Supplementary

Anonymous Author(s)
Affiliation
Address
email

1 Proof for Synthesizing Navigation Skills by Exploiting Spatial Independence

We now formally describe under what conditions we will resolve a manipulation option o for some set of starting states Z. Consider an option o that has an associated navigation symbol σ^o to characterize part of its initiation set I_o: $I_o = \text{Proj}(Z, S_b) \cap \sigma^o$. Then this implies that if the agent is in a state that is an element of Z, and only changes the robot's mobile base pose to be an element of the navigation symbol without changing anything else, then the resulting state would be an element of the initiation set of the option. We prove that if our assumptions regarding the initiation set of a manipulation option are satisfied, then we can synthesize a locomotive behavior from our navigation stack using our learned navigation symbol, which means we can generate the navigation stack to support a specific option. Since the state is Markovian, proving for the more general case where we aim to generate a navigation stack to support a manipulation plan follows from repeated applications of Theorem 1, and so we omit it.

If a manipulation option's o_i initiation set can be written using the definition of spatial independence (Equation 1) from the current set of states Z, then sampling a location l from σ^{o_i} and synthesizing and executing a path plan from the navigation stack to l from a start state in Z is sufficient for enabling the robot to execute the manipulation option o_i.

Theorem 1. *If, for a starting set of states Z, the initiation set I_{o_i} for a manipulation option $o_i \in O$ can be characterized as in Equation 1, then a location l sampled from the associated navigation symbol $l \in \sigma^{o_i}$ can be used in conjunction with a path planner to locomote the robot to a state s that is within I_{o_i} the initiation set of o_i as long as there is a collision-free path.*

Proof. By our assumptions, we know that the initiation set for the manipulation option can be decomposed into $I_o = \text{Proj}(Z, S_b) \cap \sigma^o_s$. We also assume that the agent starts in a state z element of Z ($z \in Z$). We can then use the pose l that is sampled from the navigation symbol σ^{o_i} to synthesize a navigation action n_i that starts from z and ends at location l, $n_i \in N(z, l)$ as long as there is a collision free path through the environment. The effect of executing n_i from z by definition only affects spatial state variables S_b, and so the resulting state is an element of $Proj(Z, S_b)$ and also an element of navigation symbol σ^o_s. Therefore it the resulting state is an element of the intersection of $Proj(Z, S_b)$ and σ^o_s, which is by definition the initiation set of o_i based on the Equation 1.

□

2 Simulation Experiment: Spatial Independence for Learning Symbols

Part of the model learning process requires identifying which factors are independence since there is no a priori assumption about the structure of the initiation and effect sets of the skills. Partitioning is via DBSCAN clustering [1], and the precondition classifiers are learned using a SVM [2] with an RBF kernel (hyperparameters are optimized using grid search. The effect density estimation is performed with a kernel density estimators [3, 4] with a Gaussian kernel, with a grid search over the bandwidth.

Submitted to the 7th Conference on Robot Learning (CoRL 2023). Do not distribute.

To incorporate the spatial independence assumption, we use a simple augmentation to the baseline: data collection occurs as before, but the spatial state variables are separated from all the other non-spatial state variables before learning. We then perform model learning exactly as [5], except with the separated spatial state variables already identified as an independent factor. This difference represents how the spatial independence assumption can be incorporated into an existing symbol learning pipeline.

3 Experiment: Transfer of Learned Abstractions

In the second set of experiments, our goal is to evaluate how AOSMs help transfer learned abstractions to novel environments. For these experiments, we provided a manipulation-only plan p $=\{$**PickUp(Mug)**, **ToggleOn(CoffeeMachine)**, **ToggleOff(CoffeeMachine)**, **PutIn(Mug,CoffeeMachine)**, **MakeCoffee(Mug,CoffeeMachine)**$\}$. The robot must construct the navigation symbols that enable it to generate navigation behaviors that enable those actions to be executed.

3.0.1 Approaches

For these set of experiments, all of the approaches perform a similar procedure. For a given scene and current step of the plan o, the robot 1) uses rejection sampling to sample a pose l from the associated navigation symbol σ^o 2) uses the path planner to move to location l, and 3) attempts to run the manipulation option o. If the agent fails to successfully execute the manipulation option, the location l is added as a negative sample to the dataset used to train σ^o; the robot repeats these steps until successful execution. When the robot is successful in executing the manipulation option, location l is added as a positive sample to the dataset used to train σ^o, and the robot proceeds to the next plan step. These navigation symbols are trained using Gaussian Process classifiers [6] with an RBF kernel.

There are two important design choices when learning navigation symbols that can be chosen inde-pedently of each other: 1) which spatial frame are the navigation symbols learned in, and 2) what proposal distribution is used for rejection sampling. In [7], the global map frame is used as the spatial frame and a random distribution for sampling, and we call this baseline **random global**. Learning symbols in the map frame enables the robot to leverage a path planner to generate navigation behaviors, but it means that the robot must relearn the symbols when the scene changes. To exploit the structure of object-centric skills, an object-centric spatial frame can be used to learn the symbols, which the agent can transform into a map frame given a semantic map that includes object pose. This enables the agent to effectively transfer learned information from one map to another. Using an object-centric frame with a random sampling distribution is akin to the approach in James et al. [8], which we term **random object**. However, using a uniform distribution as the proposal distribution is extremely inefficient since the robot will try manipulating objects from locations extremely far from the object. Kaelbling and Lozano-Pérez [9] proposed exploiting the nature of space using a geometric heuristic that samples poses near the object, and so we call the baseline that uses the geometric heuristic for sampling poses and learning in a global map frame **heuristic global**. The final approach learns in an object-centric spatial frame and uses the geometric heuristic to sample poses, which to our knowledge has not been used in conjunction to learn symbols. We call this baseline **heuristic object**, and it corresponds to our assumption. To give an upper-bound on performance, we also evaluate an oracle, which always samples feasible manipulation locations.

To determine how effectively each approach enables learned abstractions to be transferred to different environments, we use investigate two experimental settings: when the agent successfully finishes executing the plan, 1) the scene is reset to the initial configuration and the agent retries executing the plan (the single-scene setting), and 2) a new scene is chosen and the agent retries executing the plan (the multi-scene setting). In the single-scene setting there is no need for transfer and the choice of spatial frame does not matter. This lets us evaluate how important the chosen proposal distribution is for learning navigation symbols. In the multi-scene setting, the agent must also transfer the learned symbols to different scenes, which lets us evaluate how useful the choice of frame is for transfer.

3.0.2 Metrics

For each task execution in a scene, we report the cumulative total number of manipulation skills the robot executed, until the plan succeeded.

3.0.3 Results

Results for the single-scene and multi-scene setting for all four approaches are in Figure **??**. The approaches differ substantially between the single-scene and multi-scene setting. In the single-scene setting, the heuristic sampler quickly guides the agent towards locations that afford useful manipulations, when compared to sampling random locations. However, after around 15 episodes, all of the approaches learn to plan in the single scene, and all approach the oracle's performance (which is just the length of the plan). However, in the multi-scene setting, although the heuristic sampler with global frame starts off better than the random sampler with an object-centric frame, after about 3 episodes, the object-centric frame with random sampling starts to outperform it. This is because the global frame approach cannot port across different scenes, whereas object-centric frames can. Therefore, we see that navigation symbols in an AOSM should be a) learned in an object-centric frame to support portability to new domains, and b) learned using a sampling process with geometric information included, rather than sampling at random.

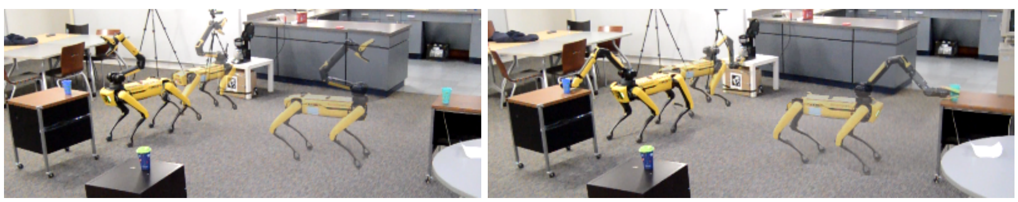

Figure 1: An example demonstration of the Spot building an AOSM and using it to prepare coffee. (Left): Spot navigates around the space, identifies objects, and constructs an AOSM. (Right): With the AOSM and a manipulation-only plan, the Spot can synthesize the navigation abstractions to locomote around the environment to successfully execute the manipulation skills.

4 Robot Hardware Demonstration

Our demonstration of using an AOSM on a real robot can be seen in full detail in Figure 1. We first manually drive the robot around and use an off-the-shelf SLAM implementation to generate a 3D geometric map of the environment which the robot can use to navigate to 3D poses. The robot then constructs a semantic map that captures the spatial pose and semantic attributes of each of the relevant objects in the scene. Once the robot is equipped with a set of manipulation skills, it generates an AOSM of the scene using hand-crafted navigation symbols, which enables it to sample navigation poses that support successfully executing each of its manipulation skills. The robot then uses a hand-specified PDDL of the coffee preparation task to generate the manipulation-only plan using Fast Downward, which results in: **PickUp(WaterCup)**,**Pour(WaterCup)**, **Place(WaterCup)**,**PickUp(CoffeeGrinds)**,**Pour(CoffeeGrinds)**, **Place(CoffeeGrinds)**, and then **CloseLid(CoffeeMachine)**, **PushButton(CoffeeMachine)**. With the AOSM, the robot can synthesize a navigation stack to support plan execution (Figure 1).

We time how long it takes the robot to construct an AOSM in 2 different environments. Navigating the environment to observe the objects and then constructing the AOSM takes an average of 82.5 seconds. Executing the plan for the coffee preparation task takes on average 140 seconds. These timings demonstrate the efficacy of AOSMs to enable a robot to rapidly generate the navigation abstractions for supporting task execution.

References

[1] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd*, volume 96, pages 226–231, 1996.

[2] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[3] M. Rosenblatt. Remarks on some nonparametric estimates of a density function. *The Annals of Mathematical Statistics*, 27(3):832, 1956.

[4] E. Parzen. On estimation of a probability density function and mode. *The Annals of Mathematical Statistics*, 33(3):1065, 1962.

[5] S. James, B. Rosman, and G. Konidaris. Autonomous learning of object-centric abstractions for high-level planning. In *International Conference on Learning Representations*, 2022.

[6] H. Nickisch and C. Rasmussen. Approximations for binary gaussian process classification. *Journal of Machine Learning Research*, 9(Oct):2035–2078, 2008.

[7] G. Konidaris, L. P. Kaelbling, and T. Lozano-Perez. From skills to symbols: Learning symbolic representations for abstract high-level planning. *Journal of Artificial Intelligence Research*, 61: 215–289, 2018.

[8] S. James, B. Rosman, and G. Konidaris. Learning portable representations for high-level planning. In *International Conference on Machine Learning*, pages 4682–4691. PMLR, 2020.

[9] L. P. Kaelbling and T. Lozano-Pérez. Hierarchical task and motion planning in the now. In *2011 IEEE International Conference on Robotics and Automation*, pages 1470–1477. IEEE, 2011.