2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

Online Adaptation of Sampling-Based Motion Planning with Inaccurate Models

Marco Faroni and Dmitry Berenson

***Abstract*— Robotic manipulation relies on analytical or learned models to simulate the system dynamics. These models are often inaccurate and based on offline information, so that the robot planner is unable to cope with mismatches between the expected and the actual behavior of the system (e.g., the presence of an unexpected obstacle). In these situations, the robot should use information gathered online to correct its planning strategy and adapt to the actual system response. We propose a sampling-based motion planning approach that uses an estimate of the model error and online observations to correct the planning strategy at each new replanning. Our approach adapts the cost function and the sampling bias of a kinodynamic motion planner when the outcome of the executed transitions is different from the expected one (e.g., when the robot unexpectedly collides with an obstacle) so that future trajectories will avoid unreliable motions. To infer the properties of a new transition, we introduce the notion of context-awareness, i.e., we store local environment information for each executed transition and avoid new transitions with context similar to previous unreliable ones. This is helpful for leveraging online information even if the simulated transitions are far (in the state-and-action space) from the executed ones. Simulation and experimental results show that the proposed approach increases the success rate in execution and reduces the number of replannings needed to reach the goal.**

I. Introduction

The advances in physics-based simulation and deep-learning models in robotics allow to achieve complex tasks such as manipulation of deformable objects and liquids [1]–[7], contact-rich manipulation [8]–[10] and safe, compliant manipulation [11], [12]. These models are often inaccurate in predicting the outcome of actions (e.g., because of the epistemic uncertainty of learned models or the sim-to-real gap of physics simulators). For example, consider the tabletop scenario in Fig. 1: A compliant manipulator moves a heavy object across the table; because of the payload and the compliant control, the trajectory execution will deviate from the planned path, possibly causing unexpected collisions. Previous works addressed this problem by propagating uncertainty throughout the search [13], [14] or by discarding actions deemed to lead to significant errors [3]. This usually requires complex modeling of the system uncertainty over the whole state-and-action space. However, even minor mismatches between the offline and the online scenario can jeopardize the effectiveness of the uncertainty-aware planner. In the example of Fig. 1, a mismatch between the simulated and real compliance would cause the planner to find trajectories that are infeasible in the real world. Suppose the real dumbbell is heavier than the simulated one; then, the trajectories would often cause the robot to collide with the environment. Even in the case of replanning, the planner would likely continue generating a trajectory that causes a collision if it does not take online data into account. Updating the system model according to the observed behavior can be computationally demanding (e.g., it could require retraining a deep network representing the dynamics) and often cannot be performed online. Adaptive methods are typically restricted to short-horizon planning and locally update the system dynamics in the neighborhood of the current configuration when the robot is stuck [15]–[17]. As a shortcoming, these methods adapt the structure of the planning problem only in a small neighborhood of the executed transitions. For example, [15] and [16] penalize state-and-action pairs within a certain distance from previous inaccurate transitions. That is, the planner will attempt to avoid transitions in a small neighborhood of the current and previous robot configurations.

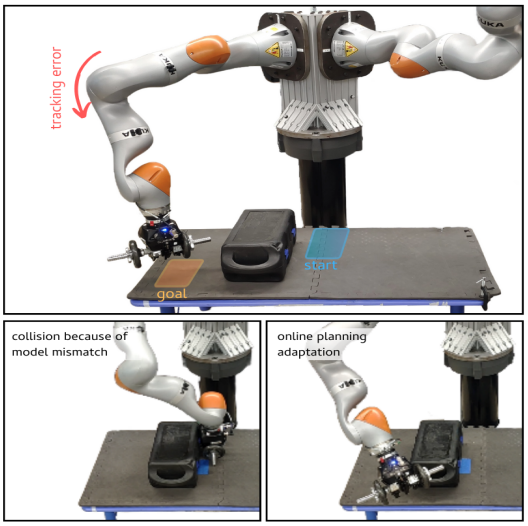


Fig. 1: A 7-degree-of-freedom manipulator carrying a weight with uncertain tracking control.

This work aims to bridge the gap between long-horizon sampling-based planning and online model adaptation to provide a robust and adaptive solution to motion planning with inaccurate models. The main idea is to formulate the problem as an optimal motion planning problem, minimizing the expected deviation between the robot model and the real system. Such expected deviation includes an offline estimate of the model inaccuracy and a residual term obtained from

This work was supported in part by the Office of Naval Research under Grant N00014-21-1-2118, and in part by the NSF under Grants IIS-1750489, IIS-2113401, and IIS-2220876. The authors are with the Robotics Department, University of Michigan, Ann Arbor, MI 48109, United States. {mfaroni; dmitryb}@umich.edu

previously executed trajectories. In particular, the residual term discourages transitions similar to those that failed in previous executions. Similarly, we adapt the sampling bias of the planner to discourage sampling such transitions. To do so, we introduce the notion of context-aware similarity, which allows the planner to infer whether new simulated transitions will be unreliable during the execution. In our examples, the context of a transition is composed of the offline-estimated model deviation and the presence of obstacles in the neighborhood of the transition. Under the assumption that transitions with a similar context will yield a similar outcome (even if they are far in the state-and-action space), the planner can infer the reliability of new simulated transitions and adapt the motion planning problem after a few executions. To do so, we update the cost function by assigning a penalty term according to the probability that a transition is unreliable (i.e., if it has a context similar to previous unreliable transitions). Then, we update the sampling distribution of our RRT-based planner to undersample transitions with high probability of being unreliable. To adapt the sampling distribution online, we build on the adaptive sampling proposed in [18], which clusters previous transitions and uses a Multi-Armed Bandit-based sampler to draw new transitions from such clusters. [18] clusters simulated transitions with respect to their cost and state-space distance. In this work, we cluster them with respect to their local context and penalize clusters containing unreliable transitions in the Multi-Armed Bandit (MAB). The result is that the planner discourages possible unreliable transitions both at sampling time and when it evaluates the cost function. We demonstrate that the proposed approach increases the execution success rate and reduces the number of necessary replannings to reach the goal in 2D examples and a 7-degree-of-freedom manipulation scenario.

II. Related Works

Motion planning under uncertainty is a long-studied topic in robotics. The goal is to find a sequence of motions that drive the robot from a start to a goal configuration despite a flawed representation of the robot and/or the environment. Because sampling-based planners such as RRT [19] and PRM [20] are the most widespread in robotics, a wide variety of sampling-based planners have been proposed also to cope with uncertainty. The most common approach is to consider an estimate of the model uncertainty while building the search tree. For example, the uncertainty estimate can be used to plan in a belief space that models the distribution of the possible outcomes of a robot action [13], [21], [22]. The shortcoming is that such planners fail when it is impossible to find a solution under all possible realizations of the uncertainty model. Moreover, they often approximate the set of reachable states in a sampling-based fashion, requiring a large number of dynamics evaluations at each iteration.

Mitrano et al. [3] proposed a different approach that learns an estimate of the model deviation to avoid solutions with large expected error. It learns a classifier that predicts whether a transition will be reliable or not when executed and uses it to discard unreliable transitions in an RRT. Similarly, [18] uses an estimate of the model error as a cost function in an RRT-like planner and shows that low-cost solutions lead to a higher success rate. Other approaches aim to find a trusted domain where a given controller can compensate for the execution error [23]. Replanning approaches have also proved effective in coping with uncertain dynamics [24] and moving obstacles [25]. All these approaches rely on offline information to quantify the model uncertainty. However, they do not reason about possible mismatches between the uncertainty estimate and the real-world outcome of the robot actions. That is, even if a robust planner is used, these methods do not leverage new observations to correct the behavior of the planner, possibly getting the robot stuck repeatedly. Moreover, updating the system model according to the observed behavior is prohibitive because it requires retraining the model after each new observation.

Outside the realm of sampling-based planning, other approaches addressed the possibility of changing the planning strategy online according to online information on the robot execution error. For example, [15] and [16] use A* to locally penalize the neighborhood of state-and-action pairs that proved to be unreliable (i.e., such that their actual execution error was different from the expected one). This prevents repeating unreliable actions at the next execution. However, the correction only applies to states and actions in a small neighborhood of the executed ones and A* is usually inefficient with large continuous spaces, as is typical of robotic manipulation. As for online planning, receding-horizon planners have been proposed to deal with errors in execution. E.g., [17] implements a receding-horizon planner that trades off the avoidance of states with unexpected dynamics and stagnation of the progress towards the goal.

This work focuses on sampling-based kinodynamic motion planning. In particular, we would like a kinodynamic planner to adapt the search according to an estimate of the model error and online observations gathered during the execution. To do so, we update the cost function and the sampling distribution of the planner. Biasing the sampling is a common strategy to improve the solution cost quickly [26]–[32]. Sampling bias may be learned *a priori* [29], [30] or adapted during the planning [18], [32]. We leverage the online-learning approach described in [18] to trade off exploitation of regions of the search space with low cost vs. less-explored ones. [18] runs `RRT` multiple times and, at the end of each run, it groups the transitions via clustering. Then, it uses a Multi-Armed Bandit (MAB) algorithm to decide whether the next transition will be drawn from a cluster or the uniform distribution. We leverage this sampling strategies with two major changes: (i) we perform clustering with respect to the context of the transitions instead of using state-and-action space distance; (ii) we modify the MAB's rewards of the clusters that contain unreliable transitions.

III. Problem Statement

Consider a dynamical system, $x_{k+1} = f(x_k, u_k)$ where $f : X \times U \rightarrow X$ and X and U are the state space

and the control space, respectively. We consider the motion planning problem from a start configuration, $x_{\text{start}} \in X_{\text{free}}$, to a configuration in the goal set, $X_{\text{goal}} \subseteq X_{\text{free}}$, where $X_{\text{free}} \subseteq X$ is the set of valid robot states.

Problem 1 (Motion planning problem): Given $x_{\text{start}} \in X_{\text{free}}$ and $X_{\text{goal}} \subseteq X_{\text{free}}$, find a sequence of controls, $\bar{u} = [\bar{u}_0, \ldots, \bar{u}_{N-1}]$, such that $x_N \in X_{\text{goal}}$ under robot dynamics $x_{k+1} = f(x_k, u_k)$.

We assume that the planning algorithm has access to (i) an approximate model of the system, $\hat{f} : X \times U \to X$, to simulate the outcome of a state-and-action pair and (ii) a model deviation estimate, $\texttt{MDE}(x, u) \approx ||f(x, u) - \hat{f}(x, u)||$. Because $\hat{f} \neq f$, the planner may find solutions that do not reach the goal or collide with the environment when executed. A possible solution to this issue is to search for valid solutions under both the true and the approximate dynamics by assuming or estimating the properties of the model uncertainty [13], [22], [23]. Another approach consists of optimistically planning with the approximated dynamics and triggering replanning after each action or when the executed trajectory deviates too much from the planned one [15], [16], [33]. In this work, we follow this second approach. In particular, we assume a safety function is available to stop the robot when it deviates from the nominal trajectory. Such a `replan&execute` paradigm is exemplified in Alg. 1.

Algorithm 1: The `replan&execute` paradigm.

Input: x_{start}, X_{goal}, $\hat{f}$, δ_{safe}

$x = x_{\text{start}}$, $\hat{x} = x_{\text{start}}$
while $x \notin X_{\text{goal}}$ **do**
  $\bar{u} = \texttt{plan}(x, X_{\text{goal}})$
  for $\bar{u}_i$ **in** $\bar{u}_0, \ldots, \bar{u}_{N-1}$ **do**
    $x = \texttt{execute}(\bar{u}_i)$
    $\hat{x} = \hat{f}(\hat{x}, \bar{u}_i)$
    if $||x - \hat{x}|| \geq \delta_{\text{safe}}$ **then**
      break

The `replan&execute` paradigm can be combined with the `MDE` to plan for trajectories that avoid high-error transitions [3], [18]. However, it can become inefficient when there is a mismatch between the information embedded in the `MDE` function and the real environment. For example, consider a planner that minimizes the `MDE` function over the trajectory, as in [18]. Suppose that the optimal solution collides with the environment during the execution. Replanning by minimizing the same `MDE` function would return a solution similar to the previous one, and we would not expect the robot to make progress toward the goal. To overcome this issue we need to change the structure of the motion planning problem, e.g., by updating the cost function with the new information.

Our goal is to adapt the planning problem by embedding online information gathered from real observations so that future re-plannings will avoid transitions whose actual error is inconsistent with the `MDE` estimate. We do so by adapting the motion planner cost function and sampling distribution so that it will discourage transitions with a high probability of being unreliable, i.e., those with large expected error or similar to previous unreliable transitions.

IV. Method

We can summarize the proposed method as follows:

1) The motion planner searches for a path that minimizes the `MDE` function;
2) The robot executes the trajectory; it stops if the deviation from the nominal path is larger than a safety threshold.
3) At each safety stop, we measure the mismatch between the execution error of each transition and its `MDE` value. We also compute the context of each executed transition, i.e., a representation of the environment in the proximity of the transition.
4) We update the cost function to be used for the next replanning. To do so, we classify previous transitions as unreliable by running an anomaly detection algorithm on the execution error. Then, we add a penalty term to transitions that are similar to previous unreliable transitions. The similarity function is based on the similarity between the local environments of the new and the executed transitions. This allows the planner to infer the probability that a transition is unreliable (i.e., anomalous) even when it is far from previous ones in the state-and-action space. This behavior is also enforced during sampling by leveraging an online learning bias strategy [18] that discourages sampling transitions with a high probability of being unreliable.

For a better understanding of the approach, consider the simplified problem in Fig. 2, where a point robot moves in a 2D environment and is subject to a drift to the right. The magnitude of the drift depends on the robot position, as shown in the left image. Our approach initially plans and executes a trajectory found by minimizing the nominal error estimate. For this reason, the path lies in the low-drift (yellow) region. During the execution, the robot deviates from the nominal trajectory and may collide with the obstacle, causing a halt. Before replanning, we evaluate the local context of each executed transition (second image from the left). The context contains information on the obstacle presence around the transition. During the replanning, we use this information to infer the reliability of new simulated transitions. For example, in the third image, the planner undersamples and assigns a large cost to transitions with a context similar to that of unreliable transitions (orange grids). The result is that the new solution will be less likely to contain transitions with unfavorable context (right image).

The following sections illustrate the unreliability detection, the adaptive cost function, the context-based similarity, and the adaptive biased planner in detail.

A. Detection of unreliable executed transitions

Consider the executed transitions, $\mathcal{T}^E = \{\tau_1^E, \ldots, \tau_M^E\}$ and their execution error $e(\tau_j^E) = ||f(\tau_j^E.x, \tau_j^E.u) - \hat{f}(\tau_j^E.x, \tau_j^E.u)||$. We consider a transition to be unreliable

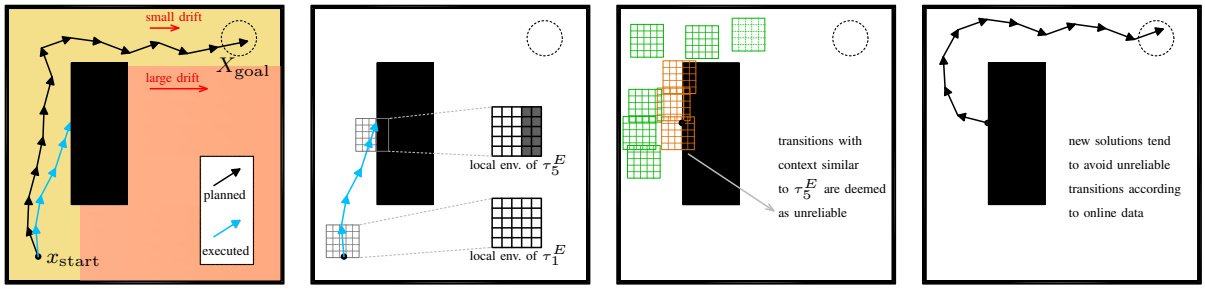


Fig. 2: Sketch of the proposed method. From left to right: (i) the robot plans and executes a trajectory minimizing the expected error. It stops when it deviates too much from the nominal path. (ii) every time it stops, we observe the actual error and the context of the executed transitions. (iii) we update the cost function and the sampling bias to avoid transitions similar to the unreliable ones. (iV) the new solution will likely avoid such transitions.

if the predicted and the actual execution error have an anomalous mismatch. Thus, we run an anomaly detection algorithm on $e(\tau_j^E)\,\forall j = 1, ..., M$, and assign an anomaly label, l_j^E, to each transition such that:

$$l_j^E = l(\tau_j^E) = \begin{cases} 1 & \text{if } e(\tau_j^E) \text{ is anomalous} \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

Anomalous transitions are those for which the MDE did not predict the execution error reliably. We will therefore try to avoid using similar transitions during the next replannings.

B. Adaptive cost function

At each replanning, we minimize the cost function

$$c(\tau) = \texttt{MDE}(\tau) + P_{\text{anomaly}}(\tau)\, C + (1 - P_{\text{anomaly}}(\tau))\, \hat{e}(\tau) \tag{2}$$

where $C > 0$ is a penalty term to discourage anomalous transitions, $P_{\text{anomaly}}(\tau)$ is an estimate of the probability that τ will be anomalous if executed, and $\hat{e}(\tau)$ is an estimate of error $e(\tau)$ if τ will be executed. The MDE term considers the nominal estimate of the model error. The second and the third term come into play after the first execution: the second term penalizes transitions with high probability of being anomalous/unreliable; the third one corrects the MDE estimate according to the error observed for similar transitions.

To compute $P_{\text{anomaly}}(\tau)$ and $\hat{e}(\tau)$ for new, simulated transitions, we define a similarity measure, $S(\tau_i, \tau_j^E) \in [0, 1]$, between transitions and use it as a proxy of the probability that $l(\tau_i) = l(\tau_j^E)$. Then, we denote by $\mathcal{T}_{\text{srt}}^E$ the sequence composed of the elements of $\mathcal{T}^E$ sorted in descending order according to $S(\tau, \tau_j^E)$ and denote by $S_{\text{srt}} = [S(\tau, \tau_{\text{srt},1}^E), ..., S(\tau, \tau_{\text{srt},M}^E)]$ the corresponding similarity vector. Thus,

$$\begin{aligned} P_{\text{anomaly}}(\tau) = S_{\text{srt},1}\, l(\tau_{\text{srt},1}^E) + (1 - S_{\text{srt},1}) S_{\text{srt},2} l(\tau_{\text{srt},2}^E) \\ + ... + \prod_{k=1}^{M-1} (1 - S_{\text{srt},k}) S_{\text{srt},M} l(\tau_{\text{srt},M}^E) \end{aligned} \tag{3}$$

P_{anomaly} approximates the probability that τ is unreliable by weighting all executed transitions according to the their similarity with τ.

We leverage the similarity measure also to compute $\hat{e}(\tau)$ as the weighted average of the observed residual errors:

$$\hat{e}(\tau) = \frac{\sum_{j=1}^{M} S(\tau, \tau_j^E)\, e(\tau_j^E)}{\sum_{j=1}^{M} S(\tau, \tau_j^E)} \tag{4}$$

C. Context-based transition similarity

Cost function (2) relies on the definition of the similarity function $S(\tau_i, \tau_j)$. Previous works on sampling-based motion planning trying to leverage similarity between transitions or configurations typically relied on state-and-action Euclidean distance [18], [34]. However, this only allows the planner to infer the properties of new transitions when they are in the local neighborhood of previous ones. To overcome the locality issue, we introduce the idea of context similarity, i.e., we deem two transitions as similar if they have similar local environment features. The definition of a suitable context-similarity function is problem-dependent. In our examples, we use a local grid centered at the robot end-effector pose and compute a context vector ζ of size $F + 1$ such that

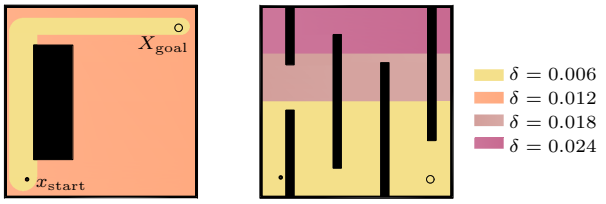
$$\zeta_i = \begin{cases} \texttt{MDE}(\tau) & \text{if } i = F + 1 \\ 1 & \text{if } i \leq F \text{ and the } i\text{th point of the local grid is in collision with the environment} \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

The similarity measure is then $S(\tau_i, \tau_j) = e^{\frac{1}{2}||\zeta(\tau_i) - \zeta(\tau_j)||} \in [0, 1]$. How to automatically define $S(\tau_i, \tau_j)$ for a given problem is an open question that we leave for future work.

D. Adaptive context-aware sampling bias

We embed our planning strategy in a kinodynamic sampling-based planner based on MAB-RRT [18]. MAB-RRT runs multiple instances of RRT sequentially and, at the end of each run, clusters previous transitions according to a reward function. Then, in the next run, it uses the clusters, $\mathcal{C}_i$, as arms of a Multi-Armed Bandit algorithm. MAB iteratively chooses from which cluster to draw the next transition, trading off the exploitation of high-reward (i.e., low-cost) clusters and less-explored regions. This results in a more efficient sampling of the state space and better path quality.

We use MAB-RRT as a planner in line 3 of Alg. 1 with reward function $r(\tau) = -c(\tau)$. Then, we modify the clustering strategy to leverage the online information



Scenario A Scenario B

Fig. 3: Scenarios for the numerical analysis in Sec. V-A.

gathered from the executed transitions. We use the similarity function $S(\tau_i, \tau_j)$ to compute the clusters; then we compute the initial reward of the ith arm of the MAB as

$$\bar{r}_i = \frac{1}{|\mathcal{C}_i|} \sum_{\tau_j \in \mathcal{C}_i} -\texttt{MDE}(\tau_j).$$

Finally, we adjust the initial reward of each arm based on the previously executed transitions. To do so, consider the set of clusters $\mathcal{C} = \{\mathcal{C}_1, ..., \mathcal{C}_H\}$, their initial rewards $\bar{r} = \{\bar{r}_1, ..., \bar{r}_H\}$, and the set of executed transitions $\mathcal{T}^E$. First, we predict whether τ_j^E belongs to the ith cluster. If so, we adjust $\bar{r}_i$ according to the probability that $\mathcal{C}_i$ contains anomalous transitions, i.e.:

$$\bar{r}_i = \bar{r}_i \left(1 - \frac{|\{\tau \in \mathcal{T}^E \text{ s.t. } b_i(\tau) = 1 \wedge l(\tau) = 1\}|}{|\{\tau \in \mathcal{T}^E \text{ s.t. } b_i(\tau) = 1\}|}\right) \quad (6)$$

where $b_i(\tau) = 1$ if τ was predicted to belong to $\mathcal{C}_i$ and 0 otherwise, and $|\cdot|$ is the cardinality of a set. The result of this operation is that sampling clusters associated with anomalous transition is discouraged during planning.

The combination of the biased sampling and the adaptive cost function (2) will avoid unreliable transitions during planning. Our approach assumes the planner minimizes the cost function during each planning. Although [18] did not provide formal guarantees on asymptotic optimality, results in Sec. V suggest the cost decreases consistently with iterations during each replanning.

V. Results

This section validates our approach in simulated 2D scenarios and a 7-DOF real-world manipulation problem, showing that it outperforms the baselines in terms of execution success rate and number of replannings to reach the goal.

A. 2D navigation example

We consider the scenarios in Fig. 3. The robot is a single integrator with a drift term $\delta > 0$ such that $x_{k+1} = f(x_k, u_k) = x_k + Tu_k + [\delta(x), 0]^T$, $X = [0, 1]^2$, $U \in [0.5, 0.5]^2$, while the model used for planning is $x_{k+1} = \hat{f}(x_k, u_k) = x_k + Tu_k$, where T is the control duration. The drift $\delta(x)$ depends on the robot position and is depicted in Fig. 3. We consider $\texttt{MDE} = \delta(x)$.

Note that there is a mismatch between the estimated error and the actual one because the robot stops as soon as it touches an obstacle. For example, in Scenario A, the low-error region near the left edge of the obstacle will actually result in a large execution error because the drift will drive the robot into contact with the obstacle. The planner should learn to avoid such regions after a few executions, even though it encompasses the best nominal solution (i.e., it minimizes the MDE function).

To define the context variable, ζ, we consider a 5x5 voxel grid centered in the robot frame. For each new transition τ, we set $\zeta_i = 1$ if the ith grid point is in collision with the environment and $\zeta_i = 0$ otherwise. According to (5), the last element of ζ is equal to $\texttt{MDE}(\tau)$.

We compare our context-aware algorithm, CTX-RRT, with its non-adaptive counterpart, MAB-RRT minimizing the MDE function ($c(\tau) = \texttt{MDE}(\tau)$), and two variants for ablation:

- CTX-RRT-static-bias, which adapts the cost function according to (2) but does not adapt clusters' rewards with the observed error;
- CTX-RRT-static-cost, which adapts the clusters' rewards according to (6) but not the cost function.

We evaluate the average number of replannings to reach the goal and the success rate (we report a failure if the robot does not reach the goal within 10 replannings). Table I shows the results for 30 trials. For both scenarios, CTX-RRT has the largest success rate and the smallest average number of replannings to reach the goal. As expected, MAB-RRT performs worse than all the planners because it tends to find similar paths near the obstacles even if such movements keep leading to a halt in execution. The two variants, CTX-RRT-static-cost and CTX-RRT-static-bias, slightly outperform MAB-RRT. However, they perform significantly worse than CTX-RRT, highlighting the importance of the combined adaptive bias and cost. Results are consistent across the two scenarios, with an expected performance decay of all methods in Scenario B because of the higher complexity of the problem.

Note that such a combined adaptation is in line with the working principle of MAB-RRT, which iteratively improves the solution by running multiple instances of RRT and choosing the best solution. However, if the cost function does not change, the planner will use the nominal MDE to select the trajectory for execution. On the other hand, if only the cost function changes, MAB-RRT's performances decay because the MAB-based sampling is no longer effective with respect to the new cost function.

B. Manipulation example

We demonstrate our approach on a manipulation task with uncertain dynamics. We consider the tabletop application in Fig. 1, where a 7-degree-of-freedom arm moves a dumbbell (3.6 kg) from a start to a goal pose on the table. A joint-space impedance controller makes the robot compliant with the environment, yet it causes a significant trajectory-tracking error because of the large payload. The state and control spaces are joint position and velocity, respectively. The MDE function is an estimate of the robot Cartesian-space tracking error depending on the robot joint positions as described in Appendix A of [18]. Note that both planners are aware of the obstacles (i.e., they perform collision checking on the simulated motions); yet collisions occur at runtime because of the trajectory tracking error discussed above. Because of

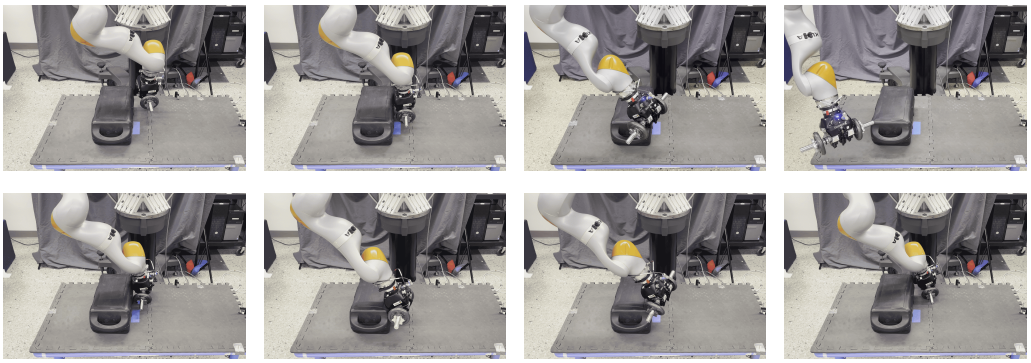


Fig. 4: Examples of executions of two trajectories planned with CTX−RRT (top) and MAB−RRT (bottom).

TABLE I: Simulation results for the 2D scenarios.

(a) Scenario A

	Success rate mean	N. replannings mean (std.dev.)
MAB−RRT	0.40	5.9(1.9)
CTX−RRT−static−cost	0.93	5.5(1.8)
CTX−RRT−static−bias	0.90	4.6(0.78)
CTX−RRT	**0.97**	**3.5(0.74)**

(b) Scenario B

	Success rate mean	N. replannings mean (std.dev.)
MAB−RRT	0.27	8.2(1.1)
CTX−RRT−static−cost	0.54	7.9(0.55)
CTX−RRT−static−bias	0.27	8.0(1.2)
CTX−RRT	**0.75**	**6.8(0.98)**

the tracking error, the robot may collide with the obstacles in the environment causing the robot to halt. The MDE is able to estimate only the error owed to the robot dynamics. Possible contacts with the obstacles and the inaccuracy of the MDE function cause a mismatch between the estimated and the measured Cartesian error.

To define the context variable, ζ, we consider a rectangular cuboid centered on the robot tool center point and aligned with tool frame. We consider a uniform grid of 25 points on each face of the cuboid. Then, we set $\zeta_i = 1$ if at least half of the points on the ith face are in collision with the environment and 0 otherwise. Finally, the last element of ζ is equal to $\texttt{MDE}(\tau)$ according to (5).

We evaluate the average number of replannings to reach the goal and the success rate (we report a failure if the robot does not reach the goal within 10 replannings). Table II shows the comparison between CTX−RRT and MAB−RRT (30 repetitions). Note that the large robot compliance causes a low overall success rate with both planners. Nonetheless, CTX−RRT significantly increases the success rate and reduces the average number of replannings needed to reach the goal. In particular, CTX−RRT leverages the first executions to understand that motions too close to the obstacles lead to a large error even if it was not predicted by the MDE. As a consequence, in the next replannings, it tends to avoid such motions and chooses paths that are less likely to result in a collision. In comparison, MAB−RRT keeps searching for the best solution according to the MDE's information, leading to

TABLE II: Experimental results for the 7D scenario.

	Success rate mean	N. replannings mean (std.dev.)
MAB−RRT	0.27	8.5(0.49)
CTX−RRT	**0.46**	**7.2(0.82)**

multiple stops. This is visible also in the accompanying video and in Fig. 4, which show executions with the two planners. Initially, both planners find solutions that minimize the MDE function. During the execution, the robot deviates from the nominal path and collides with the obstacle. CTX−RRT uses the unexpected collisions to update the cost function and the sampling distribution and is eventually able to find a path in a different region of the space to avoid further collisions (top images). In comparison, MAB−RRT keeps finding similar solutions causing several halts (bottom images).

VI. Conclusions

We presented an adaptive planning strategy that can cope with model uncertainty and mismatches between the estimated and actual execution errors. The approach adapts the cost function and sampling bias of a kinodynamic motion planner to discourage unreliable robot motions. Results on 2D examples and a 7D manipulation scenario showed that the approach improves the success rate during execution. The current work relies on the definition of a context vector that captures the local environment features of each transition. Future works will aim to learn an effective representation and similarity function for the transition context automatically.

Appendix

A. Parameter tuning

This appendix describes the tuning of the parameters used to perform the experiments in Sec. V-A and Sec. V-B.

In all experiments, we tune the parameters needed to run MAB−RRT as described in Appendix A.1 of [18]. The anomaly detection algorithm (see Sec. IV-A) uses a z-score anomaly detection with z=0.95 and we set $C = 10$ in (2). In the 2D problems (Sec. V-A), the context variable consists of a 5x5 grid with a uniform resolution equal to 0.05 and $\delta_{\text{safe}} = 0.05$. In the 7D problem (Sec. V-B), the context variable is a binary vector whose elements are associated with each face of rectangular cuboid of size 0.4m×0.3m×0.3m aligned with the x, y, and z axis of the tool frame and $\delta_{\text{safe}} = 0.2$ rad.

REFERENCES

[1] M. Lippi, P. Poklukar, M. C. Welle, A. Varava, H. Yin, A. Marino, and D. Kragic, "Enabling visual action planning for object manipulation through latent space roadmap," *IEEE Transactions on Robotics*, vol. 39, pp. 57–75, 2023.

[2] G. Nicola, E. Villagrossi, and N. Pedrocchi, "Co-manipulation of soft-materials estimating deformation from depth images," *Robotics and Computer-Integrated Manufacturing*, vol. 85, p. 102630, 2024.

[3] P. Mitrano, D. McConachie, and D. Berenson, "Learning where to trust unreliable models in an unstructured world for deformable object manipulation," *Science Robotics*, vol. 6, no. 54, 2021.

[4] X. Lin, C. Qi, Y. Zhang, Z. Huang, K. Fragkiadaki, Y. Li, C. Gan, and D. Held, "Planning with spatial-temporal abstraction from point clouds for deformable object manipulation," in *Conference on Robot Learning*, 2022.

[5] P. Mitrano, A. LaGrassa, O. Kroemer, and D. Berenson, "Focused adaptation of dynamics models for deformable object manipulation," *Robotics: Science and Systems*, 2022.

[6] J. Liu, Y. Chen, Z. Dong, S. Wang, S. Calinon, M. Li, and F. Chen, "Robot cooking with stir-fry: Bimanual non-prehensile manipulation of semi-fluid objects," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 5159–5166, 2022.

[7] Z. Sun, Z. Wang, J. Liu, M. Li, and F. Chen, "Mixline: A hybrid reinforcement learning framework for long-horizon bimanual coffee stirring task," in *International Conference on Intelligent Robotics and Applications*, 2022, pp. 627–636.

[8] J. Liang, X. Cheng, and O. Kroemer, "Learning preconditions of hybrid force-velocity controllers for contact-rich manipulation," in *Conference on Robot Learning*, 2022.

[9] E. Páll, A. Sieverling, and O. Brock, "Contingent contact-based motion planning," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2018, pp. 6615–6621.

[10] M.-T. Khoury, A. Orthey, and M. Toussaint, "Efficient sampling of transition constraints for motion planning under sliding contacts," in *IEEE International Conference on Automation Science and Engineering*, 2021, pp. 1547–1553.

[11] A. V. Vivas, A. Cherubini, M. Garabini, P. Salaris, and A. Bicchi, "Minimizing energy consumption of elastic robots in repetitive tasks," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2023.

[12] T. Marcucci, M. Garabini, G. M. Gasparri, A. Artoni, M. Gabiccini, and A. Bicchi, "Parametric trajectory libraries for online motion planning with application to soft robots," in *International Symposium on Robotics Research*. Springer, 2020, pp. 1001–1017.

[13] A. Bry and N. Roy, "Rapidly-exploring random belief trees for motion planning under uncertainty," in *IEEE International Conference on Robotics and Automation*, 2011, pp. 723–730.

[14] J. Van Den Berg, S. Patil, and R. Alterovitz, "Motion planning under uncertainty using differential dynamic programming in belief space," in *The International Symposium of Robotics Research*, 2017, pp. 473–490.

[15] A. Vemula, Y. Oza, J. A. Bagnell, and M. Likhachev, "Planning and execution using inaccurate models with provable guarantees," in *Robotics: Science and Systems*, 2016.

[16] A. Vemula, J. A. Bagnell, and M. Likhachev, "Cmax++: Leveraging experience in planning and execution using inaccurate models," in *AAAI Conference on Artificial Intelligence*, 2021, pp. 6147–6155.

[17] S. Zhong, Z. Zhang, N. Fazeli, and D. Berenson, "Tampc: A controller for escaping traps in novel environments," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1447–1454, 2021.

[18] M. Faroni and D. Berenson, "Motion planning as online learning: A multi-armed bandit approach to kinodynamic sampling-based planning," *IEEE Robotics and Automation Letters*, vol. 8, no. 10, pp. 6651–6658, 2023.

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

[20] L. E. Kavraki, P. Svestka, J. . 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.

[21] Q. H. Ho, Z. N. Sunberg, and M. Lahijanian, "Gaussian belief trees for chance constrained asymptotically optimal motion planning," in *IEEE International Conference on Robotics and Automation*, 2022, pp. 11 029–11 035.

[22] A. Wu, T. Lew, K. Solovey, E. Schmerling, and M. Pavone, "Robust-rrt: Probabilistically-complete motion planning for uncertain nonlinear systems," in *The International Symposium of Robotics Research*. Springer, 2022, pp. 538–554.

[23] C. Knuth, G. Chou, N. Ozay, and D. Berenson, "Planning with learned dynamics: Probabilistic guarantees on safety and reachability via lipschitz constants," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 5129–5136, 2021.

[24] W. Sun, S. Patil, and R. Alterovitz, "High-frequency replanning under uncertainty using parallel sampling-based motion planning," *IEEE Transactions on Robotics*, vol. 31, no. 1, pp. 104–116, 2015.

[25] C. Tonola, M. Faroni, M. Beschi, and N. Pedrocchi, "Anytime informed multi-path replanning strategy for complex environments," *IEEE Access*, vol. 11, pp. 4105–4116, 2023.

[26] J. D. Gammell and M. P. Strub, "Asymptotically optimal sampling-based motion planning methods," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 4, pp. 295–318, 2021.

[27] N. M. Amato, O. B. Bayazit, L. K. Dale, C. Jones, and D. Vallejo, "Obprm: An obstacle-based prm for 3d workspaces," in *International Workshop on Algorithmic Foundations of Robotics*, 1998, pp. 155–168.

[28] A. Attali, S. Ashur, I. B. Love, C. McBeth, J. Motes, D. Uwacu, M. Morales, and N. M. Amato, "Evaluating guiding spaces for motion planning," 2022.

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

[30] R. Cheng, K. Shankar, and J. W. Burdick, "Learning an optimal sampling distribution for efficient motion planning," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2020.

[31] 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 *IEEE International Conference on Robotics and Automation*, 2021.

[32] S. Dalibard and J.-P. Laumond, "Linear dimensionality reduction in random motion planning," *The International Journal of Robotics Research*, vol. 30, pp. 1461–1476, 2011.

[33] T. Dam, G. Chalvatzaki, J. Peters, and J. Pajarinen, "Monte-carlo robot path planning," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 11 213–11 220, 2022.

[34] O. Arslan and P. Tsiotras, "Machine learning guided exploration for sampling-based motion planning algorithms," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2015, pp. 2646–2652.