Motion Memory: Leveraging Past Experiences to Accelerate Future Motion Planning

Dibyendu Das[1]   Yuanjie Lu[1]   Erion Plaku[2]   Xuesu Xiao[1]

Abstract— When facing a new motion-planning problem, most motion planners solve it from scratch, e.g., via sampling and exploration or starting optimization from a straight-line path. However, most motion planners have to experience a variety of planning problems throughout their lifetimes, which are yet to be leveraged for future planning. In this paper, we present a simple but efficient method called *Motion Memory*, which allows different motion planners to accelerate future planning using past experiences. Treating existing motion planners as either a closed or open box, we present a variety of ways that Motion Memory can contribute to reduce the planning time when facing a new planning problem. We provide extensive experiment results with three different motion planners on three classes of planning problems with over 30,000 problem instances and show that planning speed can be significantly reduced by up to 89% with the proposed Motion Memory technique and with increasing past planning experiences.

I. INTRODUCTION

Motion planning refers to the computational process of determining a sequence of control inputs and actions to move a robot from a given start state to a desired goal location while avoiding obstacles and observing system and environment constraints. Motion planners are essential components for almost all robotic applications [1], such as autonomous navigation [2], [3] and manipulation [4], [5]. Therefore, quick, efficient, and optimal collision-free motion planning is of paramount value to the entire robotics community.

Decades of research into motion planning have made significant progress and are able to find motion-planning solutions for different robot platforms, e.g., using Probabilistic Roadmaps (PRM) [6], Expansive Spaces algorithm [7], and Rapidly-Exploring Random Trees (RRT) [8], [9] to move mobile robots or manipulator arms. Nevertheless, these planners still face challenges in complex real-world settings when real-time planning is required to assure fast and reliable motion execution. Conventional motion planners need to plan from scratch every time they encounter a new environment. This situation remains true even when robots repeatedly face similar environments, where prior experiences could be beneficial. Such repetitive planning introduces unnecessary
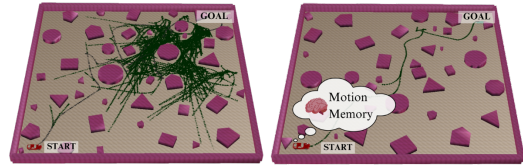
Fig. 1: Traditional motion planners require significant amount of effort to plan from scratch (left), such as large amount of samples or iterations (illustrated in green); Motion Memory utilizes past planning experiences to accelerate future planning when facing new planing problems (right).

planning time and therefore limits the robot performance in real-world environments where fast planning time can benefit the downstream tasks, such as quickly moving through highly constrained obstacle spaces.

On the other hand, advances in machine learning have demonstrated that robots are capable of learning emergent behaviors in a data-driven manner without depending on heavily engineered attributes and heuristics. One particular benefit of learning methods is the potential to continually improve with increasing real deployment experiences [10], a capability that the classical motion planners lack.

Considering the limitations of classical motion planners and the potential of learning from experiences, we present Motion Memory, a new paradigm based on past planning experiences to guide traditional motion-planning methods when facing new planning problems in order to reduce computational overhead and therefore improve planning efficiency, as robots gather more and more deployment experiences in the real world. Leveraging machine learning, Motion Memory includes an experience augmentation technique and a representation learning method that enable robots to reflect on prior planning experiences for efficient future planning as shown in Fig. 1. To be specific, the experience augmentation strategy automatically generates new planning problems, for which past motion plans are (or are not) the solutions, and thus provides Motion Memory with an extensive corpus of training data to generalize to future planning problems. Motion Memory also utilizes representation learning to enable autonomous robots to learn from augmented previous planning experiences so that motion planners can identify, store, memorize, and retrieve past planning experiences to facilitate motion planning in unseen future environments. We present different ways to integrate Motion Memory with three existing motion planners in three different categories of environments both in a closed and open box manner

to showcase the wide applicability and generalizability of the technique. Our experiments demonstrate that Motion Memory significantly reduces the motion-planning time in future unseen environments by up to 89% with increasing deployment experiences.

II. RELATED WORK

We review related work in classical motion planning and motion planning with machine learning.

A. Classical Motion Planning

Among classical motion planners, sampling-based approaches have shown to be effective in solving challenging problems in complex, unstructured environments [11], [12]. To account for dynamics [13]–[16], sampling-based motion planners often expand a motion tree whose branches correspond to collision-free and dynamically-feasible trajectories. RRT [17], [18] and its variants [19]–[21] rely on nearest neighbors to expand the motion tree toward random samples. EST [22] relies on probability distributions to push the tree toward less-explored areas. KPIECE [23] leverages a grid decomposition and interior-exterior cells. GUST [24] introduces a discrete layer based on a roadmap abstraction and relies on discrete search to expand the motion tree along shortest paths in the roadmap. The work in [25] further improves the motion-tree expansion to more aggressively follow the roadmap paths, while also increasing the clearance from the obstacles. When presented with a new planning problem, these approaches, however, have to plan from scratch as they do not learn from prior experiences. This is precisely what our Motion Memory framework addresses, enabling motion planners to leverage prior solutions to similar planning problems.

B. Learning Assisted Motion Planning

Throughout the years, the learning and planning communities have investigated a variety of strategies for implementing the concept of machine learning for efficient motion planning [26]. One intuitive approach is to apply machine learning to generate high-quality valid samples in critical regions more efficiently. Some methods produce valid samples in relevant areas by learning the representation of the configuration space [27]–[30], whereas other methods train models to bias sampling over the critical regions of predicted trajectories [31]–[33]. Another practical strategy is to minimize the number of collisions [34]–[36] by learning Gaussian mixture models [37], kernel perceptron [38], and graph neural network [39] to replace the conventional collision checker or using a learned model to determine the order of testing the nodes to identify valid paths [40]–[42].

Instead of learning a specific planning operation, other learning-based pipelines learn from prior experience to solve motion planning problems in an end-to-end manner. One strategy is to use similarity function to determine the relevant information and retrieve it from a database in the form of paths [43]–[45] or sampling distributions [46], [47]. Another approach is to train a deep neural network for efficient

motion planning by using a database of past solved problems. These methods take as input point-cloud representations of a workspace and learn to encode this point-cloud into a latent space [48]–[51]. Chamzas et al. [52] and Lien et al. [53] construct a similarity function only over the workspace to extract suitable local representations of planning problems [54].

Informed by the aforementioned works, this paper proposes to hallucinate planning problems by experience augmentation, construct a memory of past planning experiences by classifying environments as similar or dissimilar using representation learning, and retrieve relevant planning experience to assist planners in future planning. Compared to all these specific techniques developed for a specific motion planner in an ad hoc manner, our Motion Memory is designed to be a universal technique that is applicable to any motion planner and allows them to improve planning efficiency with increasing planning experiences in a principled manner.

III. MOTION MEMORY

The key inspiration behind Motion Memory is that most motion planners will encounter many different planning problems and produce different planning solutions throughout their lifetimes. Motion Memory aims to reflect on past planning experiences by generating a set of planning problems that will make the past planning solutions feasible/non-feasible. Then, utilizing representation learning, Motion Memory learns a latent space which contains efficient but also expressive information regarding the feasibility or optimality of the existing problem solutions with respect to any new planning problems in the future.

A. Problem Formulation

We formulate our motion planning problem in the state space (S-space) [55], which represents the universe of all possible robot states. For the scope of this paper, we use a 2D ground mobile robot as an example, but the general paradigm of Motion Memory has the potential to scale up to high-dimensional workspaces. A particular environment's S-space can be decomposed as $S = S_{\text{free}} \cup S_{\text{obst}}$, where $S_{\text{free}} \in \mathcal{S}_{\text{free}}$ is the set of reachable states and $S_{\text{obst}} \in \mathcal{S}_{\text{obst}}$ is the unreachable set due to obstacles, nonholonomic constraints, velocity bounds, etc. The robot state $s \in S$ is defined as in collision with obstacles when $s \in S_{\text{obst}}$ and collision free otherwise ($s \in S_{\text{free}}$). We denote the robot action to be $u \in \mathcal{U}$, e.g., commanded linear and angular velocity (v, ω), and define a motion plan $P \in \mathscr{P}$ as a sequence of such actions and resulted states $P = \{u_i, s_i \mid 1 \leq i \leq t\}$. $\mathscr{P}$ is the space of all motion plans. Notice that some motion planners may produce the resulted states only, i.e., $P = \{s_i \mid 1 \leq i \leq t\}$ and leave the action generation to a low-level controller. With the aforementioned notation, any motion planner can be defined as a function $f(\cdot)$ that can be used to produce motion plans,

$$P = f(S \mid s_{\text{s}}, s_{\text{g}}),$$
$$\text{s.t.} \quad S = S_{\text{free}} \cup S_{\text{obst}}, \quad s_{\text{s}}, s_{\text{g}}, s_i \in S_{\text{free}}, \forall s_i \in P, \tag{1}$$

that result in the robot moving from the robot's start state s_{s} to a specified goal state s_{g} without intersecting S_{obst},
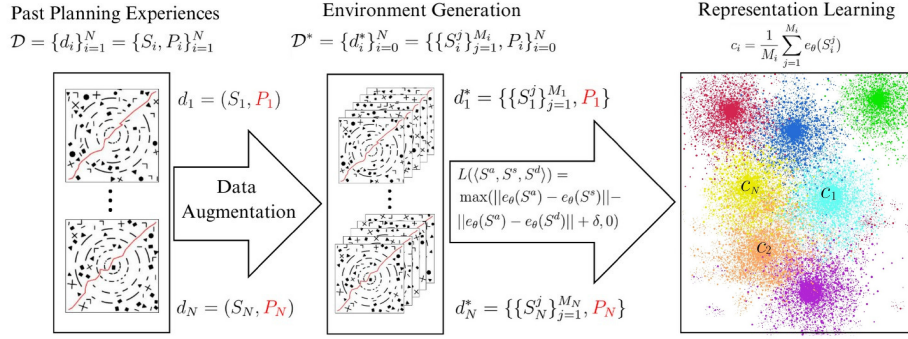
Fig. 2: Motion Memory: Using Past Planning Experiences, Environment Generation, and Representation Learning to Accelerate Future Planning.

while observing robot motion constraints and optimizing a particular cost function (e.g. distance, clearance, energy, and combinations thereof).

A motion planner $f(\cdot)$ that has been used to solve different motion planning problems in the past will have available a dataset $\mathcal{D} = \{d_i\}_{i=1}^N = \{S_i, P_i\}_{i=1}^N$, where N is the amount of currently available S-space (planning problem) and motion plan (planning solution) pairs. Here, we assume that the start and goal states are a constant distance away from each other and remain the same throughout all planning problems, e.g., by aligning them through rotation and translation of the S-space, a realistic assumption in many real-world motion planning problems where dynamics is primarily considered in the local planner, which is guided by local goals a constant distance away from the robot computed by a global planner. For example, s_s can be always expressed as the origin of the S-space, and s_g as $(50, 50)$. The goal of Motion Memory is to utilize the existing $\mathcal{D}$ from past planning experiences to accelerate future planning as shown in Fig. 2.

B. Augmenting Past Experiences with Hallucination

Although $\mathcal{D}$ may contain a variety of planning problem and solution pairs which have the potential to cover a new planning problem faced in the future, the chance of encountering exactly the same problem is still very low, considering the changing real world, sensory noise, and perceptual imperfectness. Therefore, Motion Memory bootstraps on the existing dataset $\mathcal{D}$ and augments it with hallucinated planning problems.

In contrast to the motion planning problem of finding the optimal motion plan P given a S-space S in Eqn. (1), the hallucination problem is defined as its inverse problem [56]–[58], i.e., given a motion plan P, what are the S-spaces S that assure this motion plan is optimal:

$$\{S_i\}_{i=1}^\infty = f^\dagger(P \mid s_s, s_g),$$
$$\text{s.t.} \quad S = S_{\text{free}} \cup S_{\text{obst}}, \quad s_s, s_g, s_i \in S_{\text{free}}, \forall s_i \in P. \quad (2)$$

Notice that $f^\dagger(\cdot)$ is not strictly the inverse function of $f(\cdot)$, because $f(\cdot)$ is non-injective, i.e., different S-spaces can map to the same optimal plan. Therefore, instead of one S-space S, we need to find the set of all S-spaces $\{S_i\}_{i=1}^\infty$, where

the motion plan P is optimal. In most cases, such a set is an infinity set, except for finitely discretized S.

Compared to the forward motion planning problem, which needs to run in real time onboard computation-limited robot platforms, the inverse hallucination problem in Eqn. (2) is much easier to solve. Researchers have proposed different ways to approximate the infinity set $\{S_i\}_{i=1}^\infty$ using representative obstacle states, e.g., the maximal [56], a minimal [57], or a learned distribution of [58] obstacle set in the S-space. With these hallucination techniques, each member of the original pairwise motion planning problem-solution dataset can be augmented from $d_i = \{S_i, P_i\}$ to $d_i^* = \{\{S_i^j\}_{j=1}^{M_i}, P_i\}$, where M_i is the number of total generated planning problems for solution i. Thus, the original dataset $\mathcal{D}$ is augmented to

$$\mathcal{D}^* = \{d_i^*\}_{i=0}^N = \{\{S_i^j\}_{j=1}^{M_i}, P_i\}_{i=0}^N,$$

in which each past planning solution P_i is no longer only paired with one original planning problem S_i, but a set of M_i planning problems $\{S_i^j\}_{j=1}^{M_i}$, for which P_i is optimal. While $P_1, P_2, ..., P_N$ indicate N existing motion plans, P_0 corresponds to an empty set: It is also possible to produce a set of motion planning problems in which no existing motion planning solutions are feasible or optimal, e.g., by adding obstacles to intersect with at least one robot state in every existing motion plan, which will yield $d_0^* = \{\{S_0^j\}_{j=1}^{M_0}, P_0\}$, which is also included in $\mathcal{D}^*$.

C. Representation Learning

The augmented dataset $\mathcal{D}^*$ contains more planning problems for each past planning solution, where the solution is feasible or optimal. Such an augmented dataset is used to learn an efficient latent representation space, which contains critical information regarding the feasibility or optimality of each existing motion plan solution with respect to any motion planning problem. Specifically, we adopt a triplet loss to enforce solution invariance in the learned latent space so that all environments where solution P_i is feasible or optimal stay close to each other in the learned embedding space. We generate triplet training data of anchor, similar, and dissimilar planning problems $\langle S^a, S^s, S^d \rangle$ by sampling S^a and S^s from the same set of planning problems where
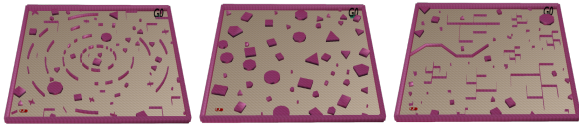
Fig. 3: Different Planning Problem Classes : Curves (Left), Random (Middle), Trap (Right)

solution P_i is feasible or optimal, i.e., $\{S_i^j\}_{j=1}^{M_i}$, and sample S^d from other problem sets and assure P_i is not feasible or optimal for S^d. The planning problem (S-space) encoder $e_\theta(\cdot)$ is then trained to minimize a triplet loss,

$$L(\langle S^a, S^s, S^d \rangle) =$$
$$\max(||e_\theta(S^a) - e_\theta(S^s)|| - ||e_\theta(S^a) - e_\theta(S^d)|| + \delta, 0),$$

with θ as the learnable parameters. The representation space will contain data points from N different latent clusters corresponding to the N existing motion plans $\{P_i\}_{i=0}^N$, including a cluster for S-spaces where none of the N plans are feasible or optimal. The cluster centroids are computed as

$$c_i = \frac{1}{M_i} \sum_{j=1}^{M_i} e_\theta(S_i^j).$$

D. Accelerating Future Planning

Using representation learning, the planning problem encoder is able to project any new motion planning problem S_{N+1} into the latent space, i.e., $l_{N+1} = e_\theta(S_{N+1})$. The existing motion plan associated with the closest cluster centroid P_{i*} is likely the closest to the actual motion planning solution of the problem S_{n+1}:

$$i^* = \underset{i}{\operatorname{argmin}} ||e_\theta(S_{n+1}) - c_i||.$$

For existing motion planners, a motion plan P_{i*} potentially very close to the actual solution can be used to accelerate planning in different ways. For example, for sampling-based motion planners, the states in P_{i*} can be used to bias sampling. For optimization-based approaches, both the actions and states can be used as an initial guess, highlighting the potential of Motion Memory to serve as an effective seed [59], although this specific application is not demonstrated in our experiments. For motion planners treated as a closed-box or an open-box, P_{i*} can be collision checked and be used as the final solution if no collisions are detected, or utilize the closed-box planner to only fix the plan segments which are in collision. In cases where P_0 is the closest in the representation space, the motion planner can start planning from scratch.

IV. EXPERIMENTS

We present extensive experimental results by integrating our Motion Memory technique with three different motion planners in both a closed-box and an open-box manner to solve three different classes of motion-planning problems. We present detailed results on the three planners' performances with or without Motion Memory for each problem

class and for all three classes combined into one. Furthermore, we present evidence demonstrating Motion Memory allows motion planners' planning efficiency to improve with increasing past planning experiences. Finally, we conduct an ablation study to show that the improved planning efficiency is not only due to more past planning experiences, rather the Motion Memory technique itself is an indispensable part.

A. Experimental Setup

Considering Motion Memory is designed to be a universal paradigm that is agnostic to any underlying planner, we integrate Motion Memory with three different motion planners, in both a closed-box and an open-box manner to show its general applicability (denoted as "C" and "O" in all experiment results). In the closed-box integration, we do not assume any access to or knowledge about the underlying planner, and only interface Motion Memory with it using the Motion Memory output as a potential solution after taking a new planning problem as input. The open-box integration assumes access to and knowledge about the underlying planner and depends on which components can be benefited from the Motion Memory output, e.g., sampling distribution or initial optimization guess. We describe the three underlying planners with their closed-box and open-box Motion Memory integration as follows.

1) Baseline Planners: We experiment with different sampling-based motion planners. We first use RRT [17], [18], one of the most popular methods. RRT uses random sampling and nearest neighbors to guide the motion-tree expansion. We also use GUST [24], which was specifically designed for motion-planning problems for vehicles with dynamics. GUST introduces a discrete layer obtained by building a roadmap to guide the motion-tree expansion. The roadmap is then used to induce a partition of the motion-tree into groups based on their nearest roadmap node. During the motion-tree expansion, priority is given to those groups associated with short paths to the goal. Our third planner, which we refer to as Follow [25], further improves GUST by more aggressively following the roadmap paths, and dynamically adjusting the weights based on the progress made during the motion-tree expansion.

a) Closed-box Versions of the Motion Planners: We first use these planners as black boxes in conjunction with the motion-memory framework. Specifically, let $\zeta_1, \ldots, \zeta_k$ be the top k-predictions obtained by the motion-memory framework for a new motion-planning problem. Each of these predictions corresponds to a dynamically-feasible trajectory (retrieved from the database). The black-box version, referred to as $MP_{ClosedBox}$, first checks these trajectories $\zeta_1, \ldots, \zeta_k$ in order for collisions. If some ζ_i is not in collision, then $MP_{ClosedBox}$ returns ζ_i as the solution. Otherwise, $MP_{ClosedBox}$ runs MP and returns the solution (if any) found by MP.

b) Open-box Versions of the Motion Planners: We can better leverage the predictions made by the motion-memory framework by looking inside the motion planners to better leverage the predictions. We refer to these versions as $MP_{OpenBox}$. $MP_{OpenBox}$, as the closed-box version, starts by
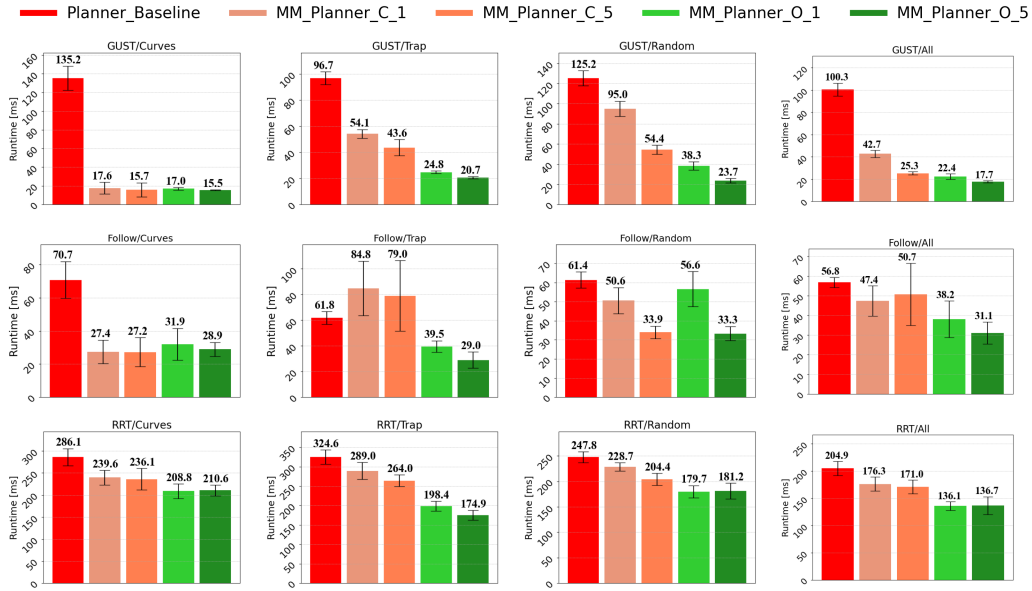
Fig. 4: Different Motion Memory configurations improve three planners in three classes of motion planning problems.

checking if any of the predicted trajectories $\zeta_1, \dots, \zeta_k$ is not in collision. If all of them are in collision, then $\text{MP}_{\text{OpenBox}}$ runs a modified version of MP, as described below.

$\text{RRT}_{\text{OpenBox}}$ is obtained by changing the sampling distribution from which the target is drawn. In the original RRT, the target is sampled with probability b_{goal} (a predetermined probability, usually $b_{\text{goal}} = 0.1$) from an area near the goal, and with probability $1 - b_{\text{goal}}$ from the entire state space. To leverage the predictions, we bias the sampling to be along the predicted trajectories $\zeta_1, \dots, \zeta_k$. In particular, let $b_1, \dots, b_k, b_{\text{goal}}, b_{\text{other}}$ such that $b_1 + \dots + b_k + b_{\text{goal}} + b_{\text{other}} = 1$. Generally, $b_1 > \dots > b_k > b_{\text{goal}} > b_{\text{other}}$. We sample the target with probability b_i along ζ_i, with probability b_{goal} from near the goal, and with probability b_{other} from the entire state space. To generate a target along ζ_i, we first select an intermediate state in ζ_i and then sample around it. In this way, $\text{RRT}_{\text{OpenBox}}$ biases the exploration toward the predicted trajectories $\zeta_1, \dots, \zeta_k$.

$\text{GUST}_{\text{OpenBox}}$ differs from GUST only in the roadmap construction. Specifically, while GUST generates a roadmap node by sampling from the entire space, $\text{GUST}_{\text{OpenBox}}$ biases the sampling along $\zeta_1, \dots, \zeta_k$. Specifically, to generate a roadmap node, $\text{GUST}_{\text{OpenBox}}$ selects ζ_i with probability b_i, selects a state s uniformly at random from ζ_i, and generates a roadmap node by sampling near s. This results in a roadmap that captures the connectivity along the predicted trajectories $\zeta_1, \dots, \zeta_k$. Note that the roadmap is collision free, even if parts of $\zeta_1, \dots, \zeta_k$ are in collision (since samples that result in collision are discarded during the roadmap construction). This biased roadmap construction allows for a more efficient expansion of the motion tree along the predicted trajectories.

$\text{Follow}_{\text{OpenBox}}$ differs from Follow only in the roadmap construction. In fact, $\text{Follow}_{\text{OpenBox}}$ uses the same procedure as $\text{GUST}_{\text{OpenBox}}$ for constructing the roadmap. $\text{Follow}_{\text{OpenBox}}$ then seeks to closely follow the shortest roadmap path from the start to the goal by aggressively expanding the motion tree to follow the nodes in the shortest path in succession.

2) Planning Problem Classes: Fig. 3 shows the three different classes of planning problems used in our experiments, referred to as "Curves," "Random," and "Trap." These problem classes are parametrized, and the obstacles are procedurally generated. Placement (location and orientation) and size of the obstacles are drawn from a probability distribution so that numerous instances can be generated for each problem class. An instance corresponds to a specific placement of the obstacles, as shown in Fig. 3. For example, for the "Curves" problem class we can vary the number of curves, number of segments per curve, separation among segments, and so on. For the "Random" problem class, we can vary the density of the obstacles as well as their shape and placement. For the "Trap" problem class we can perturb the placement of the major obstacles and also of the random obstacles spread throughout the environment.

3) Motion Memory Implementation: For each of the three motion planning problem classes, we assume our Motion Memory has access to 100 planning problems and their corresponding motion planning solutions as its past planning experiences, i.e., $\mathcal{D} = \{d_i\}_{i=1}^{100} = \{S_i, P_i\}_{i=1}^{100}$. We augment each data point in $\mathcal{D}$ with 999 more planning problems and generate an augmented dataset $\mathcal{D}^* = \{d_i^*\}_{i=1}^{100} = \{\{S_i^j\}_{j=1}^{1000}, P_i\}_{i=1}^{100}$, a dataset of 100,000 problems per class with 100 solutions, by slightly rearranging obstacles close to the motion plan and randomly shuffling obstacles in other places. For simplicity, we omit the set of problems where no existing motion plan is a solution, i.e., d_0^*. To train the planning problem encoder $e_\theta(\cdot)$, we use a Convolutional Neural Network which takes motion planning problems as input and outputs a 30-dimensional latent space. Specifically, the input to our CNN is a discretized version of the workspace, represented as a 2D grid where each cell encodes

the presence or absence of an obstacle, and the network architecture comprises sequential layers of convolution and pooling, and the final feature map is processed through fully connected layers to produce the 30-dimensional embedding. For testing, we generate another 10,000 unseen planning problems per class for the motion planners to solve.

We present experiments comparing closed-box and open-box integrations with both top-one and top-five predictions for new problems. While top-five predictions provide more information, they also incur additional computational costs due to the need to process multiple predicted solutions.

B. Improvement for Different Planners and Problems

In Fig. 4, we show individual planning performance of the three underlying planners (GUST, Follow, and RRT in row 1, 2, and 3 respectively) in the three classes of motion planning problems (Curves, Random, and Trap in column 1, 2, and 3 respectively). The fourth column is the results of a large Motion Memory which does not distinguish among different motion planning problem classes, with 300,000 training and 30,000 testing data points. In each figure, we show the average planning time of default baseline planner without Motion Memory (`Planner_Baseline`), the baseline planner assisted by Motion Memory in a closed-box manner with top-one or top-five prediction(s) (`MM_Planner_C_1` and `MM_Planner_C_5`), and also assisted in an open-box manner (`MM_Planner_O_1` and `MM_Planner_O_5`).

The results show that Motion Memory can significantly reduce planning time compared to all three default baseline planners, with the only exception of using Motion Memory in a closed-box manner with Follow in the Trap environment. In each of the 12 sub-figures in Fig. 4, we can observe increasing performance from left to right. Using Motion Memory in a open-box manner achieves similar or more improvement compared to using it in a closed-box manner. In most cases, using the top-five Motion Memory predictions outperforms using the top-one prediction, despite the potentially more computation to process four extra predictions. Comparing the first three columns, Motion Memory can achieve the most significant improvement for Curves environments with GUST and Follow, while RRT enjoys the most Motion Memory benefits in the Trap environments. In the last column where Motion Memory does not distinguish among the three classes of planning problems, it also outperforms all three baseline planners. Comparing the three rows, Motion Memory is the most helpful for GUST, while the difference made by Motion Memory to RRT is relatively smaller.

C. Continual Improvement with Increasing Experience

We also study the performance improvement of Motion Memory with respect to increasing past planning experiences. In Fig. 5, we show how the runtime of the $GUST_{ClosedBox}$ planner to solve planning problems in Curves environments changes when having access to more available motion plans. For each bar in Fig. 5, we go through the entire Motion Memory pipeline, including environment generation
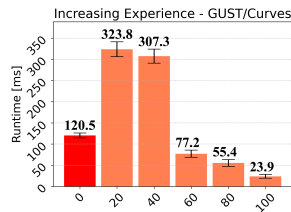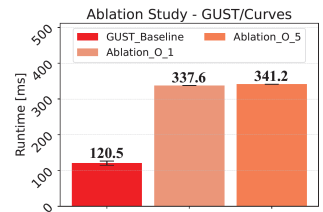

Fig. 5: Improvement with Increasing Experiences.

Fig. 6: Ablation Study: Random Path Selection.

and representation learning, with a limited number of available motion plans (20, 40, 60, 80, or 100). By integrating the Motion Memory model produced by the corresponding amount of past experiences with $GUST_{ClosedBox}$, we test the planner performance on the 10,000 unseen test problems. With only 20 or 40 available past motion plans, Motion Memory underperforms the baseline GUST, because a limited set of available motion plans is not able to sufficiently cover the variety of new planning problems. With increasing experiences, we see a significant reduction in runtime and improvement in planning efficiency. Such experiment results confirm our hypothesis that a motion planner can continually improve when having access to increasing amount of past planning experiences with the assistance of Motion Memory.

D. Ablation Study

Finally, to demonstrate the necessity of the Motion Memory technique, in addition to the access to a large amount of past planning experiences, we conduct an ablation study, in which we do not use the environment generation and representation learning in Motion Memory to decide which past plan may be most helpful for a future planning problem, but randomly pick a past plan to assist the motion planner. Fig. 6 shows that randomly picking one or five past plans to assist GUST in Curves environments in a open-box manner will increase the planning time. Such results indicate it is necessary to use Motion Memory to decide which past planning experience is useful to accelerate future planning.

V. DISCUSSION

We present Motion Memory, a universal technique which is agnostic to different underlying motion planners but helps them to accelerate their future planning runtime with past planning experiences. By augmenting past experiences and using representation learning, Motion Memory avoids unnecessary and repetitive replanning from scratch when facing similar future planning problems. We demonstrate the efficacy of Motion Memory by integrating it with different planners in a closed-box and open-box fashion, and solving different classes of motion-planning problems more efficiently. One possible direction for future research is to extend Motion Memory for manipulation planning, where the robot interacts with the objects in the environment. Another direction is to consider a heterogeneous team of robots and how Motion Memory can facilitate planning for different types of robot, possibly adpating plans from one robot type to another.

REFERENCES

[1] J. Canny, *The complexity of robot motion planning*. MIT press, 1988.

[2] Y. Kuwata, J. Teo, G. Fiore, S. Karaman, E. Frazzoli, and J. P. How, "Real-time motion planning with applications to autonomous urban driving," *IEEE Transactions on control systems technology*, vol. 17, no. 5, pp. 1105–1118, 2009.

[3] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics & Automation Magazine*, vol. 4, no. 1, pp. 23–33, 1997.

[4] S. Murray, W. Floyd-Jones, Y. Qi, D. J. Sorin, and G. D. Konidaris, "Robot motion planning on a chip." in *Robotics: Science and Systems*, vol. 6, 2016.

[5] R. Volpe, "Rover functional autonomy development for the mars mobile science laboratory," in *Proceedings of the 2003 IEEE Aerospace Conference*, vol. 2, 2003, pp. 643–652.

[6] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[7] D. Hsu, J.-C. Latombe, and R. Motwani, "Path planning in expansive configuration spaces," in *Proceedings of international conference on robotics and automation*, vol. 3. IEEE, 1997, pp. 2719–2726.

[8] S. LaValle, "Rapidly-exploring random trees: A new tool for path planning," *Research Report 9811*, 1998.

[9] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The international journal of robotics research*, vol. 30, no. 7, pp. 846–894, 2011.

[10] X. Xiao, B. Liu, G. Warnell, and P. Stone, "Motion planning and control for mobile robot navigation using machine learning: a survey," *Autonomous Robots*, vol. 46, no. 5, pp. 569–597, 2022.

[11] H. Choset, K. M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun, *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT Press, 2005.

[12] S. M. LaValle, *Planning Algorithms*. Cambridge, MA: Cambridge University Press, 2006.

[13] X. Xiao, J. Biswas, and P. Stone, "Learning inverse kinodynamics for accurate high-speed off-road navigation on unstructured terrain," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 6054–6060, 2021.

[14] H. Karnan, K. S. Sikand, P. Atreya, S. Rabiee, X. Xiao, G. Warnell, P. Stone, and J. Biswas, "Vi-ikd: High-speed accurate off-road navigation using learned visual-inertial inverse kinodynamics," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 3294–3301.

[15] P. Atreya, H. Karnan, K. S. Sikand, X. Xiao, S. Rabiee, and J. Biswas, "High-speed accurate robot control using learned forward kinodynamics and non-linear least squares optimization," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 11 789–11 795.

[16] A. Datar, C. Pan, and X. Xiao, "Learning to model and plan for wheeled mobility on vertically challenging terrain," *arXiv preprint arXiv:2306.11611*, 2023.

[17] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," *International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, 2001.

[18] S. M. LaValle, "Motion planning: The essentials," *IEEE Robotics & Automation Magazine*, vol. 18, no. 1, pp. 79–89, 2011.

[19] W. Xinyu, L. Xiaojuan, G. Yong, S. Jiadong, and W. Rui, "Bidirectional potential guided RRT* for motion planning," *IEEE Access*, vol. 7, pp. 95 046–95 057, 2019.

[20] A. Shkolnik, M. Walter, and R. Tedrake, "Reachability-guided sampling for planning under differential constraints," in *IEEE International Conference on Robotics and Automation*, 2009, pp. 2859–2865.

[21] D. Devaurs, T. Simeon, and J. Cortés, "Enhancing the transition-based RRT to deal with complex cost spaces," in *IEEE International Conference on Robotics and Automation*, 2013, pp. 4120–4125.

[22] D. Hsu, J.-C. Latombe, and R. Motwani, "Path planning in expansive configuration spaces," in *IEEE International Conference on Robotics and Automation*, vol. 3, 1997, pp. 2719–2726.

[23] I. A. Şucan and L. E. Kavraki, "A sampling-based tree planner for systems with complex dynamics," *IEEE Transactions on Robotics*, vol. 28, no. 1, pp. 116–131, 2012.

[24] E. Plaku, "Region-guided and sampling-based tree search for motion planning with dynamics," *IEEE Transactions on Robotics*, vol. 31, pp. 723–735, 2015.

[25] E. Plaku, E. Plaku, and P. Simari, "Clearance-driven motion planning for mobile robots with differential constraints," *Robotica*, vol. 36, pp. 971–993, 2018.

[26] T. McMahon, A. Sivaramakrishnan, E. Granados, K. E. Bekris *et al.*, "A survey on the integration of machine learning with sampling-based motion planning," *Foundations and Trends® in Robotics*, vol. 9, no. 4, pp. 266–327, 2022.

[27] G. S. Aoude, B. D. Luders, J. M. Joseph, N. Roy, and J. P. How, "Probabilistically safe motion planning to avoid dynamic obstacles with uncertain motion patterns," *Autonomous Robots*, vol. 35, pp. 51–76, 2013.

[28] B. Burns and O. Brock, "Sampling-based motion planning using predictive models," in *Proceedings of the 2005 IEEE international conference on robotics and automation*. IEEE, 2005, pp. 3120–3125.

[29] Z. Kingston, M. Moll, and L. E. Kavraki, "Exploring implicit spaces for constrained sampling-based planning," *The International Journal of Robotics Research*, vol. 38, no. 10-11, pp. 1151–1178, 2019.

[30] G. Sutanto, I. R. Fernández, P. Englert, R. K. Ramachandran, and G. Sukhatme, "Learning equality constraints for motion planning on manifolds," in *Conference on Robot Learning*. PMLR, 2021, pp. 2292–2305.

[31] I. Baldwin and P. Newman, "Non-parametric learning for natural plan generation," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2010, pp. 4311–4317.

[32] M. Zucker, J. Kuffner, and J. A. Bagnell, "Adaptive workspace biasing for sampling-based planners," in *2008 IEEE International Conference on Robotics and Automation*. IEEE, 2008, pp. 3757–3762.

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

[34] K. Hauser, "Lazy collision checking in asymptotically-optimal motion planning," in *2015 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2015, pp. 2951–2957.

[35] A. Mandalika, S. Choudhury, O. Salzman, and S. Srinivasa, "Generalized lazy search for robot motion planning: Interleaving search and edge evaluation via event-based toggles," in *Proceedings of the International Conference on Automated Planning and Scheduling*, vol. 29, 2019, pp. 745–753.

[36] J. Bialkowski, M. Otte, S. Karaman, and E. Frazzoli, "Efficient collision checking in sampling-based motion planning via safety certificates," *The International Journal of Robotics Research*, vol. 35, no. 7, pp. 767–796, 2016.

[37] J. Huh and D. D. Lee, "Learning high-dimensional mixture models for fast collision detection in rapidly-exploring random trees," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 63–69.

[38] N. Das and M. Yip, "Learning-based proxy collision detection for robot motion planning applications," *IEEE Transactions on Robotics*, vol. 36, no. 4, pp. 1096–1114, 2020.

[39] C. Yu and S. Gao, "Reducing collision checking for sampling-based motion planning using graph neural networks," *Advances in Neural Information Processing Systems*, vol. 34, pp. 4274–4289, 2021.

[40] J. Pan, S. Chitta, and D. Manocha, "Faster sample-based motion planning using instance-based learning," in *Algorithmic Foundations of Robotics X: Proceedings of the Tenth Workshop on the Algorithmic Foundations of Robotics*. Springer, 2013, pp. 381–396.

[41] M. Bhardwaj, S. Choudhury, B. Boots, and S. Srinivasa, "Leveraging experience in lazy search," *Autonomous Robots*, vol. 45, pp. 979–996, 2021.

[42] B. Hou, S. Choudhury, G. Lee, A. Mandalika, and S. S. Srinivasa, "Posterior sampling for anytime motion planning on graphs with expensive-to-evaluate edges," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 4266–4272.

[43] D. Berenson, P. Abbeel, and K. Goldberg, "A robot path planning framework that learns from experience," in *2012 IEEE International Conference on Robotics and Automation*. IEEE, 2012, pp. 3671–3678.

[44] È. Pairet, C. Chamzas, Y. Petillot, and L. E. Kavraki, "Path planning for manipulation using experience-driven random trees," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3295–3302, 2021.

[45] D. Coleman, I. A. Şucan, M. Moll, K. Okada, and N. Correll, "Experience-based planning with sparse roadmap spanners," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 900–905.

[46] C. Chamzas, A. Shrivastava, and L. E. Kavraki, "Using local experi-

ences for global motion planning," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 8606–8612.

[47] S. Finney, L. P. Kaelbling, and T. Lozano-Pérez, "Predicting partial paths from planning problem parameters." in *Robotics: Science and Systems*. Citeseer, 2007.

[48] A. H. Qureshi, A. Simeonov, M. J. Bency, and M. C. Yip, "Motion planning networks," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 2118–2124.

[49] J. J. Johnson, L. Li, F. Liu, A. H. Qureshi, and M. C. Yip, "Dynamically constrained motion planning networks for non-holonomic robots," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 6937–6943.

[50] R. Strudel, R. G. Pinel, J. Carpentier, J.-P. Laumond, I. Laptev, and C. Schmid, "Learning obstacle representations for neural motion planning," in *Conference on Robot Learning*. PMLR, 2021, pp. 355–364.

[51] A. H. Qureshi, J. Dong, A. Choe, and M. C. Yip, "Neural manipulation planning on constraint manifolds," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6089–6096, 2020.

[52] C. Chamzas, Z. Kingston, C. Quintero-Peña, A. Shrivastava, and L. E. Kavraki, "Learning sampling distributions using local 3d workspace decompositions for motion planning in high dimensions," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 1283–1289.

[53] J.-M. Lien and Y. Lu, "Planning motion in environments with similar obstacles." in *Robotics: Science and systems*, 2009.

[54] C. Chamzas, A. Cullen, A. Shrivastava, and L. E. Kavraki, "Learning to retrieve relevant experiences for motion planning," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 7233–7240.

[55] J.-C. Latombe, *Robot motion planning*. Springer Science & Business Media, 2012, vol. 124.

[56] X. Xiao, B. Liu, G. Warnell, and P. Stone, "Toward agile maneuvers in highly constrained spaces: Learning from hallucination," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1503–1510, 2021.

[57] X. Xiao, B. Liu, and P. Stone, "Agile robot navigation through hallucinated learning and sober deployment," in *2021 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2021, pp. 7316–7322.

[58] Z. Wang, X. Xiao, A. J. Nettekoven, K. Umasankar, A. Singh, S. Bommakanti, U. Topcu, and P. Stone, "From agile ground to aerial navigation: Learning from learned hallucination," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 148–153.

[59] T. S. Lembono, A. Paolillo, E. Pignat, and S. Calinon, "Memory of motion for warm-starting trajectory optimization," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2594–2601, 2020.