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

Francesco Grothe, Valentin N. Hartmann, Andreas Orthey, Marc Toussaint

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 time spaces 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).

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...
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 $\mathcal{X} = \mathcal{Q} \times T$, where $\mathcal{Q}$ is the underlying configuration state space and $T$ is the time state space. Note that $\mathcal{X}$ can be unbounded in time. Let $\mathcal{X}_{\text{free}} \subseteq \mathcal{X}$ be the obstacle-free subset of states, $x_{\text{start}}$ the start state, and $\mathcal{X}_{\text{goal}} = \mathcal{Q}_{\text{goal}} \times T_{\text{goal}}$ 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 $v_{\text{max}} \in \mathbb{R}^{|\mathcal{Q}|}$ as a vector containing the maximum velocity for each space component.

The goal is to compute a continuous path $p : [0, 1] \rightarrow \mathcal{X}_{\text{free}}$, 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 $v_{\text{max}}$ and it is not possible to move backwards in time. Thus, we define our distance function $d$
between two states, \( x_1 = (q_1, t_1) \) and \( x_2 = (q_2, t_2) \) as

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

where \( d_Q \) is the intrinsic metric of the configuration space, \( \lambda \in (0, 1) \) weights the importance of \( d_Q \) 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 \( X \), \( x_{\text{start}}, X_{\text{goal}}, \) and \( d \) it requires a planner termination condition \( \text{PTC} \), a time bound \( t_{\text{max}} \in (0, \infty) \), 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 \( p_{\text{goal}} \) (Line 3) & 4). Then, a random state \( x_{\text{rand}} \) is sampled (Line 5). If possible, the current tree \( T_a \) is expanded by the new state \( x_{\text{new}} \) (i.e., the extension between \( x_{\text{rand}} \) and \( x_{\text{new}} \)) and a connection from \( x_{\text{new}} \) to the other tree \( T_b \) is attempted (Line 3) & 4). In case of a successful connection, the solution is updated (Line 7). 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 3), and samples new goals for the goal tree (Line 6).
- **Conditional Sampling** (Line 7), which first samples a state from \( Q \), and then samples a corresponding valid time, with which \( x_{\text{rand}} \) 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).](image)

- **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 \( T \) 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 \( B \): \( B.\text{timeRange} \) determines the time bound for goal sampling and \( B.\text{batchSize} \) determines after how many generated samples the expansion takes place. When a batch is full, \( B.\text{timeRange} \) is increased by \( P.\text{rangeFactor} \) and \( B.\text{batchSize} \) 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 \( B.\text{batchProbability} \) and \( 1 - B.\text{batchProbability} \), respectively, to ensure a uniform distribution over the total space.

Precisely, the **Progressive Goal Region Expansion** works as follows: The parameters \( P.\text{rangeFactor} \), \( P.\text{initialBatchSize} \), and \( P.\text{sampleRatio} \) are user-specified. All variables of \( B \) are initialized at the start (Algorithm 2) and updated during execution. While \( B.\text{timeRange} \) is used when the current goal region is sampled, \( B.\text{newTimeRange} \) is used to sample the newly expanded one. After the first expansion, \( B.\text{newTimeRange} \) is always higher than \( B.\text{timeRange} \) by a factor equal to \( P.\text{rangeFactor} \) (Alg. 3 Line 2) & 3). The minimum amount of the new batch size is given by \( (P.\text{rangeFactor} - 1) \cdot B.\text{totalSamples} \). 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
Algorithm 2 InitializeBoundVariables

Input: $P$

1. $B.timeRange ← P.rangeFactor$
2. $B.newTimeRange ← P.rangeFactor$
3. $B.batchSize ← P.initialBatchSize$
4. $B.samplesInBatch ← 0$; $B.totalSamples ← 0$
5. $B.batchProbability ← 1$
6. $B.goals ← ∅$; $B.newGoals ← ∅$
7. return $B$

Algorithm 3 UpdateGoalRegion

Input: $B, P, t_{\text{max}}$

1. if $t_{\text{max}} = \infty$ and $B.samplesInBatch = B.batchSize$ then
2. $B.timeRange ← B.newTimeRange$
3. $B.newTimeRange ← P.rangeFactor$
4. $B.batchSize ← \frac{P.rangeFactor - 1 \cdot B.totalSamples}{P.sampleRatio}$
5. $B.batchProbability ← \frac{1 - P.sampleRatio}{P.rangeFactor}$
6. $B.goals ← B.goals \cup B.newGoals$
7. $B.newGoals ← ∅$; $B.samplesInBatch ← 0$
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. 5 for an illustration) meet this requirement. Thus, similar to

Algorithm 4 SampleGoal

Input: $x_{\text{start}}, \mathcal{A}_{\text{goal}}, T_{\text{goal}}, t_{\text{max}}, B$

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

The specific calculation of $t_{\text{lb}}, t_{\text{ub}}$ is dependent on whether the current (Line 3) 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, $x_{\text{new}}$, after tree expansion. Contrary to RRT*, rewiring is only performed in the goal trees.

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.

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 3). 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 $x_{\text{start}}$ 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 $t_{\text{lb}}, t_{\text{ub}}$ are obtained by the minimal arrival time from the start configuration $q_{\text{start}}$ until $q$ (Line 3) and the maximum valid time given by:

$$\text{MAXVALIDTIME}(q_{\text{end}}, G) = \max_{(q_{g}, t_{g}) \in G} \left( t_{g} - \min_{i} \frac{d_{\mathcal{Q}}(q_{\text{end}}, q_{g})}{v_{i}^{\text{max}}} \right)$$

The specific calculation of $t_{\text{lb}}, t_{\text{ub}}$ is dependent on whether the current (Line 4) or the new region is sampled (Line 7).
Algorithm 5 SampleConditionally

Input: $x_{\text{start}}, \mathcal{X}, B$

1. repeat
2. $q \leftarrow \text{SAMPLEUNIFORM}(\mathcal{Q})$
3. $t_{\text{min}} \leftarrow t_{\text{start}} + \text{LOWERBOUNDARRIVALTIME}(f_{\text{start}}, q)$
4. if RANDOM(0, 1) < $B.b\text{atchProbability}$ then
5. $t_{\text{lb}} \leftarrow t_{\text{min}}$
6. $t_{\text{ab}} \leftarrow \text{MAXVALIDTIME}(q, B, \text{goals})$  $\triangleright$ eq $(2)$
7. else
8. $t_{\text{min}}^* \leftarrow \text{MAXVALIDTIME}(q, B, \text{goals})$
9. $t_{\text{lb}} \leftarrow \text{MAX}(t_{\text{min}}, t_{\text{min}}^*)$
10. $t_{\text{ab}} \leftarrow \text{MAXVALIDTIME}(q, B, \text{newGoals})$
11. until $t_{\text{lb}} < t_{\text{ab}}$
12. $t \leftarrow \text{SAMPLEUNIFORM}(t_{\text{lb}}, t_{\text{ab}})$
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 and RRT* in space-time using their OMPL implementations in default configuration. Since RRT* and RRT-Connect algorithms cannot 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.

A. Scenarios

We evaluate the method on the following scenarios:

(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 ($\mathbb{R}^3$) 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 $6^\text{th}$ and the $11^\text{th}$ agent (i.e. the previous 5, and 10 agents, respectively, already have a trajectory).

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).

1Videos of the scenarios, and the paths are in the supplementary material.
B. Experimental Results

We analyze the results of both the abstract experiments (Fig. 5(a) - Fig. 5(d)) and the simulated robot experiments (Fig. 5(e) - Fig. 5(h)). 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 as well as 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].
