Zeng, and Y. Zhang. Matterport3D: Learning from RGB-D data in indoor environments. In *3DV*, 2017.

[39] A. Kumar*, S. Gupta*, D. Fouhey, S. Levine, and J. Malik. Visual memory for robust path following. In *Advances in Neural Information Processing Systems*, 2018.

[40] T. Swedish and R. Raskar. Deep visual teach and repeat on path networks. In *CVPRW*, 2018.

[41] P. Anderson, A. Chang, D. S. Chaplot, A. Dosovitskiy, S. Gupta, V. Koltun, J. Kosecka, J. Malik, R. Mottaghi, M. Savva, and A. Zamir. On evaluation of embodied navigation agents. *arXiv preprint arXiv:1807.06757*, 2018.

[42] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[43] A. Murali, T. Chen, K. V. Alwala, D. Gandhi, L. Pinto, S. Gupta, and A. Gupta. Pyrobot: An open-source robotics framework for research and benchmarking. *arXiv preprint arXiv:1906.08236*, 2019.