Learning Navigation Subroutines from Egocentric Videos

Supplementary Material

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

A1 Subroutines and Affordances Full 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. We use cross-entropy loss between the actual action and the predicted action. We use Adam [1] with 64 batch size and 0.001 learning rate. Ablations over number of interaction samples by varying number of starting locations and number of steps per starting location is shown in Figure A4.

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

Subroutine Training: We slice each of the $217K$ videos into clips of length 10 steps with a sliding window of 5 (ablations over the length of subroutines is shown in Figure A4). 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) and show ablations over this hyper-parameter Figure A4. 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 in length 10 trajectory by minimizing cross-entropy loss over the inferred z label.

A2 Consistency and Diversity Visualizations

We unroll different subroutines from different locations in the test environment $\mathcal{E}_{test}$, and visualize the trajectories followed by each of them in the top view in Figure A2. We show multiple rollouts of each subroutine from each of the starting locations. Randomness in behavior comes from the sampling of the actions from the network output. The three top view figures in each column of Figure A2 correspond to one subroutine at three different starting locations and we rollout 8 trajectories from each starting location. Thus, each column demonstrates that a specific subroutine does similar things when initialized at different locations, showing the consistency of our learned subroutines. For example, SubR1 always turns right, SubR2 always turns left. Rollouts shown in different rows of Figure A2 show that different subroutines show diverse behaviors when started from the same location. This shows the diversity of across our learned subroutines.

We also quantitatively compute disentanglement: we unroll the different subroutines from the same starting location and compute the intersection over union between trajectories from $\texttt{SubR}_i$ and $\texttt{SubR}_j$ (for example, IoU between the green region and blue region in plots in the top row of Figure A2). A higher IoU implies similar areas are traversed by two sets of sub-policies, lower IoU implies the two sub-policies are distinct. Thus, we should expect a higher IoU between the trajectories from the same subroutine and a lower IoU between different subroutines. The average IoU between different subroutines is 0.42, and the average IoU between trajectories from the same subroutines is 0.58. Thus, indeed different subroutines are disentangled.

3rd Conference on Robot Learning (CoRL 2019), Osaka, Japan.

A3 Affordance Model Entropy Visualization

We also look at the entropy of the output of affordance model in Figure A1 in top view. The arrow shows the direction in which the agent is facing and we plot the entropy of the prediction of the affordance model when the first-person (egocentric) observation is given as input. A higher entropy implies that more subroutines apply in the given scenario. The observed entropy is consistent with our expectations, as explained in the figure caption.

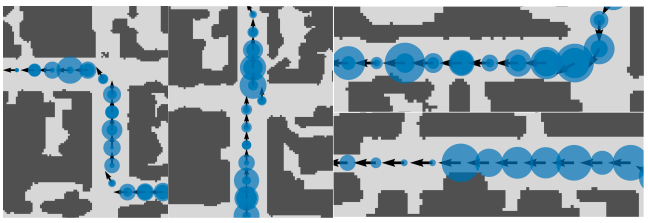

Figure A1: Multi-modality in Affordance Predictions: We visualize the entropy of the distribution output by the affordance model in the *test* environment. A larger circle denotes a higher entropy meaning more subroutines can be invoked at that location. We observe that the affordance model has a higher entropy as the agent approaches hallway intersections, or room entrances. This multi-modality collapses as the agent crosses the decision junctions.

A4 Baselines for Exploration

1. *Random Policy*: We randomly sample an action from the four possible actions (stay, left, right, forward) at every step.
2. *Forward Bias Policy*: Since motion is typically dominated by forward motion, we compare to another policy that samples the forward action more preferably. We use the distribution of actions in the MP3D Walks Dataset, probabilities for `stop, turn left, turn right and forward` were $[0.0, 0.17, 0.17, 0.66]$ respectively.
3. *Always Forward, Rotate on Collision*: This baseline repeats the following procedure: rotate by a random angle sampled from $(-\pi, \pi]$, move straight till collision.
4. *Diversity Policy (DIAYN) [2]*: We use the state-of-the-art RL-based unsupervised skill learning algorithm from Eysenbach *et al.* [2] to learn 4 diverse skills on $\mathcal{E}_{train}$ environments. We test the learned skills for exploration by randomly sampling a skill, and then executing it for 10 steps, where we sample actions from the probabilities output by the selected skill. Policy architecture is same as those for our subroutines, discriminator is based of a ResNet 18 model. Both models are initialized from ImageNet. Policy is trained for over 10 million interaction, best performance occurs at around 1M interaction samples.
5. *Curiosity Policy [3]*: We train a curiosity-based agent that seeks regions of space where its forward model has high prediction error [3]. Policy architecture is same as that for our subroutines (except that it does not take in the latent vector z), and initialized from ImageNet. Forward model is learned in the $conv5$ average pooled feature space of a fixed Resnet 18 model pre-trained on ImageNet. Trajectories are executed by sampling from the action probabilities output by the policy. Once again, policy is trained for over 10 million interaction, best performance occurs at around 1M interaction samples.

Curiosity Model: Pathak *et al.* [3] proposed use of prediction error of a forward model as an intrinsic reward for learning skills using RL. We were surprised at the rather poor performance for the curiosity model. We found that the model converges to the policy of simply rotating in-place. Such a degenerate solution makes sense as rotating in-place has higher prediction error than staying in-place and moving forward. In-place rotations cause new parts of the environment to become visible which makes for a harder prediction task. Staying-in-place and moving forward cause only minor changes to the image or no changes at all. Thus, the curiosity model rightly learns to simply rotate in-place. We saw this same behavior across different runs with different hyper-parameters and different architectures: policies will collapse to outputting just the rotation actions. Entropy based regularization is used to prevent such a collapse. We used such regularization and cross-validated various choices for the trade-offs in loss between entropy regularization and policy gradient loss, but didn't find it to alleviate this issue. We selected the best model for the task of exploration across different runs and different number of training iterations. This selected model ended up being a heavily regularized model that would pick actions almost uniformly at random, as that would get higher performance than simply rotating in-place. As both extremes (taking actions randomly, or

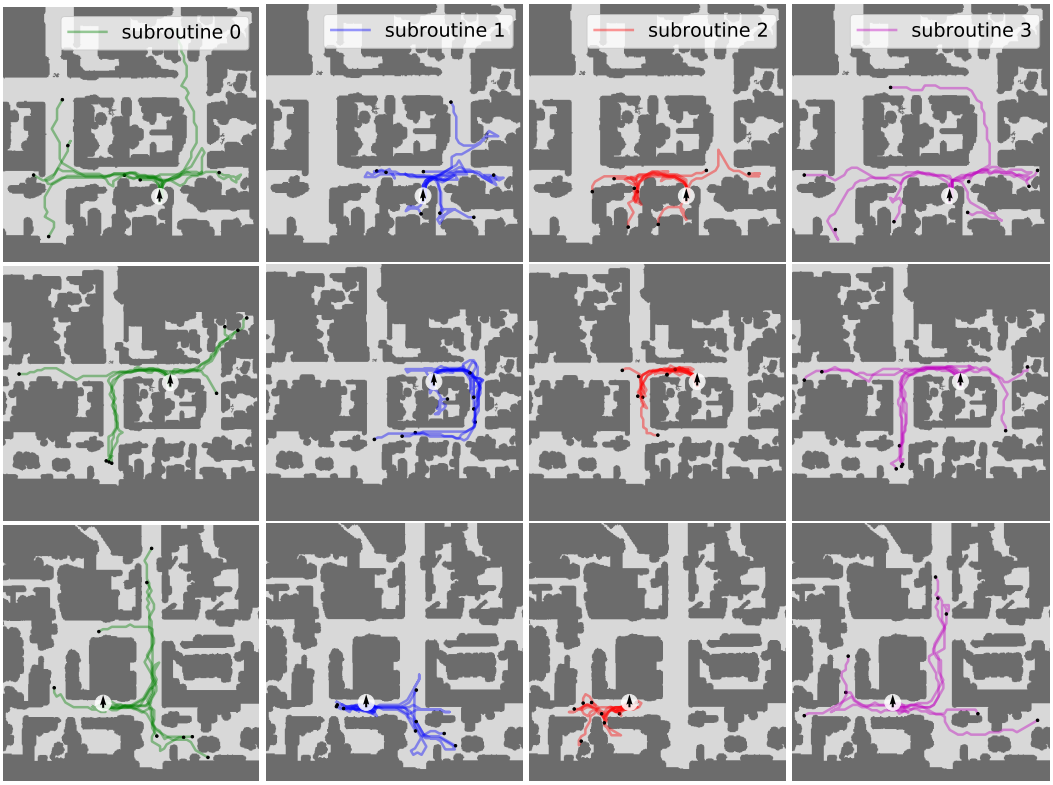

Figure A2: Subroutine Consistency and Diversity: Each top-view figure shows multiple roll-outs of a subroutine from a given location. The black arrow in white circle shows starting position and the black dots shows the ending location of the rollouts. Columns show the same subroutine over different starting locations, illustrating the consistency of our subroutines while rows show different subroutines unrolled from the same location illustrating their diversity. It appears that `SubR1` prefers turning right and, `SubR2` prefers turning left. Note that policies only use first person views.

picking only the rotate in-place action) are trivial solutions, the curiosity model starts to ignore the image and consequently performs on-par with uninitialized models for reinforcement learning tasks.

Diversity Model: The diversity model from Eysenbach *et al.* [2] seeks to classify states with the skill id that was used to get to it (see Algorithm 1 in [2]). While this works well for the environments studied in [2], it breaks down for visual navigation. This is because, the same state can be reached via different skills depending on the starting state. This causes the skill classifiers q to only perform at chance. Consequently, the reward for the skill policies is uniform, causing the policies to collapse (all actions produce the same reward, and hence no learning happens). We observed this empirically in our experiments as well: accuracy for state classification was at chance (25% for four skills), and the reward stayed constant. Best performing policy (based on validation for exploration metrics) always predicted the following probabilities for different actions for different skills: $[0.246, 0.232, 0.237, 0.285]$ (for `stop, left, right, forward` respectively). As this can be done without looking at the image, the policy learns to ignores the image. Thus, the model perform on-par with uninitialized models for hierarchical reinforcement learning experiments.

A5 Exploration Visualization

We show the coverage for each method in Figure A3. Figure 10 overlay trajectories executed by different policies onto the map (only used for visualization). We see a wider coverage for VMSR over other methods and also observe that the trajectories avoid the walls of the hallways when going down them.

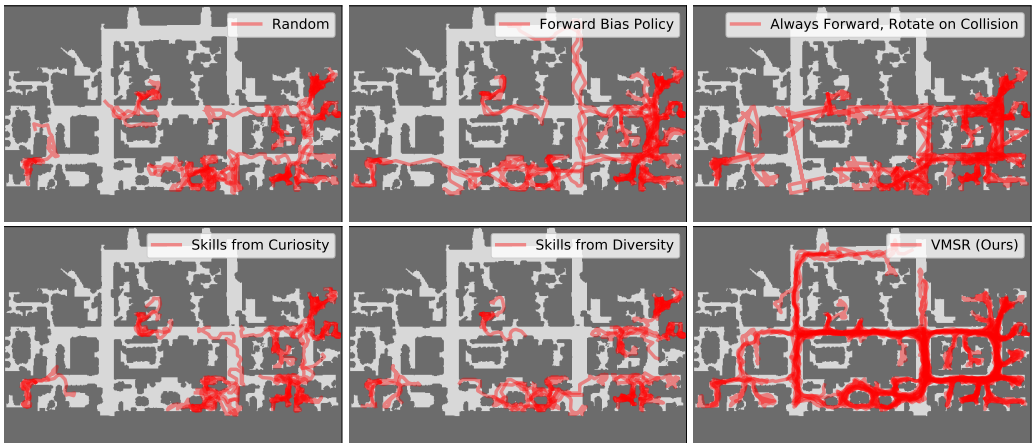

Figure A3: Coverage Visualization: We show coverage of the overall space after sampling 20 roll-outs from 11 different locations in the the test environment $\mathcal{E}_{test}$. Note that VMSR covers more of the environment. It is able to come out of rooms and different roll-outs go towards different areas. Curiosity, diversity and Random policies spend most of their time inside rooms. Policies that are biased to move forward do come out, but do not show diverse behavior. Visualizations show top view, however policies only use first person views.

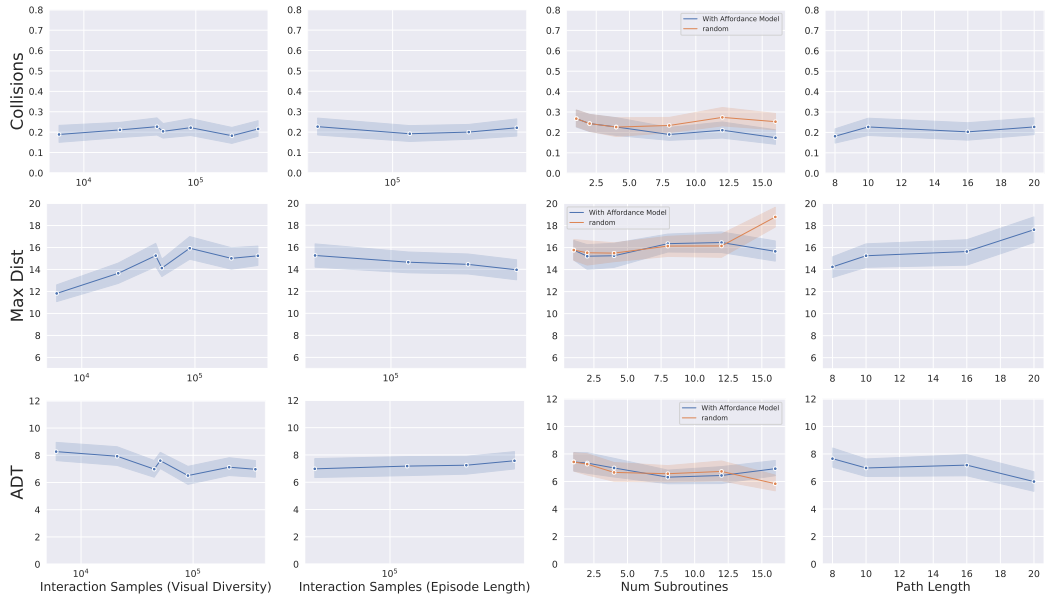

Figure A4: Dependence on active environment interaction samples, length of reference videos and number of subroutines specified: **Column 1 and 2**: We plot the exploration metrics against the number of self supervision interaction samples. There are two orthogonal ways of achieving this – increasing the number of restarts while keeping each episode length fixed (Col 1) and increasing the length of each self supervision episode while keeping the number of restarts fixed (Col 2). We see that visual diversity improves performance on Max Dist metric, but saturates at 45K interaction samples (1500 restarts with 30 steps each). Performance roughly remains the same as we increase the episode length. **Column 3**: We change the number of subroutines learned on the x-axis and compare the use of affordance model for sampling subroutines to randomly sampling subroutines. Affordance model shows improvement in collision rate over random sampling, indicating that the affordance model better respects the constraints of the physical space. We don't see an improvement in the exploration metric or max distance metric. **Column 4**: We observe improvements as we increase the path length of the reference trajectories. Longer trajectories presumably allow VMSR to learn more complex subroutines.

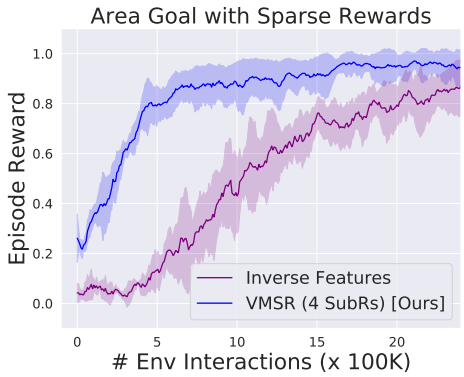

Figure A5: We compare VMSR initialization to initializing the image features of the sub-policies with the features from the inverse model for the downstream HRL task of PointGoal with sparse rewards. We see that VMSR is 3x more sample efficient compared to this baseline.

A6 Ablations

We show ablations over 4 hyper-parameters in Figure A4 (see caption for more details). We compare VMSR initialization to inverse features initialization for downstream HRL task in Figure A5. We show the ability of the trained model to generalize across various camera heights of the reference images in Figure A6.

A7 RL Experimental Setup

We use $\mathcal{E}_{test}$ for RL experiments. We use A2C to train all our algorithms on Point Goal task and Area Goal task.

- **Area Goal:** The task is to find the nearest washroom. $\mathcal{E}_{test}$ contains 2 washroom, and we start the agent 10-23 steps away from the nearest washroom. We randomly start the agent at a different location for every episode.
- **Point Goal:** We specify the goal coordinates relative to the start position, and randomly sample the start and the goal locations every episode. The goal is 10-17 steps away from the start location.

A8 Video Results

The enclosed video `vmsr.mp4` contains video results of real robot deployment followed by an explanation of our method. We use [4] for real robot deployment. Note that along with the 4 primitive actions (rotate left, rotate right, move forward and stay in place), the robot also moves slightly backward incase of a collision.

A9 Area Splits and Agent Settings

We give details of area splits and the action space of the experts which generate reference videos in Table A1. In the table, step size (x) refers to the length of a single forward step, θ refers to the rotation angle for left/right turn, ϕ refers to the elevation angle of the onboard RGB camera from the horizontal and h refers to the height of the robot from the ground.

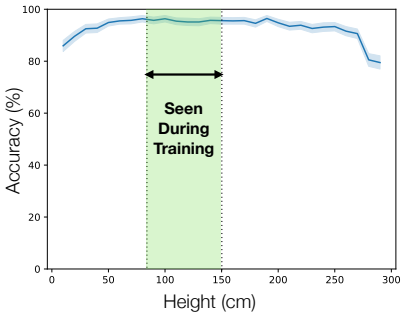

Figure A6: We test the generalization of the learned inverse model on images from $\mathcal{E}_{val}$, which is unseen during training. We plot the prediction accuracy (y axis) as we increase the camera height from the ground (x axis). The agent is trained on heights from 90cm to 150 cm during training in $\mathcal{E}_{train}$ and evaluated on heights from 10cm to 300cm. We observe a very consistent performance even in the range not seen during training. Note that the agent starts touching the ceiling of the room in some places at 300cm.

Table A1: Split of environments between different sets used in the paper. These environments are from Stanford Building Parser Dataset (SBPD) [5] and Matterport 3D Dataset (MP3D) [6]. We fix a step size (x) and rotation angle (θ) for each area by randomly sampling from the list. For elevation angle and height of the robot, we resample a value from the given ranges for *every video*.

Split	Environments	Agent Settings			
		Step Sizes (x in cm)	Rotation Angles (θ)	Elevations (ϕ)	Height (h in cm)
$\mathcal{E}_{train}$	`area1, area6, B6ByNegPMKs, Vvot9Ly1tCj`	20, 50, 80	36°, 24°, 18°	[-25°, 5°]	[90, 150]
$\mathcal{E}_{video}$	`area5a, area5b, p5wJjkQkbXX, VFuaQ6m2Qom, 2n8kARJN3HM, SN83YJsR3w2`	30, 60, 90	40°, 30°, 24°, 20°	[-35°, -5°]	[80, 160]
$\mathcal{E}_{val}$	`area3`	40	30°	-15°	120
$\mathcal{E}_{test}$	`area4`	40	30°	-15°	120

References

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

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

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

[4] 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.

[5] 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.

[6] 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.