2019 International Conference on Robotics and Automation (ICRA)
Palais des congres de Montreal, Montreal, Canada, May 20-24, 2019

Safe and Efficient High Dimensional Motion Planning in Space-Time with Time Parameterized Prediction

Shen Li[1] and Julie A. Shah[1]

Abstract— **In this work, we propose an algorithm that can plan safe and efficient robot trajectories in real time, given time-parameterized motion predictions, in order to avoid fast-moving obstacles in human-robot collaborative environments. Our algorithm is able to reduce the robot configuration space and the time domain significantly by constructing a Lazy Safe Interval Probabilistic Roadmap based on a pre-planned path. The algorithm then plans efficient obstacle-avoidance strategies within the space-time roadmap. We benchmarked our algorithm by evaluating the performance of a simulated 6-joint manipulator attempting to avoid a quickly moving human hand, using a dataset collected from human experiments. We compared our algorithm's performance with those of 8 variations of prior state-of-the-art planners. Results from this empirical evaluation indicate that our method generated safe plans in 97.5% of the evaluated situations, achieved a planning speed 30 times faster than the benchmarked methods that planned in the time domain without space reduction, and accomplished the minimal solution execution time among the benchmarked planners with a similar planning speed.**

I. Introduction

In many domains, it is more productive and efficient for robots to collaborate with humans in close physical proximity to one another, rather than for either to work on their own [1]. However, it is significantly challenging for a robot to plan efficient motion within a high-dimensional configuration space while avoiding nearby humans who are moving rapidly. For example, in a collaborative tabletop manipulation task described by Lasota and Shah [1], humans moved at speeds of up to 141cm/s while operating within a distance as short as 6cm from a robot [2].

Motion planning algorithms usually take as input a series of predicted locations of obstacles in the near future, representing the obstacles' swept volumes. These planning algorithms then consider all swept volumes as obstacles to be avoided [3, 4]. This conservative volume significantly reduces the flexibility a planner is allowed when finding an efficient path while also maintaining safety constraints.

Given the time of each predicted obstacle location, motion planners can potentially reduce the conservative volume, because the robot is not required to consider the obstacle locations in the past due to time monotonicity [5]. With a human motion predictor capable of providing obstacle trajectory with time parameterization — as described, for example, in work by Lasota and Shah [6] — planners can further optimize robot trajectories in space and time in order to generate novel behaviors to avoid dynamic obstacles with improved economy of motion. For example, in the collaborative manipulation scenario depicted in Fig. 1, it would be safer and more efficient if the robot temporarily halted motion as the obstacles moved out of the robot's way.

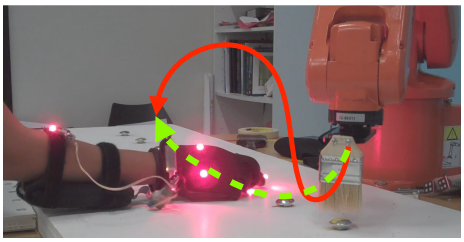


Fig. 1: A robot moves to the other side of a human hand. Given time-parameterized prediction of the hand's position, a safer and more efficient robot behavior would be to temporarily halt motion while the hand moves out of the way (as indicated by the path in green), rather than to follow a lengthy detour around the hand (as indicated by the path in red).

However, this behavior would require the ability to reason about obstacle avoidance within both high-dimensional robot configuration space ($\mathcal{C}$) and time domain ($\mathcal{T}$), which is computationally challenging [7–9]. Current planning algorithms treat time parameterizations as fixed or prespecified and do not optimize them during the planning process [5]. These algorithms usually apply separate post-processing techniques for time parameterization, such as shortcutting or smoothing out redundant or jerky motions [10–12], and constraining trajectories under kinodynamic constraints, such as velocity, acceleration, and torque limits [13–15]. However, the purpose of these techniques is generally not to avoid obstacles by manipulating time [5], as avoidance is handled separately by manipulating the path in space during the planning process. Some algorithms reason about collision avoidance in both space and time [7, 16], but it can be challenging to generalize these algorithms to high-dimensional manipulators.

In this work, we propose a space-time representation called the *Lazy Safe Interval Shortcut Probabilistic Roadmap* (Lazy SISPRM), which is applied to a pre-planned path for further optimization in $\mathcal{C} \times \mathcal{T}$ to generate novel obstacle-avoidance behaviors with greater variety and efficiency. The Lazy SISPRM planner significantly reduces the high-dimensional search space, $\mathcal{C} \times \mathcal{T}$, via a two-step method. It first reduces $\mathcal{C}$ to a shortcut probabilistic roadmap (SPRM): a graph composed of waypoints along the pre-planned path alongside a set of shortcuts for that path. Next, the planner incorporates time and reduces the joint space, SPRM $\times \mathcal{T}$, to a "Lazy SISPRM" by grouping safe robot configurations through a consecutive series of time steps, in the spirit of the

[1]The authors are with the Computer Science and Artificial Intelligence Intelligence Laboratory, Massachusetts Institute of Technology. shenli@mit.edu, julie_a_shah@csail.mit.edu

"safe interval path planning (SIPP)" Phillips and Likhachev [7]. Within this Lazy SISPRM, our algorithm conducts an optimal search to find the locally optimal trajectory. (The method is "lazy" in that it will not evaluate the feasibility of the SISPRM until necessary during the search [17].)

We demonstrate that, compared with prior art, our method for planning among moving obstacles improves the economy of motion by optimizing trajectory in the SISPRM. Incorporating human motion data from prior work by Lasota and Shah [1], we simulated a scenario involving human reaching and retracting motions. We compared the Lazy SISPRM planner with a set of state-of-the-art planners, including search-based planners [7, 18, 19], sampling-based planners [20], and functional gradient descent methods [3, 5, 10, 11]. The results indicate that by significantly reducing the search space, the Lazy SISPRM planner finished within a planning-time budget as short as 171ms on average — which is sufficient for real-time replanning when coupled with state-of-the-art motion predictors [6]. By optimizing paths in space-time, the Lazy SISPRM planner produced collision-free trajectories for 97.5% of the situations evaluated, and achieved the lowest solution execution time among the benchmarked planners.

II. Related Work

A. Motion Planning in Dynamic Environments

In the field of motion planning, many algorithms successfully enable robots to avoid dynamic obstacles in real time by bending trajectories within $\mathcal{C}$, such as sampling-based [21], discrete-time trajectory optimization [3], and continuous-time trajectory optimization [22] methods. However, time parameterizations are usually fixed, or not specified at all, during the planning process [5]. To ensure safety, the resulting paths are typically conservative and avoid the entire set of predicted obstacle volumes across time. With time-parameterized predictions and planning in $\mathcal{C}\times\mathcal{T}$, our method is able to improve efficiency while maintaining safety.

B. Time Parameterization for Kinodynamic Constraints

In kinodynamic planning, researchers have developed methods to produce time-optimal trajectories via numerical integration [23–25], reachability analysis [26], and convex optimization [27]. Yi et al. [14] utilized Markov Chain Monte Carlo to improve the sampling and planning efficiencies for Informed RRT*. Pham et al. [15] developed admissible velocity propagation to reduce the dynamic state space. However, these methods focused primarily on manipulating time to constrain trajectory under kinodynamic constraints. Our method incorporates time parameterization to develop safe and efficient strategies for avoiding dynamic obstacles.

C. Time Parameterization for Dynamic Obstacle Avoidance

One way to avoid obstacles by manipulating time is to plan within $\mathcal{T}$ as an extra dimension besides $\mathcal{C}$, which is computationally challenging [7–9]. Prior work has focused on reducing the search space; for example, Kushleyev and Likhachev [8] proposed a data structure — a time-bounded lattice — to merge short-term planning within time and long-term planning without time. Phillips and Likhachev [7] reduced the space $\mathcal{T}$ by grouping safe configurations throughout a period of time as a single "safe interval." They developed a search algorithm, SIPP, to find optimal trajectories based on these safe intervals. Gonzalez et al. [9] used state dominance relationships to improve the efficiency of SIPP. Narayanan et al. [18] extended SIPP to an anytime search algorithm by integrating ARA* and weighted A*. Unhelkar et al. [28] enabled SIPP to reuse information from previous planning processes. Chen et al. [16] decomposed $\mathrm{SE}(2)\times\mathcal{T}$ into geometric shapes and conducted A* search to plan optimal trajectories. These methods guarantee completeness and optimality in low-dimensional spaces ($e.g.$, the space for robot navigation), but remains computationally challenging to scale to high-dimensional planning problems [12]. Our planner significantly reduces the high-dimensional $\mathcal{C}\times\mathcal{T}$ to a Lazy SISPRM, where searching for optimal trajectories is conducted efficiently within 171ms, a time brief enough to support real-time replanning applications [6].

Other approaches [13] support planning in high-dimensional spaces to avoid obstacles in $\mathcal{C}\times\mathcal{T}$, but have other trade-offs. T-CHOMP jointly optimizes a trajectory coupled with the timestamps of waypoints in $\mathcal{C}\times\mathcal{T}$ via functional gradient descent. The resulting optimization bends a trajectory in order to avoid obstacles and/or varies speed when near obstacles [5]; however, the optimization objectives of T-CHOMP do not include minimizing execution time. In practice, although T-CHOMP shifts the timestamp of each waypoint during optimization, the total execution time is prespecified and fixed [5]. By optimizing for execution time, the Lazy SISPRM planner is able to automatically reason about the "wait/walk dilemma." This refers to the decision about whether to take a long detour to avoid obstacles (such as the red path in Fig. 1) or to stop, wait and take a more direct path (as shown in green in Fig. 1). In contrast, for T-CHOMP, the tradeoff between trajectory smoothness and proximities to obstacles is determined by the user.

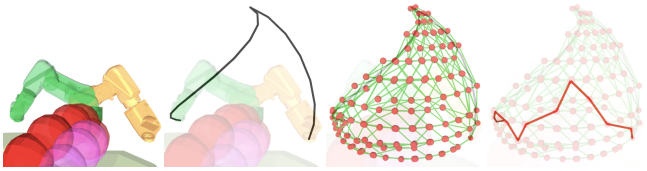
III. Methods

We define our problem as finding a collision-free trajectory with minimal execution time in a short time budget.

A. Problem Definition

Let $\mathcal{C}\subseteq\mathbb{R}^{|\mathcal{C}|}$ denotes the $|\mathcal{C}|$-dimensional robot configuration space, $\mathcal{W}\subseteq\mathbb{R}^3$ denote the 3-dimensional work space, and $\mathcal{T}\subseteq\mathbb{R}$ denote the time domain. We define a robot path ξ with a sequence of n waypoints as $[q_1, q_2, \ldots, q_n]$, where $q_i\in\mathcal{C}$. We define a robot trajectory ζ with n waypoints as the integration between ξ and the timestamp $t\in\mathcal{T}$ associated with each $q\in\xi$, $i.e.$, $\zeta=[(q_1,t_1),(q_2,t_2),\ldots,(q_n,t_n)]$.

Our goal is to find a locally optimal ζ^* that safely carries the robot from the start configuration q_s at $t_1=0$ to the goal configuration q_g with minimal execution time t_n. Unlike previous works, in which t_i is determined by kinodynamic constraints, the Lazy SISPRM planner manipulates t_i to search for trajectories that efficiently avoid dynamic



(a) (b) (c) (d)

Fig. 2: (a) depicts a simulation of the scene shown in Fig. 1. The robot configuration in orange is q_s; the one in green is q_g. The light green volumes near the bottom edge indicate static obstacles (representing the table). The sphere-shaped dynamic obstacle (representing the human hand) first moves toward the robot (as shown in red), then away (as shown in pink). (b) shows a path sample ξ in black planned by RRT-connect [20]. ξ avoids all obstacle volumes across $\mathcal{T}$, as described in Sec. III-B.2. (c) depicts the SPRM constructed from the ξ in (b), as described in Sec. III-B.3. Our Lazy SISPRM is initialized using the SPRM, as described in Sec. III-C. (d) A* is employed to find ζ^* with minimized execution time within Lazy SISPRM, as described in Sec. III-D.

obstacles. Planning in $\mathcal{C} \times \mathcal{T}$ is computationally challenging, particularly for real-time replanning applications. To reduce the computational complexity in planning in $\mathcal{C} \times \mathcal{T}$, our Lazy SISPRM planner reduces the search space via a two-step process before searching for ζ^*. As depicted in Alg. 1, during the first step (lines 1-22), the Lazy SISPRM planner samples a SPRM [29] to represent $\mathcal{C}$, as presented in Sec. III-B. In the second step (lines 24-26), the planner constructs a Lazy SISPRM [17] by aggregating the SPRM with safe intervals [7], as depicted in Sec. III-C.

Algorithm 1: Lazy SISPRM Planning

Input: Start state q_s, Goal state q_g, Dynamic obstacles $\mathcal{O}$, Interpolation resolution res
Output: Trajectory ζ

1 # Sec. III-B.2 Path Sampling:
2 $obsAllTime \leftarrow \emptyset$; # obstacle volumes across $\mathcal{T}$
3 **foreach** $O \in \mathcal{O}$ **do**
4 **foreach** $(o, t) \in Pred(O)$ **do**
5 $obsAllTime = obsAllTime \cup [o]$;
6 **end**
7 **end**
8 $\xi \leftarrow \rho(q_s,\ q_g,\ obsAllTime)$; # Planning using ρ
9 $n = ||\xi||$;
10 # Sec. III-B.3 SPRM Construction:
11 SPRM $\leftarrow \emptyset$;
12 **for** $i \leftarrow 1\ :\ (n-1)$ **do**
13 $pt_i \leftarrow \xi[i]$;
14 SPRM.$AddEdge(pt_i,\ \xi[i+1])$; # Add ξ to SPRM
15 **for** $j \leftarrow (i+1)\ :\ n$ **do**
16 $pt_j \leftarrow \xi[j]$;
17 $\lambda \leftarrow Interp(pt_i,\ pt_j,\ res)$; # A shortcut
18 **for** $k \leftarrow 1\ :\ ||\lambda|| - 1$ **do**
19 SPRM.$AddEdge(\lambda[k],\ \lambda[k+1])$;
20 **end**
21 **end**
22 **end**
23 # Sec. III-C Search Space Reduction via Safe Interval:
24 LazySISPRM $\leftarrow$ SPRM ;
25 # Sec. III-D Optimal Search within Lazy SISPRM:
26 $\zeta^* = SIPP(q_s,\ q_g,\ \mathcal{O},$ LazySISPRM) ;
27 **return** ζ^* ;

B. Search Space Reduction via SPRM

The Lazy SISPRM planner constructs an SPRM to represent $\mathcal{C}$ by querying an obstacle/human motion prediction system and a path sampler.

1) Human Motion Prediction: Our algorithm assumes the use of a motion prediction system that provides a time-parameterized predicted trajectory for each dynamic obstacle within a lookahead time, such as provided by the multiple predictor system (MPS) [6]. Let $\mathcal{O}$ be the set of dynamic obstacles. The prediction of an obstacle $O \in \mathcal{O}$ is a sequence of n_{do} volumes wherein each volume o is associated with a timestamp t — *i.e.*, $pred(O) = [(o_1, t_1), ..., (o_{n_{do}}, t_{n_{do}})]$. Each volume o_i contains a set of position vectors of the points $\in \mathcal{W}$ that compose o_i at t_i. The prediction is referred to as a function $Pred(\cdot)$ at line 4 in Alg. 1.

2) Path Sampling: Given q_s, q_g, and $pred(\cdot)$, the Lazy SISPRM planner queries a motion planner ρ for a feasible path ξ from q_s to q_g. ξ avoids the entire set of obstacle volumes while disregarding the timestamps — *i.e.*, $\forall q \in \xi, \forall O \in \mathcal{O}, \forall (o,t) \in pred(O), s.t.\ RV(q) \cap o = \emptyset$, where RV is a mapping from q to the set of robot volumes $\subseteq \mathcal{W}$. In the event that the Lazy SISPRM planner fails to find a more-efficient ζ^* than ξ later in Sec. III-D, ξ will thus serve as a contingency plan [13, 21]. Path sampling is depicted from line 1 to 8 in Alg. 1. In practice, ρ must be fast to ensure a short planning time for applications that require online replanning [30] in response to changing motion predictions.

3) SPRM Construction: Given a feasible ξ, our planner samples a set of shortcuts [23] along ξ. Intuitively, a shortcut $\lambda_{i,j}$ between two waypoints $q_i, q_j \in \xi$ is a direct path that bridges q_i and q_j, which the robot can take to skip the waypoints along ξ. The Lazy SISPRM planner generates a $\lambda_{i,j}$ for $\forall q_i, q_j \in \xi$ via linear interpolation and collects a set of shortcuts denoted as Λ_ξ based on ξ. Note that we do not limit the interpolation methods; other methods, such as cubic polynomial interpolation, could be employed for smoother motions.Practically, the method and resolution are determined by the computational resource and time budget.

Our planner then constructs a SPRM: an undirected graph where each node n represents a $q \in \lambda \in \Lambda_\xi$, denoted as $n \sim q$, where q doesn't need to be collision free against $pred(O)$ for $\forall O \in \mathcal{O}$. The planner connects $n_i \sim q_i$ and $n_j \sim q_j$ if q_i and q_j are connected in any $\lambda \in \Lambda_\xi$. Our roadmap is *probabilistic* because it is deterministically grown from a randomly sampled initial path ξ [29]. SPRM construction is formulated from line 10 to line 22 in Alg. 1. In addition to Λ_ξ, our planner adds the waypoints with their connections along ξ to the SPRM. Consequently, in the event that the Lazy SISPRM planner fails to return a more efficient ζ^*, the robot can still retrieve the contingency plan ξ from the SPRM. Fig. 2c depicts the process of constructing the SPRM from ξ (Fig. 2b). Each dot in red represents a node, and each line segment in green represents an edge in the SPRM.

C. Search Space Reduction via Safe Interval

At this point, our planner has reduced $\mathcal{C} \times \mathcal{T}$ to SPRM $\times \mathcal{T}$ by sampling and shortcutting a path sample ξ. However, it is still computationally challenging to search in SPRM $\times \mathcal{T}$ due to the potentially long horizon and fine resolution of $\mathcal{T}$. To address this issue, our algorithm further reduces the search space by aggregating safe intervals [7].

A time interval θ is a contiguous period of time during which a robot configuration is collision-free, while the robot configurations before and after θ are not [7]. For each node

$n \in$ SPRM where $n \sim q$, the Lazy SISPRM planner identifies a set of safe intervals, denoted as Θ_q. Each safe interval is defined as $\theta_q = (q, [t, t'])$, with $t, t' \in \mathcal{T}$, wherein q is collision-free within $[t, t']$ but not before t or after t'. The entire set of safe intervals in the SPRM is defined as $\Theta = \cup_{q \sim n \in \text{SPRM}} \Theta_q$. Next, our planner constructs a SISPRM in which each node $n \in$ SISPRM represents a $\theta_q = (q, [t, t']) \in \Theta$, denoted as $n \sim \theta_q$. Two nodes, $n_1 \sim \theta_{q_1} = (q_1, [t_1, t_1'])$ and $n_2 \sim \theta_{q_2} = (q_2, [t_2, t_2'])$, are connected if q_1 and q_2 are connected in the SPRM and if it is safe for the robot to reach q_2 at $t' \in [t_2, t_2']$ from q_1 at $t \in [t_1, t_1']$. In this way, our planner reduces SPRM $\times \mathcal{T}$ to a SISPRM, making the search for ζ^* much more efficient.

However, constructing a SISPRM requires costly collision checking between $RV(q)$ for each $n \in$ SPRM $\sim q$ and all obstacle volumes across $\mathcal{T}$ from the prediction. To reduce the computational burden, our planner takes the "lazy" approach [17]: the planner only initializes the SISPRM with the SPRM, as shown at line 24 in Alg. 1, and defer finding Θ until it becomes necessary during the optimal search for ζ^* within the SISPRM, as discussed in Sec. III-D.

D. Optimal Search within Lazy SISPRM

We adapt SIPP Phillips and Likhachev [7] to search for a ζ^* that safely traverses the Lazy SISPRM between the start node $\theta_{q_s} = (q_s, [0, \cdot])$, and one of the goal nodes $\theta_{q_g} = (q_g, [\cdot, \cdot])$, with the minimal execution time. With the assumption of infinite joint accelerations, the search guarantees optimality and completeness of ζ^* within the Lazy SISPRM [9]. Specifically, we assume the robot follows a straight-line path at a constant speed, similar to Phillips and Likhachev [7] and Chen et al. [16]. The travel time between q_i and q_j, denoted as $\Delta_t(q_i, q_j)$, is computed as $||q_i - q_j||$/joint_vel, where joint_vel is a pre-specified joint velocity within limits. Fig. 2d depicts the ζ^* found by our algorithm, and Fig. 2b depicts the initial, longer and less efficient path sample, ξ.

IV. Experiments

We compared our Lazy SISPRM planner with 8 state-of-the-art algorithms for planning safe and efficient trajectories in dynamic environments. We simulated a collaborative manipulation task during which a robot planned to reach a goal while avoiding a human hand. As depicted in Fig. 3, a human picked up a screw from s^H, reached one of the goals $g_{\{1,2,\ldots,7,8\}}$ for screw placement, and retracted his/her hand back to s^H. Meanwhile, the robot placed sealant onto each screw at $g_{\{1,2,\ldots,7,8\}}$ using a brush attached to its end effector. In the experiment, we fixed the robot start state $q_s = g_7$, the robot goal state $q_g = g_2$, and the human goals $g_{\{3,4,5,6\}}$ to encourage motion conflicts between the human and robot.

We used a dataset of human motions collected via real-time hand tracking as subjects worked with the robot while it executed a fixed sequence of motions to place the sealant [1, 2, 6]. In this dataset, human motion consisted of two short movements: reaching toward one of the goals $g_{\{3,4,5,6\}}$ and retracting back to s^H. Between the reaching and retracting

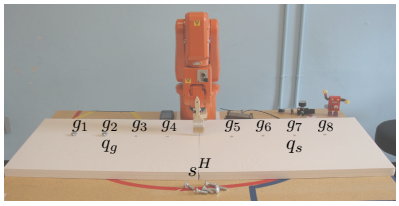


Fig. 3: In this task [2], the human starts from s^H, brings a screw to one of $g_{\{1,2,\ldots,7,8\}}$, and retracts back to s^H. Meanwhile, the robot starts from q_s, and places sealant onto a screw at q_g safely and efficiently.

motions, the human spent 3.76s, on average, with his/her hands stationary as he/she twisted the screws into place. We removed the data frames during which the hand was stationary from the dataset to focus our evaluation on the dynamic aspects of the task. We down-sampled the prediction to a frequency of 10Hz. We included 20 hand trajectories from the dataset for each goal $\in g_{\{3,4,5,6\}}$, yielding 80 trials in total. The average time the person took to reach and retract was together 1.2s. In each trial, each planner would plan robot trajectories to avoid human hands based on the perfect time-parameterized prediction of the hand motion. Similar to [11], we approximated the robot body, the human hand, and the table as spheres to reduce the computational overhead in collision checking, as depicted in Fig. 2a.

We evaluated the performance of the benchmarked algorithms according to the following metrics: (1) Average execution time: This measured the efficiency of a solution, which was one objective the Lazy SISPRM planner was optimizing for. (2) Average planning time: This indicated whether a planner was fast enough to work with motion predictors for replanning. For example, motion predictors with a lookahead time of 0.5s require a planning budget of 0.25s [30]. (3) Average path length (L^2-Norm [31]) in $\mathcal{C}$ among all joints. (4) Success rate. A trajectory was considered successful if there were no collisions along the trajectory and the last waypoint was close to the goal with a tolerance. The experiments were conducted in a machine with Intel i7-7700K, 4.20 GHz CPU and 16 GB RAM.

V. Results

We compared the performance of the Lazy SISPRM planner to 8 variants of state-of-the-art planners across 80 trials, corresponding to the 80 human motions in the dataset. In each trial, the robot planned a trajectory from q_s to q_g while avoiding the human hand and static obstacles, as described in Sec. IV. We set the resolutions for collision checking to 0.15rad and 0.1s in $\mathcal{C}$ and $\mathcal{T}$ for all planners. We used the planner implementation from OMPL [32], ITOMP [33], and STOMP [34]. In the following sections, we discuss the implementation details and the comparison results between the Lazy SISPRM planner and benchmarked planners.

A. The Lazy SISPRM Planner

We chose RRT-connect [20] as our path sampler ρ. RRT-connect is fast, but sometimes generates overly conservative solutions, which Lazy SISPRM is able to further optimize. When constructing the SPRM, we set the interpolation resolution, $res = 0.2$rad, as an input in Alg. 1. The heuristic

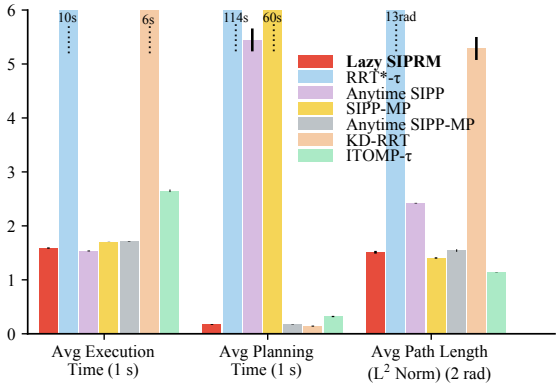


Fig. 4: Avg. execution time, planning time, and path length for each planner.

employed in the A* search (described in Sec. III-D) was computed as the travel time from a state q to the goal q_g in a straight-line path in $\mathcal{C}$. As mentioned in Sec. III-D, we used $\Delta_t(q, q_g)$ with a joint_vel of 1.5rad/s to compute the travel time between q and q_g.

Among the 80 trials, Lazy SISPRM planner succeeded at a rate of 97.50%. The failures were due to the discretization of $\mathcal{C} \times \mathcal{T}$ for collision checking during the A* search. As shown in Fig. 4, the Lazy SISPRM planner found efficient solutions with an average execution time of 1.59s, an average path length of 3.02rad, and an average planning time of 171ms. Note that the planning time includes the time for path sampling, SPRM construction, and search. The path shown in Fig. 5a demonstrates the robot was able to choose an efficient and direct path by manipulating time.

B. Comparisons with Search-Based Methods

1) SIPP: Our method reduces $\mathcal{C}$ to a SPRM, while SIPP [7] and Anytime SIPP [18] search on the full $\mathcal{C}$. We implemented SIPP and Anytime SIPP with the same resolution res and heuristic as mentioned in Sec. V-A, which yielded a discretized $\mathcal{C}$ with $53,745,120$ states. The action space was composed of combinations of six joints' actions, where each joint could move 1, 0, or -1 grid space at a time; consequently, the size of the action space became $3^6 = 729$. Among the 80 trials, SIPP failed to identify solutions within 5min — potentially due to the large state and action space — so we further tested 3 faster variations of SIPP.

2) Anytime SIPP (A-SIPP): A-SIPP [18] ran a series of weighted A* searches with decreasing ϵ. We set the initial $\epsilon = 100$, and evaluated the first solution. Among the 80 trials, although A-SIPP significantly improved the planning speed, the success rate was 57.50% due to timeouts. In Fig. 4, the average execution time was 1.53s, which was slightly (although statistically significantly) lower than that of the Lazy SISPRM planner (1.59s; $p < 0.01$). However, the average path length and planning time of A-SIPP (4.84rad, $5,446$ms) were both significantly larger than those of the Lazy SISPRM planner (3.02rad, 171ms) (both $p < 0.01$) An example of the first solution is depicted in Fig. 5b. These findings indicate that by reducing $\mathcal{C}$ to an SPRM, the Lazy SISPRM planner significantly improved planning time without substantially sacrificing solution quality.

3) SIPP with Motion Primitives (SIPP-MP): SIPP-MP plans with 12 predefined basic primitives and 8 compound primitives [19]. Across the 80 trials, the success rate of SIPP-MP was 98.73%. In Fig. 4, SIPP-MP produced an average path length of 2.81rad, which was significantly lower than that of the Lazy SISPRM planner (3.02rad; $p < 0.01$). However, the average execution time and planning time of SIPP-MP (1.70s and $60,281$ms) were all significantly higher than those of the Lazy SISPRM planner (1.59s and 171ms) (both $p < 0.01$). This indicates that SIPP-MP substantially reduced the branching factor, but planning time was still much slower than with our planner.

4) Anytime SIPP-MP (A-SIPP-MP): To improve the planning speed, we assessed the anytime version of SIPP-MP with an initial $\epsilon = 5$. Across the 80 trials, A-SIPP-MP succeeded 100.00% of the time. A-SIPP-MP and Lazy SISPRM planner had similar planning times (173ms, 171ms) and path lengths (3.08rad, 3.02rad). Anytime SIPP-MP had an average execution time of 1.71s, which was significantly larger than that of the Lazy SISPRM planner (1.59s, $p < 0.01$). These results indicate that reducing the action space through motion primitives compromised the efficiency of the solutions.

C. Comparisons with Sampling-Based Methods

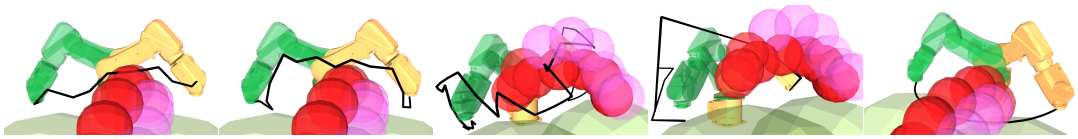
We benchmarked two algorithms from the family of RRT that respectively considered time in implicitly and explicitly.

1) Kinodynamic RRT-τ (KD-RRT-τ): KD-RRT [35] reasons about time implicitly without having time as a dimension. To reduce the curse of dimensionality, our state space was composed of joint positions without velocities. The timestamp for each state was implicitly saved as an internal variable. The planner updated a state's timestamp during state propagation by applying Δ_t as mentioned in Sec. V-A. This planner, referred to as KD-RRT-τ, only considers the obstacle volumes at time τ for a state with a timestamp τ.

Across the 80 trials, KD-RRT-τ succeeded 86.00% of the time. As shown in Fig. 4, KD-RRT-τ had an average planning time of 142ms, which was significantly lower than that of the Lazy SISPRM planner (171ms; $p = 0.02$). The average execution time and path length of KD-RRT-τ (6.49s and 10.5751rad) were both significantly higher than those of the Lazy SISPRM planner (1.59s and 3.02rad) (both $p < 0.01$).

2) RRT-τ:* We adapted RRT* [37] to reason about time explicitly and plan within $\mathcal{C} \times \mathcal{T}$. We implemented a pseudo-metric to give infinite costs to the connections with reversed timestamps. This planner, referred to as RRT*-τ, only considers obstacles at time τ for the states with timestamp τ.

Here we reported the results of the first found solution. Among the 80 trials, 14% of the time, RRT*-τ found a solution within 4min. As shown in Fig. 4, RRT*-τ had an average planning time of $114,008$ms, execution time of 10.47s, and path length of 12.8308rad, all of which were significantly higher than those of the Lazy SISPRM planner (171ms, 1.59s, 3.02rad, respectively) (all $p < 0.01$). RRT*-τ has a long planning time with low efficiency in its solutions, possibly due to the extra dimensionality of the time domain and the time irreversibility constraint.



(a) (b) (c) (d) (e)

Fig. 5: Trajectories planned by the benchmarked planners with the same q_s, q_g, obstacles, and predictions. These trajectories were planned by the following: (a) The Lazy SISPRM planner. (b) A-SIPP (1st solution) [18]. (c) KD-RRT-τ [35]. (d) RRT*-τ (1st solution) [36]. (e) ITOMP-τ [3].

The trajectories produced by the Lazy SISPRM planner (depicted in Fig. 5a) were more efficient in space and time than those generated by KD-RRT-τ (Fig. 5c) and RRT*-τ (Fig. 5d). We cannot apply retimers or smoothers to further improve the solutions of KD-RRT-τ and RRT*-τ, since their collision checking is dependent on timestamps.

D. Comparisons with Functional Gradient Descent Methods

We benchmarked ITOMP [3] without replanning to represent the functional gradient descent (FGD) methods, such as CHOMP [11], T-CHOMP [5], and STOMP [10]. ITOMP parameterizes a trajectory as a set of waypoints spaced equally in time. The execution times are prespecified, and the timestamp of each waypoint ($i.e.$, *pseudo time*) is fixed during optimization [5]. For the sake of fairness, we didn't benchmark ITOMP's execution time.

We developed a variation of ITOMP, *ITOMP-*τ, that only considers obstacles at time τ_i for waypoints with timestamp τ_i. Across the 80 trials, ITOMP-τ succeeded at a rate of 92.50%. In Fig. 4, the average path length of ITOMP-τ was 2.27rad, which was significantly shorter than that of the Lazy SISPRM planner (3.02rad; $p < 0.01$). One reason was that the functional gradient on the smoothness of ITOMP-τ further reduced path length. The average planning time of ITOMP-τ (320ms) was significantly longer than that of the Lazy SISPRM planner (171ms; both $p < 0.01$). Fig. 5e depicts a direct and short trajectory planned by ITOMP-τ, where the robot is able to avoid the obstacles by varying its speed. However, ITOMP-τ took significantly longer planning time, and it was unable to improve execution time.

VI. Discussion And Future Direction

The results from our empirical evaluation demonstrate that the Lazy SISPRM planner has a high success rate and quickly generates collision-free trajectories with short execution times. The Lazy SISPRM planner conducts an optimal search around a path sample, while SIPP and its variations search from scratch within the high dimensional $\mathcal{C} \times \mathcal{T}$. The space reduction to SISPRM increased planning speed by more than 30 times, compared with anytime SIPP. Compared with sampling-based planners, conducting an optimal search in Lazy SISPRM enabled our planner to develop shortcuts for dynamic obstacle avoidance, significantly improving the efficiency and flexibility of solutions in both space and time. Compared with FGD methods for which execution time is prespecified, our method is able to minimize execution time.

FGD methods, as mentioned in Sec. V-D, can produce optimized behaviors wherein the robot first slows down, waits for an obstacle to move out of the way, then speeds up via the objective functionals on obstacles and smoothness [3, 11], as depicted in Fig. 5e. T-CHOMP [5] optimizes both space and time, and could converge more quickly to a desirable trajectory (similar to Fig. 5e); however, FGD methods cannot change the prespecified execution time. Moreover, whether to adjust speed along a direct path or choose a lengthy detour is determined by the weight assignments between the functional objectives on obstacles and smoothness. Similar to optimizing smoothness in $\mathcal{C}$ to implicitly improve path length, one future direction is to develop objectives to implicitly or explicitly improve execution time.

The Lazy SISPRM planner, alongside SIPP [7], requires the assumption of infinite acceleration for optimality and completeness guarantees. To execute trajectories produced by our planner, one would need to employ a quasistatic constraint on the robot. One future work is integrating kinodynamic constraints [14–16] and objectives on smoothness [19] into our Lazy SISPRM to plan time-optimal behaviors for collision avoidance, joint constraints, and smoothness. It would be useful to build reusable PRMs that handle kinodynamic constraints and static obstacles to reduce the online computational burden. Other improvements could be accomplished via using prior planning experiences [38], developing adaptive motion primitives [39] and dimensionality [39], and minimizing the updates to the SPRM [28], and plan [3], shortcut [23], and search [18] in anytime manners.

VII. Conclusion

In this work, we presented the Lazy SISPRM planner, which plans in $\mathcal{C} \times \mathcal{T}$ for high-dimensional manipulators to avoid dynamic obstacles. We demonstrated that our approach outperformed 8 variations of prior state-of-the-art planners in terms of trajectory execution time, path length, and planning time in scenarios from a human-robot interaction dataset. Our planner builds a Lazy SISPRM based on a path sample to represent $\mathcal{C}$, which enables it to plan with an extra dimension $\mathcal{T}$, in a time budget sufficient for real-time replanning. Empirical results indicate that our method was 30 times faster than benchmarked methods that plan in $\mathcal{T}$ without space reduction, and without substantially reducing solution quality. Within the Lazy SISPRM, our method is able to shortcut the path sample and find solutions with a short execution time and path length. Empirical findings show that our method achieved the minimal execution time among all the benchmarked planners with a similar planning speed.

REFERENCES

[1] P. A. Lasota and J. A. Shah, "Analyzing the effects of human-aware motion planning on close-proximity human–robot collaboration," *Human factors*, 2015.

[2] P. A. Lasota, G. F. Rossano, and J. A. Shah, "Toward safe close-proximity human-robot interaction with standard industrial robots," in *CASE*, 2014.

[3] C. Park, J. Pan, and D. Manocha, "Itomp: Incremental trajectory optimization for real-time replanning in dynamic environments." in *ICAPS*, 2012.

[4] J. S. Park, C. Park, and D. Manocha, "Intention-aware motion planning using learning based human motion prediction," in *RSS*, 2017.

[5] A. Byravan, B. Boots, S. S. Srinivasa, and D. Fox, "Space-time functional gradient optimization for motion planning," in *ICRA*, 2014.

[6] P. A. Lasota and J. A. Shah, "A multiple-predictor approach to human motion prediction," in *ICRA*, 2017.

[7] M. Phillips and M. Likhachev, "Sipp: Safe interval path planning for dynamic environments," in *ICRA*, 2011.

[8] A. Kushleyev and M. Likhachev, "Time-bounded lattice for efficient planning in dynamic environments," in *ICRA*, 2009.

[9] J. P. Gonzalez, A. Dornbush, and M. Likhachev, "Using state dominance for path planning in dynamic environments with moving obstacles," in *ICRA*, 2012.

[10] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "Stomp: Stochastic trajectory optimization for motion planning," in *ICRA*, 2011.

[11] M. Zucker, N. Ratliff, A. D. Dragan, M. Pivtoraiko, M. Klingensmith, C. M. Dellin, J. A. Bagnell, and S. S. Srinivasa, "Chomp: Covariant hamiltonian optimization for motion planning," *IJRR*, 2013.

[12] J. Schulman, Y. Duan, J. Ho, A. Lee, I. Awwal, H. Bradlow, J. Pan, S. Patil, K. Goldberg, and P. Abbeel, "Motion planning with sequential convex optimization and convex collision checking," *IJRR*, 2014.

[13] D. Hsu, R. Kindel, J.-C. Latombe, and S. Rock, "Randomized kinodynamic motion planning with moving obstacles," *IJRR*, 2002.

[14] D. Yi, R. Thakker, C. Gulino, O. Salzman, and S. Srinivasa, "Generalizing informed sampling for asymptotically-optimal sampling-based kinodynamic planning via markov chain monte carlo," in *ICRA*, 2018.

[15] Q.-C. Pham, S. Caron, P. Lertkultanon, and Y. Nakamura, "Admissible velocity propagation: Beyond quasi-static path planning for high-dimensional robots," *IJRR*, 2017.

[16] C. Chen, M. Rickert, and A. Knoll, "Kinodynamic motion planning with space-time exploration guided heuristic search for car-like robots in dynamic environments," in *IROS*, 2015.

[17] R. Bohlin and L. E. Kavraki, "Path planning using lazy prm," in *ICRA*, 2000.

[18] V. Narayanan, M. Phillips, and M. Likhachev, "Anytime safe interval path planning for dynamic environments," in *IROS*, 2012.

[19] B. J. Cohen, S. Chitta, and M. Likhachev, "Search-based planning for manipulation with motion primitives," in *ICRA*, 2010.

[20] J. J. Kuffner and S. M. LaValle, "Rrt-connect: An efficient approach to single-query path planning," in *ICRA*, 2000.

[21] K. Hauser, "On responsiveness, safety, and completeness in real-time motion planning," *Autonomous Robots*, 2012.

[22] J. Dong, M. Mukadam, F. Dellaert, and B. Boots, "Motion planning as probabilistic inference using gaussian processes and factor graphs," in *RSS*, 2016.

[23] K. Hauser and V. Ng-Thow-Hing, "Fast smoothing of manipulator trajectories using optimal bounded-acceleration shortcuts," in *ICRA*, 2010.

[24] Q.-C. Pham, "A general, fast, and robust implementation of the time-optimal path parameterization algorithm," *T-RO*, 2014.

[25] T. Kunz and M. Stilman, "Probabilistically complete kinodynamic planning for robot manipulators with acceleration limits," in *IROS*, 2014.

[26] H. Pham and Q.-C. Pham, "A new approach to time-optimal path parameterization based on reachability analysis," *T-RO*, 2018.

[27] K. K. Hauser, "Fast interpolation and time-optimization on implicit contact submanifolds." in *RSS*, 2013.

[28] V. V. Unhelkar, R.-D. Buhai, and J. A. Shah, "Planning in dynamic environments using evolving time-indexed graphs," 2016.

[29] L. Kavraki, P. Svestka, and M. H. Overmars, *Probabilistic Roadmaps For Path Planning In High-dimensional Configuration Spaces*, 1994.

[30] V. V. Unhelkar, P. A. Lasota, Q. Tyroller, R.-D. Buhai, L. Marceau, B. Deml, and J. A. Shah, "Human-aware robotic assistant for collaborative assembly: Integrating human motion prediction with planning in time," *IEEE Robotics and Automation Letters*, 2018.

[31] R. Burger, M. Bharatheesha, M. van Eert, and R. Babuška, "Automated tuning and configuration of path planning algorithms," in *ICRA*, 2017.

[32] I. A. Şucan, M. Moll, and L. E. Kavraki, "The Open Motion Planning Library," *IEEE Robotics & Automation Magazine*, 2012.

[33] "Code for the itomp planner," https://github.com/pjsdream/itomp_exec.

[34] "Ros package of the stomp planner," https://github.com/ros-industrial/industrial_moveit/tree/indigo-devel/stomp_core.

[35] S. M. LaValle and J. J. Kuffner Jr, "Randomized kinodynamic planning," *IJRR*, 2001.

[36] S. Karaman, M. R. Walter, A. Perez, E. Frazzoli, and S. Teller, "Anytime motion planning using the rrt*," in *ICRA*, 2011.

[37] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *IJRR*, 2011.

[38] M. Phillips, B. J. Cohen, S. Chitta, and M. Likhachev, "E-graphs: Bootstrapping planning with experience graphs." in *RSS*, 2012.

[39] B. J. Cohen, G. Subramania, S. Chitta, and M. Likhachev, "Planning for manipulation with adaptive motion primitives," in *ICRA*, 2011.