Efficient sampling-based approaches to optimal path planning in complex cost spaces

Didier Devaurs, Thierry Simeon, Juan Cortés

Efficient Sampling-based Approaches to Optimal Path Planning in Complex Cost Spaces

Didier Devaurs, Thierry Siméon, and Juan Cortés

CNRS, LAAS, 7 avenue du colonel Roche, F-31400 Toulouse, France
Univ de Toulouse, LAAS, F-31400 Toulouse, France
{devaurs,nic,jcortes}@laas.fr

Abstract. Sampling-based algorithms for path planning have achieved great success during the last 15 years, thanks to their ability to efficiently solve complex high-dimensional problems. However, standard versions of these algorithms cannot guarantee optimality or even high-quality for the produced paths. In recent years, variants of these methods, taking cost criteria into account during the exploration process, have been proposed to compute high-quality paths (such as T-RRT), some even guaranteeing asymptotic optimality (such as RRT*). In this paper, we propose two new sampling-based approaches that combine the underlying principles of RRT* and T-RRT. These algorithms, called T-RRT* and AT-RRT, offer probabilistic completeness and asymptotic optimality guarantees. Results presented on several classes of problems show that they converge faster than RRT* toward the optimal path, especially when the topology of the search space is complex and/or when its dimensionality is high.

Keywords: optimal path planning · anytime path planning · cost space path planning · sampling-based path planning

1 Introduction

Robot path-planning methods have traditionally focused on solving the *feasible path planning* problem, i.e. on finding a collision-free path for a robot moving in a complex environment. This relies on a classical framework abstracting the workspace of a robot system into a *configuration space*. In many application fields, however, generating feasible solution paths might not be sufficient. It may be required to obtain a high-quality solution path with respect to a given cost criterion, i.e. a low-cost path. One might even be looking for the optimal solution path with respect to this cost criterion, i.e. the path minimizing the cost. This amounts to solving an *optimal path planning* problem.

The first cost criterion to be considered was path length [4],[10, 11],[14, 15]. More interesting problems can be addressed with more sophisticated criteria, based on the definition of a cost function over the configuration space, which is then referred to as a *cost space*. Early work in cost-space path planning only involved discrete, coarse-grained cost functions [5],[10]. Our work focuses on continuous cost functions, which is more challenging. As an example, in outdoor

navigation problems, the cost of a configuration can be the elevation of the position of the robot within a 2-D terrain. When high-clearance paths are desirable, the cost of a configuration can be the inverse of the distance between the robot and the closest obstacle [2],[8]. Even more complex cost functions can appear in robotic problems [1],[13] and structural-biology problems [7].

When applied to the optimal path planning problem, classical grid-based methods, such as A* or D*, can compute resolution-optimal solution paths [16]. However, these methods are limited to problems involving low-dimensional spaces that can be discretized without leading to a combinatorial explosion. On the other hand, sampling-based algorithms, such as the Rapidly-exploring Random Tree (RRT) [12], have been successful at solving complex path-planning problems in high-dimensional spaces. Besides, they are conceptually simple and achieve probabilistic completeness. Nevertheless, these algorithms originally targeted feasible path planning, and usually produce sub-optimal solutions. Smoothing methods can be used to improve solution paths in a post-processing phase [6], but they often provide only local improvement, and offer no guarantee of convergence toward the global optimum. With the aim of taking a configuration-cost function into account during the space exploration, a variant of RRT called the Transition-based RRT (T-RRT) was proposed [8]. It extends RRT by integrating a Metropolis-like transition test favoring the exploration of low-cost regions of the space. It has been successfully applied to diverse robotic problems [1],[2],[8] and structural-biology problems [7], but it offers no optimality guarantee. Another variant of RRT, called RRT*, was devised to solve the optimal path planning problem [10]. RRT* has been shown to guarantee *asymptotic optimality*, and has been applied to various robotic problems [9–11]. However, it has been suggested that RRT* might converge slowly in high-dimensional spaces [2]. Finally, more recent approaches focus on asymptotic near-optimality [4],[14].

In this paper, we combine two approaches, namely RRT* and T-RRT, to devise new algorithms inheriting their respective strengths. The first algorithm, called *Transition-based RRT** (T-RRT*), consists of integrating the transition test of T-RRT into RRT*. The motivation is to favor the exploration of low-cost regions of the space, while maintaining the asymptotic properties of RRT*. The second algorithm, called *Anytime T-RRT* (AT-RRT), consists of enhancing T-RRT with an anytime behavior enabled by the integration of a procedure adding useful cycles (based on the path-cost criterion) to the graph built over the space [15]. The motivation is to quickly obtain a first high-quality solution-path and, then, carry on the exploration for the solution to continually improve and converge toward the optimal path.

In what follows, we present a simple formulation of the feasible and optimal path planning problems (Section 2). Then, we describe T-RRT* and AT-RRT in greater details (Section 3); we prove that both algorithms are probabilistically complete and asymptotically optimal (Section 4). Finally, we evaluate T-RRT* and AT-RRT on several path planning problems, and show that they converge toward the optimal path faster than RRT* (Section 5). Thanks to the filtering properties of the transition test they include, T-RRT* and AT-RRT can effi-

ciently solve difficult problems featuring complex cost spaces, on which RRT* converges very slowly. We present several such examples, illustrating various aspects that make a path planning problem difficult to solve. 1) If the problem features a large-scale workspace, even in low dimension, favoring low-cost regions avoids wasting time exploring the whole space. 2) If the space features several homotopic classes between which it is difficult to jump, even in low dimension, using the transition test can bias the search toward the class containing the optimal path and avoid being trapped in a sub-optimal class. 3) If the problem is high-dimensional, it is inherently complex because the search space is intrinsically large and can potentially contain many homotopic classes.

2 Problem Formulation

2.1 Feasible Path Planning

The classical formulation of the path planning problem relies on abstracting the workspace of a robotic system into a configuration space $\mathcal{C}$, also called $\mathcal{C}$-space. A configuration $q \in \mathcal{C}$ describes the position and volume occupied by the robotic system in the workspace. The subset of $\mathcal{C}$ containing the configurations inducing collisions with some obstacles in the workspace is denoted $\mathcal{C}_{\text{obst}}$. Assuming that its complement in $\mathcal{C}$ is an open set, we denote by $\mathcal{C}_{\text{free}}$ the set $cl(\mathcal{C} \setminus \mathcal{C}_{\text{obst}})$ of configurations producing no collision, where $cl()$ denotes the closure of a set. Given an initial configuration $q_{\text{init}} \in \mathcal{C}_{\text{free}}$ and a goal configuration $q_{\text{goal}} \in \mathcal{C}_{\text{free}}$, a path planning problem can be defined as a triplet $(\mathcal{C}, q_{\text{init}}, q_{\text{goal}})$. A path over the $\mathcal{C}$-space is a continuous function $\pi : [0,1] \to \mathcal{C}$. It is said to be collision-free if for all $t \in [0,1]$, $\pi(t) \in \mathcal{C}_{\text{free}}$, i.e. $\pi : [0,1] \to \mathcal{C}_{\text{free}}$. Let Π denote the set of all paths over $\mathcal{C}$ and Π_{free} the set of collision-free paths in Π. The *feasible path planning problem* is classically defined as follows:

Definition 1 (Feasible path planning). *Given a path planning problem $(\mathcal{C}, q_{\text{init}}, q_{\text{goal}})$, find a path $\pi \in \Pi_{\text{free}}$ such that $\pi(0) = q_{\text{init}}$ and $\pi(1) = q_{\text{goal}}$, if one exists, or report failure otherwise.*

Let Π_{feas} denote the set of paths in Π_{free} satisfying this feasibility condition. Among the path planning problems having a solution, the analysis we present requires to focus on problems satisfying the *robust feasibility* property [10]. Several algorithms have been proposed in the robotics community to solve the feasible path planning problem. Among them, sampling-based approaches are not complete, but satisfy a property called *probabilistic completeness*, that can be interpreted as a notion of "almost-sure" success.

Definition 2 (Probabilistic completeness). *An algorithm $\mathcal{A}$ is probabilistically complete if, for any robustly feasible path planning problem $(\mathcal{C}, q_{\text{init}}, q_{\text{goal}})$, the probability that $\mathcal{A}$ fails to return a solution when one exists decays to zero as the running time of $\mathcal{A}$ approaches infinity.*

The analysis we present in Section 4 is based on the fact that T-RRT and RRT* have been shown to be probabilistically complete [8, 10].

2.2 Optimal Path Planning

Let $c : \mathcal{C} \to \mathbb{R}_+$ denote a continuous cost function associating to each configuration of the $\mathcal{C}$-space a positive cost value. Being enriched with this function, $\mathcal{C}$ is referred to as a cost space, and we talk about cost-space path planning. When exploring a cost space, instead of only looking for a feasible solution path, one might search for a high-quality path with respect to a given path-cost criterion. Let $c_{\mathrm{p}} : \Pi_{\mathrm{free}} \to \mathbb{R}_+$ denote this cost criterion, associating to each collision-free path a positive cost value. It can be defined in several ways, the most common being to consider the *integral of the cost* along a path. As a discrete approximation of the integral of the cost with constant step size $\delta = \frac{1}{n}$ (where n is the number of subdivisions of the path), the cost of a path π can be defined as

$$c_{\mathrm{p}}(\pi) = \frac{\mathrm{length}(\pi)}{n} \sum_{k=1}^{n} c\left(\pi\left(\frac{k}{n}\right)\right) \quad . \tag{IC}$$

As an alternative, the *mechanical work* of a path can be defined as the sum of the positive cost variations along the path, which can be interpreted as summing the "forces" acting against the motion. It has been shown that the mechanical work can assess path quality better than the integral of the cost in many situations [8]. As a discrete approximation of the mechanical work with constant step size $\delta = \frac{1}{n}$, the cost of a path π can be defined as

$$c_{\mathrm{p}}(\pi) = \sum_{k=1}^{n} \max\left\{0\,,\; c\left(\pi\left(\frac{k}{n}\right)\right) - c\left(\pi\left(\frac{k-1}{n}\right)\right)\right\} \quad . \tag{MW}$$

We could consider other criteria to evaluate path quality, such as the maximal cost along the path, or the average cost. In the case of feasible planning, path length could be considered. However, this is not a good choice when planning in a cost space because this criterion ignores the costs of the configurations along the path. Which criterion is the most suited depends on the planning problem and on the characteristics of its expected optimal solution. Comparing cost criteria is out of the scope of this paper. We use both IC and MW not to limit ourselves to a single criterion, which could bias the interpretation of the results.

The *optimal path planning problem* can now be defined as follows:

Definition 3 (Optimal path planning). *Given a path planning problem $(\mathcal{C}, q_{\mathrm{init}}, q_{\mathrm{goal}})$, a continuous configuration-cost function $c : \mathcal{C} \to \mathbb{R}_+$, and a monotonic, bounded path-cost criterion $c_p : \Pi_{\mathrm{free}} \to \mathbb{R}_+$, find a path $\pi^* \in \Pi_{\mathrm{feas}}$ such that $c_{\mathrm{p}}(\pi^*) = \min\{c_{\mathrm{p}}(\pi) \,|\, \pi \in \Pi_{\mathrm{feas}}\}$ if one exists, or report failure otherwise.*

With these notations, an optimal path planning problem is defined as a quintuplet $(\mathcal{C}, q_{\mathrm{init}}, q_{\mathrm{goal}}, c, c_{\mathrm{p}})$. If it admits a solution π^*, then π^* is called the optimal path. Note that the analysis we present requires to focus on optimal path planning problems admitting a *robustly optimal* solution [10]. In the context of optimal path planning, the evaluation of a sampling-based algorithm should be based not only on probabilistic completeness, but also on the concept of *asymptotic optimality*. This property can be interpreted as a notion of "almost-sure" convergence toward the optimal path, and has been defined as follows [10]:

Definition 4 (Asymptotic optimality). *An algorithm $\mathcal{A}$ is asymptotically optimal if, for any optimal path planning problem $(\mathcal{C}, q_{\text{init}}, q_{\text{goal}}, c, c_{\text{p}})$ admitting a robustly optimal solution path with finite cost $c^* \in \mathbb{R}_+$, the cost of the solution path produced by $\mathcal{A}$ (this cost being infinite if no solution is available yet) decreases toward c^* as the running time of $\mathcal{A}$ approaches infinity.*

The analysis in Section 4 is based on the asymptotic optimality of RRT* [10].

3 Algorithms

The Rapidly-exploring Random Tree (RRT) [12] is a popular sampling-based algorithm that can solve the feasible path planning problem. Starting from the initial configuration q_{init}, RRT iteratively builds a tree $\mathcal{T}$ on the $\mathcal{C}$-space. At each iteration, a configuration q_{rand} is randomly sampled in $\mathcal{C}$, and an extension toward q_{rand} is attempted, starting from its nearest neighbor, q_{near}, in $\mathcal{T}$. If the extension succeeds, a new configuration q_{new} is added to $\mathcal{T}$, and connected by an edge to q_{near}. The criteria on when to stop the exploration can be reaching the goal configuration q_{goal}, a given number of nodes in $\mathcal{T}$, a given number of iterations, or a given running time.

Several algorithms have been devised as extensions of RRT to explore cost spaces. Among them, the Transition-based RRT (T-RRT) consists of integrating in RRT a transition test that favors the exploration of low-cost regions of $\mathcal{C}$ [8]. This transition test is used to accept or reject the move from q_{near} to q_{new} based on their respective costs. Even though it yields high-quality (i.e. low-cost) paths when solving the feasible path planning problem on a cost space, T-RRT offers no guarantee to solve the optimal path planning problem. The other variant of RRT we consider here, named RRT*, has been specifically developed to solve the optimal path planning problem [10]. In RRT*, instead of being linked to q_{near}, q_{new} is linked to the configuration (among its neighbors in $\mathcal{C}$) minimizing the cost of the path in $\mathcal{T}$ between q_{init} and q_{new}. Furthermore, if, as a parent in $\mathcal{T}$, q_{new} allows one of its neighbors in $\mathcal{C}$ to be connected to q_{init} via a lower-cost path than the one currently available, some rewiring is performed in $\mathcal{T}$. By deciding how to create and remove edges of $\mathcal{T}$ based on the costs of the paths between q_{init} and every node in $\mathcal{T}$, RRT* enables the cost of the solution extracted from $\mathcal{T}$ to decrease with time. However, despite its asymptotic-optimality guarantees, RRT* may converge slowly in high-dimensional spaces [2].

In this work, we combine the beneficial concepts underlying these extensions of RRT: 1) the filtering properties of the transition test in T-RRT, favoring the creation of new nodes in low-cost regions of $\mathcal{C}$, and 2) the cost-based management of edges in RRT*, allowing the cost of the solution path to decrease with time. We do this in two different ways, by proposing an extension to RRT* named *Transition-based RRT** (T-RRT*) and an extension to T-RRT named *Anytime T-RRT* (AT-RRT). Both algorithms can solve the optimal path planning problem and offer asymptotic-optimality guarantees (cf. Section 4). They allow us to efficiently explore complex cost spaces, yielding high-quality solution paths that improve with time in an anytime fashion.

Algorithm 1: Transition-based RRT* (T-RRT*)

input : the optimal path planning problem $(\mathcal{C}, q_{\text{init}}, q_{\text{goal}}, c, c_{\text{p}})$, the dimension d of the $\mathcal{C}$-space, and the γ constant derived from the volume of $\mathcal{C}_{\text{free}}$ [10]
output: the graph $\mathcal{G}$

1 $\mathcal{G} \leftarrow$ `initGraph`(q_{init})
2 **while not** `stoppingCriteria`$(\mathcal{G})$ **do**
3 $q_{\text{rand}} \leftarrow$ `sampleRandomConfiguration`$(\mathcal{C})$
4 $q_{\text{near}} \leftarrow$ `findNearestNeighbor`$(\mathcal{G}, q_{\text{rand}})$
5 $q_{\text{new}} \leftarrow$ `extend`$(q_{\text{near}}, q_{\text{rand}})$
6 **if** $q_{\text{new}} \neq$ null **and** `transitionTest`$(\mathcal{G}, c(q_{\text{near}}), c(q_{\text{new}}))$ **then**
7 `addNewNode`$(\mathcal{G}, q_{\text{new}})$
8 $n \leftarrow$ `numberOfNodes`$(\mathcal{G})$
9 $Q_{\text{near}} \leftarrow$ `nearestNeighbors`$(\mathcal{G}, q_{\text{new}}, \gamma\,(\log(n)\,/\,n)^{1\,/\,d})$
10 $q_{\text{par}} \leftarrow$ `parentMinimizingCostFromInit`$(q_{\text{new}}, q_{\text{near}}, Q_{\text{near}}, c_{\text{p}})$
11 `addNewEdge`$(\mathcal{G}, q_{\text{par}}, q_{\text{new}})$
12 **foreach** $q_{\text{n}} \in Q_{\text{near}}$ **do**
13 $\pi \leftarrow$ `pathInSpace`$(q_{\text{new}}, q_{\text{n}})$
14 **if** `costFromInit`$(q_{\text{new}}) + c_{\text{p}}(\pi) <$ `costFromInit`(q_{n}) **and** `isCollisionFree`(π) **then**
15 `removeEdge`$(\mathcal{G}$, `parent`$(q_{\text{n}}), q_{\text{n}})$
16 `addNewEdge`$(\mathcal{G}, q_{\text{new}}, q_{\text{n}})$
17 **return** $\mathcal{G}$

Algorithm 2: transitionTest $(\mathcal{G}, c_{\text{i}}, c_{\text{j}})$

input : the current temperature T and its increase rate T_{rate}
output: *true* if the transition is accepted, *false* otherwise

1 **if** $c_{\text{j}} \leq c_{\text{i}}$ **then return** True
2 **if** $\exp(-(c_{\text{j}} - c_{\text{i}})\,/\,T) > 0.5$ **then**
3 $T \leftarrow T\,/\,2^{(c_{\text{j}} - c_{\text{i}})\,/\,\texttt{costRange}(\mathcal{G})}$; **return** True
4 **else**
5 $T \leftarrow T \cdot 2^{T_{\text{rate}}}$; **return** False

3.1 Transition-based RRT* (T-RRT*)

The pseudo-code of T-RRT* is shown in Algorithm 1. T-RRT* extends RRT* by integrating the transition test (line 6) originally developed for T-RRT [8]. This transition test is used to accept or reject the move from q_{near} to q_{new} based on their respective costs. If the move is accepted, T-RRT* behaves exactly like RRT*. First, a new node is created in $\mathcal{G}$ to store q_{new} (line 7). Then, a search in $\mathcal{G}$ is performed to compute the set Q_{near} of configurations contained in a neighborhood of q_{new} of radius $\gamma\,(\log(n)\,/\,n)^{1\,/\,d}$ (line 9). As defined for RRT*, this radius depends on the dimension d of $\mathcal{C}$, on a constant γ derived from the volume of $\mathcal{C}_{\text{free}}$, and on the number n of nodes in $\mathcal{G}$ [10]. This dependency on

n ensures that the radius decreases as $\mathcal{G}$ grows. The next step of the algorithm consists of finding the configuration q_{par} in $Q_{\text{near}} \cup \{q_{\text{near}}\}$ to which q_{new} should be connected (line 10): the parent of q_{new} is chosen as the configuration via which the path between q_{init} and q_{new} has minimal cost. This is done by computing, for all $q_{\text{n}} \in Q_{\text{near}} \cup \{q_{\text{near}}\}$, the cost $c_{\text{p}}(\pi_{\text{n}}^{\mathcal{G}}) + c_{\text{p}}(\pi_{\text{n}}^{\mathcal{C}})$, where $\pi_{\text{n}}^{\mathcal{G}}$ is the path between q_{init} and q_{n} in $\mathcal{G}$, and $\pi_{\text{n}}^{\mathcal{C}}$ is the path between q_{n} and q_{new} in $\mathcal{C}$. Finally, since the addition of a new node in $\mathcal{G}$ potentially leads to the appearance of new paths having lower costs than those currently in $\mathcal{G}$, some rewiring might be performed (lines 12–16). For each $q_{\text{n}} \in Q_{\text{near}}$, if the cost of the path going from q_{init} to q_{n} via q_{new} is lower than the cost of the current path between q_{init} and q_{n} in $\mathcal{G}$, q_{new} becomes the new parent of q_{n} in $\mathcal{G}$.

The `transitionTest` involved in the T-RRT* algorithm is presented in Algorithm 2. The transition between two configurations is evaluated on the basis of their costs c_{i} and c_{j}, c_{i} being the cost of the source configuration and c_{j} the cost of the target configuration. A downhill move ($c_{\text{j}} \leq c_{\text{i}}$) in the cost landscape is always accepted. An uphill move is accepted or rejected based on the probability $\exp(-(c_{\text{j}} - c_{\text{i}}) \,/\, T)$ that decreases exponentially with the cost variation $c_{\text{j}} - c_{\text{i}}$. In that case, the level of selectivity of the transition test is controlled by the *temperature* T, which is an adaptive parameter of the algorithm. Low temperatures limit the expansion to gentle slopes of the cost landscape, and high temperatures enable it to climb steep slopes. After each accepted uphill move, T is decreased to avoid over-exploring high-cost regions: it is divided by $2^{(c_{\text{j}} - c_{\text{i}}) \,/\, \texttt{costRange}(\mathcal{G})}$, where `costRange(`$\mathcal{G}$`)` is the cost difference between the highest-cost and the lowest-cost configurations stored in the nodes of $\mathcal{G}$. After each rejected uphill move, T is increased to facilitate the exploration and avoid being trapped in a local minimum of the cost landscape: it is multiplied by $2^{T_{\text{rate}}}$, where $T_{\text{rate}} \in (0, 1]$ is the temperature increase rate.

3.2 Anytime Transition-based RRT (AT-RRT)

AT-RRT, whose pseudo-code is presented in Algorithm 3, also features the `transitionTest` (line 6), and extends T-RRT by offering an anytime behavior. Before any feasible path is found between q_{init} and q_{goal}, AT-RRT behaves exactly like T-RRT. As opposed to T-RRT, however, after a solution path is found, the exploration is allowed to continue and a cycle-addition procedure is activated (lines 9–10). This leads to the creation in $\mathcal{G}$ of new paths that can be of higher quality than the one found so far. This procedure is based on the notion of *useful cycles*, as described in [15].

The `addUsefulCycles` procedure is presented in Algorithm 4. When a new configuration q_{new} is added to $\mathcal{G}$, we consider all configurations in $\mathcal{G}$, within a neighborhood of q_{new}, as potential candidate targets for new edges. The radius of this neighborhood depends on the dimension d of $\mathcal{C}$ and on a constant γ derived from the volume of $\mathcal{C}_{\text{free}}$, as is done for RRT* [10]. This radius also decreases with the number n of nodes in $\mathcal{G}$. Within the candidate set Q_{near}, we are interested in the configurations that are "close" to q_{new} in $\mathcal{C}$, but "far" from q_{new} in $\mathcal{G}$, not in terms of distance but of path cost. For each candidate $q_{\text{n}} \in Q_{\text{near}}$, if the

Algorithm 3: Anytime Transition-based RRT (AT-RRT)

input : the optimal path planning problem $(\mathcal{C}, q_{\text{init}}, q_{\text{goal}}, c, c_{\text{p}})$
output: the graph $\mathcal{G}$

$\mathcal{G} \leftarrow$ `initGraph`(q_{init})
while not `stoppingCriteria`$(\mathcal{G})$ **do**
  $q_{\text{rand}} \leftarrow$ `sampleRandomConfiguration`$(\mathcal{C})$
  $q_{\text{near}} \leftarrow$ `findNearestNeighbor`$(\mathcal{G}, q_{\text{rand}})$
  $q_{\text{new}} \leftarrow$ `extend`$(q_{\text{near}}, q_{\text{rand}})$
  if $q_{\text{new}} \neq$ null **and** `transitionTest`$(\mathcal{G}, c(q_{\text{near}}), c(q_{\text{new}}))$ **then**
    `addNewNode`$(\mathcal{G}, q_{\text{new}})$
    `addNewEdge`$(\mathcal{G}, q_{\text{near}}, q_{\text{new}})$
    if `solutionPathExists`$(\mathcal{G}, q_{\text{init}}, q_{\text{goal}})$ **then**
      `addUsefulCycles`$(\mathcal{G}, q_{\text{new}}, c_{\text{p}})$
return $\mathcal{G}$

Algorithm 4: addUsefulCycles $(\mathcal{G}, q_{\text{new}}, c_{\text{p}})$

input: the dimension d of the $\mathcal{C}$-space
  the γ constant derived from the volume of $\mathcal{C}_{\text{free}}$ (as in RRT* [10])

$n \leftarrow$ `numberOfNodes`$(\mathcal{G})$
$Q_{\text{near}} \leftarrow$ `nearestNeighbors`$(\mathcal{G}, q_{\text{new}}, \gamma\,(\log(n) / n)^{1/d})$
foreach $q_{\text{n}} \in Q_{\text{near}}$ **do**
  $\pi_{\text{g}} \leftarrow$ `pathInGraph`$(\mathcal{G}, q_{\text{new}}, q_{\text{n}})$
  $\pi_{\text{s}} \leftarrow$ `pathInSpace`$(q_{\text{new}}, q_{\text{n}})$
  if $c_{\text{p}}(\pi_{\text{s}}) < c_{\text{p}}(\pi_{\text{g}})$ **and** `isCollisionFree`(π_{s}) **then**
    `addNewEdge`$(\mathcal{G}, q_{\text{new}}, q_{\text{n}})$

cost of the local path π_{s} between q_{new} and q_{n} in $\mathcal{C}$ is less than the cost of the lowest-cost path π_{g} between q_{new} and q_{n} in $\mathcal{G}$, and if π_{s} is collision-free, we add an edge to $\mathcal{G}$ between q_{new} and q_{n}, thus creating a useful cycle.

4 Analysis

We now review the properties of T-RRT* and AT-RRT, in terms of probabilistic completeness and asymptotic optimality (cf. Section 2). It has already been proven that T-RRT and RRT* are probabilistically complete [8, 10]. In the case of T-RRT, this property is directly derived from the probabilistic completeness of RRT, despite the integration of the transition test. A similar reasoning allows us to state that T-RRT* is probabilistically complete, thanks to the probabilistic completeness of RRT*. Furthermore, as AT-RRT behaves like T-RRT before a solution path is found, it satisfies the same properties.

Theorem 1 (Probabilistic completeness). *The T-RRT* and AT-RRT algorithms are probabilistically complete.*

Let us assume in the sequel that the γ constant involved in T-RRT* and AT-RRT, and originally introduced in RRT*, satisfies

$$\gamma > 2\left(1+\frac{1}{d}\right)^{\frac{1}{d}}\left(\frac{\mu(\mathcal{C}_{\text{free}})}{\zeta_d}\right)^{\frac{1}{d}} \quad , \tag{1}$$

where d is the dimension of $\mathcal{C}$, ζ_d is the volume of the unit ball in the d-dimensional Euclidean space, and $\mu()$ is an operator measuring volumes. Under this assumption, RRT* has proven to be asymptotically optimal [10].

The only difference between T-RRT* and RRT* is the presence of a transition test filtering configurations based on their costs. The consequence of applying such rejection sampling is that the samples cannot be assumed to be drawn from a uniform distribution on $\mathcal{C}$. Even though the asymptotic optimality of RRT* was proven under a "uniform distribution" assumption, this result can be extended to any continuous probability distribution with density bounded away from zero [10]. As the probability of a sample to be accepted by the transition test is never zero, the samples drawn by T-RRT* follow such distribution. Therefore, T-RRT* is also asymptotically optimal.

Let us recall that the interesting properties of RRT* come from its ability to replace existing edges in $\mathcal{G}$ by new edges enabling lower-cost paths to appear. This allows the cost of the solution path produced by RRT* to decrease with time. Furthermore, the "almost-sure" convergence toward the optimal solution path is ensured by the fact that the cost-based decisions on connections are made for configurations within neighborhoods of radii based on a value of γ satisfying (1). The lower bound on γ expressed in (1) is the minimal value allowing RRT* to be asymptotically optimal. Keeping in mind that increasing the value of γ raises the computational cost of an iteration of RRT* (because of the increased number of neighbors to consider), this lower bound represents the optimal tradeoff between efficiency and asymptotic optimality.

Clearly, AT-RRT and T-RRT* use the same procedure to create and filter nodes, based on the extension mechanism of RRT and on the transition test of T-RRT. The difference between them lies in the management of edges. In AT-RRT, no edge is removed, thus leading to the creation of cycles, but this has no impact on the current analysis. The main point is that, in both algorithms, alternative paths are created based on cost improvement. Where they differ is on the criterion that an edge has to satisfy to be considered useful in terms of cost improvement. In T-RRT*, this criterion is based on whether an edge allows a configuration to be connected to q_{init} via a path in $\mathcal{G}$ having minimal cost. In AT-RRT, this criterion is based on whether an edge allows two configurations to be connected via a path in $\mathcal{C}$ whose cost is lower than the costs of the existing paths in $\mathcal{G}$. It is clear that both criteria achieve the same goal: they both allow the cost of the solution path to decrease with time. Finally, as the cost-based decisions on the addition of useful cycles happen in neighborhoods of radii based on a value of γ satisfying (1), AT-RRT is also asymptotically optimal.

Theorem 2 (Asymptotic optimality). *The T-RRT* and AT-RRT algorithms are asymptotically optimal.*

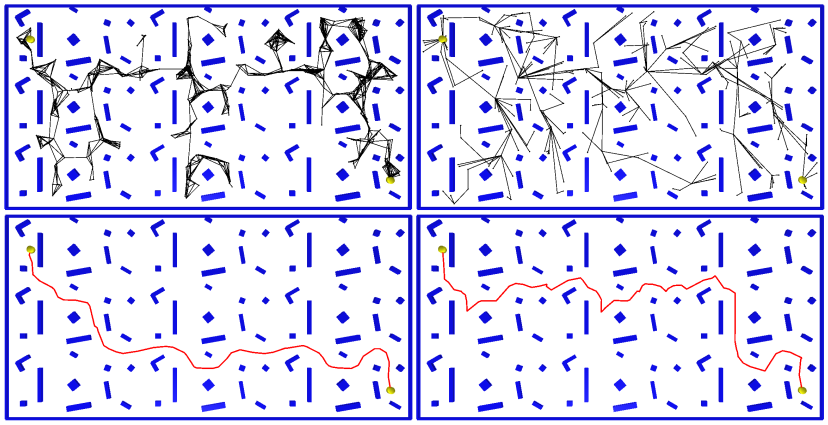

Fig. 1. *Stones* problem: 2-DoFs disk moving among rectangular obstacles, while maximizing its clearance. Top row: graphs built by AT-RRT (left) and T-RRT* (right) after a runtime of 0.5 s. Bottom row: solution paths produced by T-RRT* when minimizing IC (left) or MW (right) after a runtime of 10 s. Paths produced by AT-RRT are similar.

5 Evaluation

5.1 Path Planning Problems

We have evaluated T-RRT* and AT-RRT on several optimal path-planning problems that differ in terms of $\mathcal{C}$-space dimensionality, geometrical complexity and configuration-cost function type. The *Stones* problem (illustrated in Fig. 1) is a 2-degrees-of-freedom (DoFs) example in which a disk has to go through a space cluttered with rectangular-shaped obstacles. The objective is to maximize clearance, so the cost function c associates to each position of the disk the inverse of the distance between the disk and the closest obstacle.

The *Inspection* problem deals with industrial inspection in a dense environment, and involves an aerial robot, as shown in Fig. 2. The featured quadrotor is modeled as a 3-DoFs sphere (i.e. a free-flying sphere) representing the security zone around it. Assuming that motions are performed quasi-statically, we restrict the problem to planning in position (controllability issues lie outside the scope of this paper). For safety reasons, the quadrotor has to move in this environment trying to maximize clearance for the security sphere. The specificity of this problem is its large-scale workspace.

The *Transport* problem features aerial robots, and deals with the collaborative transport of objects, as shown in Fig. 3. Two quadrotors have to carry an H-looking object and go through one of two holes in a wall. The robotic system comprises the quadrotors themselves (and not safety spheres around them), the 3-R planar manipulator arms attached below them, and the carried object. A configuration of this system is defined by the position and orientation of the object in space, and the relative positions of the quadrotors with respect to the object. This problem is restricted to planning in position for the quadrotors

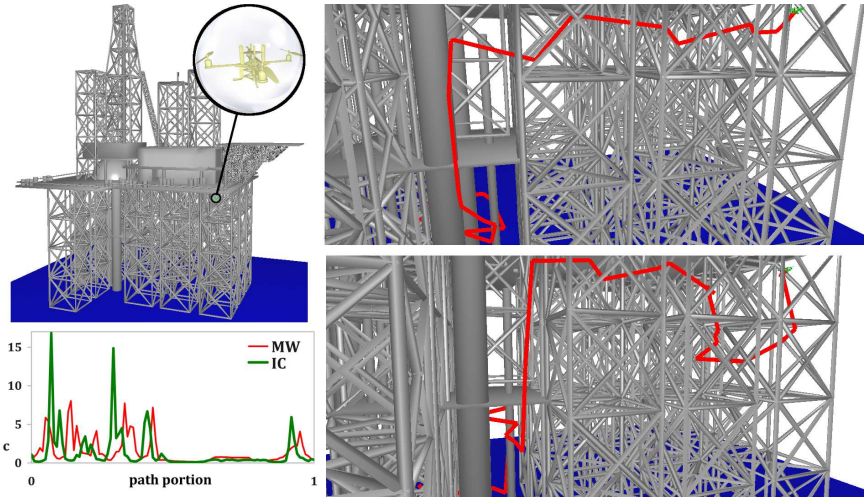

Fig. 2. *Inspection* problem: quadrotor (whose close-up is shown in yellow) inspecting an oil-rig (top left). The cost function is based on the clearance of the 3-DoFs safety sphere around the quadrotor. Right column: paths produced by AT-RRT when minimizing IC (top) or MW (bottom), after a running time of 10 s. The cost profiles of the two paths are also shown (bottom left). Paths produced by T-RRT* are similar.

because of the quasi-static assumption made on their motions. We consider a planar version of the problem, thus disregarding translations along the Y axis and rotations around the X and Z axes. Besides, the revolute joints of the arms are passive DoFs in constraints related to the closure of the kinematic chain. Therefore, the system can be described with 7 DoFs: 3 DoFs for the object (two translations along the X and Z axes, and a rotation around the Y axis) and 2 DoFs for each quadrotor (two translations along the X and Z axes). In this example, different cost functions can be defined. The notion of clearance could be considered, but we will use a cost function based on the notion of "balance" in our experiments. Assuming the initial configuration is stable, the idea is to maintain it as much as possible, while allowing a complete freedom of movement for the object with respect to the translations along the X and Z axes. To achieve that, the cost of a configuration is defined as the sum of the differences to the initial values for the rotation of the object and the translations of the quadrotors. The specificity of the *Transport* problem lies in the fact that it features two very distinct homotopic classes. The two holes in the wall constitute narrow passages of similar difficulty in terms of purely geometrical planning: despite being wider, the lower hole is partly obstructed by the second wall. However, when planning in the cost space with the clearance-based cost function, paths going through the lower hole are favored because it is larger than the other one. On the contrary, when planning in the cost space with the balance-based cost function, paths going through the upper hole are favored because going through the lower one requires the robotic system to tilt sharply.

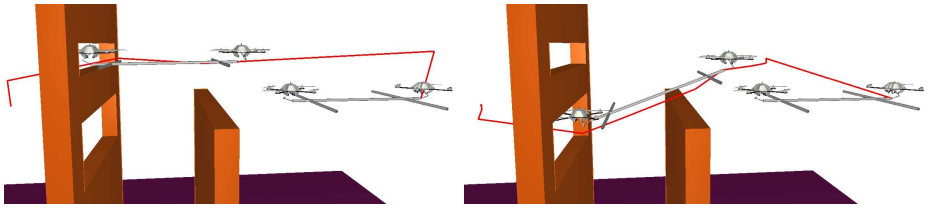

Fig. 3. *Transport* problem: the two quadrotors have to transport an object and go through one of the holes in the wall, while maintaining the balance of the whole system. Both images show an intermediate and the final configurations along paths obtained after 50 s. Left: path produced by T-RRT* when minimizing MW. Paths produced when minimizing IC, and paths produced by AT-RRT are similar. Right: path produced by RRT* when minimizing IC. Paths produced when minimizing MW are similar.

The *Snake* problem (illustrated in Fig. 4) involves a snake-like object constituted of 10 identical cylinders between which 9 revolute joints are defined. We also consider two translations and a rotation in the planar workspace, which adds up to 12 DoFs. The cost function is defined as the sum of the absolute differences between the angular values of consecutive revolute joints, added to the absolute value of the first revolute joint. The objective is to favor a straight configuration of the robot, or configurations in which all revolute joints have the same value, which correspond to a regular coiling of the robot. This problem enables us to analyze the behavior of the algorithms in higher dimension.

5.2 Settings

Before using T-RRT* and AT-RRT, their parameters have to be set. Following [2], T_{rate} is set to 0.1 and T is initialized to 10^{-6}. Finding a good value for γ happens to be a real issue. As already mentioned, the lower bound for γ expressed in (1) is the optimal value with respect to the tradeoff between efficiency and asymptotic optimality. However, computing this value requires to estimate the volume of $\mathcal{C}_{\text{free}}$. This is possible in low-dimensional spaces when the robotic system and the obstacles are represented with simple geometric models, but this is not realistic otherwise. To ensure that γ satisfies (1), we set:

$$\gamma = 2\left(1+\frac{1}{d}\right)^{\frac{1}{d}}\left(\frac{\mu(\mathcal{C})}{\zeta_d}\right)^{\frac{1}{d}} \quad . \tag{2}$$

On the *Stones* and *Inspection* problems, since $\mathcal{C}$ is an Euclidean space, its volume can easily be computed using the validity interval of every DoF. However, this is not straightforward on the *Transport* and *Snake* problems because of the revolute joints. For a DoF corresponding to such joint, its angular range is multiplied by the length of the associated rigid body.

T-RRT* and AT-RRT have been implemented in the motion planning platform *Move3D*. To fairly assess them, no smoothing is performed on the solution paths. Values of IC and MW are averaged over 100 runs. Results have been obtained on an Intel Core i5 processor at 2.6 GHz with 8 GB of memory.

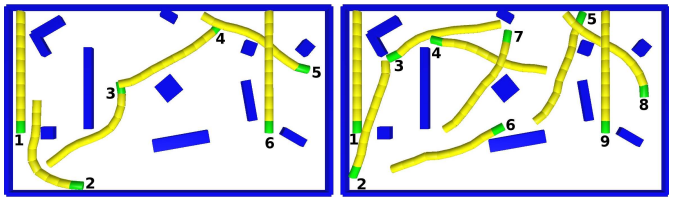

Fig. 4. Selected configurations along paths produced by AT-RRT when minimizing IC (left) or MW (right), after a running time of 100 s, on the *Snake* problem. A snake-like object has to move among rectangular obstacles. The cost function favors straight configurations, and regular over irregular coiling. T-RRT* provides similar results.

5.3 Results

T-RRT* and AT-RRT build graphs over $\mathcal{C}$ in different ways because they involve different strategies to create (and potentially remove) edges. This is illustrated in Fig. 1 on the *Stones* problem. The upper left figure clearly shows the cycles created by AT-RRT, and the redundancy in paths. As can been seen in the upper right figure, the tree built by T-RRT* is much sparser, because high-cost edges are removed. The numerical results we present show that these differences in behavior do not create significant differences in performance. Also, the solution paths produced by the two algorithms usually look very similar.

Differences in solution paths are mainly due to the choice of the cost criterion: IC or MW. This is clearly visible in Fig. 1 and 2. Minimizing IC tends to favor shorter paths along which the maximal cost can be quite high (as shown by Fig. 2, bottom left), and minimizing MW sometimes produces strangely convoluted paths. Another drawback of MW (not illustrated here) is that, if the cost of q_{init} is high, MW can be low even for paths going through high-cost configurations. A better cost criterion could probably be defined by combining IC and MW, but this goes beyond the scope of this paper. Note that, on some problems, such as *Transport*, the choice of the cost criterion has little impact on the results.

To evaluate the performance of T-RRT* and AT-RRT, we analyze the evolution over time of the costs of the solution paths they produce. As a reference, we compare both algorithms to RRT* [10]. To obtain the best results with RRT*, we use the *conditional activation* and *branch-and-bound* heuristics when they are beneficial. The *conditional activation* heuristic consists of planning with a regular RRT until the first solution is found, and only then activating the procedures specific to RRT* [9]. The *branch-and-bound* heuristic consists of trimming the nodes in $\mathcal{G}$ that cannot allow finding paths with costs lower than that of the current solution path, which is assessed using a cost-to-go function [11]. Both heuristics are beneficial on the *Transport* and *Snake* problems.

Numerical results obtained on the four path planning problems (each one being tested with a given pair (q_{init}, q_{goal}) of configurations) are reported in Fig. 5 for IC, and Fig. 6 for MW. They clearly show that T-RRT* and AT-RRT converge faster than RRT* toward the optimum. Even on a problem as simple as *Stones*, if only little time is available, T-RRT* and AT-RRT yield better-quality

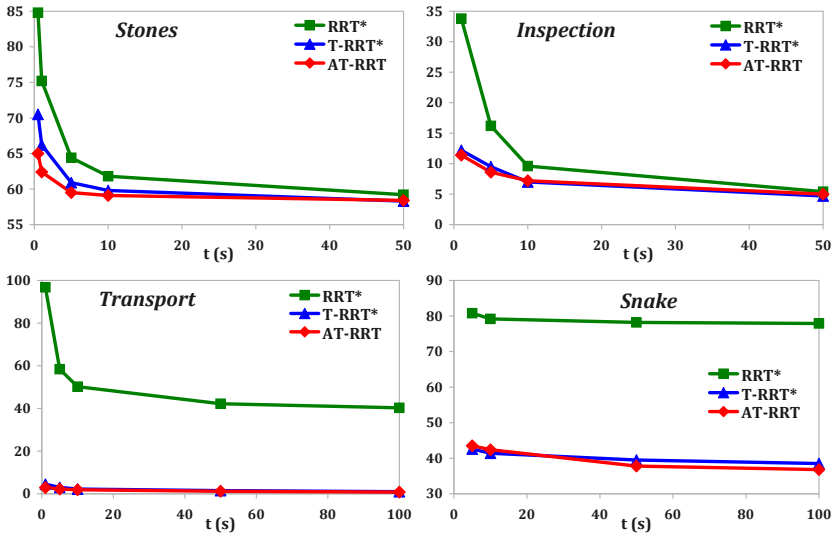

Fig. 5. Evolution over time of the costs (IC) of the solution paths produced by RRT*, T-RRT* and AT-RRT, on the four path planning problems.

solutions than RRT*. But, given enough time, all algorithms produce paths of similar quality. When the size of the workspace is larger, as in the *Inspection* problem, the dominance of T-RRT* and AT-RRT is even clearer. It appears that the filtering properties of the transition test help focus the search on the most relevant (i.e. low-cost) parts of the workspace: graphs produced by RRT* contain numerous nodes in high-cost regions of the space, contrary to graphs produced by T-RRT* or AT-RRT (not shown here due to space limitations). When the problem is even more complex, as is the case of *Transport*, the weaknesses of RRT* start to translate into a very low rate of convergence. Thanks to the transition test, the search performed by T-RRT* or AT-RRT is usually guided toward the homotopic class containing the optimal path (i.e. the upper hole, when using the balance-based cost function, as shown by Fig. 3, left). On the contrary, the first solution produced by RRT* can belong to any of the two homotopic classes; if it is found in the sub-optimal one (i.e. the lower hole), RRT* gets stuck in this class and into optimizing a low-quality solution (as shown by Fig. 3, right). Finally, on high-dimensional problems, such as *Snake*, RRT* usually converges very slowly. Looking at Fig. 5 and 6, one may think that this is also the case for T-RRT* and AT-RRT. To check that, we have let all algorithms run for 12 hours while minimizing MW. We have obtained solutions of costs 3.42, 2.41 and 2.24 for RRT*, T-RRT* and AT-RRT respectively. Looking at Fig. 6, it means that, after 100 s, T-RRT* and AT-RRT are already close to the optimum, contrary to RRT*.

Finally, to assess whether what we observe is consistent across the domains corresponding to the four path planning problems, we have evaluated the algo-

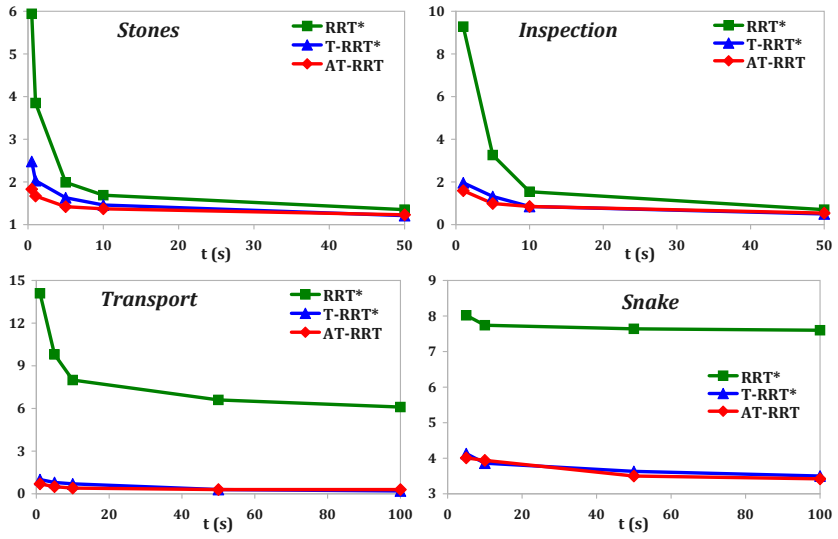

Fig. 6. Evolution over time of the costs (MW) of the solution paths produced by RRT*, T-RRT* and AT-RRT, on the four path planning problems.

rithms on instances of these problems involving different pairs (q_{init}, q_{goal}) of configurations. The results we have obtained (not presented here due to space limitations) are similar to what we report above.

6 Conclusion

In this paper, we have proposed two novel sampling-based algorithms to solve the optimal path planning problem, by combining the underlying principles of T-RRT and RRT*, the goal being to benefit from their respective strengths while overcoming their weaknesses. On the positive side, T-RRT can efficiently explore a cost space thanks to the filtering properties of its transition test, and RRT* is asymptotically optimal. On the negative side, T-RRT is not asymptotically optimal, and RRT* may converge slowly on complex cost spaces. The two hybrid methods are: 1) the Transition-based RRT* (T-RRT*), which is an extension of RRT* integrating the transition test of T-RRT, and 2) the Anytime T-RRT (AT-RRT), which is an extension of T-RRT integrating a useful-cycle addition procedure. We have proven that T-RRT* and AT-RRT are both probabilistically complete and asymptotically optimal. We have evaluated them on several optimal path-planning problems featuring complex cost spaces, and compared them to RRT*. Results show that they converge faster than RRT* toward the optimal path, sometimes orders of magnitude faster.

Results tend to show that AT-RRT performs slightly better than T-RRT*. As future work, it would be interesting to analyze further how the two algorithms behave, to pinpoint which strategy works best at solving particular classes of

optimal path planning problems. Disregarding computational performance, a clear advantage of AT-RRT over T-RRT* is that it can easily be extended into a multiple-tree algorithm, similar to the *Multi T-RRT* [3]. Another interesting aspect of AT-RRT is that it builds a graph containing cycles, therefore providing alternative paths over the space. This could be leveraged when path replanning is required due to errors in the model or moving obstacles.

Acknowledgments

This work has been partially supported by the European Community under Contract ICT 287617 "ARCAS". The authors would like to thank Sertac Karaman for helpful discussions on the RRT* algorithm.

References

1. Berenson, D., Siméon, T., Srinivasa, S.: Addressing cost-space chasms in manipulation planning. In: IEEE ICRA. pp. 4561–4568 (2011)
2. Devaurs, D., Siméon, T., Cortés, J.: Enhancing the transition-based RRT to deal with complex cost spaces. In: IEEE ICRA. pp. 4105–4110 (2013)
3. Devaurs, D., Siméon, T., Cortés, J.: A multi-tree extension of the Transition-based RRT: Application to ordering-and-pathfinding problems in continuous cost spaces. In: IEEE/RSJ IROS (2014)
4. Dobson, A., Bekris, K.: Sparse roadmap spanners for asymptotically near-optimal motion planning. Int. J. Robot. Res. 33(1), 18–47 (2014)
5. Ferguson, D., Stentz, A.: Anytime RRTs. In: IEEE/RSJ IROS. pp. 5369–5375 (2006)
6. Geraerts, R., Overmars, M.: Creating high-quality paths for motion planning. Int. J. Robot. Res. 26(8), 845–863 (2007)
7. Jaillet, L., Corcho, F., Pérez, J.J., Cortés, J.: Randomized tree construction algorithm to explore energy landscapes. J. Comput. Chem. 32(16), 3464–3474 (2011)
8. Jaillet, L., Cortés, J., Siméon, T.: Sampling-based path planning on configuration-space costmaps. IEEE Trans. Robot. 26(4), 635–646 (2010)
9. Jeon, J., Karaman, S., Frazzoli, E.: Anytime computation of time-optimal off-road vehicle maneuvers using the RRT*. In: IEEE CDC. pp. 3276–3282 (2011)
10. Karaman, S., Frazzoli, E.: Sampling-based algorithms for optimal motion planning. Int. J. Robot. Res. 30(7), 846–894 (2011)
11. Karaman, S., Walter, M., Perez, A., Frazzoli, E., Teller, S.: Anytime motion planning using the RRT*. In: IEEE ICRA. pp. 1478–1483 (2011)
12. LaValle, S., Kuffner, J.: Rapidly-exploring random trees: progress and prospects. In: Algorithmic and Computational Robotics: New Directions, pp. 293–308. A. K. Peters, Wellesley, MA (2001)
13. Manubens, M., Devaurs, D., Ros, L., Cortés, J.: Motion planning for 6-D manipulation with aerial towed-cable systems. In: RSS (2013)
14. Marble, J., Bekris, K.: Asymptotically near-optimal planning with probabilistic roadmap spanners. IEEE Trans. Robot. 29(2), 432–444 (2013)
15. Nieuwenhuisen, D., Overmars, M.: Useful cycles in probabilistic roadmap graphs. In: IEEE ICRA. pp. 446–452 (2004)
16. Stentz, A.: Optimal and efficient path planning for partially-known environments. In: IEEE ICRA. pp. 3310–3317 (1994)