Informed Multi-Representation Multi-Heuristic A*

***Abstract*— Generating motion plans for robots, like humanoids, with many degrees of freedom is a challenging problem because of the high-dimensionality of the resulting search space. To circumvent this, many researchers have made the observation that large parts of the solution plan are often much lower dimensional in nature. Some recent algorithms exploit this by either planning on a graph with adaptive dimensionality or leveraging a decoupling in the robot's action space. Often, it is possible to gain more fine-grained information about the local dimensionality of the plan from any robot state to the goal which can then be used to inform search. In this work, we present a heuristic-search-based planning algorithm that admits such information as a prior in the form of lower dimensional manifolds (called representations) and a probabilistic mapping (conditioned on the world and the goal) from robot states to these representations. We train a Conditioned Variational Autoencoder (CVAE) for every representation and use them to compute the required probabilitic mapping. Using this additional domain knowledge, our motion planner is able to generate high quality bounded-suboptimal plans. Experimentally, we validate the practicability and efficiency of our approach on the challenging 10 degree-of-freedom mobile manipulation domain.**

I. INTRODUCTION

We propose to implement a learning-based planning method that utilizes successful plans generated from expensive sources during training time (such as time-consuming planners or humans in given environments), in order to better inform search during test time in unseen environments. [1] presents an approach that uses conditional variational autoencoders over states along available plans and learns a distribution that captures the validity/desirability of states conditioned on the environment map, start and goal. This distribution is used to bias samples for sampling-based planning algorithms and guide their search in a more informed direction, for example, expert plans through narrow passageways or trajectories that avoid obstacles. In our work we explore the idea of controlling expansions made by the Multi-Representation Multi-Heuristic A* (MR-MHA*) algorithm [2]. MR-MHA* is a variant of MHA* in which each queue focuses on only a small set of dimensions (called a representation). We will make use of the model proposed in [1] to learn a distribution over different representations so as to better schedule expansions from the queues.

II. RELATED WORK

The idea of dividing a robotic system into kinematic subsystems has been used extensively in motion planning for humanoids. [3] uses an RRT-based planner to plan for a 43 dof humanoid robot. Planning is initially attempted using a small subsystem of the robot. Additional dof are adaptively added based on the distance of the end-effector of the robot to the goal or if planning using a smaller subsystem of the robot failed. Further, every collision situation corresponds to a fixed lower dimensional representation of the robot. [4] uses a similar idea of using the RRT planner in an adaptively dimensional C-space by adaptively selecting joints based on the distance of the corresponding link from the goal and obstacles.

In the heuristic-search community, a host of algorithms targeted at graphs with large branching factors have been developed. Adaptive Dimensionality proposed in [5] maintains two kinds of state-spaces- a high dimensional space S^{hd} and a lower-dimensional space S^{ld} that is a projection of S^{hd} onto a lower dimensional manifold. The planner then seeks to plan in S^{ld} as much as possible and expand the search into S^{hd} only if it is necessary to ensure the feasibility of the resulting path. Enhanced Partial Expansion A* [6] seeks to improve planning time by doing only partial expansions of nodes. Given some domain and heuristic-specific knowledge, it brings down the effective branching factor of the graph by generating only *some* successors of a vertex.

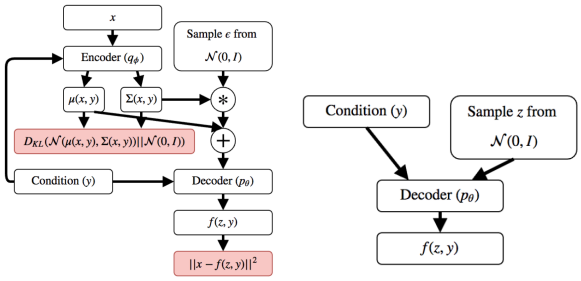
III. METHODOLOGY

A. Planning Formulation

Our problem formulation consists of a robot (R) that has a total of 10 degrees of freedom. The arm consists of 7 degrees $[R_A = (\theta_1, ..., \theta_7)]$ while the base consists of 3 degrees $[R_B = (x, y, \phi)]$. The robot is tasked with navigation and manipulation in a given 3D environment that consists of narrow passages in the forms of gaps between walls (ENV $= [(x_1, y_1), ..., (x_n, y_n)]$). The environment may or may not consist tables with objects present on them. The start state of the robot in this environment corresponds to its current base location and configuration of the arm's joint angles ($R_{start} =< R_A, R_B >$). The goal state of the robot is the position and rotation of the arm's end effector ($R_{goal} =< RE_x, RE_y, RE_z, RE_{\text{roll}}, RE_{\text{pitch}}, RE_{\text{yaw}} >$). The vanilla MR-MHA* algorithm [2] finds a path from R_{start} to R_{goal}. However as opposed to MHA* which would expand the full 10 dimensional robot state, MR-MHA* expands either R_A or R_B. In our implementation, to further speed up the search, we try to inform the selection of which dimension to expand based on where in the environment the search is at a given point.

B. CVAE Architecture

Our architecture is inspired from the work in [1] and resembles the standard CVAE architecture that is used in literature. As shown in Figure 1(a), it consists of an encoder that takes a training sample and conditioning variable as input. The encoder consists of 3 fully connected layers with 400 neurons each. The samples are encoded into a latent distribution represented by vectors $\mu(x, y), \Sigma(x, y)$, each of dimension 3. The vectors are then concatenated with noise sampled from $\mathbf{N}(0, 1)$ and passed into the decoder which again has 3 layers of size 400 each. The reconstructed output



(a) (b)

Fig. 1: (a) CVAE Training (b) CVAE Testing [1]

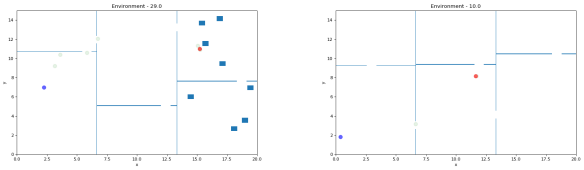


Fig. 2: Top views of 2 Randomly generated map variants used for testing of CVAE and MR-MHA* pipeline (lines are walls in 3Ds and squares are tables)

from the decoder and $\mu(x, y), \Sigma(x, y)$ from the encoder are combined into the following loss function :

$$\mathbf{L} = \|x - f(z, y)\|^2 + D_{KL}(\mathcal{N}(\mu(x, y), \Sigma(x, y))\|\mathcal{N}(0, I)) \tag{1}$$

As noted earlier, for informing MR-MHA* using the knowledge of the environment, we want to predict environment locations where base should be expanded during search and where arm should be expanded. For this purpose we train two CVAEs, one for predicting locations where base should be used and another for predicting where arm dimension should be used. The input to either CVAE for training is the (x,y) location of the robot's base concatenated with the conditioning variable which in our case is the concatenation of : the (x,y) location of robot's base in R_{start}, the (x,y) location of the robot's base in R_{goal}, the (x,y) location of all the narrow gaps in the map.

C. CVAE Training

Data Generation We require that the trained CVAE model is able to generalise to different start/goal pairs and different environments where positions of narrow gaps and obstacles change. Thus we developed a methodology to generate random maps with certain given specifications such as maximum number of gaps, height of the tables, width of the narrow gaps etc. A total of 20 maps were generated and 3 out of those were used for testing (as shown in Figure 2). The test maps were not shown to the network during training phase. After generating the maps, the next step was to generate random collision free start/goal pairs for each map which could then be used by vanilla MR-MHA* to generate trajectories and data for the CVAEs. We generated 500 start/goal pairs for each map by sampling random locations. For maps that contain tables, the goal was sampled to a location above each table, as would be required in a planning for manipulation task. The combination of generated maps and start/goal pairs requires the robot to traverse the map, go through narrow passages and finally position and orient its end-effector at the required goal location. Due to the complex nature of this task, the final generated trajectories contain expansions in arm as well as base dimension and all the trajectory data together serves as an input for training the two CVAEs. Observation of training also revealed, that the final plan does include arm expansions near the narrow gaps (where the robot may have to move its arm to get through the door) and near the goal where a table may or may not be located. Thus, if the CVAEs are trained to generate this data, they can be used during test time to inform MR-MHA* as described in the next section.

D. CVAE Testing

Running CVAE The first step to test the planner with incorporated CVAE output is to get predictions from the CVAE. For this purpose, the start/goal pair from the planning query and the narrow gap locations in the environment are concatenated together into the conditioning variable. This condition along with noise sampled from $\mathbf{N}(\mathbf{0}, \mathbf{1})$ is passed into the trained decoder models of arm and base CVAE to (x,y) locations on the map where arm and base should be expanded (as shown in Figure 1(b)). Figure 3(left) shows samples generated by arm and base decoders for a given map and start/goal locations. As we can see that the CVAE has been well conditioned on the gaps and start/goal pairs, generating samples going though the gaps in an unseen test environment. In addition the arm CVAE output in Figure 3(a) shows high point density near the narrow gaps, start and goal, indicating a greater likelihood for arm use near these, a behaviour exhibited by our training planner on environments such as these. The base CVAE output in 3(b) on the other hand is more uniform and spread throughout the path. For our purposes, we generate a set of 1000 samples every time.

Computing Expansion Probabilities Once the CVAE outputs have been obtained, it is required to convert the information into probabilities in a way that for a given (x,y) the probability indicates which dimension should be more likely to be expanded. For this purpose, we utilise a nearest neighbour search in both CVAE outputs. More specifically, each CVAE output is stored as a KD-Tree (T_{arm} and T_{base}). Now for a given (x,y) in the map, we search both KD-Trees in a pre-specified radius (for out tests this was set to 0.5m) and calculated the probabilities as follows :

$$P_{arm}(x, y) = \frac{N(T_{arm})}{N(T_{arm}) + N(T_{base})} \tag{2}$$

$$P_{base}(x, y) = \frac{N(T_{base})}{N(T_{arm}) + N(T_{base})} \tag{3}$$

Here $N(T_{arm})$ and $N(T_{base})$ denote the number of nearest neighbours found in the arm and base KD-Trees respectively in the pre-specified radius. This is shown in Figure 3 for an environment and start/goal pair. As we can see in the plots of P(arm) and P(base), P(arm) is higher nearer the narrow

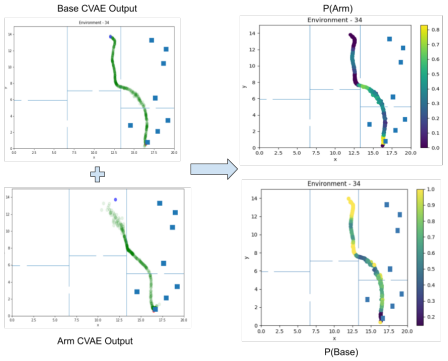


Fig. 3: Probabilities of expanding arm or base (along the base CVAE points) obtained by using nearest neighbours on CVAE outputs

gaps and near the table at the goal while P(base) is higher in the remaining trajectory paths.

Running MR-MHA* The last step in our algorithm is incorporating the computed expansion probabilities into MR-MHA*. The original algorithm is designed to handle queues corresponding to multiple heuristics. The state at the top of the queues (state with min f-value) is picked for expansion in a round-robin fashion. We modify this round-robin iteration to a discrete sampling based queue selection. As shown in Figure 4, the probability of each queue depends on its existing label (arm or base) and the probability of expansion of the state at the top for that label. Thus for Q_1 in Figure 4, $p(x_1)$ will correspond to the probability $P_{x_1}(arm)$ while for Q'_1, $p(x'_1)$ will correspond to $P_{x'_1}(base)$ and so on for other queues. These probabilities are combined to create a discrete probability distribution Q_t from which a queue is then sampled to be expanded. Intuitively, we can see that this approach will automatically expand more in the arm dimension if the min f-value states in the queues have a higher $P(arm)$ and expand more in the base dimension if the min f-value states in the queues have a higher $P(base)$.

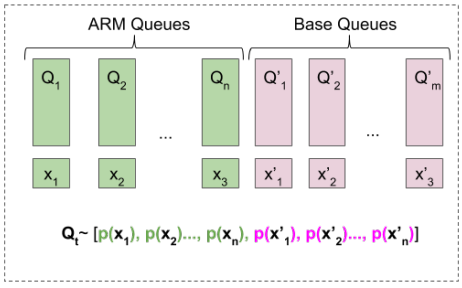


Fig. 4: MR-MHA* heuristic queues with expansion probabilities

IV. EXPERIMENTS

In order to evaluate our CVAE and planner pipeline, we tested on 3 environments similar to ones shown in Figure 2, running the planner over 300 times in total. Environment 1 and 3 had tables while 2 did not. A comparison of the original planner and the informed one we have developed can be seen in Table I and in the **YouTube video**. From the results, across the 3 environments we get an average runtime improvement

Algorithm 1 Informed MR-MHA* with m representations and n inadmissible heuristics.

procedure CHOOSEQUEUE
  for $i \in \{1, ..., n\}$ **do**
    $r \leftarrow \text{REPID}(i)$
    $s \leftarrow \text{OPEN}_i.\text{TOP}()$
    $Q \leftarrow P_r(s)$
  return $i \sim Q$
procedure INFORMED MR-MHA*(m, n)
  $g(s_{start}) \leftarrow 0$, $g(s_{goal}) \leftarrow \infty$
  $\text{OPEN}_0 \leftarrow \phi$, $\text{CLOSED}_0 \leftarrow \phi$ ▷ Anchor
  $\text{OPEN}_i \leftarrow \phi$ for $i \leftarrow 1, \ldots, n$ ▷ Inadmissible
  $\text{CLOSED}_i \leftarrow \phi$ for $i \leftarrow 1, \ldots, m$ ▷ Inadmissible
  for $i \leftarrow 0, \ldots, n$ **do**
    Insert s_{start} into OPEN_i with $\text{KEY}(s_{start}, i)$
  while $g(s_{goal}) > w_2 \cdot \text{OPEN}_0.\text{MINKEY}()$ **do**
    $i \leftarrow$ CHOOSEQUEUE()
    if $\text{OPEN}_i.\text{MINKEY}() \leq w_2 \cdot \text{OPEN}_0.\text{MINKEY}()$ **then**
      $s \leftarrow \text{OPEN}_i.\text{TOP}()$
      $\text{EXPAND}(s, i)$
      for $j \leftarrow 1, ..., m$ **do**
        if $\text{REP}(j) \subseteq \text{REP}(\text{REPID}(i))$ **then**
          Insert s in CLOSED_j
    else
      $s \leftarrow \text{OPEN}_0.\text{TOP}()$
      $\text{EXPAND}(s, 0)$
      Insert s in CLOSED_0

TABLE I: Mean Reductions with 3 environments

Environment	Number of Expansions		Runtime (in seconds)		Success Rate (%)	
	MR	I-MR	MR	I-MR	MR	I-MR
1	3482	2156	7.10	6.57	90.00	93.33
2	2619	2359	6.84	5.98	86.67	83.33
3	3162	2551	6.37	6.72	83.00	81.00

of around 5%. The reduction in number of expansions is more significant and is around 23%. The higher reduction in expansions vs runtime can be attributed to the extra time spent in KD-Tree construction and KD-Tree search done at every expansion step, which increases the runtime though overall expansions are significantly less. In addition, we also find that reduction in planning time is highly dependant on CVAE output. If the CVAE output covers points that are explored during the search and CVAE points go through the narrow gaps, the runtime reduction is significantly high. Infact, in our experiments, the maximum runtime reduction we obtained across environments was greater than 90%. This shows that this method can indeed reduce the search runtime significantly provided the CVAE output is as per search requirements. In our future work, we can improve the CVAE output by showing it more environments, thus improving the conditioning on gaps. In addition we can train the CVAE not just on the final planner generated trajectory but in general on the states that were "good" expansions for base and arm individually.

V. Video Link

https://youtu.be/1qIXPbPmPNw

References

[1] B. Ichter, J. Harrison, and M. Pavone, "Learning Sampling Distributions for Robot Motion Planning," Tech. Rep., 2018.

[2] D. Youakim, A. Dornbush, M. Likhachev, and P. Ridao, "Motion Planning for an Underwater Mobile Manipulator by Exploiting Loose Coupling," Tech. Rep., 2018.

[3] N. Vahrenkamp, C. Scheurer, T. Asfour, J. Kuffner, and R. Dillmann, "Adaptive motion planning for humanoid robots," *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*, pp. 2127–2132, 2008.

[4] D. H. Kim, Y. S. Choi, T. Park, J. Y. Lee, and C. S. Han, "Efficient path planning for high-DOF articulated robots with adaptive dimensionality," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2015-June, no. June, pp. 2355–2360, 2015.

[5] K. Gochev, B. Cohen, J. Butzke, A. Safonova, and M. Likhachev, "Path Planning with Adaptive Dimensionality," *Proceedings of the International Symposium on Combinatorial Search (SoCS)*, pp. 52–59, 2011.

[6] M. Goldenberg, A. Felner, R. Stern, G. Sharon, N. Sturtevant, R. C. Holte, and J. Schaeffer, "Enhanced partial expansion A*," *Journal of Artificial Intelligence Research*, vol. 50, no. 9, pp. 141–187, 2014. [Online]. Available: http://www.jair.org/media/4171/live-4171-7932-jair.pdf