Planning with Learned Subgoals Selected by Temporal Information

Xi Huang[1], Gergely Sóti[1,2], Christoph Ledermann[1], Björn Hein[1,2], and Torsten Kröger[1]

Abstract—Path planning in a changing environment is a challenging task in robotics, as moving objects impose time-dependent constraints. Recent planning methods primarily focus on the spatial aspects, lacking the capability to directly incorporate time constraints. In this paper, we propose a method that leverages a generative model to decompose a complex planning problem into small manageable ones by incrementally generating subgoals given the current planning context. Then, we take into account the temporal information and use learned time estimators based on different statistic distributions to examine and select the generated subgoal candidates. Experiments show that planning from the current robot state to the selected subgoal can satisfy the given time-dependent constraints while being goal-oriented.

I. INTRODUCTION

Modern robotic applications aim to place the robots into a collaborative environment rather than in a confined workstation, e.g. encapsulated by a safety fence. Such collaborative workspaces are open, allowing humans, robots, and objects to enter them freely. From the perspective of a robot, task-unrelated agents entering its workspace are obstacles that need to be detected and avoided while executing its own task. This avoidance problem is an extension of the classical collision avoidance for static environments, where part of the workspace is blocked by non-moving obstacles. Previously, being blocked or not was a static spatial property, based on the spatial occupation and position of the obstacles. Now, in a collaborative environment, this property becomes time-dependent, highly increasing the complexity of the problem.

In this work, we propose to simplify the problem posed by changing environments with the assumption that a changing environment is static over a small period of time, where the changes are so small that they do not affect path planning. Hence, an overall path can be planned and executed step by step, always assuming that the current environment state is static. For static environments, a variety of sampling-based planning approaches have been proven to be efficient. Among recent methods, some try to estimate meaningful spatial distributions for sampling [1]–[6], some propose to approximate the cost-to-go function [7], while some neural planners imitate the behavior of the optimal planners and plan a path [8] or output an action directly [9]. Despite the diversity of these methods, all these methods rely on the spatial domain alone to perform their computations. However, our approach has an assumption that the environment is fixed

The authors are with [1]Institute for Anthropomatics and Robotics, Karlsruhe Institute of Technology, 76131 Karlsruhe, Germany, and [2]Robotics and Autonomous Systems, Institute of Applied Research, Karlsruhe University of Applied Sciences, 76133 Karlsruhe, Germany
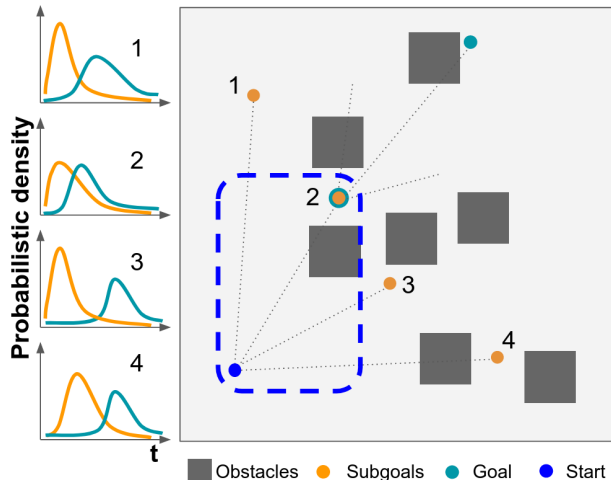x.huang@kit.edu

Fig. 1: Planning with learned subgoals: with the estimated probabilistic distribution of planning time from the start to subgoal candidates (orange), and from the goal to subgoal candidates (turquoise), candidate 2 is selected. Planning ranges are adapted to the blue dashed bounding box after the selection for planning efficiency.

for only a small time window, imposing a time constraint for the planning. None of the approaches above allow us to incorporate this time constraints directly.

Therefore, in our work, we aim to integrate temporal information into learning to make the algorithm aware of our time budget while solving a planning problem. In this context, temporal information refers to the planning time used to plan from the current robot state to intermediate subgoals. Together with this temporal information, the recent methods, which only learn from the spatial information, can be extended and applied to scenarios where there are constraints regarding the planning time, such as changing environments.

Combining the spatial and temporal information, we propose Planning with Learned Subgoals (PLS), which outputs subgoals to incrementally lead the robot to the final goal. Unlike the methods that use a learned spatial distribution to bias the sampling process, the subgoals offered by our approach are more like milestones. These milestones decompose a complex planning problem into small and easily solvable pieces. For small and easy problems, we can further tailor the planning range, in which we sample and plan, to achieve better performance. As illustrated in Fig. 1, given a planning problem with obstacles, we generate a batch of candidates for subgoals, marked in orange. Then, based on the estimation

of the planning time from the start to the candidates and from candidates to the goal, we select the most suitable candidate with index 2 as a subgoal according to the metrics introduced in III-D. After the selection, the range of the planning problem presented in III-E shrinks to the blue dashed box for better planning performance. Experiments show that the integration of the temporal information makes the planning algorithm significantly more effective and can be applied to reactive planning scenarios where the planner has limited time to find a solution. Our contributions are mainly:

- Leveraging a generative model to predict spatial subgoals that decompose a complex planning problem.
- Using learned time estimators to capture the temporal information regarding the planning time given the planning problem.
- Designing two metrics based on the time estimation to select suitable and goal-directed subgoal candidates.
- Conducting an ablation study to compare the performance based on different assumptions of the distribution for time estimators.

II. RELATED WORK

Recent approaches combining machine learning and sampling-based motion planning fall roughly into two categories, i.e., learning the heuristics or distributions and neural planners. We review them in the following and highlight the differences between these methods.

A. Learning heuristics

Heuristics are crucial for speeding up the planning process in the sampling-based methods. They serve as an oracle to guide the search or sampling [10]–[13]. Recent approaches attempt to learn various kinds of heuristics to achieve faster planning. Planning delay is featured as a heuristic in [14] to keep the planner away from the local minimum, which usually has a high delay. Uncertainty is used as a heuristic to guide the planner to explore in [15] by updating probabilistic models using maximum posterior every time after gaining new information. Given a planning problem, [7] learns the cost function as heuristics and uses the gradient of the cost function to find a path between the start and the goal.

B. Learning distributions

While learning heuristics aims to estimate a value given a sample or region and use this value to prioritize the planning or searching, learning distribution directly predicts the regions likely to solve the problem. Using a conditional variational autoencoder (CVAE) to capture latent representations, [5] learns a sampling distribution of the optimal path given the planning requests. [6] applies the connectivity in graph theory to identify the critical configurations in a roadmap. These critical configurations improve the capability of a roadmap to solve narrow passage problems. LEGO [16] proposes a method to collect the bottleneck configurations. Then, they leverage a CVAE as a generative model to generate bottleneck configurations during the inference.
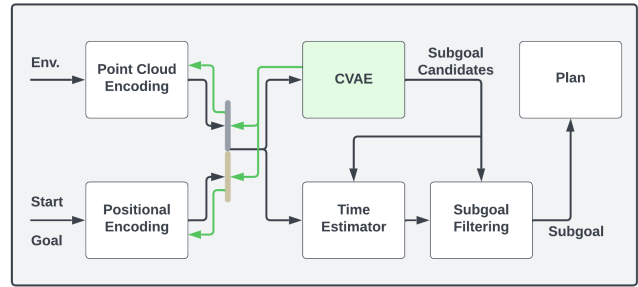

Fig. 2: The PLS architecture. The green arrows show the direction of gradient descent. During training, the backpropagation of the time estimator does not reach the encoding blocks.

These bottleneck configurations serve the same purpose as in [6]. The methods above learn some predefined features and metrics using a supervised manner. The authors of [2] notice the gap between sampling and the downstream planner performance. They propose a generator-critic architecture, similar to actor-critic in reinforcement learning. The critic serves as a proxy to quantify whether the samples contribute to finding a solution. This links the sampling distribution directly to the planning performance, e.g., planning time and path quality. Learning distributions from experience can also be found in [1] [17]. While the distribution is explicitly learned and deployed during the sampling phase, [3] learns an agent to reject implausible samples. Implicitly, the agent learned the knowledge of the plausible region. Rather than learning the distribution from collected data, [18] applies the Stein variational inference to shifting samples toward the free space by optimizing a posteriori.

C. Neural planners

The methods above learn to predict the distribution of the sampling or heuristics for planning and eventually use a traditional planner to find a solution. Different from that, neural planners try to mimic the behavior of traditional planners and learn directly to plan. Motion planning network [8] imitates the behavior of the RRT* planner and outputs a complete path by planning in a bi-directional fashion. It utilizes offline learning and active learning to reinforce the planning performance. Motion policy networks [9] mimic the behavior of diverse planners in both task and configuration space, including AIT* [12] and Geometric Fabrics [19]. Instead of learning to capture the whole path, it learns to generate the delta, i.e. the joint velocity given the current planning problem.

Our approach goes beyond existing methods that focus solely on spatial information by incorporating temporal data into the planning process. We train models to predict the time required for planning using a normal or log-normal distribution. These predictions help us identify the most useful samples produced by the generative model, ensuring that the planning process stays within a specific time frame while still goal-oriented. As the planning time is strongly

correlative to the number of collision checks, estimating the planning time is a new and closer way to link the samples to the planning performance.

III. METHOD

Given a planning problem, including the start and goal joint configuration, and the information of the surrounding environment, the objective of this work is to generate joint configurations of a robot manipulator as subgoals that not only lead the robot towards the final goal configuration but also can be planned within a given time constraints. We address this problem by using a generative model to output a batch of candidates for an intermediate subgoal instead of planning a complete path. Then, we employ the temporal estimation regarding the planning time in two different aspects to select the proper one as the subgoal. Fig. 2 shows the architecture of PLS.

A. Dataset

To integrate both temporal and spatial factors, we have assembled a dataset containing over one million planning requests using OMPL [20]. These requests were planned using a UR10e robot in various environments with at least one feasible solution. Each request provides a unique, collision-free starting and ending configuration that lies within $[-2\pi, 2\pi]$ for all joints. Each data entry in our dataset contains information about the environment, the start and final configurations, the intermediate waypoints between them, and the time needed for planning from the starting point to each waypoint. Multiple runs were conducted to obtain a comprehensive distribution of planning times.

For the generation of spatial waypoints connecting the start and goal configurations, we employ the AIT* optimal planner. Subsequently, we compute the time required to traverse from the start configuration to each waypoint. We choose RRTConnect [21] over AIT* for the computation of planning times. This choice is informed by the characteristics of AIT* as an optimal planner that continually refines its solutions until the given time budget is over. Our observations indicate that AIT* generally takes a bit more time to return an initial solution compared to RRTConnect, which is consistent with the results reported in the original AIT* paper [12].

B. Generating Spatial Subgoals

Planning to subsequent subgoals offers several benefits compared to directly aiming for the global goal. First, the planner typically needs less planning time to return a solution. This efficiency arises from the reduced distance between the starting point and the subgoal. Empirical evidence suggests that planning time tends to grow exponentially with the increase in distance. The subgoals decompose complex planning problems into pieces and are usually in closer proximity to the current robot state. While the environment changes, paths leading to subgoals in closer proximity are preferable. This is because paths to global goals, which are usually longer, are more vulnerable to becoming invalid due to these changes. An additional advantage is that we can
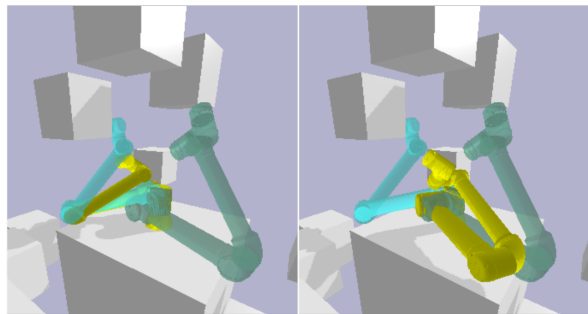

Fig. 3: An example of the dataset. The start and goal configurations are marked in cyan and green, respectively. The subgoals, namely ground truths are shown in yellow.

accordingly shape and narrow the planning range based on the subgoals, thereby minimizing the search effort. This is especially beneficial in spaces with high degrees of freedom. Note that these are advantages only under that condition that the subgoals are plausible and goal-oriented, meaning they can progressively guide the robot toward its final goal.

Previous works [4], [5] showcase that a conditional variational autoencoder (CVAE) [22] is capable of capturing the environment representation and generating samples that are similar to the dataset, e.g. generating samples along the optimal path or in the bottleneck region. Similar to them, we employ a CVAE to generate plausible subgoals given the planning problem as conditions. Different from them, we take temporal information into account and train our CVAE in such a way that the model tends to generate subgoals that can meet the time constraints. To achieve this goal, during the training, we only use the waypoints in the dataset mentioned above with a constraint for maximal planning time of 0.05s. Theoretically, this constraint for planning time should be small to enable reactive planning in changing environments. Practically, however, the smaller the range is, the fewer generated data can fit into it and thus training a model properly becomes harder.

Given a planning problem $\mathbf{c} \in \mathcal{C}$, including the start and goal configuration and the surrounding environment, we aim to generate the subgoals that can lead the robot to reach the goal. The CVAE model samples $\mathbf{z} \in \mathbb{R}^m$ from a m-dimensional i.i.d. latent space and produces n-dimensional joint configurations $\mathbf{x} \in \mathbb{R}^n$. The objective is to maximize the probability that the model generates the data that are likely to be in the dataset $\mathcal{X}_{sg}$ over the whole $\mathbf{z}$-space

$$p(\mathbf{x}|\mathbf{c}) = \int p(\mathbf{x}|\mathbf{z}, \mathbf{c})p(\mathbf{z})\mathrm{d}\mathbf{z}. \tag{1}$$

As we aim to generate data that are similar to the dataset $\mathcal{X}_{sg}$, the distribution of the latent variables for generating the synthetic data $p(\mathbf{z}|\mathbf{x}, \mathbf{c})$ should be similar to the distribution $q(\mathbf{z}|\mathbf{x}, \mathbf{c})$ represented by the dataset. The Kullback-Leibler (KL) divergence between these two distributions is

$$\mathcal{D}[q(\mathbf{z}|\mathbf{x}, \mathbf{c})||p(\mathbf{z}|\mathbf{x}, \mathbf{c})] = \mathbb{E}_{\mathbf{z} \sim q}[\log q(\mathbf{z}|\mathbf{x}, \mathbf{c}) - \log p(\mathbf{z}|\mathbf{x}, \mathbf{c})]. \tag{2}$$

In the following, we use $\mathcal{D}[\cdot]$ to denote the KL divergence. The evidence lower bound (ELBO) of $\log p(\mathbf{x}|\mathbf{c})$ can be written as:

$$\log p(\mathbf{x}|\mathbf{c}) \geq \mathbb{E}_{\mathbf{z}\sim q}[\log p(\mathbf{x}|\mathbf{z},\mathbf{c})] - \mathcal{D}[q(\mathbf{z}|\mathbf{x},\mathbf{c})||p(\mathbf{z}|\mathbf{x},\mathbf{c})]. \quad (3)$$

While we maximize the evidence lower bound, the $\log p(\mathbf{x}|\mathbf{c})$ will increase accordingly. We suggest the reader refer to [23] for further information regarding CVAE. The first term of the right-hand side characterizes the likelihood of a data point $\mathbf{x}$ given the $\mathbf{z}$ and $\mathbf{c}$. Given the assumption that $p(\mathbf{x}|\mathbf{z},\mathbf{c}) \sim \mathcal{N}(f(\mathbf{z},\mathbf{c}),\boldsymbol{\Sigma})$, where $\boldsymbol{\Sigma}$ denotes diagonal covariant matrices and $f(\mathbf{z},\mathbf{c})$ is the generated sample, we have

$$\log p(\mathbf{x}|\mathbf{z},\mathbf{c}) \propto -||\mathbf{x} - f(\mathbf{z},\mathbf{c})||_2^2 \quad (4)$$

The assumption $p(\mathbf{x}|\mathbf{z},\mathbf{c}) \sim \mathcal{N}(f(z,\mathbf{c}),\boldsymbol{\Sigma})$ has been used for various tasks and we consider it valid within the scope of this paper. We modify Eq. 4 to be

$$\log p(\mathbf{x}|\mathbf{z},\mathbf{c}) \propto -||g(\mathbf{x}) - g(f(\mathbf{z},\mathbf{c}))||_2^2, \quad (5)$$

with a non-linear function capturing both the forward kinematics $g_{FK}(\cdot)$ and joint positions $g_{joint}(\cdot)$ of a robot using a weighting factor α

$$g(\mathbf{x}) = \alpha g_{FK}(\mathbf{x}) + (1-\alpha)g_{joint}(\mathbf{x}). \quad (6)$$

Our model directly outputs joint configurations. However, predicting wrong values at the first joint and at the end-effector can generally lead to significantly different effects. The changes at the first joint usually will result in a larger placement of the robot. Therefore, the g_{FK} is a weighting function that reflects the kinematics of the robot. However, for a robot with a range of $[-2\pi, 2\pi]$ at every joint, the forward kinematics can be the same although the joint configurations are different. To address this problem, we introduce the positional encoding with l levels

$$g_{joint}(\mathbf{x}) = [\mathbf{x}, \cos\mathbf{x}, \sin\mathbf{x}, ..., \cos(2^l\mathbf{x}), \sin(2^l\mathbf{x})]. \quad (7)$$

Positional encoding is used in recent methods like Neural Radiance Field (NeRF) [24] to capture the high-frequency part of the data. In our case, positional encoding provides the model with more information to understand the planning problem. By assuming the prior of the latent space to be $p(\mathbf{z}|\mathbf{x},\mathbf{c}) \sim \mathcal{N}(\mathbf{0},\mathbf{I})$, the final loss of the CVAE becomes

$$\mathcal{J} = ||g(\mathbf{x}) - g(f(\mathbf{z},\mathbf{c}))||_2^2 + \beta\mathcal{D}[q(\mathbf{z}|\mathbf{x},\mathbf{c})||\mathcal{N}(\mathbf{0},\mathbf{I})], \quad (8)$$

with a weighting factor β [25]. Two neural networks are used to capture approximate the function $q(\mathbf{z}|\mathbf{x},\mathbf{c})$ and $f(\mathbf{z},\mathbf{c})$, respectively. Using the reparametrization trick [26], we apply stochastic gradient descent to train the networks.

C. Learning Temporal Distributions

While using a generative model to learn subgoals, it is desired that the model can provide diverse and multi-modal subgoals, which can solve the problem differently. On the other hand, diversity can lead to a problem because the planning time toward the subgoal may not fit the time constraints. To address this problem, we employ a time estimator
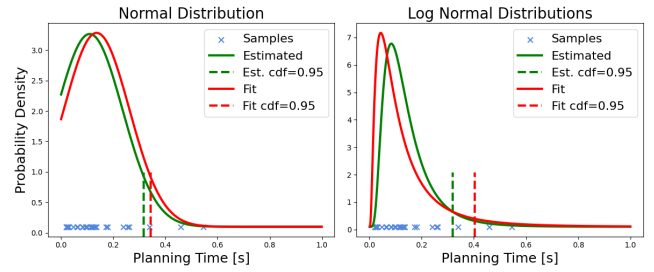

Fig. 4: Samples of planning time described by normal and log-normal distributions. Red: statically determined. Green: estimated by a model.

to select the proper subgoals regarding the constraints. The time estimator takes the environment, the start configuration, and the subgoal as input and outputs a distribution of the planning time. This temporal distribution serves as a proxy of the planning effort a planner needed for the problem and it is strongly correlative to the number of collision checks.

After analyzing the data, we consider normal and log-normal distribution. Log-normal distribution describes the data whose logarithm is normally distributed. In other words, given the normally distributed data X, log-normal distribution describes the data $Y = e^X$. The data from a log-normal distribution is always positive and skewed. As shown in Fig 4, the data has a long tail in the positive direction.

As mentioned before, the objective of learning temporal distribution is to verify if the predicted subgoal can be planned within the time constraints. Intuitively, the time distribution depends on the complexity of a planning problem, which is defined by the start-goal query and the environment. Therefore, we use the same encoded latent representations in III-B as a starting point to learn the time distribution. A multi-layer perceptron (MLP) takes this encoded representation as input and outputs the parameters $\hat{\theta}$ of a 1-D distribution. To match the planning time distribution for each planning instance i in the dataset depicted III-A, we form the loss function based on negative log-likelihood

$$J_i(T_i, \hat{\theta}_i) = -\frac{1}{N}\sum_{j=0}^{N}\mathcal{L}(t_{i,j}|\hat{\theta}) + w||\theta_i - \hat{\theta}_i||_2^2, \quad (9)$$

where $T_i \in \mathbb{R}$ is a random variable and $t_{i,j} \in T_i$ is a planning time collected by the dataset. In addition to the negative log-likelihood, we add a mean square error (MSE) loss between the parameter θ empirically derived from the data t_j and the output of the time estimator $\hat{\theta}$, weighted by w. As shown in Fig. 2, the gradient of the time estimator does not backpropagate to the encoding blocks. We employ this time estimator to inform the selection of subgoals, described in the following in III-D.

D. Subgoal Selection based on Temporal Estimation

We designed two selection metrics based on the temporal estimation to select the subgoals that are likely to be planned within the time constraints, and that are goal-oriented, respectively. Selecting goal-oriented subgoals is important,

because, in an extreme case, any model that copies the start configuration from the condition and uses it as output, would meet the time constraints.

1) Start-to-sample metric: The start-to-sample metric estimates how likely the planning time from the start and generated subgoal candidates will exceed the user-defined time constraints. With the estimated parameters $\hat{\theta}$, we can analytically determine the cumulative density function (CDF) $c(\cdot)$ of the estimated distribution. As shown in Fig. 4, the vertical dashed lines indicate the position t_{95} where the $c(t_{95}) = 0.95$, meaning that 95% of the values drawn from this distribution are supposed to be smaller than t_{95}. We see t_{95} as a threshold to classify if the samples can meet the time constraints. Depending on the needs, one can set this confidence level to a higher value.

2) Goal-to-sample metric: While the start-to-sample metric classifies if the samples can be planned within the time constraint, it does not quantify how goal-oriented the samples are. In search algorithms, the cost-to-go heuristics depict the estimated cost of reaching the goal and are used to prioritize the exploration. These heuristics usually are spatial distances such as the Manhattan distance. Similar to the cost to go, we can estimate the time distribution from the goal to the subgoal candidates using the same time estimator model. We further use the t_{95} to describe how are these candidates close to the goal.

These two metrics can be applied to the selection separately or together. In the experiment described in IV-A, we introduce two selection criteria, named best-effort and goal-oriented, respectively. The best-effort selection first computes the t_{95} value of all subgoal candidates and randomly chooses a candidate from the ones with $t_{95} \leq t_d$, where t_d denotes that the time constraints for planning. If no candidates meet the constraints, it selects the one with the smallest t_{95}. The goal-oriented selection builds on the best-effort selection. Instead of randomly choosing the qualified candidates with $t_{95} \leq t_d$, it ranks the samples by the goal-to-sample metric and selects the one with the smallest value, i.e. the most goal-oriented one.

E. Planning range shaping

A larger planning range usually means a larger search space and therefore a longer planning time for sampling-based motion planners. For some complex planning problems, we have to keep the planning range large. As the subgoals break down the complex problems into pieces, we are able to shape the range accordingly for every small planning problem. This concentrates the samples in the region where the planner can find a solution. A similar concept can be found in batch-informed trees (BIT) [10]. The difference is that the methods in the BIT family draw the bounds formed by high-dimensional ellipsoids after an initial solution is found. In our case, we directly set the planning range based on the planning problems.

For an n-DoF robot arm, we set the lower bounds $\mathbf{b}_l \in \mathbb{R}^n$ and upper bounds $\mathbf{b}_u \in \mathbb{R}^n$ for joints depending on the mobility of each joint and the planning problem. We

constrain the search space with greater paddings for the joints that can produce large motion, usually the ones close to the robot base, depending on the robot's kinematics.

IV. EXPERIMENTS AND RESULTS

A. Quantitative Evaluation

We verify and evaluate the effectiveness of our proposed method within two steps. First, we verify that the model can generate samples that can be planned within a short period of time. As an ablation study, we compare the results of the generated samples with and without the adaptive planning range. The difference between the selctions based on normal and log-normal distribution is shown as well. Secondly, we examine the capability of CVAE-generated subgoals in guiding the robot toward the final goal. In this part, we contrast the planning time as well as the quality of the paths formed by a sequence of subgoals against established baselines.

For the first aspect, our focus is to verify the capability of the model and outline the contribution of integrating the subgoal selection based on the time estimation and the planning range shaping, respectively. The planning problems used for the experiment are first planned with the RRTConnect planner for 30 runs. All runs of these planning problems result in a planning time of at least 0.05s. Therefore, we can see that RRTConnect has 0% in the case of less than 0.05s. As summarized in Table I, we evaluate the CVAE model using random selection and four other variants. These variants differ from each other based on their assumption of distributions (L or N) and the inclusion or exclusion of planning range shaping (S). In the random selection, we randomly select a candidate from the batch generated by the CVAE model without selection and use it as the subgoal.

	Distribution [%]			Time [s]	
	0.05	0.1	0.2	Mean	Std.
RRTConnect	0	3.8	12.9	1.153	1.86
Random	65.4	74.9	83.2	0.143	0.328
B-L-S	85.1	90.0	94.1	0.056	0.203
B-N-S	**89.2**	**93.8**	**96.7**	**0.035**	**0.137**
B-L	67.0	73.6	79.9	0.198	0.434
B-N	71.3	78.9	86.2	0.122	0.211

TABLE I: Planning to a subgoal. The letter "B" denotes the best-effort selection. The letter "L" and "N" indicates the selection using the log-normal and the normal distributions, respectively, while "S" signifies the planning range shaping.

	Succ. [%]	Distribution [%]			Time [s]		Length [rad]
		0.05	0.1	0.2	Path	Subgoal	
Baseline	**100**	0	3	11	1.153	-	**17.65**
Random	61.7	82.6	87.4	91.9	0.517	0.061	50.67
B-L-S	81.5	90.1	93.2	95.9	0.208	0.029	25.79
B-N-S	69.0	**95.2**	**97.0**	**98.4**	0.149	**0.016**	23.55
G-L-S	85.2	89.3	93.3	96.2	0.172	0.030	25.51
G-N-S	82.3	88.5	92.2	96.2	0.214	0.039	25.56

TABLE II: Planning to fianl goals. For the baseline statistics, we list the path length of the optimal planner AIT* with a planning budget of five seconds and the planning time of RRTConnect for the same reason mentioned in III-A.

Each subgoal is planned using RRTConnect with 30 runs as well. For the four variants, instead of the random selection, we use the best-effort selection mentioned in III-D without considering whether the candidates are goal-oriented, marked as "B" in Table I. The desired time bound t_d is set to be 0.05s. Three key values are defined to capture the distribution of the planning time, namely 0.05, 0.1, and 0.2. For example, the column of 0.05 shows the percentage of the planning time $t \leq 0.05s$. The mean and standard deviation of the planning time are also listed. Compared to the random selection with planning range shaping, the best-effort metric successfully selects better candidates. Meanwhile, planning range shaping contributes to better performance as well.

For the second aspect, we investigate if our proposed method can progressively navigate the robot to the final goal. Subsequently, we plan paths from the starting point or the last subgoal to the newly generated subgoal. The process terminates if the goal or the maximum number of generation trials of ten is reached. We replicate this process ten times for each problem. Other than the random and best-effort selection mentioned in the first aspect, we add the goal-oriented selection mentioned in III-D for further comparison, marked as "G" in Table II. Importantly, the success rate of reaching the *final* goal is compared. Similar to the first aspect, we investigate how the planning time distribution of subgoal changes given different selection criteria. In addition, we compare the mean planning time for each subgoal and the complete path and employ path length in [rad] to depict the path quality. As the evaluation dataset, we select the planning problems whose planning time using RRTConnect is at least 0.05s in all runs. This leads to the result that the baseline method has a success rate of 100%. Results are listed in Table II.

The results with the random selection indicate that a model relying solely on spatial information is not effective in either meeting time constraints or successfully guiding the robot to its goal configuration. Using the temporal information to select the candidates generally improves the performance in both aspects. The G-L-S variant generates sub-goals that can be planned with 0.030 seconds on average while the baseline method needs 1.15 seconds on average. Accumulating the planning time between all subgoals, our method achieved a result of 0.149 seconds on average, which is still way smaller than the baseline. Compared to the AIT* solution, our approach exhibits a longer path length. An obvious reason is that we use the RRTConnect algorithm without path smoothing in the experiment, meaning that the path to the generated subgoals is not optimal. To examine the optimality of the subgoals, other metrics are needed.

B. Two robot arms experiment

In this experiment, we apply the models trained by dataset mentioned in III-A to a completely different scenario. We use two UR10e robot arms for goal reaching task, assuming the target configuration of the upstream manipulation task is given. These two robots are operating separately. We apply our approach to one robot as motion planner, seeing the
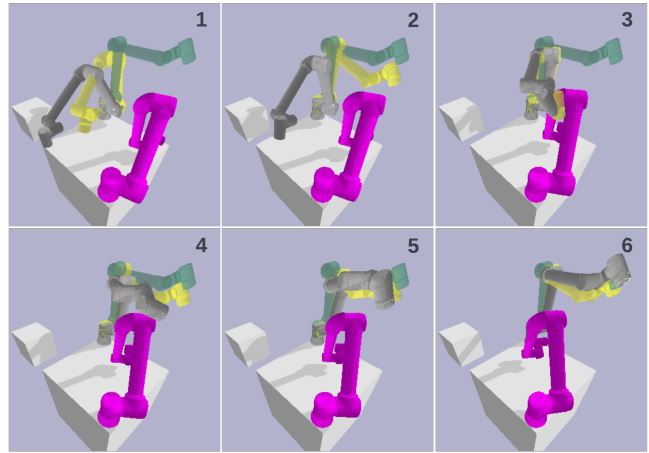

Fig. 5: Two robot arms experiment. Gray: current G-robot statek. Green: final goal. Yellow: selected subgoals by PLS. Pink: P-robot as a moving object running a fixed trajectory.

other robot as changing environment. This motion planner outputs a parameterized trajectory to the generated subgoals and finally completes the goal-reaching task without causing any collision.

As shown in Fig 5, six snapshots are captured in chronological order. During the execution, the gray robot, we call it G-robot in the following, has to reach the final goal state presented in green. The yellow robot visualizes the selected subgoal. We use the log-normal time estimator and the goal-oriented metrics (G-L-S) for the selection. Meanwhile, the robot marked in pink, we call it P-robot in the following, follows a fixed trajectory to move from the right to the left. The snapshot labeled with "1" presents the initial setup. The first subgoal chosen generally steers the G-robot's end-effector toward its base link, as it appears to be the shortest path to reach the goal when the P-robot is still at a considerable distance. In snapshot "2", the model predicts a subgoal close to the final goal region right away. As the P-robot approaches the center of the table in snapshot "3", the newly generated subgoal directs the G-robot to elevate to avoid a collision. Finally, it maneuvers over the P-robot and reaches the final goal region.

V. Conclusion

In this work, we propose PLS that aims to generate subgoals that not only can lead a robot to its final goal but also can be planned from the current robot configuration within the desired period of time. We extend the existing generative models that use only spatial information by integrating temporal information to select proper subgoals from a set of candidates. Although experiments show that PLS can achieve shorter planning time for both subgoals and the accumulated path, we want to address two issues in the future, including the consideration of more temporal information including the movement of the obstacles and the previously generated subgoals, and the integration of time estimation closer into the generation.

<div style="text-align: center;">REFERENCES</div>

[1] 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.

[2] Y. Lee, C. Chamzas, and L. E. Kavraki, "Adaptive experience sampling for motion planning using the generator-critic framework," 2022.

[3] C. Zhang, J. Huh, and D. D. Lee, "Learning implicit sampling distributions for motion planning," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 3654–3661.

[4] R. Kumar, A. Mandalika, S. Choudhury, and S. S. Srinivasa, "Lego: Leveraging experience in roadmap generation for sampling-based planning," *arXiv preprint arXiv:1907.09574*, 2019.

[5] 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.

[6] B. Ichter, E. Schmerling, T.-W. E. Lee, and A. Faust, "Learned critical probabilistic roadmaps for robotic motion planning," *arXiv preprint arXiv:1910.03701*, 2019.

[7] J. Huh, G. Xing, Z. Wang, V. Isler, and D. D. Lee, "Learning to generate cost-to-go functions for efficient motion planning," in *International Symposium on Experimental Robotics*. Springer, 2020, pp. 555–565.

[8] A. H. Qureshi, Y. Miao, A. Simeonov, and M. C. Yip, "Motion planning networks: Bridging the gap between learning-based and classical motion planners," *IEEE Transactions on Robotics*, 2020.

[9] A. Fishman, A. Murali, C. Eppner, B. Peele, B. Boots, and D. Fox, "Motion policy networks," in *Conference on Robot Learning*. PMLR, 2023, pp. 967–977.

[10] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Batch informed trees (BIT*): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 3067–3074.

[11] C. Dellin and S. Srinivasa, "A unifying formalism for shortest path problems with expensive edge evaluations via lazy best-first search over paths with edge selectors," in *Proceedings of the International Conference on Automated Planning and Scheduling*, vol. 26, 2016.

[12] M. P. Strub and J. D. Gammell, "Adaptively Informed Trees (AIT*) and Effort Informed Trees (EIT*): Asymmetric bidirectional sampling-based path planning," *The International Journal of Robotics Research*, vol. 41, no. 4, pp. 390–417, 2022.

[13] X. Huang, G. Sóti, H. Zhou, C. Ledermann, B. Hein, and T. Kröger, "HIRO: Heuristics informed robot online path planning using pre-computed deterministic roadmaps," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 8109–8116.

[14] J. Kaur, I. Chatterjee, and M. Likhachev, "Speeding up search-based motion planning using expansion delay heuristics," in *Proceedings of the International Conference on Automated Planning and Scheduling*, vol. 31, 2021, pp. 528–532.

[15] 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.

[16] Y. Lee, P. Cai, and D. Hsu, "MAGIC: Learning macro-actions for online POMDP planning," *arXiv preprint arXiv:2011.03813*, 2020.

[17] C. Chamzas, A. Cullen, A. Shrivastava, and L. E. Kavraki, "Learning to retrieve relevant experiences for motion planning," *arXiv preprint arXiv:2204.08550*, 2022.

[18] A. Lambert, B. Hou, R. Scalise, S. S. Srinivasa, and B. Boots, "Stein Variational Probabilistic Roadmaps," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 11 094–11 101.

[19] M. Xie, K. Van Wyk, A. Li, M. A. Rana, Q. Wan, D. Fox, B. Boots, and N. Ratliff, "Geometric fabrics for the acceleration-based design of robotic motion," *arXiv preprint arXiv:2010.14750*, 2020.

[20] I. A. Şucan, M. Moll, and L. E. Kavraki, "The Open Motion Planning Library," *IEEE Robotics & Automation Magazine*, vol. 19, no. 4, pp. 72–82, December 2012, https://ompl.kavrakilab.org.

[21] J. J. Kuffner and S. M. LaValle, "Rrt-connect: An efficient approach to single-query path planning," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, vol. 2. IEEE, 2000, pp. 995–1001.

[22] K. Sohn, H. Lee, and X. Yan, "Learning structured output representation using deep conditional generative models," *Advances in neural information processing systems*, vol. 28, 2015.

[23] C. Doersch, "Tutorial on variational autoencoders," *arXiv preprint arXiv:1606.05908*, 2016.

[24] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "Nerf: Representing scenes as neural radiance fields for view synthesis," *Communications of the ACM*, vol. 65, no. 1, pp. 99–106, 2021.

[25] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner, "beta-vae: Learning basic visual concepts with a constrained variational framework," in *International conference on learning representations*, 2016.

[26] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.