Learning Contextual Actions for Heuristic Search-Based Motion Planning

Dhruv Mauria Saxena[1] and Maxim Likhachev[1]

***Abstract*— Heuristic search-based motion planning can be computationally costly in large state and action spaces. In this work we explore the use of generative models to learn contextual actions for successor generation in heuristic search. We focus on cases where the robot operates in similar environments, i.e. environments drawn from some underlying distribution. Intuitively, in such cases the robot is bound to observe similar looking local regions of the environment over the course of its operation. We evaluate the use of a *conditional variational autoencoder* (CVAE) to learn a distribution over contextual actions given this local map and a goal location. These contextual actions are used to help the search make faster progress towards the goal, and avoid or get out of local minima along the way. We show simulation results for kinematic planning problems in a variety of 2D environments for motion planning for a point-robot and a planar arm with up to 5 degrees-of-freedom. Our approach outperforms traditional search-based planning algorithms in terms of computational cost (number of expansions) while maintaining bounds on suboptimality.**

I. Introduction

There are many tasks for robots that require solving motion planning problems in similar environments[1]. Robots are routinely and repeatedly used on the floor in warehouses [1], for pick-and-place tasks in factories [2], [3], for navigation in agricultural fields [4], and for navigation among dense crowds in the case of service robots [5]. In such environments, we can reduce the computational cost of future planning queries by leveraging prior experience. This work presents a simple and straightforward initial approach to incorporate the use of generative models within a heuristic search framework. Specifically, we focus on reducing planning time as measured by the number of state expansions in the context of heuristic search, while maintaining guarantees on bounded solution suboptimality.

In this work, we explore the use of *contextual actions* for this purpose. We define these as actions $\vec{a}$ that are conditioned on local information for a given robot state. Figure 1 contains an illustrative example that shows the utility of such actions. We consider the problem of learning the distribution $p(\vec{a}|\phi(x), \vec{x}_{\text{goal}})$ using a Conditional Variational Autoencoder (CVAE) [6], where $\vec{a}$ is the contextual action, x is the current state, $\phi(x)$ is a feature encoding of the current state which includes information about the local environment, and $\vec{x}_{\text{goal}}$ is a vector towards the desired goal state from the current state. Since we consider similar environments in which we repeatedly solve planning problems between different start and goal states, we know the robot will visit parts of the state-space with local environments similar to those encountered previously. This insight helps us model the distribution $p(\vec{a}|\phi(x), \vec{x}_{\text{goal}})$, given prior experience data.

[1] Carnegie Mellon University, USA. {dsaxena, mlikhach}@andrew.cmu.edu

[1]We say environments are *similar* if they are drawn from some underlying distribution e.g. 2D environments with convex obstacles, or pick-and-place environments with different cubby-hole configurations.

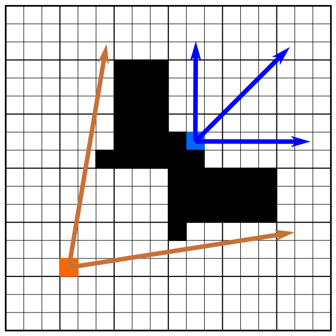
Fig. 1: Contextual actions, represented by arrows, can help a heuristic search algorithm both avoid local minima (from orange state) and escape local minima (from the blue state).

For any new planning problem the robot is given, we can sample from the learned distribution during the search process to take actions that make quicker progress towards the goal. As such, our approach benefits from being able to continuously learn and refine its estimate of the distribution. In our approach, we rely on a corpus of prior experience data in the form of (x, x', x_{goal}) tuples, where x and x' are states along a bounded suboptimal path to goal x_{goal}[2]. In this preliminary work, we consider the simple case where contextual actions $\vec{a} = \overrightarrow{xx'}$ are vectors in the state space. Furthermore, the approach in this paper requires computation of the feature encoding of a state $\phi(x)$ at every expansion during the heuristic search. This is computationally expensive and can increase overall planning clock times even though it significantly decreases the number of states expanded. We believe that this work nicely incorporates the use of learned models in heuristic search and motivates further research to resolve such simplifying assumptions and shortcomings.

II. Preliminaries

A. Problem Formulation

In this paper we focus on kinematic path planning problems in 2D workspaces $\mathcal{X} \sim \mathcal{D}$ drawn from a distribution $\mathcal{D}$ as a proof-of-concept for using generative models in

[2]In this notation we assume that x' is further along the path (i.e. closer to x_{goal}) than x.

conjunction with heuristic search. Our goal is to reduce the number of states expanded by the heuristic search algorithm, while providing guaranteed bounds on solution suboptimality. The effect of modeling the distribution $p(\vec{a}|\phi(x), \vec{x}_{\text{goal}})$ from experience is that actions $\vec{a}$ result in longer edges in our search graph that cut across free space. In the case where such edges are invalid or infeasible, the search can always progress by expanding states generated using the usual action space of the robot. We rely on the new edges to reduce state expansions, and the original action space to maintain bounds on solution suboptimality.

B. Heuristic Search-Based Motion Planning

As test domains, we consider point-robots and planar arms with up to 5 degrees-of-freedom operating in 2D workspaces $\mathcal{X} = \mathcal{X}_{\text{free}} \cup \mathcal{X}_{\text{obs}}$, where $\mathcal{X}_{\text{free}}$ is the free space that the robot can move in, and $\mathcal{X}_{\text{obs}}$ is the space occupied by obstacles. A planning problem is defined by the tuple $(\mathcal{X}, x_{\text{start}}, x_{\text{goal}}, \mathcal{A})$, which includes a start configuration x_{start}, a desired goal configuration x_{goal}, and the action space $\mathcal{A}$. For every state $x \in \mathcal{X}_{\text{free}}$ that the search algorithm considers, it evaluates all possible successors $x' \in \{\vec{a}(x) \mid \vec{a} \in \mathcal{A}\}$. The output of $\vec{a}(x)$ includes all the states the robot enters while taking action $\vec{a}$ from state x. However, by x' we specifically refer to the final state in these sequences (we use $x_{\vec{a}}$ to refer to intermediate states in Algorithm 1.). Each edge along the final path found by the search is an action $\vec{a} \in \mathcal{A}$.

C. Conditional Variational Autoencoders

The core of our work relies on using a conditional variational autoencoder (CVAE) to learn a distribution over contextual actions for the search algorithm. CVAEs are latent variable models [6], [7] that factor an unknown posterior distribution over a latent variable of fixed dimension. We model the distribution $p(\vec{a}|\phi(x), \vec{x}_{\text{goal}})$ with a CVAE using information from previous planning problems that the robot has solved as this lets us reason about edges (actions) which are beneficial in any given state more succintly. For discrete search algorithms like the ones we use for path planning, identifying such beneficial edges and determining which ones to add to the search graph can be computationally expensive. By learning a generative model in continuous space, we can sample a sparse set of edges to add to our search graph.

III. Approach

For both the point-robot and planar arm, we use 200×200 2D workspaces $\mathcal{X}$ from an existing open-source dataset [8]. Our approach proceeds in two distinct phases. First, we solve several planning problems in these workspaces in order to create our training dataset. This dataset is used to train our CVAE in order to learn the distribution $p(\vec{a}|\phi(x), \vec{x}_{\text{goal}})$. In the second phase, the CVAE is used as part of the heuristic-search algorithm for previously unseen planning problems.

A. Dataset Collection and Training

We solve M planning problems each in N 2D workspaces using Weighted A* search [9]. A planning problem is defined

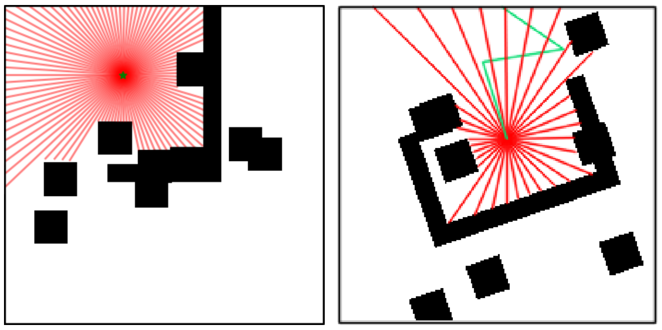
(a) (b)

Fig. 2: Features $\phi(x)$ for a (a) point-robot, and (b) planar arm with 3 degrees-of-freedom. The simulated laser readings (in red) are used as the feature vector. For a planar arm, rays are cast from all joints and the readings are concatenated to form the feature vector.

as the tuple $(\mathcal{X}, x_{\text{start}}, x_{\text{goal}}, \mathcal{A})$ where $\mathcal{X} \sim \mathcal{D}$, $x_{\text{start}}, x_{\text{goal}} \in \mathcal{X}_{\text{free}}$, and $\mathcal{A}$ is the action space. For both the point-robot and the planar arm, for a state x, we use Euclidean distance $h(x) = \omega \cdot \|x - x_{\text{goal}}\|$ in the configuration space as the heuristic, inflated by a factor of $\omega = 5$. The solved paths are post-processed by shortcutting [10]. For each state in the shortened path, we compute a feature representation $\phi(x)$ comprised of 100 simulated laser readings in the workspace $\mathcal{X}$. Examples of $\phi(x)$ for the point-robot and planar arm (3 degrees-of-freedom) are shown in Figure 2. Given a state x, we compute the contextual action $\vec{a} = \overrightarrow{xx'}$ as the vector to the next state along the shortened path x'. Additionally, we also compute the vector to the goal state $\vec{x}_{\text{goal}}$. Thus we convert all raw tuples (x, x', x_{goal}) from the NM solved planning problems to datapoints $(\phi(x), \vec{a}, \vec{x}_{\text{goal}})$ which are appended to our dataset Φ. The CVAE is trained to minimise the well-known evidence lower bound for the posterior distribution $p(\vec{a}|\phi(x), \vec{x}_{\text{goal}})$ given this dataset Φ [7]. By conditioning the posterior on $\vec{x}_{\text{goal}}$, our goal is to predict contextual actions likely to lead to the specified goal state x_{goal}. Alternatively, for our point-robot experiments, we also consider training our CVAE to model the posterior $p(\vec{a}|\phi(x))$ in an attempt to predict contextual actions likely to lead to *any* goal in the workspace $\mathcal{X}$. Our results in Section IV show that both these models help us reduce the number of state expansions in heuristic search.

B. Contextual Actions for Heuristic Search

Contextual actions can be easily incorporated into a generic search-based motion planning framework. We only modify the Expand function for these algorithms. In addition to generating successor states from the regular action space $\mathcal{A}$ of the robot, we also generate successors, if feasible, by sampling actions from the posterior learned by the CVAE. This process is described in Algorithm 1. Lines 7- 12 sample k contextual actions from the posterior distribution modeled by the CVAE, evaluate whether these actions are feasible in $\mathcal{X}$, and if so, add the final state x' to the search graph as a

Algorithm 1 State Expansion with Contextual Actions

```
1: procedure EXPAND($x, k$)
2:     $\mathcal{O} \leftarrow \mathcal{O} \setminus \{x\}$                    ▷ $\mathcal{O}$ is the search frontier
3:     for $x' \in \{\vec{a}(x) \mid \vec{a} \in \mathcal{A}\}$ do
4:         if $\vec{a}(x)$ is collision-free in $\mathcal{X}$ then
5:                                     ▷ True if $x_{\vec{a}} \in \mathcal{X}_{\text{free}} \forall x_{\vec{a}} \in \vec{a}(x)$
6:             $\mathcal{O} \leftarrow \mathcal{O} \cup \{x'\}$
7:     for $i \in 1, \cdots, k$ do
8:         Sample $a \sim p(\vec{a} | \phi(x), \vec{x}_{\text{goal}})$
9:         if $\vec{a}(x)$ is collision-free in $\mathcal{X}$ then
10:            $x' \leftarrow a(x)$
11:            if $x' \notin \mathcal{O}$ then
12:                $\mathcal{O} \leftarrow \mathcal{O} \cup \{x'\}$
```

successor. We ensure that a sampled contextual action results in a successor state that lies on the discrete lattice used by our underlying heuristic search algorithm, by snapping x' to the closest lattice point. The additional step in Line 11 checks whether a newly computed x' has already been added to the set $\mathcal{O}$, which represents the search frontier, during the current call to EXPAND.

Theoretical Analysis: Our Weighted A* search graph consists of two types of edges: those that correspond to an action from the original action space $\mathcal{A}$ which we represent as $e_{\mathcal{A}}$, and those that correspond to a contextual action generated by the CVAE, which we represent as e_C. Since we use Weighted A* as the backbone search algorithm, we preserve guarantees on *completeness* and *suboptimality* with respect to the original graph (without learned contextual actions), while using learned contextual actions during the search to reduce state expansions.

Theorem 1 (Completeness)**.** *Heuristic search-based motion planning with learned contextual actions is complete, i.e. it will find a solution if one exists.*

Proof (sketch). Weighted A* without contextual actions is complete [11]. Since we only add edges e_C to the search graph, any solution that might be found by Weighted A* without contextual actions can be found by Weighted A* with contextual actions. □

Theorem 2 (Suboptimality)**.** *Weighted A* with learned contextual actions using an inflation factor $\omega \geq 1$ is ω-optimal, i.e. the cost of any solution π returned by the algorithm is no worse than ω times the cost of the optimal solution, $c(\pi) \leq \omega \cdot c(\pi^*)$ where $c(\cdot)$ is the edge cost function*[3].

Proof (sketch). Weighted A* without contextual actions is ω-optimal [11]. A feasible edge e_C connects two states x and x'. Since this edge is feasible in $\mathcal{X}$, there exists a path $\pi(x, x')$ from x to x' comprised only of edges $e_{\mathcal{A}}$. Thus, $c(e_C) \leq \sum_{e_{\mathcal{A}} \in \pi(x,x')} c(e_{\mathcal{A}})$ as edge costs are non-negative and additive. This preserves the ω-optimality of Weighted A* with contextual actions. □

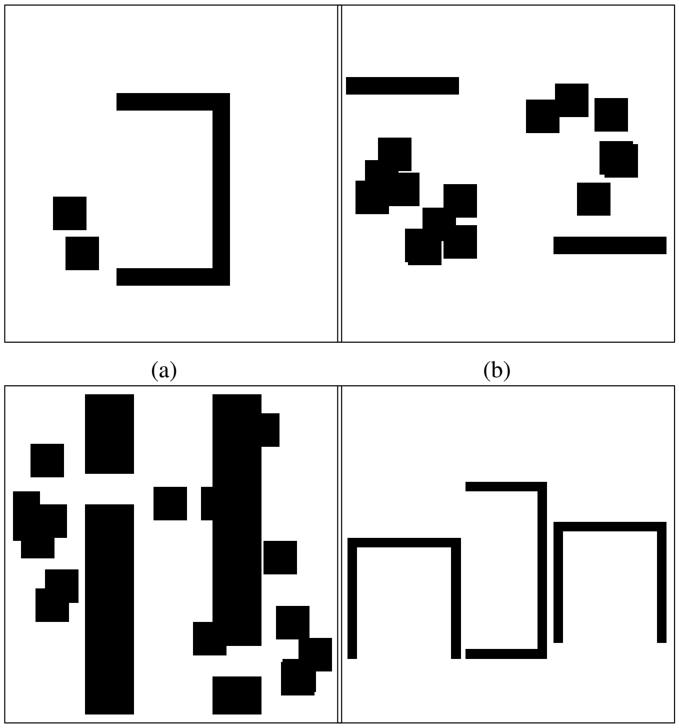

Fig. 3: Sample instances of the four different workspace types: (a) Bugtrap Forest, (b) Forest, (c) Gaps and Forest, (d) Multiple bugtraps.

IV. Experimental Results

We used four types of 2D workspaces $\mathcal{X}$ for our point-robot experiments. Figure 3 shows one example each from these categories. For planar arm planning, we used environments from category Figure 3a with random rotations applied to each workspace. We solved $M = 10$ paths with randomly generated start and goal states in $N = 800$ environments for each category. To shorten our solution paths, we randomly sampled 25 pairs of states in the path, and removed all states in between if those two could be connected via line-of-sight.

Our CVAE architecture and training procedure is implemented using PyTorch $v1.1.0$ [12]. The encoder has two hidden layers with $(128, 64)$ neurons respectively, and ReLU activations. The decoder is a mirror of the encoder. We use latent variables with two dimensions. The network architecture was kept constant for all experiments, and we did not spend any time optimising the architecture. Both the point-robot and planar arm experiments use 100 simulated laser readings as $\phi(x)$. We did not append $\vec{x}_{\text{goal}}$ to the feature vector for point-robot planning, but we did for planar arm planning.

A. Simulation Experiments

We present results for point-robot and planar arm motion planning with heuristic search. Our experiments show that

[3]The cost of a path is sum of edge costs of the edges that make up the path, $c(\pi) = \sum_{e \in \pi} c(e)$

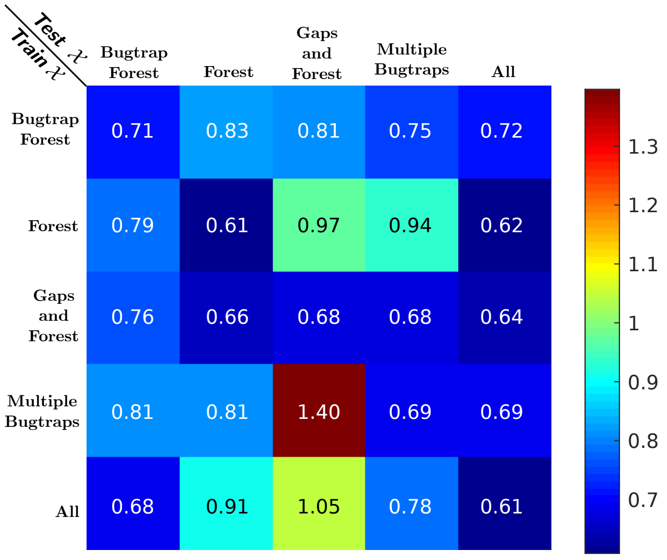

Fig. 4: Performance of heuristic search with learned contextual actions for different combinations of train and test workspaces $\mathcal{X}$.

our approach can be used to improve the computational cost of search-based planning problems across a spectrum of state spaces from the relatively simple 2D state space of a point-robot, to much more complicated and unintuitive configuration spaces for planar arms. For the point-robot experiments, we first sample $1,000$ contextual actions from $p(\vec{a}|\phi(x))$, and then cluster the resulting successor states x' into $k = 4$ clusters using K-means clustering [13]. The contextual actions a corresponding to these cluster centers x' are the ones we use in Line 8 of Algorithm 1. For the planar-arm, we directly sample $k = 1$ contextual actions from $(\phi(x), \vec{a}, \vec{x}_{\text{goal}})$ in Line 8 of Algorithm 1.

We present quantitative results by comparing the number of states expanded by Weighted A* with and without learned contextual actions across 50 test planning problems, along with the cost of the solution paths. Specifically, we calculate the ratios of these numbers with and without learned contextual actions. Metric **M1** is the ratio of state expansions, and metric **M2** is the ratio of solution costs.

Table I contains experimental results for point-robot planning while Table II contains results for planar arm planning. In each table, a separate CVAE was trained for the results in each row. For point-robot planning, Weighted A* performs roughly 30-40% fewer state expansions with learned contextual actions than without. In the case of planar arm planning, contextual actions help save roughly 10-20% state expansions on average. Figure 5 visualises the states expanded by Weighted A* with and without learned contextual actions for 2D point-robot planning.

In order to test the flexibility of our approach, for point-robot planning, we also tested CVAEs trained on one workspace on planning problems in other workspaces from Figure 3. Furthermore, we also trained a CVAE on samples from all workspaces as part of this approach (we refer to

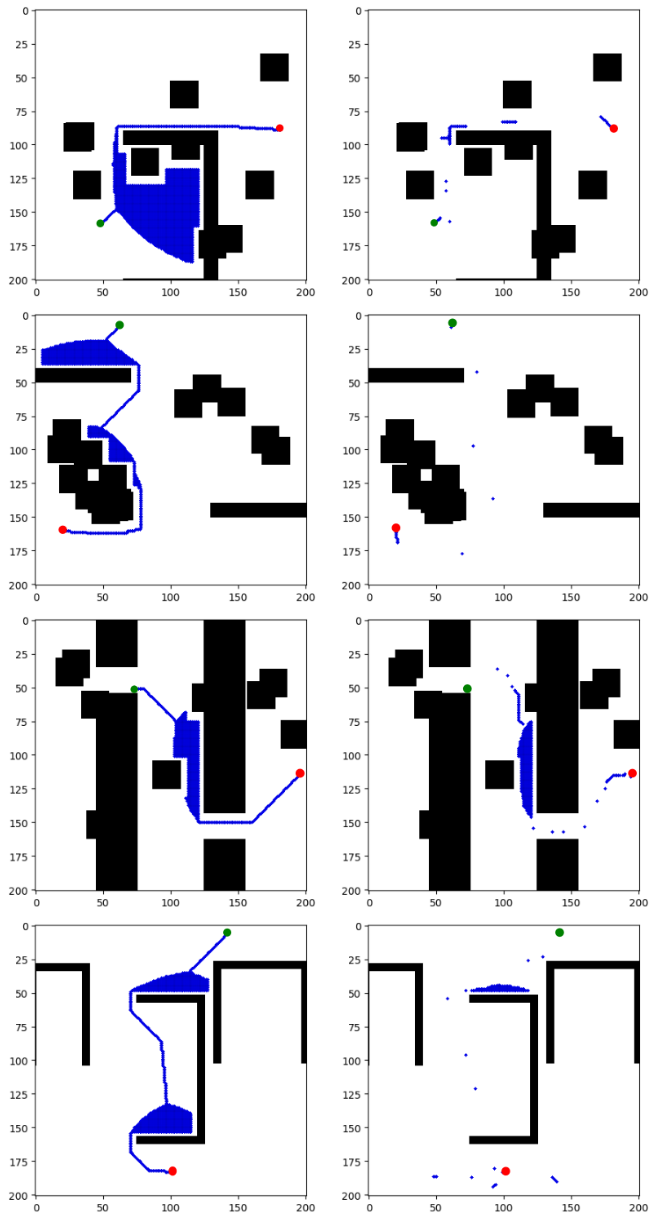

Fig. 5: Qualitative results for point-robot planning. State expansions are shown in blue for Weighted A* without learned contextual actions (*left*), and with learned contextual actions (*right*). Start state is in green, goal state in red.

the combination of all other workspaces as "all"). Figure 4 presents the results of these experiments. Each cell in the grid is the ratio of the numbers of states expanded by Weighted A* with and without learned contextual actions during the test problems. We see that heuristic search with learned contextual actions is robust to a mismatch between the train and test workspaces. We attribute this to the fact that our learned distribution $p(\vec{a}|\phi(x))$ is conditioned on local features of the state $\phi(x)$. If the feature representation is consistent across workspaces, our model will capture the inherent dependency of contextual actions on these features. The local features also help us avoid the pitfall of latent variable models that usually need samples from the entire state space to learn a distribution that generalises well [14].

TABLE I: Performance for point-robot motion planning

2D Workspace	**M1**	**M2**
Bugtrap Forest	0.71 ± 0.23	1.05 ± 0.06
Forest	0.61 ± 0.19	1.05 ± 0.08
Gaps and Forest	0.68 ± 0.24	1.04 ± 0.07
Multiple Bugtraps	0.69 ± 0.23	1.05 ± 0.08

TABLE II: Performance for planar arm motion planning

Degrees-of-Freedom	**M1**	**M2**
2	0.84 ± 0.55	1.19 ± 0.09
3	0.93 ± 0.34	1.05 ± 0.06
4	0.78 ± 0.65	1.20 ± 0.10
5	0.81 ± 0.48	1.28 ± 0.12

V. Related Work

A. Heuristic Search

Heuristic search-based motion planning has been used for a diverse set of applications including aggressive quadrotor flight [15], humanoid motion planning [16], and multi-agent pathfinding [17]. Phillips et al. [18] present an approach that can reuse previously planned paths or expert demonstrations to build an *experience graph*. The goal is for the search to latch on to this graph to find new solutions quickly. The work of Bhardwaj et al. [8] is perhaps most closely related to ours within the heuristic search framework. They rely on imitating an optimal search algorithm to select the next state to be expanded. The search is then biased towards expanding states that are more likely to lie on the path to the goal. In our work however, we do not learn such a selection policy over state expansions, and use contextual actions to bypass unnecessary state expansions by augmenting the search graph.

B. Action Priors

Similar to our intuition of using past experiences to learn a generative model over contextual actions, existing work has considered simply learning a prior probability of action utility in any given state. This prior over the action space is then used to either eliminate actions that will not be useful for exploration [19], or incentivise the use of actions that will be useful to achieve a specified goal, given a goal-conditioned prior distribution over actions [20].

C. Neural Motion Planning

The use of machine learning to aid classical motion planning has been addressed within the emerging field of *neural motion planning*, and has given rise to numerous approaches to solve the problem. A number of works look at learning a latent space representation of the robot's state space or observation space [21], [22], [23], [24], [25]. The underlying assumption is that this learned latent space encapsulates all features important for planning, and makes motion planning faster as it is a lower-dimensional space by construction. Since most of these techniques focus on learning global state distributions, they have most commonly been used within sampling-based planning algorithms [26].

D. Goal-Conditioned Reinforcement Learning

The seminal work of Kaelbling [27] has spawned research into using reinforcement learning to learn policies conditioned on the specified goal, in addition to the state of the robot. One early attempt looked at learning goal-conditioned value functions [28] where the robot state is augmented with the goal location at each timestep. A number of different works have used local controllers or navigation policies to move between waypoints learned via goal-conditioned reinforcement learning. These waypoints can be selected by running a shortest-path graph search over a graph where each node is a previously visited location in the environment [29], [30]. A recent method computes a complete trajectory of waypoints in a recursive manner by iteratively predicting midpoint subgoals given an intial start and goal location [31]. Subgoals can also be extracted by using *successor representations*, which computes the action-value function of a state in terms of the expected discounted occupancy of states in the future [32], [33].

VI. Conclusion and Discussion

We present an approach for combining generative models in the form of Conditional Variational Autoencoders (CVAEs) with heuristic search-based motion planning algorithms for learning contextual actions, given local feature information about the state of a robot. Our results show that our approach helps save 10-40% of computation in terms of numbers of states expanded. In the work we present here, computing the feature vector $\phi(x)$ for a state x can be time consuming for our planning algorithms, especially since we do so for every state expanded as per Algorithm 1. Instead of querying our learned model for actions at every state expansion, we would like to look into policies that determine when to query such a model. In addition, the successors generated by contextual actions could be used as attractor states (or waypoints or subgoals) in the search space, and we might reduce state expansions while avoiding heavy featurisation costs by biasing the search frontier progression towards such attractors. We argue that augmenting traditional heuristic search with learned contextual actions can result in considerable improvements in performance. While we restrict ourselves to simple test domains in this preliminary work, we hope to extend it to more complicated environments and robots, and apply this approach to kinodynamic planning problems in the future.

Acknowledgements

The authors would like to thank Yash Oza and Tushar Kusnur for their help with the implementation of some of the experiments used in this work.

References

[1] H. Ma, S. Koenig, N. Ayanian, L. Cohen, W. Hönig, T. Kumar, T. Uras, H. Xu, C. Tovey, and G. Sharon, "Overview: Generalizations of multi-agent path finding to real-world scenarios," *arXiv preprint arXiv:1702.05515*, 2017.

[2] N. Correll, K. E. Bekris, D. Berenson, O. Brock, A. Causo, K. Hauser, K. Okada, A. Rodriguez, J. M. Romano, and P. R. Wurman, "Analysis and observations from the first amazon picking challenge," *IEEE Transactions on Automation Science and Engineering*, vol. 15, no. 1, pp. 172–188, 2016.

[3] C. Eppner, S. Höfer, R. Jonschkowski, R. Martín-Martín, A. Sieverling, V. Wall, and O. Brock, "Lessons from the amazon picking challenge: Four aspects of building robotic systems." in *Robotics: Science and Systems*, 2016.

[4] D. Bochtis, C. Sørensen, and S. Vougioukas, "Path planning for in-field navigation-aiding of service units," *Computers and Electronics in Agriculture*, vol. 74, no. 1, pp. 80–90, 2010.

[5] P. Trautman, J. Ma, R. M. Murray, and A. Krause, "Robot navigation in dense human crowds: Statistical models and experimental studies of human–robot cooperation," *The International Journal of Robotics Research*, vol. 34, no. 3, pp. 335–356, 2015.

[6] K. Sohn, H. Lee, and X. Yan, "Learning structured output representation using deep conditional generative models," in *Advances in Neural Information Processing Systems 28*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds. Curran Associates, Inc., 2015, pp. 3483–3491.

[7] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

[8] M. Bhardwaj, S. Choudhury, and S. Scherer, "Learning heuristic search via imitation," in *1st Annual Conference on Robot Learning, CoRL 2017, Mountain View, California, USA, November 13-15, 2017, Proceedings*, 2017, pp. 271–280.

[9] I. Pohl, "Heuristic search viewed as path finding in a graph," *Artificial intelligence*, vol. 1, no. 3-4, pp. 193–204, 1970.

[10] R. Geraerts and M. H. Overmars, "Creating high-quality paths for motion planning," *I. J. Robotics Res.*, vol. 26, no. 8, pp. 845–863, 2007. [Online]. Available: https://doi.org/10.1177/0278364907079280

[11] M. Likhachev, G. J. Gordon, *et al.*, "ARA*: Formal Analysis," 2010.

[12] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in PyTorch," in *NIPS Autodiff Workshop*, 2017.

[13] D. Arthur and S. Vassilvitskii, "k-means++: the advantages of careful seeding," in *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2007, New Orleans, Louisiana, USA, January 7-9, 2007*, N. Bansal, K. Pruhs, and C. Stein, Eds. SIAM, 2007, pp. 1027–1035. [Online]. Available: http://dl.acm.org/citation.cfm?id=1283383.1283494

[14] D. Ghosh, A. Gupta, and S. Levine, "Learning actionable representations with goal-conditioned policies," *CoRR*, vol. abs/1811.07819, 2018.

[15] S. Liu, K. Mohta, N. Atanasov, and V. Kumar, "Search-based motion planning for aggressive flight in se(3)," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 2439–2446, July 2018.

[16] A. Dornbush, K. Vijayakumar, S. Bardapurkar, F. Islam, M. Ito, and M. Likhachev, "A single-planner approach to multi-modal humanoid mobility," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, May 2018, pp. 1–9.

[17] M. Liu, H. Ma, J. Li, and S. Koenig, "Task and path planning for multi-agent pickup and delivery," in *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, ser. AAMAS '19. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2019, pp. 1152–1160.

[18] M. Phillips, B. Cohen, S. Chitta, and M. Likhachev, "E-graphs: Bootstrapping planning with experience graphs," in *In Proceedings of the Robotics: Science and Systems Conference (RSS 2012)*, July 2012.

[19] B. Rosman and S. Ramamoorthy, "What good are actions? accelerating learning using learned action priors," in *2012 IEEE International Conference on Development and Learning and Epigenetic Robotics, ICDL-EPIROB 2012, San Diego, CA, USA, November 7-9, 2012*. IEEE, 2012, pp. 1–6. [Online]. Available: https://doi.org/10.1109/DevLrn.2012.6400810

[20] D. Abel, D. E. Hershkowitz, G. Barth-Maron, S. Brawner, K. O'Farrell, J. MacGlashan, and S. Tellex, "Goal-based action priors," in *Proceedings of the Twenty-Fifth International Conference on Automated Planning and Scheduling, ICAPS 2015, Jerusalem, Israel, June 7-11, 2015*, R. I. Brafman, C. Domshlak, P. Haslum, and S. Zilberstein, Eds. AAAI Press, 2015, pp. 306–314. [Online]. Available: http://www.aaai.org/ocs/index.php/ICAPS/ICAPS15/paper/view/10589

[21] R. Kumar, A. Mandalika, S. Choudhury, and S. S. Srinivasa, "LEGO: leveraging experience in roadmap generation for sampling-based planning," *CoRR*, vol. abs/1907.09574, 2019. [Online]. Available: http://arxiv.org/abs/1907.09574

[22] B. Ichter and M. Pavone, "Robot motion planning in learned latent spaces," *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 2407–2414, July 2019.

[23] A. H. Qureshi, Y. Miao, A. Simeonov, and M. C. Yip, "Motion planning networks: Bridging the gap between learning-based and classical motion planners," *CoRR*, vol. abs/1907.06013, 2019. [Online]. Available: http://arxiv.org/abs/1907.06013

[24] B. Ichter, J. Harrison, and M. Pavone, "Learning sampling distributions for robot motion planning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, May 2018, pp. 7087–7094.

[25] T. M. Moerland, J. Broekens, and C. M. Jonker, "Learning multimodal transition dynamics for model-based reinforcement learning," *arXiv preprint arXiv:1705.00470*, 2017.

[26] S. M. LaValle and J. James J. Kuffner, "Randomized kinodynamic planning," *The International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, 2001. [Online]. Available: https://doi.org/10.1177/02783640122067453

[27] L. P. Kaelbling, "Learning to achieve goals," in *Proceedings of the 13th International Joint Conference on Artificial Intelligence. Chambéry, France, August 28 - September 3, 1993*, R. Bajcsy, Ed. Morgan Kaufmann, 1993, pp. 1094–1099.

[28] T. Schaul, D. Horgan, K. Gregor, and D. Silver, "Universal value function approximators," in *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, ser. JMLR Workshop and Conference Proceedings, F. R. Bach and D. M. Blei, Eds., vol. 37. JMLR.org, 2015, pp. 1312–1320. [Online]. Available: http://proceedings.mlr.press/v37/schaul15.html

[29] B. Eysenbach, R. Salakhutdinov, and S. Levine, "Search on the replay buffer: Bridging planning and reinforcement learning," in *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*, H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, and R. Garnett, Eds., 2019, pp. 15 220–15 231. [Online]. Available: http://papers.nips.cc/paper/9660-search-on-the-replay-buffer-bridging-planning-and-reinforcement-learning

[30] N. Savinov, A. Dosovitskiy, and V. Koltun, "Semi-parametric topological memory for navigation," in *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. [Online]. Available: https://openreview.net/forum?id=SygwwGbRW

[31] T. Jurgenson, O. Avner, E. Groshev, and A. Tamar, "Sub-goal trees – a framework for goal-based reinforcement learning," 2020.

[32] R. Ramesh, M. Tomar, and B. Ravindran, "Successor options: An option discovery framework for reinforcement learning," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, S. Kraus, Ed. ijcai.org, 2019, pp. 3304–3310. [Online]. Available: https://doi.org/10.24963/ijcai.2019/458

[33] T. D. Kulkarni, A. Saeedi, S. Gautam, and S. J. Gershman, "Deep successor reinforcement learning," *CoRR*, vol. abs/1606.02396, 2016. [Online]. Available: http://arxiv.org/abs/1606.02396