ST-RRT*: Asymptotically-Optimal Bidirectional Motion Planning through Space-Time

Francesco Grothe[1], Valentin N. Hartmann[1,2], Andreas Orthey[1], Marc Toussaint[1]

Abstract— **We present a motion planner for planning through space-time with dynamic obstacles, velocity constraints, and unknown arrival time. Our algorithm, Space-Time RRT* (ST-RRT*), is a probabilistically complete, bidirectional motion planning algorithm, which is asymptotically optimal with respect to the shortest arrival time. We experimentally evaluate ST-RRT* in both abstract (2D disk, 8D disk in cluttered spaces, and on a narrow passage problem), and simulated robotic path planning problems (sequential planning of 8DoF mobile robots, and 7DoF robotic arms). The proposed planner outperforms RRT-Connect and RRT* on both initial solution time, and attained final solution cost. The code for ST-RRT* is available in the Open Motion Planning Library (OMPL).**

I. Introduction

Motion planning is a fundamental challenge in robotics [1]. In many real-world applications, obstacles change positions over time and goals are only valid at specific times. For applications such as multi-robot assembly, multiple motion scheduling subproblems need to be solved [2]. Assuming that obstacle trajectories are given a priori, the subproblems can be modelled as *navigation through dynamic environments*. Mathematically, this is formulated as planning through a space-time state space.

Efficient and optimal planning through space-time raises three fundamental challenges. First, since goal arrival times are unknown upfront, it becomes difficult, yet crucial, to define and adjust the time range in a coordinated and meaningful way. The second challenge is the representation of kinodynamic constraints in the planning model. Whether a movement is possible depends on kinematic parameters, velocity, and acceleration. Lastly, robots should minimize arrival time. Arrival time is crucial for long-horizon planning problems, where optimization of intermediate arrival times is one of the central challenges [2]. These challenges make planning through space-time a demanding problem. We are not aware of any sampling-based method which either operates in unbounded space-time or is asymptotically optimal with respect to shortest arrival time.

To address those challenges, we develop Space-Time RRT* (ST-RRT*). The basic operating principle of ST-RRT* is illustrated in Fig. 1: (**a**) We compute an initial estimate of a feasible goal time (blue dashed line), and grow both a forward tree from the start state (blue), and a set of reverse trees from the goal regions (red). If no solution is found given a certain number of samples, the upper time limit in which we generate samples is increased (**b**). If a solution (orange) is found (**c**), the parts of the trees that can not lead to an improved solution are pruned. This process continues to improve the solution path, and tighten the upper time bound until a termination condition is reached (**d**).

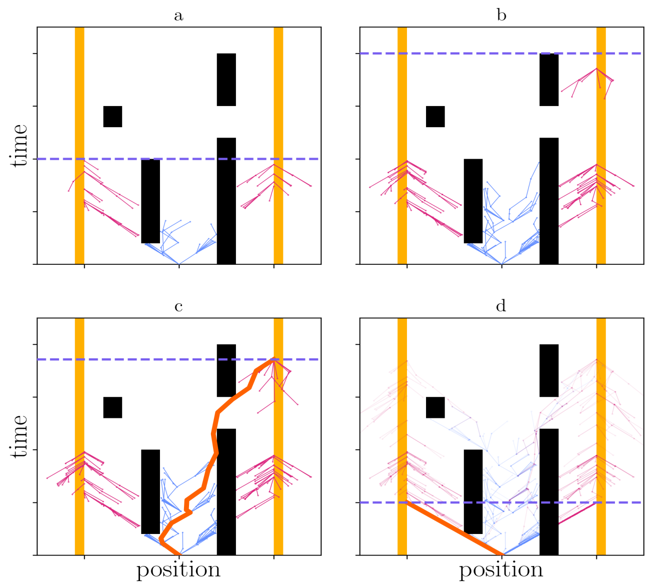


Fig. 1: Four snapshots of ST-RRT* in $\mathbb{R}^{1+1}$ (one space, plus one time dimension). The forward tree is blue, the backward trees are red, obstacles are black, and the goal regions are yellow. (**a**) Using the initial batch of samples, no solution was found. (**b**) The upper bound of the time space (dashed line) is expanded, more goal nodes are sampled and the trees are grown. (**c**) An initial solution is found (orange), and the upper bound is decreased accordingly. (**d**) Parts of the trees that can not contribute to the solution anymore are pruned (lower opacity), and the final solution after convergence.

ST-RRT* is a bidirectional motion planning algorithm that is probabilistically complete and asymptotically optimal with respect to shortest arrival time. ST-RRT* is able to operate in unbounded time spaces and model velocity constraints. ST-RRT* is inspired by RRT-Connect [3], with three changed components to attain the stated qualities in space-time. Our main contributions are:

- *Progressive Goal Region Expansion*: ST-RRT* gradually increases the sampled time range to efficiently operate in unbounded time spaces. Simultaneously, we adjust the sampling densities over the time dimension

The research has been supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy – EXC 2120/1 – 390831618 "IntCDC".

[1]Learning and Intelligent Systems Group, TU Berlin, Germany
[2]Machine Learning & Robotics Lab, University of Stuttgart, Germany
`valentin.hartmann@ipvs.uni-stuttgart.de`

to ensure a more uniform sampling distribution.

- *Conditional Sampling*: We develop a novel sampling method that prevents the sampling of states which cannot be part of a solution path due to velocity constraints.
- *Simplified Rewiring*: To obtain optimal solutions, states are rewired similar to RRT* [4]. In contrast to RRT*, we perform a simplified rewiring step, where only nodes in the set of goal trees are rewired.

We demonstrate our algorithm on both abstract (planning for a disk in up to $\mathbb{R}^{8+1}$), and simulated robotic motion planning problems (both robotic arms and mobile robots).

II. Related Work

In the following two sections, we review literature on planning in dynamic environments and time-optimal path planning. For an exhaustive discussion of path planning methods we refer to [5] and [6] for an overview on (asymptotically optimal) sampling based path planning methods.

A. Planning in Dynamic environments

Planning in dynamic environments can be roughly divided in two approaches. First, we have reactive methods, which work with the assumption that the trajectories of the moving obstacles are unknown, whereas the second category assumes full knowledge of the obstacles' trajectories.

Reactive methods such as Execution-extended RRT [7], Closed-loop RRT [8], [9], RRTX [10] or Real-time RRT* [11] are methods specifically developed for rapid replanning. Rapid replanning is necessary when previously computed paths become invalid during execution. Risk-RRT [12] incorporates predictions about the obstacles' movement, and computes partial motion paths to keep the probability of a collision under a given threshold. However, frequent replanning is still needed as only partial paths are returned. Various methods exist to enable efficient replanning, i.e. to reuse as much prior work as possible from previously planned paths, or to establish coarse connectivity of the space, and only replan for dynamic obstacles ([13], [14], [15], [16]).

Contrary to reactive methods, the following methods assume full knowledge of obstacle trajectories, and thus do not rely on replanning. Time-Based RRT [17] expands the configuration state space by the time dimension and plans unidirectionally to a set of known goal states. However, knowledge of the specific time for each goal configuration is assumed, and only unidirectional planning is supported. Safe Interval Path planning [18] finds optimal paths with respect to shortest time by constructing a discrete search space with states defined by their configuration and a corresponding 'safe interval'. However, a graph needs to be constructed for the entire state space, and thus it suffers the inherent problems: it is only feasible for problems with few dimensions.

In this work, we assume full knowledge of all paths of the moving obstacles, but no a priori knowledge of the arrival time, as is the case in multi-robot assembly planning tasks [2]. Thus, our method does not require replanning and is able to efficiently find feasible and time-optimal paths. Our method also enables to plan bidirectionally in unbounded time spaces, leading to a more efficient planner than other RRT-based planners in the space-time setting.

B. Time-optimal Trajectory planning

A common approach to find kinodynamically feasible paths is based on path-velocity decomposition: first find a geometrically feasible path, and then find a valid time-parametrization for this path [19]. Extensions to this approach were presented e.g. in [20], which relaxes the quasi-static requirement. However, this approach is inapplicable here, as obstacles are dynamic and the time optimization on a fixed path might render it infeasible.

Other approaches to planning include optimization approaches (e.g. STOMP [21], or sequential convex optimization [22]), or extending the configuration space with velocity coordinates [23]. Optimization based approaches work well to incorporate complex constraints, but suffer from the well known non-convexity of the general planning problem. Furthermore, optimizing for arrival time is not straightforward. In general, these methods are not complete and therefore can not achieve global optimality.

Sampling based kinodynamic planning on the other hand, doubles the dimensionality of the state space we plan in, and thus makes planning with high DoF-robots slow or even infeasible. Since time is not taken into account explicitly, planning with dynamic obstacles is not straightforward.

By extending the configuration space with a time component, and planning and optimizing in this space-time state space, we retain these guarantees. Through usage of bidirectional planning, conditional sampling, and simplified rewiring, we achieve a high efficiency.

III. The Space-Time RRT* Algorithm

We consider the motion planning problem in space-time with unbounded arrival time. Our objective is to minimize arrival time under given velocity constraints. By adding a time dimension to the configuration space we obtain the *Space-Time* state-space $\mathcal{X} = \mathcal{Q} \times \mathcal{T}$, where $\mathcal{Q}$ is the underlying configuration state space and $\mathcal{T}$ is the time state space. Note that $\mathcal{X}$ can be unbounded in time. Let $\mathcal{X}_{\text{free}} \in \mathcal{X}$ be the obstacle-free subset of states, x_{start} the start state, and $\mathcal{X}_{\text{goal}} = \mathcal{Q}_{\text{goal}} \times \mathcal{T}_{\text{goal}}$ the goal region. In the following, we assume full knowledge of the obstacles' trajectories, and plan for holonomic robots with a given maximum velocity. We define $v_{\max} \in \mathbb{R}^{|\mathcal{Q}|}$ as a vector containing the maximum velocity for each space component.

The goal is to compute a continuous path $p : [0,1] \to \mathcal{X}_{\text{free}}$, such that $p(0) = x_{\text{start}}$, $p(1) \in \mathcal{X}_{\text{goal}}$, and the velocity constraints are satisfied. We are interested in finding not only feasible, but paths which minimize the arrival-time, $c(p) = t_1$ with t_1 being the time element of $p(1) = (q_1, t_1)$.

In space-time, the distance that can be covered in a given time is constrained by $v_{\max}$ and it is not possible to move backwards in time. Thus, we define our distance function d

Algorithm 1 ST-RRT*

```
Input: $\mathcal{X}, x_{\text{start}}, \mathcal{X}_{\text{goal}}, d, \text{PTC}, t_{\max}, p_{\text{goal}}, P$
 1: $T_a \leftarrow \{x_{\text{start}}\}$;  $T_b \leftarrow \emptyset$
 2: $B \leftarrow$ INITIALIZEBOUNDVARIABLES$(P)$
 3: while $\neg$PTC do
 4:     $B \leftarrow$ UPDATEGOALREGION$(B, P, t_{\max})$
 5:     if $p_{\text{goal}} \geq$ RND$(0,1)$ then
 6:         $B \leftarrow$ SAMPLEGOAL$(x_{\text{start}}, \mathcal{X}_{\text{goal}}, T_{\text{goal}}, t_{\max}, B)$
 7:     $x_{\text{rand}} \leftarrow$ SAMPLECONDITIONALLY$(x_{\text{start}}, \mathcal{X}, B, d)$
 8:     if not EXTEND$(T_a, x_{\text{rand}}, d) = Trapped$ then
 9:         $B.samplesInBatch$ += 1
10:         $B.totalSamples$ += 1
11:         REWIRETREE$(T_a, T_{\text{goal}}, x_{\text{new}})$
12:         if CONNECT$(T_b, x_{\text{new}}, d) = Reached$ then
13:             $solution \leftarrow$ UPDATESOLUTION$(x_{\text{new}})$
14:             $t_{\max} \leftarrow$ COSTPATH$(solution)$
15:             $B.batchProbability \leftarrow 1$
16:             PRUNETREES$(t_{\max}, T_a, T_b)$
17:     SWAP$(T_a, T_b)$
18: return $solution$
```

between two states, $x_1 = (q_1, t_1)$ and $x_2 = (q_2, t_2)$ as

$$d(x_1,x_2)=\begin{cases}\lambda d_{\mathcal{Q}}(q_1,q_2)+(1-\lambda)(t_2-t_1), & \\ \quad \text{if } t_1 < t_2,\ v^i \leq v^i_{\max}\ \forall i \in [1,|\mathcal{Q}|] & (1)\\ \infty, & \text{else.}\end{cases}$$

where $d_{\mathcal{Q}}$ is the intrinsic metric of the configuration space, $\lambda \in (0,1)$ weights the importance of $d_{\mathcal{Q}}$ with respect to the time-distance (but does not influence optimality), and v^i is the required speed in dimension i, such that q_2 can be reached from q_1 in time $t_2 - t_1$. As d is not symmetric, it is only a pseudometric.

A. Algorithm

The algorithmic details of ST-RRT* are shown in Algorithms 1–5. In addition to $\mathcal{X}$, x_{start}, $\mathcal{X}_{\text{goal}}$, and d it requires a planner termination condition PTC, a time bound $t_{\max} \in (0, \infty]$, a probability to sample a new goal $p_{\text{goal}} \in (0,1]$, and several bound parameters contained in P (see Section III-A.1). The basic framework is similar to RRT-Connect [3]: In each iteration a new goal is sampled with probability p_{goal} (Line 5 & 6). Then, a random state x_{rand} is sampled (Line 7). If possible, the current tree T_a is expanded by the new state x_{new} (i.e. the extension between x_{near} and x_{rand}) and a connection from x_{new} to the other tree T_b is attempted (Line 8 & 12). In case of a successful connection, the solution is updated (Line 13). Finally, T_a and T_b are swapped and the next iteration begins (Line 17). Our extensions to RRT-Connect are:

- *Progressive Goal Region Expansion*, which progressively enlarges the time component of the space (Line 4), and samples new goals for the goal tree (Line 6),
- *Conditional Sampling* (Line 7), which first samples a state from $\mathcal{Q}$, and then samples a corresponding valid time, with which x_{rand} is constructed, and

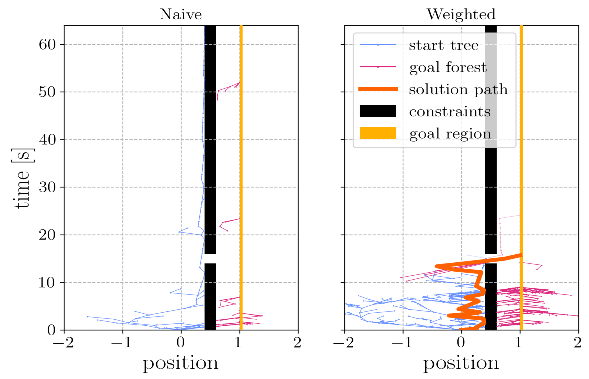


Fig. 2: Illustration of the search trees after the same computation time with naive and weighted sampling strategy with similar numbers of samples (for naive sampling, not all samples are visible, and the time bound was increased beyond the shown range).

- *Simplified Rewiring*, which improves the solution (Line 11) by optimizing for minimal arrival time.

We also prune the trees (Line 16) to remove parts which cannot improve the solution anymore.

1) Progressive Goal Region Expansion: If the time-space $\mathcal{T}$ is unbounded it is difficult to generate samples distributed throughout the whole space. However, when imposing an arbitrary time-bound, the problem might become infeasible [24], [25]. Therefore, we expand the sampled goal region progressively whenever a new batch of samples is added. To do that, we introduce several parameters contained in the bound struct B: B.*timeRange* determines the time bound for goal sampling and B.*batchSize* determines after how many generated samples the expansion takes place. When a batch is full, B.*timeRange* is increased by P.*rangeFactor* and B.*batchSize* is increased accordingly.

With an increasing time-bound, the sample density is higher at the lower time values due to the previously generated samples. Figure 2 shows how naive sampling may lead to cases where it becomes increasingly unlikely to find any solution. Thus we use weighted sampling, where the old and newly expanded region are explicitly sampled with probability B.*batchProbability* and $1 - B$.*batchProbability*, respectively, to ensure a uniform distribution over the total space.

Precisely, the *Progressive Goal Region Expansion* works as follows: The parameters P.*rangeFactor*, P.*initialBatchSize*, and P.*sampleRatio* are user-specified. All variables of B are initialized at the start (Algorithm 2) and updated during execution. While B.*timeRange* is used when the current goal region is sampled, B.*newTimeRange* is used to sample the newly expanded one. After the first expansion, B.*newTimeRange* is always higher than B.*timeRange* by a factor equal to P.*rangeFactor* (Alg. 3, Line 2 & 3). The minimum amount of the new batch size is given by $(P$.*rangeFactor* $- 1) \cdot B$.*totalSamples*. That is, when all samples of the new batch are placed in the new region, the overall distribution would be uniform over the time-space. To ensure that the old region

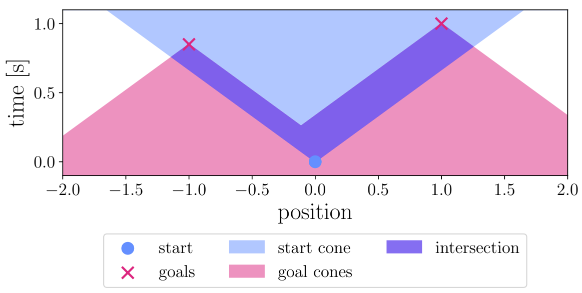


Fig. 3: The start and goal cones contain all states that can be reached from the start or can reach the goal respectively. The intersection contains all states that can be part of a solution.

is also sampled, $B.batchSize$ is further increased by $P.sampleRatio \in (0, 1)$ (Line 4). The probability to sample the old batch $B.batchProbability$ is calculated in dependence of $P.rangeFactor$ and $P.sampleRatio$ (Line 5). Due to the exponential growth of the batch size, the choice of the configuration parameters is important for performance.

To sample a goal state, its space component q is sampled first (Alg 4, Line 1). The lower and upper bounds for the time, t_{lb} and t_{ub}, are calculated in dependence on whether the time is explicitly bounded (Line 4), the current region is sampled (Line 6), or the newly expanded one is sampled (Line 8). The sampling of nongoal-states is subject to the sampled goal states and therefore implicitly bounded by the time value of the sampled goal states (Section III-A.2).

Algorithm 2 InitializeBoundVariables

```
Input: $P$
1: $B.timeRange \leftarrow P.rangeFactor$
2: $B.newTimeRange \leftarrow P.rangeFactor$
3: $B.batchSize \leftarrow P.initialBatchSize$
4: $B.samplesInBatch \leftarrow 0$;   $B.totalSamples \leftarrow 0$
5: $B.batchProbability \leftarrow 1$
6: $B.goals \leftarrow \emptyset$;   $B.newGoals \leftarrow \emptyset$
7: return $B$
```

Algorithm 3 UpdateGoalRegion

```
Input: $B, P, t_{\max}$
1: if $t_{\max} = \infty$ and $B.samplesInBatch = B.batchSize$ then
2:     $B.timeRange \leftarrow B.newTimeRange$
3:     $B.newTimeRange \mathrel{*}= P.rangeFactor$
4:     $B.batchSize \leftarrow \frac{(P.rangeFactor-1)B.totalSamples}{P.sampleRatio}$
5:     $B.batchProbability \leftarrow \frac{1-P.sampleRatio}{P.rangeFactor}$
6:     $B.goals \leftarrow B.goals \cup B.newGoals$
7:     $B.newGoals \leftarrow \emptyset$;   $B.samplesInBatch \leftarrow 0$
8: return $B$
```

2) Conditional Sampling: Any state that can be part of a solution path must have a finite distance d to the start and at least one goal state. Due to velocity-constraints, only states in the intersection of the start and goal cones (see Fig. 3 for an illustration) meet this requirement. Thus, similar to Informed RRT* [26], we only sample the region that can produce solutions. Ideally, one would sample directly from the union of intersections of start and goal velocity-cones.

Algorithm 4 SampleGoal

```
Input: $x_{\text{start}}, \mathcal{X}_{\text{goal}}, T_{\text{goal}}, t_{\max}, B$
1: $q \leftarrow$ SAMPLEUNIFORM($\mathcal{Q}_{\text{goal}}$)
2: $t_{\min} \leftarrow$ LOWERBOUNDARRIVALTIME($q_{\text{start}}, q$)
3: SAMPLEOLDBATCH $\leftarrow$ RND$(0,1) \leq B.batchProbability$
4: if $t_{\max} \neq \infty$ then
5:     $t_{\text{lb}} \leftarrow t_{\min}$;   $t_{\text{ub}} \leftarrow t_{\max}$
6: else if SAMPLEOLDBATCH then
7:     $t_{\text{lb}} \leftarrow t_{\min}$;   $t_{\text{ub}} \leftarrow t_{\min} \cdot B.timeRange$
8: else
9:     $t_{\text{lb}} \leftarrow t_{\min} \cdot B.timeRange$
10:    $t_{\text{ub}} \leftarrow t_{\min} \cdot B.newTimeRange$
11: if $t_{\text{ub}} > t_{\text{lb}}$ then
12:    $t \leftarrow$ SAMPLEUNIFORM($t_{\text{lb}}, t_{\text{ub}}$)
13:    $T_{\text{goal}} \leftarrow T_{\text{goal}} \cup \{(q, t)\}$
14:    if SAMPLEOLDBATCH then
15:        $B.goals \leftarrow B.goals \cup \{(q, t)\}$
16:    else
17:        $B.newGoals \leftarrow B.newGoals \cup \{(q, t)\}$
18:    return $B$
```

However, as the explicit computation of the intersection is not possible for multiple goal states, we use *Conditional Sampling*: We first uniformly sample a configuration q (Alg 5, Line 2). Using q, we then sample a feasible time from the range of possible times conditioned upon q. The range of possible times is dependent on x_{start} and the previously sampled goal states. To sample more uniformly, we use two goal sets: $B.goals$ for the current goal states and $B.newGoals$ for the goal states in the newly expanded region. The time bounds t_{lb}, t_{ub} are obtained by the minimal arrival time from the start configuration q_{start} until q (Line 3) and the maximum valid time given by:

$$\text{MAXVALIDTIME}(q_{\text{rnd}}, G) = \max_{(q_g, t_g) \in G} \left(t_g - \min_i \frac{d_{\mathcal{Q}}(q^i_{\text{rnd}}, q^i_g)}{v^i_{\max}} \right) \quad (2)$$

The specific calculation of t_{lb}, t_{ub} is dependent on whether the current (Line 4) or the new region is sampled (Line 7).

3) Simplified Rewiring: To compute time-optimal solutions ST-RRT* uses similar methods as RRT* and preserves its property of asymptotic optimality. Equal to RRT*, ST-RRT* tries to rewire a set of states near to the newly added state, x_{new}, after tree expansion. Contrary to RRT*, rewiring is only performed in the goal trees. This is due to the fact that rewiring nodes in the start tree can never lead to a better arrival time in the path. Rewiring states in the start tree can not change their arrival time, whereas in the goal trees a node can be rewired to a root node with a smaller time value. One more deviation is the check of which nodes should be rewired. For all nodes in the goal trees simply the time value of their respective root node has to be considered.

Algorithm 5 SampleConditionally

```
Input: $x_{\text{start}}, \mathcal{X}, B$
 1: repeat
 2:     $q \leftarrow$ SampleUniform($\mathcal{Q}$)
 3:     $t_{\min} \leftarrow t_{\text{start}}+$LowerBoundArrivalTime($q_{\text{start}}, q$)
 4:     if Random$(0,1) < B.batchProbability$ then
 5:         $t_{\text{lb}} \leftarrow t_{\min}$
 6:         $t_{\text{ub}} \leftarrow$ MaxValidTime($q, B.goals$)        ▷ eq (2)
 7:     else
 8:         $t^*_{\min} \leftarrow$ MaxValidTime($q, B.goals$)
 9:         $t_{\text{lb}} \leftarrow$ Max($t_{\min}, t^*_{\min}$)
10:         $t_{\text{ub}} \leftarrow$ MaxValidTime($q, B.newGoals$)
11: until $t_{\text{lb}} < t_{\text{ub}}$
12: $t \leftarrow$ SampleUniform($t_{\text{lb}}, t_{\text{ub}}$)
13: return $(q, t)$
```

B. Proof Sketches

To prove probabilistic completeness in space-time, we distinguish between two cases. In case of bounded time, planning with a quasi-metric reverts to kinodynamic planning, where we refer to results from [27] and [28] for completeness proofs.

The second case is unbounded time: If a solution exists, there needs to be a feasible goal region at a finite time. Since we iteratively increase the upper bound, we will, eventually, have increased the goal region to include the feasible goal region. Due to the use of uniform sampling of the time range, there will be positive probability that the feasible goal region will be sampled. Since *conditional sampling* always gives a positive probability of sampling any open set, this makes ST-RRT* retain probabilistic completeness [29].

Apart from probabilistic completeness, ST-RRT* is also asymptotically optimal with respect to arrival time. Since ST-RRT* is modelled after RRT-Connect, it can be made asymptotically optimal by tree rewiring [4], [30]. Inside the rewiring step, we connect newly added states to the nearest goal tree which minimizes arrival time. This ensures asymptotic optimality with respect to final arrival time.

IV. Evaluation

We compared ST-RRT* to other planners on 4 different scenarios using the benchmarking capabilities of OMPL [31]. All evaluations were performed over 100 runs with different pseudorandom seeds of 30s each (if not stated otherwise). ST-RRT* is compared to RRT-Connect[1] and RRT* in space-time using their OMPL implementations in default configuration. Since RRT* and RRT-Connnect algorithms can not operate on unbounded time, three different time bounds are measured. The lowest time bound was determined according to the best solutions of ST-RRT* and set to a higher value to ensure feasibility. Without knowing a solution this is generally not possible. For planning through Space-Time, most of the planners in OMPL [32] do not work either due to only working with metric spaces, only working with euclidean spaces, not supporting asymmetric distance function (e.g. due to using undirected graph structures), or were never able to find solutions in the specified runtime.

[1]The metric had to be changed to be symmetric for distance calculation, but remained as stated for motion validation (this change did not help in the other planners).

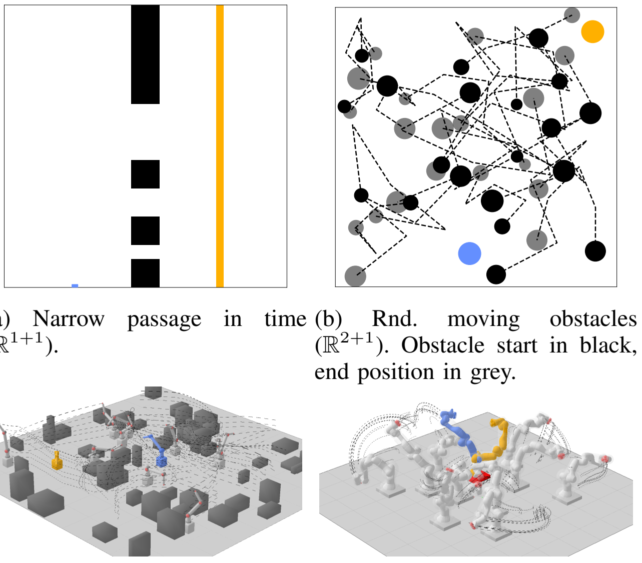

(a) Narrow passage in time ($\mathbb{R}^{1+1}$). (b) Rnd. moving obstacles ($\mathbb{R}^{2+1}$). Obstacle start in black, end position in grey. (c) Mobile robots. (d) Robotic arms.

Fig. 4: Illustrations of the scenarios: Starts are shown in blue, goals and goal regions in yellow, and obstacles in black. The dashed lines are the paths of the moving obstacles.

A. Scenarios

We evaluate the method on the following scenarios[2]:

(i) *Narrow passage*: A point has to move from start configuration q_0 to goal configuration q_F in an environment where the configuration space is split into two parts by an obstacle up to a certain point in time except for three narrow periods of time (Fig. 4a).

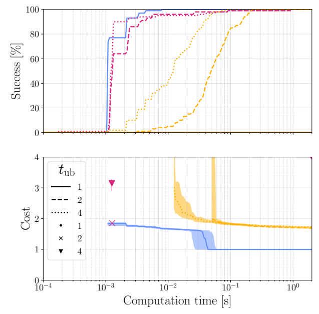
(ii) *Cluttered space*: A (hyper-)sphere has to move from q_0 to q_F in an environment with randomly moving obstacles (Fig. 4b).

(iii) *Sequential mobile robot planning*: A robot with a mobile base and a robot arm on top ($\mathbb{R}^8$) has to move from q_0 to q_F in an environment with randomly distributed obstacles, and other moving mobile robots that move on a fixed trajectory (Fig. 4c). This is a common subproblem in prioritized multi robot planning [33].

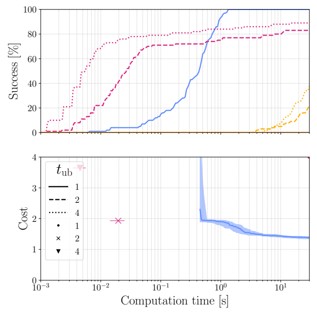
(iv) *Sequential robot arm planning*: A robotic arm ($\mathbb{R}^7$) has to move from configuration q_0 to q_F in an environment with previously planned panda robotic arms (Fig. 4d). Such a scenario may arise in e.g. simultaneous bin-picking with multiple robots.

We show the narrow passage problem in $\mathbb{R}^{1+1}$ and $\mathbb{R}^{8+1}$ and the cluttered env. in $\mathbb{R}^{2+1}$ and $\mathbb{R}^{8+1}$. For the robotic settings, we test the planners in the 6th and the 11th agent (i.e. the previous 5, and 10 agents, respectively, already have a trajectory).

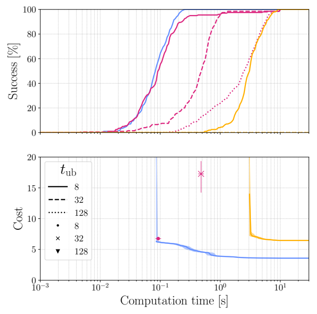
[2]Videos of the scenarios, and the paths are in the supplementary material.



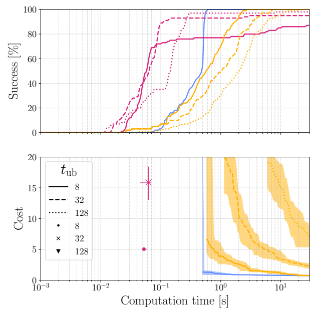
(a) Narrow passage in time: $\mathbb{R}^{1+1}$



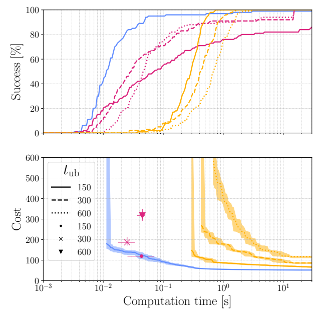
(b) Narrow passage in time: $\mathbb{R}^{8+1}$



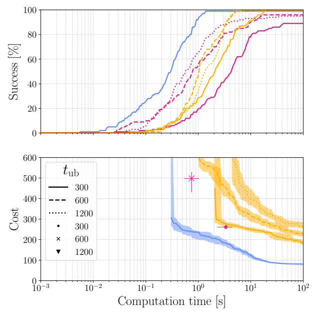
(c) Rnd. moving obstacles: $\mathbb{R}^{2+1}$



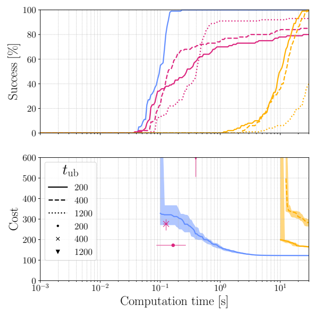
(d) Rnd. moving obstacles: $\mathbb{R}^{8+1}$



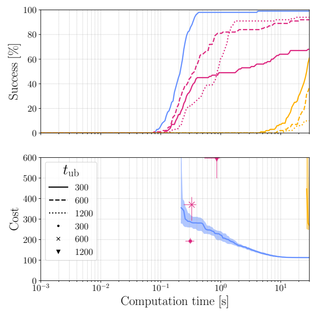
(e) Mobile robots: 6th agent



(f) Mobile robots: 11th agent: 100s



(g) Robot arms: 6th arm



(h) Robot arms: 11th arm

Fig. 5: Success rates and cost plots for the experiments (Section IV-A) for 🟦 ST-RRT*, 🟥 RRT-Connect, and 🟧 RRT* over 100 runs. RRT-Connect and RRT* were run with 3 different upper bounds, t_{ub} for the time (indicated in the figure), since they can not operate in unbounded time-spaces. The thick line is the median, and the shaded area in the cost plot shows the 95% nonparametric confidence interval. Cost for RRT-Connect is shown as the median with error bars for the 95% nonparametric confidence interval. Unsuccessful runs are treated as infinite cost. The upper time limits for RRT* and RRT-Connect are listed in the figures. Planners that are not shown were not able to find any solution in the given time.

B. *Experimental Results*

We analyze the results of both the abstract experiments (Fig. 5a - Fig. 5d), and the simulated robot experiments (Fig. 5e - Fig. 5h). We compare the success rates and the cost-convergence plots of the different algorithms.

1) Initial solution time: In almost all cases the median initial solution time of ST-RRT* is lower than for both RRT-Connect and RRT*, even with the tightest time-bound. This can be attributed to the conditional sampling, which helps avoid exploring areas that are clearly not reachable.

2) Success Rate: A low time bound helps to more quickly find solutions for RRT-Connect and RRT*; however, it can lead to the inability to find solutions at all. This is especially problematic for RRT-Connect which stops sampling goal states at some point, leading to RRT-Connect sometimes not reaching 100% success rate even though the time bound is specified such that a solution would be attainable.

3) Cost: ST-RRT* converges to the best found solution more quickly than RRT*. Additionally, while the initial cost of the solution of ST-RRT* is sometimes higher than RRT-Connect's solution, the final solution cost of ST-RRT* is in all cases lower or equal than for the other methods.

Summarizing the results, a special treatment of the time-space is clearly necessary in a planner to achieve good performance in the motion planning process and ST-RRT* outperforms the other planners on the tested problems.

V. Conclusion

We proposed ST-RRT*, a planning algorithm that is able to efficiently deal with unbounded time spaces and optimizes for arrival time in an environment with moving obstacles on known trajectories. We guarantee probabilistic completeness and asymptotic optimality by introducing progressive expansion of the goal space and generate new samples accordingly. Our algorithm efficiently deals with many goals and converges to the optimal path quickly by making use of conditional sampling and shrinking the goal spaces.

The current implementation of ST-RRT* still has two limitations: the batch size and the expansion factor must be chosen in the beginning with a crude estimate of when the goal can be reached. In practice this is not a large limitation since real settings usually impose some upper limit on the acceptable maximum time to reach a goal state. Additionally, acceleration and more complex kinodynamic constraints (e.g. torque limits) are not taken into account. While this does not pose a problem in our applications, it would not be applicable to robots which have to be in quasi-static equilibrium.

We experimentally demonstrated that ST-RRT* scales well to high dimensions on both abstract and simulated robotic experiments. Our algorithm outperforms state of the art algorithms on both initial solution time and convergence to the optimal solution. An initial version of ST-RRT* was used in work on large-scale multi-robot coordination [2].

References

[1] B. Siciliano and O. Khatib, *Springer handbook of robotics*. Springer, 2016.

[2] V. N. Hartmann, A. Orthey, D. Driess, O. S. Oguz, and M. Toussaint, "Long-horizon multi-robot rearrangement planning for construction assembly," *ArXiv*, vol. abs/2106.02489, 2021.

[3] J. J. Kuffner and S. M. LaValle, "Rrt-connect: An efficient approach to single-query path planning," in *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, vol. 2, 2000, pp. 995–1001.

[4] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[5] 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.

[6] M. Elbanhawi and M. Simic, "Sampling-based robot motion planning: A review," *Ieee access*, vol. 2, pp. 56–77, 2014.

[7] J. Bruce and M. M. Veloso, "Real-time randomized path planning for robot navigation," in *Robot Soccer World Cup*. Springer, 2002, pp. 288–295.

[8] B. D. Luders, S. Karaman, E. Frazzoli, and J. P. How, "Bounds on tracking error using closed-loop rapidly-exploring random trees," in *American Control Conference*, 2010, pp. 5406–5412.

[9] Y. Kuwata, J. Teo, S. Karaman, G. Fiore, E. Frazzoli, and J. How, "Motion planning in complex environments using closed-loop prediction," in *AIAA Guidance, Navigation and Control Conference and Exhibit*, 2008, p. 7166.

[10] M. Otte and E. Frazzoli, "Rrtx: Asymptotically optimal single-query sampling-based motion planning with quick replanning," *International Journal of Robotics Research*, vol. 35, no. 7, pp. 797–822, 2016.

[11] K. Naderi, J. Rajamäki, and P. Hämäläinen, "Rt-rrt* a real-time path planning algorithm based on rrt," in *ACM SIGGRAPH Conference on Motion in Games*, 2015, pp. 113–118.

[12] C. Fulgenzi, A. Spalanzani, C. Laugier, and C. Tay, "Risk based motion planning and navigation in uncertain dynamic environment," Research Report, Oct. 2010.

[13] L. Jaillet and T. Siméon, "A prm-based motion planner for dynamically changing environments," in *IEEE International Conference on Intelligent Robots and Systems*, vol. 2, 2004, pp. 1606–1611.

[14] Y. Yang and O. Brock, "Elastic roadmaps—motion generation for autonomous mobile manipulation," *Autonomous Robots*, vol. 28, no. 1, p. 113, 2010.

[15] D. Ferguson, N. Kalra, and A. Stentz, "Replanning with rrts," in *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2006, pp. 1243–1248.

[16] M. Zucker, J. Kuffner, and M. Branicky, "Multipartite rrts for rapid replanning in dynamic environments," in *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2007, pp. 1603–1609.

[17] A. Sintov and A. Shapiro, "Time-based rrt algorithm for rendezvous planning of two dynamic systems," in *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2014, pp. 6745–6750.

[18] M. Phillips and M. Likhachev, "Sipp: Safe interval path planning for dynamic environments," in *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2011, pp. 5628–5635.

[19] K. Kant and S. W. Zucker, "Toward efficient trajectory planning: The path-velocity decomposition," *International Journal of Robotics Research*, vol. 5, no. 3, pp. 72–89, 1986.

[20] Q.-C. Pham, S. Caron, P. Lertkultanon, and Y. Nakamura, "Admissible velocity propagation: Beyond quasi-static path planning for high-dimensional robots," *International Journal of Robotics Research*, vol. 36, no. 1, pp. 44–67, 2017.

[21] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "Stomp: Stochastic trajectory optimization for motion planning," in *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2011, pp. 4569–4574.

[22] 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," *International Journal of Robotics Research*, vol. 33, no. 9, pp. 1251–1270, 2014.

[23] D. J. Webb and J. Van Den Berg, "Kinodynamic rrt*: Asymptotically optimal motion planning for robots with linear dynamics," in *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2013, pp. 5054–5061.

[24] 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 *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2015, pp. 3067–3074.

[25] J. D. Gammell, "Informed anytime search for continuous planning problems," Ph.D. dissertation, University of Toronto, Feb. 2017.

[26] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Informed RRT*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic," in *Proc. of the IEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2014, pp. 2997–3004.

[27] M. Kleinbort, K. Solovey, Z. Littlefield, K. E. Bekris, and D. Halperin, "Probabilistic completeness of rrt for geometric and kinodynamic planning with forward propagation," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. x–xvi, 2018.

[28] L. Janson, E. Schmerling, A. Clark, and M. Pavone, "Fast marching tree: A fast marching sampling-based method for optimal motion planning in many dimensions," *International Journal of Robotics Research*, vol. 34, no. 7, pp. 883–921, 2015.

[29] L. Janson, B. Ichter, and M. Pavone, "Deterministic sampling-based motion planning: Optimality, complexity, and performance," *International Journal of Robotics Research*, vol. 37, no. 1, pp. 46–61, 2018.

[30] O. Salzman and D. Halperin, "Asymptotically near-optimal RRT for fast, high-quality motion planning," *IEEE Transactions on Robotics*, vol. 32, no. 3, pp. 473–483, 2016.

[31] M. Moll, I. A. Şucan, and L. E. Kavraki, "Benchmarking motion planning algorithms: An extensible infrastructure for analysis and visualization," *IEEE Robotics and Automation Magazine*, vol. 22, no. 3, pp. 96–102, September 2015.

[32] I. A. Şucan, M. Moll, and L. E. Kavraki, "The Open Motion Planning Library," *IEEE Robotics and Automation Magazine*, vol. 19, no. 4, pp. 72–82, December 2012.

[33] A. Orthey, S. Akbar, and M. Toussaint, "Multilevel motion planning: A fiber bundle formulation," 2020, arXiv:2007.09435 [cs.RO].