A multi-tree extension of the Transition-based RRT: Application to ordering-and-pathfinding problems in continuous cost spaces

Didier Devaurs, Thierry Simeon, Juan Cortés

A Multi-Tree Extension of the Transition-based RRT: Application to Ordering-and-Pathfinding Problems in Continuous Cost Spaces

Didier Devaurs, Thierry Siméon and Juan Cortés

***Abstract*— The Transition-based RRT (T-RRT) is a variant of RRT developed for path planning on a continuous cost space, i.e. a configuration space featuring a continuous cost function. It has been used to solve complex, high-dimensional problems in robotics and structural biology. In this paper, we propose a multiple-tree variant of T-RRT, named *Multi-T-RRT*. It is especially useful to solve ordering-and-pathfinding problems, i.e. to compute a path going through several unordered waypoints. Using the Multi-T-RRT, such problems can be solved from a purely geometrical perspective, without having to use a symbolic task planner. We evaluate the Multi-T-RRT on several path planning problems and compare it to other path planners. Finally, we apply the Multi-T-RRT to a concrete industrial inspection problem involving an aerial robot.**

I. Introduction

Sampling-based path planning has traditionally aimed at finding collision-free paths to solve complex planning problems in high-dimensional spaces [1], [2]. However, beyond feasible solutions, in many applications it is important to compute high-quality paths with respect to a given cost criterion. When a cost function is defined on the configuration space, we call the latter a cost space.

Several approaches to sampling-based cost-space path planning have been proposed based on the Rapidly-exploring Random Tree (RRT) algorithm [1], such as RRT* [3] or the Transition-based RRT (T-RRT) [4]. T-RRT combines the exploratory strength of RRT with a transition test favoring low-cost regions. It has been successfully applied to diverse robot path-planning problems [4]–[8] (some involving human–robot interactions [5]) and structural biology problems [7], [9]. Contrary to RRT*, T-RRT does not offer asymptotic-optimality guarantees, but, in high-dimensional spaces, it may converge faster than RRT* [7], [8].

In this paper, we propose a multi-tree variant of T-RRT, named *Multi-T-RRT*. Since there exist numerous multi-tree path planners that involve RRT [10]–[22], we have evaluated existing techniques and selected the most effective ones. The Multi-T-RRT is particularly useful when looking for a path going through a given set of unordered waypoints. Such ordering-and-pathfinding problems involve two aspects: a low-level path planning problem that consists of connecting pairs of waypoints, and a high-level ordering problem that consists of finding an efficient way to visit all the waypoints (which is a simple kind of task planning problem).

All authors are with CNRS, LAAS, 7 avenue du colonel Roche, F-31400 Toulouse, France and Univ de Toulouse, LAAS, F-31400 Toulouse, France (e-mails: devaurs@laas.fr, nic@laas.fr, jcortes@laas.fr)

This work has been partially supported by the European Community under Contract ICT 287617 “ARCAS”.

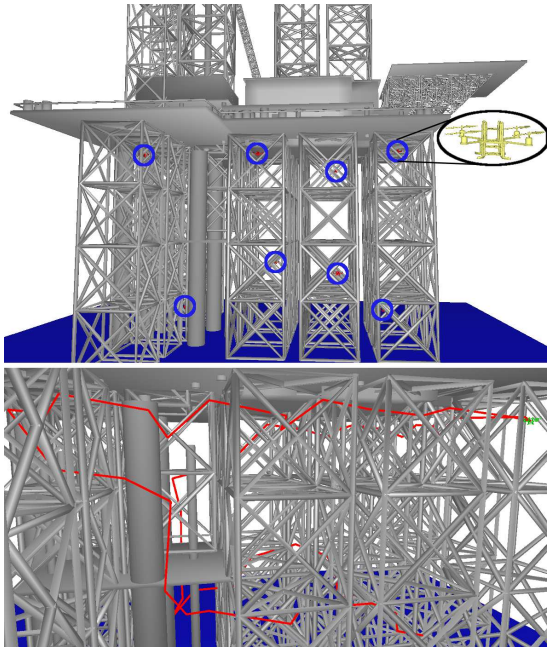

Fig. 1. Top: eight waypoints (shown in red and circled in blue) defined for a quadrotor (whose close-up is shown in yellow) inspecting an oil rig. Bottom: example of a trajectory (going through these waypoints) produced in 50 s by the Multi-T-RRT enhanced with useful-cycle addition.

Hybrid approaches to solve task-and-path planning problems are often based on decoupling the two aspects: a symbolic task planner computes a high-level plan (possibly based on geometrical data) that is refined by a path planner computing precise low-level paths [23]–[25]. In some cases, when tasks are simple enough, the overall problem possesses a purely geometrical formulation, and no symbolic task planner is needed [26]. In this work, we also follow a purely geometrical approach: the geometric path planner (i.e. Multi-T-RRT) yields high-quality high-level solutions based on the costs of the low-level paths it computes between waypoints. To achieve that, we have enhanced the Multi-T-RRT with a useful-cycle addition mechanism enabling it to continually improve the solution path in an anytime fashion (which is illustrated by the accompanying video).

After a brief review of T-RRT (Section II), we present the Multi-T-RRT, based on the strategies we have selected to expand and connect trees (Section III). Then, we report some evaluation results and compare the Multi-T-RRT to planners involving the Bidirectional T-RRT [8] (Section IV). Finally, we apply the Multi-T-RRT to ordering-and-pathfinding problems, including a concrete industrial inspection problem involving an aerial robot (Fig. 1, Section V).

Algorithm 1: Transition-based RRT

input : the configuration space $\mathcal{C}$, the cost function $c : \mathcal{C} \to \mathbb{R}_+$ and the initial configuration q_{init}
output: the tree $\mathcal{T}$

```
T ← initTree(q_init)
while not stoppingCriteria(T) do
    q_rand ← sampleRandomConfiguration(C)
    q_near ← findNearestNeighbor(T, q_rand)
    q_new ← extend(q_near , q_rand)
    if q_new ≠ null and
    transitionTest(T, c(q_near), c(q_new)) then
        addNewNodeAndEdge(T, q_near , q_new)
```

Algorithm 2: transitionTest ($\mathcal{T}$, c_i , c_j)

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

```
if c_j ≤ c_i then return True
if exp(−(c_j − c_i) / T) > 0.5 then
    T ← T / 2^((c_j − c_i) / costRange(T)) ;  return True
else
    T ← T · 2^(T_rate) ;  return False
```

II. Transition-based RRT (T-RRT)

Starting from an initial configuration q_{init}, RRT iteratively builds a tree $\mathcal{T}$ on the configuration space $\mathcal{C}$ [1]. 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 a given target configuration q_{goal}, a given number of nodes in the tree, a given number of iterations, or a given running time.

T-RRT (shown in Algorithm 1) is a variant of RRT used to explore cost spaces [4], [8]. It extends RRT by integrating a transition test favoring the exploration of low-cost areas of the space. The `transitionTest` presented in Algorithm 2 is used to accept or reject the move from q_{near} to q_{new} based on their respective costs. A downhill move ($c_j \leq c_i$) is always accepted. An uphill move is accepted or rejected based on the probability $exp(-(c_j - c_i) / T)$ that decreases exponentially with the cost variation $c_j - c_i$. 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, 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 $2^{(c_j - c_i) / \texttt{costRange}(\mathcal{T})}$, where `costRange`$(\mathcal{T})$ is the cost difference between the highest-cost and the lowest-cost configurations in the tree. After each rejected uphill move, T is increased to facilitate the exploration and to avoid being trapped in a local minimum: it is multiplied by $2^{T_{rate}}$, where $T_{rate} \in (0, 1]$ is the temperature increase rate. Following [8], we set T_{rate} to 0.1 and initialize T to 10^{-6}.

III. Multi-tree variants of T-RRT

This section introduces our multi-tree variant of T-RRT, named *Multi-T-RRT*. To develop it, we have surveyed several techniques proposed in similar work on multi-tree approaches to sampling-based path planning [10]–[22]. Some approaches aim at solving single-query problems, the way RRT usually works, but involve the construction of several RRTs to reach a solution [10]–[17]. Others are multiple-query approaches similar to the Probabilistic Road-Map (PRM), where RRT is used as a local planner [18]–[20]. Others focus on dynamic environments and build several RRTs at different points in time [21], [22]. The version of the Multi-T-RRT we present here is a single-query planner building several T-RRTs to find a path. We do not deal with multiple queries or dynamic environments.

Growing multiple trees on the configuration space can be done in various ways. The aim can be to have several RRTs rooted in different regions of the space to ensure a broader exploration [12]–[17]. In this context, trees are initialized and grown rather independently of one another. Other approaches aim at maintaining a road-map of RRTs over the space [19]–[22]. In this case, trees can be created or modified as a result of merging, splitting or pruning operations. Other approaches make use of sub-trees produced by previous queries [19], [22]. Others build RRTs in different subspaces, independently of each other [10], [11]. Finally, RRTs can be reduced to local connections between components of a large road-map [18]. In this work, we focus on growing several T-RRTs rooted at given waypoints.

When building several trees, controlling their number and the timing of the connection attempts are difficult issues [2]. First, the number of trees can be unbounded [19]–[21]. It can also be subjected to a pre-defined bound [13]–[16] or implicitly limited at runtime [17], [22]. Second, the tree roots can be sampled a priori [13] or at runtime [20]–[22]. They can also be strategic states discovered at runtime, such as configurations in narrow passages [14]–[17]. We focus here on the case where the number of trees is fixed and equal to the number of waypoints.

A. Multi-T-RRT

The pseudo-code of the *Multi-T-RRT* is presented in Algorithm 3. Instead of building a single tree, we build n trees rooted at n given waypoints q_{init}^k, $k = 1..n$. At each iteration, a tree $\mathcal{T}'$ is chosen for expansion in a round-robin fashion among the trees $\mathcal{T}_k$, $k = 1..n$. Then, an extension is attempted toward a randomly sampled configuration q_{rand}, starting from its nearest neighbor, q'_{near} , in $\mathcal{T}'$. We use an *Extend* function and not a *Connect* one, as recommended in [8]. If the extension succeeds and the transition test is satisfied, the new configuration q_{new} is added to $\mathcal{T}'$ and connected to q'_{near}. Then, we look for the configuration q''_{near} (and the tree $\mathcal{T}''$ containing it), which is the closest to q_{new} within all trees other than $\mathcal{T}'$. A connection between q_{new} and q''_{near} is attempted in both directions, by calling twice the `attemptLink` function. The exploration continues until all

Algorithm 3: Multi-T-RRT

input : the configuration space $\mathcal{C}$, the cost function $c : \mathcal{C} \to \mathbb{R}_+$ and the waypoints q_{init}^k, $k = 1..n$
output: the tree $\mathcal{T}$

```
for k = 1..n do
    T_k ← initTree(q_init^k)
while not stoppingCriteria({T_k | k = 1..n}) do
    T' ← chooseNextTreeToExpand()
    q_rand ← sampleRandomConfiguration(C)
    q'_near ← findNearestNeighbor(T', q_rand)
    q_new ← extend(q'_near, q_rand)
    if q_new ≠ null and
    transitionTest(T', c(q'_near), c(q_new)) then
        addNewNodeAndEdge(T', q'_near, q_new)
        (T'', q''_near) ← findNearestTree(q_new)
        T ← attemptLink(T', q_new, T'', q''_near, n)
        if T = null then
            T ← attemptLink(T'', q''_near, T', q_new, n)
```

Algorithm 4: attemptLink($\mathcal{T}_1$, q_1, $\mathcal{T}_2$, q_2, n)

input : the extension step-size δ
output: the tree $\mathcal{T}$

```
if distance(q1, q2) < 10 · δ then
    q_cur ← q1 ;  q_next ← extend(q1, q2)
    while q_next ≠ null and c(q_next) ≤ c(q_cur) do
        q_cur ← q_next ;  q_next ← extend(q_cur, q2)
    if q_cur = q2 then
        T ← linkAndMerge(T1, q1, T2, q2) ;  n ← n − 1
```

trees are merged or another stopping condition (number of nodes, number of expansions, running time) is met.

The `attemptLink` function (shown in Algorithm 4) was developed to attempt connections between both trees of the *Bidirectional T-RRT* [8]. If the configurations q_1 and q_2 are closer than ten times the extension step-size δ, and if the cost along the local path between them decreases monotonically (which is checked after every step of size δ), the trees $\mathcal{T}_1$ and $\mathcal{T}_2$ are merged, and the number of trees is decreased by 1. A distance threshold of $10 \cdot \delta$ represents a good trade-off between 1) wasting time checking edges that are unlikely to be valid (if the threshold is too high) and 2) having difficulties connecting trees (if the threshold is too low), as shown by the experiments presented in [8].

Using the transition test of T-RRT and testing tree connections based on cost constraints enables the Multi-T-RRT to favor low-cost regions of the space, and thus to yield low-cost paths. The cost of a path can be defined in several ways based on the costs of the configurations along the path, as we will show in the examples.

B. Other Multi-Tree Variants

To develop the Multi-T-RRT we addressed several points for which we had to choose among various alternatives. The first point was to decide which tree(s) to expand at a given step of the exploration process. The simplest strategy is to grow all trees at each iteration toward the same configuration q_{rand} [19], [21]. By having a single tree grown at each iteration, chosen in a round-robin fashion [13], [15], the trees are expanded toward different configurations q_{rand}, which appears to work better. Another strategy is to expand the tree that is the closest to q_{rand} [22]. However, when testing this approach, we have found that it can be difficult to expand trees that are growing close to the boundaries of the configuration space. A more sophisticated approach consists of choosing the tree to be expanded based on some probabilities that can be fixed [14] or adaptive [16]. But, we have found that such strategies show no clear benefit in terms of improving running time or path quality.

The second point was to decide when to attempt to link trees. The simplest strategy is to try after each successful expansion of a tree [12], [13], [15], [19], [21], [22]. Other, more sophisticated approaches consist of attempting a connection only when the bounding box of the expanded tree has increased in size [14], or when some stochastic test is satisfied, based on fixed or adaptive probabilities [16]. However, we have found that these approaches lead to many missed good opportunities for connection.

The third point was about how to perform the link attempt after a tree has been successfully expanded. This can involve a single tree, usually the nearest one [13], [16], [22], or a randomly chosen one. It can also involve all the other trees [12], [14], [17], [19], [21] or a subset of these trees, containing, e.g., some of the closest ones and some randomly chosen ones [20]. When a tree is chosen for the link attempt, we have to decide which node in this tree we will try to connect the new node of the expanded tree to. Again, it can be the nearest one or a randomly chose one. After evaluation, we have found that random choices are not beneficial. It works better to attempt a connection between the new node and its nearest neighbor within the nearest tree.

C. Useful-Cycle Addition

Even though the paths the Multi-T-RRT returns have low cost, from a higher-level ordering perspective, they might not represent the most efficient way to visit a set of waypoints. To address this issue, we propose a simple approach based on the anytime paradigm and the addition of useful cycles: after all trees are connected, we allow the exploration to continue and we activate a cycle-addition procedure. This leads to the appearance of new paths that can be of better quality (with respect to the sequence of visited waypoints) than the one found so far. Adding cycles works as follows: When a new configuration q_{new} is added to the graph, we consider all other configurations within a pre-specified distance in $\mathcal{C}$ as potential candidate targets for new edges. Among these candidates, we are interested in those that are "close" to q_{new} in $\mathcal{C}$, but "far" from q_{new} in the graph: for each candidate q_c, if the cost of the local path between q_{new} and q_c in $\mathcal{C}$ is strictly less than the cost of the lowest-cost path between q_{new} and q_c in the graph, we add an edge between q_c and q_{new}, thus creating a useful cycle. For more details on the anytime variant of T-RRT, and for a theoretical analysis, the interested reader is referred to [27].

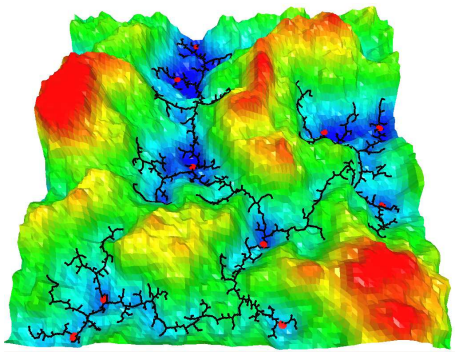

Fig. 2. Search tree built by the Multi-T-RRT on the *Landscape* problem. On this 2D cost-map, the cost is color-coded (from blue to red) and represented by the elevation. The ten waypoints are materialized by red disks.

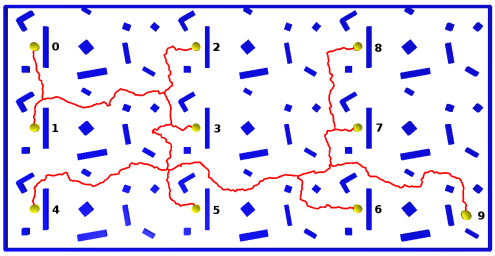

Fig. 3. Path computed by the Multi-T-RRT on the *Stones* problem, going through all ten waypoints. The cost is the inverse of the distance between the 2-DoF yellow disk and the blue rectangular-shaped obstacles.

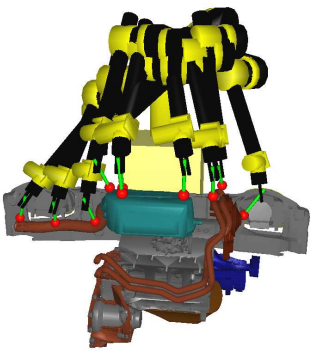

Fig. 4. Ten waypoints defined for the *Inspection* problem. The 6-DoF manipulator arm holds a sensor (the red sphere) that has to follow the surface of the car engine.

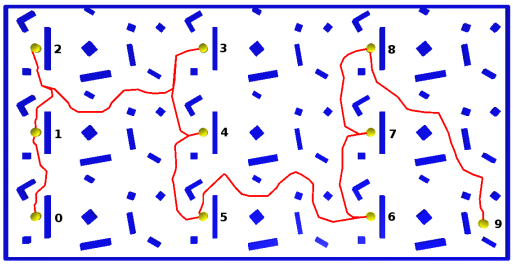

Fig. 5. Path produced by the Multi-T-RRT enhanced with useful-cycle addition on the *Stones* problem, for a running time of 5 s. The labels show the order in which the waypoints are visited.

IV. Evaluation Results

We have evaluated the Multi-T-RRT on several academic path planning problems that differ in terms of C-space dimensionality, geometrical complexity and cost-function type. We report results for three of them here. To fairly compare all algorithms, we first set aside the enhancement involving useful-cycle addition. For each example, we define ten waypoints that have to be visited in a pre-defined order (only to facilitate the evaluation of the algorithms). The *Landscape* problem is the 2D cost-map illustrated by Fig. 2, in which the cost is the elevation. The *Stones* problem (presented in Fig. 3) is a 2-degrees-of-freedom (DoF) problem in which a disk goes through a space cluttered with rectangular-shaped stones. The objective is to maximize clearance, so the cost function is the inverse of the distance between the disk and the closest obstacle. The *Inspection* problem (shown in Fig. 4) involves a 6-DoF manipulator arm holding a sensor with a spherical extremity, used to inspect a car engine. The objective is to keep the sensor as close as possible to the engine, so the cost function is the distance between the sphere and the engine surface.

The Multi-T-RRT has been implemented in the motion planning platform *Move3D* [28]. To fairly assess it, no smoothing is performed on the solution paths. On all problems, we record the running time t (in seconds), the number of expansion attempts X, the number of nodes N in the produced tree, and several quality criteria evaluated on the extracted path (with steps of size δ): the average cost $avgC$, the maximal cost $maxC$, the mechanical work MW, and the integral of the cost IC. The *mechanical work* of a path is the sum of the positive cost variations along this path [4]. For all variables, we give values averaged over 100 runs. Results were obtained on an Intel Core i5 processor at 2.6 GHz with 8 GB of memory.

We have compared two variants of the Multi-T-RRT to the basic version presented in Algorithm 3. We report the results of this comparison in Table I. The first variant involves having a local temperature associated to each tree, as opposed to having a global temperature. After evaluation, it seems that this modification has barely any influence on the results. The second variant is based on ensuring that all trees remain balanced (in terms of number of nodes) during the exploration. After evaluation, it is unclear whether this modification is advantageous or not. It appears to have sometimes a positive impact (e.g., on the *Stones* problem) and sometimes a negative impact (e.g., on the *Landscape* problem) on performance.

We have compared the Multi-T-RRT to the Bidirectional T-RRT [8] in two ways. First, in a simple scheme involving the Bidirectional T-RRT, we compute paths between pairs of consecutive waypoints, starting from scratch each time, and concatenate them to obtain the full path visiting all waypoints. Second, in an incremental scheme involving the Bidirectional T-RRT, we compute paths between pairs of consecutive waypoints, while keeping the tree built so far instead of deleting it as in the simple scheme. Results obtained with these two schemes are reported in Table I. As expected,

TABLE I

EVALUATION OF THE MULTI-T-RRT AND BIDIRECTIONAL T-RRT ON THE *Landscape*, *Stones* AND *Inspection* PROBLEMS.

		$avgC$	$maxC$	MW	IC	t (s)	N	X
Landscape	Multi-T-RRT - basic	11	22	240	10,000	0.06	1,100	6,000
	Multi-T-RRT - balanced	11	22	230	9,900	0.18	1,300	13,000
	Multi-T-RRT - local temperature	11	22	240	10,000	0.08	1,200	9,000
	simple Bidirectional T-RRT	11	22	230	9,800	0.12	2,700	20,000
	incremental Bidirectional T-RRT	11	22	240	9,900	0.12	1,400	10,000
Stones	Multi-T-RRT - basic	2.4	5.9	110	27,000	0.38	2,000	18,000
	Multi-T-RRT - balanced	2.4	5.9	110	29,000	0.28	1,100	8,000
	Multi-T-RRT - local temperature	2.5	5.9	110	29,000	0.38	2,100	18,000
	simple Bidirectional T-RRT	2.3	5.9	97	25,000	0.57	4,500	38,000
	incremental Bidirectional T-RRT	2.2	5.9	104	28,000	0.43	2,100	18,000
Inspection	Multi-T-RRT - basic	4.4	19	470	20,000	0.8	500	14,000
	Multi-T-RRT - balanced	4.1	19	480	20,000	0.9	600	16,000
	Multi-T-RRT - local temperature	4.3	19	470	19,000	1	600	18,000
	simple Bidirectional T-RRT	3.8	19	400	16,000	2	1,100	39,000
	incremental Bidirectional T-RRT	3.7	19	540	20,000	1.5	700	28,000

Average values over 100 runs are given for:

$avgC$	average cost
$maxC$	maximal cost
MW	mechanical work
IC	integral of the cost
t	running time
N	number of nodes in the tree
X	number of expansion attempts

the Multi-T-RRT is faster than the planners involving the Bidirectional T-RRT. Moreover, in spite of performing a quicker exploration of the space, the Multi-T-RRT produces paths whose costs are only slightly worse than those of paths produced by the planners involving the Bidirectional T-RRT. Therefore, the performance improvement is not achieved at the expense of path quality.

V. APPLICATION OF THE MULTI-T-RRT

In Section IV, we used the Multi-T-RRT to compute a high-quality path visiting an ordered set of waypoints, but only for evaluation purposes. In practice, the waypoints are not ordered a priori. Such ordering-and-pathfinding problems encompass two levels: a low-level path planning problem aiming at connecting the waypoints (which is solved by a geometric path planner) and a high-level ordering problem aiming at finding an efficient way to visit all the waypoints, based on the costs of the low-level paths (which is a simple kind of task planning problem). No symbolic task planner is required if we consider that the latter problem is an instance of the Traveling Salesman Problem (TSP) involving the complete graph (when path-cost is the integral of the cost) or digraph (when path-cost is the mechanical work) whose nodes are the waypoints. The distance associated with an edge of this graph (or digraph) can be estimated as the cost of the lowest-cost path between two waypoints in the tree built by the Multi-T-RRT. When only few waypoints are defined, the TSP is solved by an exhaustive search among all sequences. When more waypoints are involved, the Nearest-Neighbor or Multi-Fragment heuristics are used [29].

Based on this approach, we present an industrial inspection problem involving an aerial robot in a dense environment, as illustrated by Fig. 1. This example is typical of those addressed by the ARCAS project (http://www.arcas-project.eu). One of the goals of this project is to develop robot systems for the inspection and maintenance of industrial installations difficult to access for humans. In this example, a quadrotor is used to inspect an oil rig, going through eight waypoints defined a priori without explicit order (cf. Fig. 1). The quadrotor is modeled as a 3-DoF sphere (i.e. a free-flying sphere) representing the security zone around it. For safety reasons, it has to move in this environment trying to maximize clearance. The cost function is thus the inverse of the distance between the quadrotor and the obstacles. Assuming that the motions of the quadrotor are performed quasi-statically, we restrict the problem to planning in position (controllability issues lie outside the scope of this paper). Even though this example features a large-scale workspace, the Multi-T-RRT can quickly provide a first solution path: in about 5 s on average over 100 runs.

As already mentioned, a path produced by the Multi-T-RRT does not necessarily provide the best-quality solution visiting all the waypoints (see Fig. 3). The variant of the Multi-T-RRT involving useful-cycle addition was developed to enable continual improvement of the solution quality. As an example, Fig. 5 shows the achieved benefit on the *Stones* problem, especially when compared to Fig. 3. The path in Fig. 5 is representative of what we obtain for a running time of 5 s (as observed over 100 runs). Its cost is about half the cost of the path in Fig. 3 (obtained in 0.3 s). As another example, Fig. 1 shows a high-quality solution path obtained in 50 s on the *Oil rig* problem. Its cost is about half the cost of the first path obtained in 5 s.

To quantify the benefits of adding useful cycles, we evaluate the evolution of the quality of the solution path over time on the four examples (see Fig. 6). On the *Landscape* problem, path quality is measured by the mechanical work, MW, and on the other problems it is measured by the integral of the cost, IC. We compare the rate of convergence of the quality of paths produced by the Multi-T-RRT to the rates of convergence observed when planning with versions of PRM [30] and $\mathrm{RRT}_{\mathrm{obst\,way}}$ [13] creating cycles. Fig. 6 shows that the Multi-T-RRT yields a slightly better rate of convergence than $\mathrm{RRT}_{\mathrm{obst\,way}}$. The poor results of PRM, especially in high-dimensional spaces, are due to the fact that it does not involve any cost constraint. As it features a similar convergence rate, IRS [31] would probably not perform better. RRT* [3] would provide a better point of comparison, but it would require a multi-tree extension, which is out of the scope of this paper.

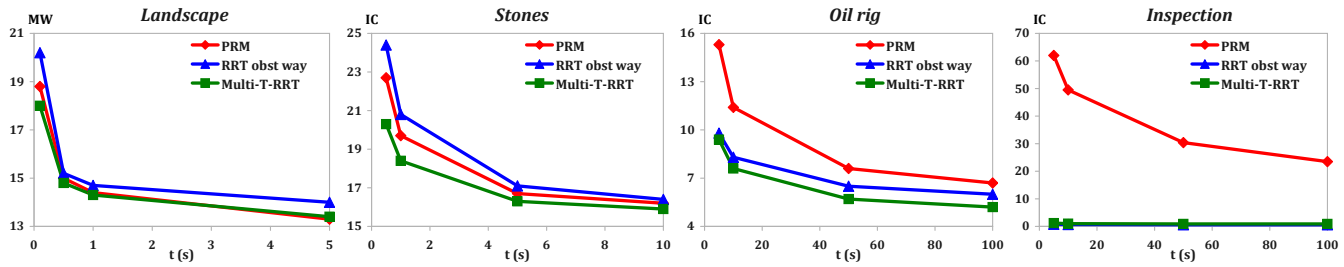

Fig. 6. Evolution of the quality of paths produced by the Multi-T-RRT, $\mathrm{RRT}_{\mathrm{obst\ way}}$ and PRM over time on problems of increasing dimensionality.

VI. Conclusion

We have presented a multi-tree variant of T-RRT named *Multi-T-RRT*. To achieve the highest efficiency, we have selected the best techniques involved in other multi-tree path planners. We have studied several path planning problems with different cost functions, geometrical complexity, and configuration-space dimensionality. When looking for a path going through an ordered list of waypoints, we have shown that the Multi-T-RRT is faster than planners based on the Bidirectional T-RRT, and that it yields paths of similar quality. When the planning problems involve both ordering and path planning aspects (i.e. when the order of the waypoints is not defined a priori), an anytime variant of the Multi-T-RRT enhanced with useful-cycle addition enables the solution to be continually improved. This allows us to visit the waypoints following a high-quality path computed (without the use of a symbolic task planner) based on the costs of the low-level paths connecting pairs of waypoints. Finally, we have applied the Multi-T-RRT to a realistic industrial inspection problem. We have also shown that the convergence rate of the solution quality is better with the Multi-T-RRT than with PRM and $\mathrm{RRT}_{\mathrm{obst\ way}}$.

The approach we have proposed to solve ordering-and-pathfinding problems in continuous cost spaces is a general one. This paper has focused on enhancing the T-RRT algorithm to solve such problems, by applying the multiple-tree, anytime and useful-cycle paradigms. It would be interesting to enhance other path planners in a similar way and compare their performance to that of the Multi-T-RRT. The poor results obtained with PRM highlight that such path planners should involve cost constraints. Therefore, a good candidate seems to be RRT* [3], in a multiple-tree version.

References

[1] S. LaValle and J. Kuffner, "Rapidly-exploring random trees: progress and prospects," in *Algorithmic and Computational Robotics: New Directions*. A. K. Peters, Wellesley, MA, 2001.

[2] S. LaValle, *Planning Algorithms*. Cambridge University Press, 2006.

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

[4] L. Jaillet, J. Cortés, and T. Siméon, "Sampling-based path planning on configuration-space costmaps," *IEEE Transactions on Robotics*, vol. 26, no. 4, 2010.

[5] J. Mainprice, A. Sisbot, L. Jaillet, J. Cortés, R. Alami, and T. Siméon, "Planning human-aware motions using a sampling-based costmap planner," in *Proc. IEEE ICRA*, 2011.

[6] D. Berenson, T. Siméon, and S. Srinivasa, "Addressing cost-space chasms in manipulation planning," in *Proc. IEEE ICRA*, 2011.

[7] R. Iehl, J. Cortés, and T. Siméon, "Costmap planning in high dimensional configuration spaces," in *Proc. IEEE/ASME AIM*, 2012.

[8] D. Devaurs, T. Siméon, and J. Cortés, "Enhancing the transition-based RRT to deal with complex cost spaces," in *Proc. IEEE ICRA*, 2013.

[9] L. Jaillet, F. Corcho, J.-J. Pérez, and J. Cortés, "Randomized tree construction algorithm to explore energy landscapes," *Journal of Computational Chemistry*, vol. 32, no. 16, 2011.

[10] J. Esposito, J. Kim, and V. Kumar, "Adaptive RRTs for validating hybrid robotic control systems," in *Algorithmic Foundations of Robotics VI*. Springer-Verlag, 2005.

[11] C. Belta, J. Esposito, J. Kim, and V. Kumar, "Computational techniques for analysis of genetic network dynamics," *The International Journal of Robotics Research*, vol. 24, no. 2-3, 2005.

[12] M. Clifton, G. Paul, N. Kwok, and D. Liu, "Evaluating performance of multiple RRTs," in *Proc. IEEE/ASME MESA*, 2008.

[13] A. Ettlin and H. Bleuler, "Randomised rough-terrain robot motion planning," in *Proc. IEEE/RSJ IROS*, 2006.

[14] M. Strandberg, "Augmenting RRT-planners with local trees," in *Proc. IEEE ICRA*, 2004.

[15] W. Wang, X. Xu, Y. Li, J. Song, and H. He, "Triple RRTs: an effective method for path planning in narrow passages," *Advanced Robotics*, vol. 24, no. 7, 2010.

[16] W. Wang, Y. Li, X. Xu, and S. Yang, "An adaptive roadmap guided Multi-RRTs strategy for single query path planning," in *Proc. IEEE ICRA*, 2010.

[17] D. Flavigné and M. Taïx, "Improving motion planning in weakly connected configuration spaces," in *Proc. IEEE/RSJ IROS*, 2010.

[18] M. Morales, S. Rodríguez, and N. Amato, "Improving the connectivity of PRM roadmaps," in *Proc. IEEE ICRA*, 2003.

[19] T.-Y. Li and Y.-C. Shie, "An incremental learning approach to motion planning with roadmap management," in *Proc. IEEE ICRA*, 2002.

[20] E. Plaku, K. Bekris, B. Chen, A. Ladd, and L. Kavraki, "Sampling-based roadmap of trees for parallel motion planning," *IEEE Transactions on Robotics*, vol. 21, no. 4, 2005.

[21] R. Gayle, K. Klinger, and P. Xavier, "Lazy Reconfiguration Forest (LRF) - an approach for motion planning with multiple tasks in dynamic environments," in *Proc. IEEE ICRA*, 2007.

[22] M. Zucker, J. Kuffner, and M. Branicky, "Multipartite RRTs for rapid replanning in dynamic environments," in *Proc. IEEE ICRA*, 2007.

[23] S. Cambon, R. Alami, and F. Gravot, "A hybrid approach to intricate motion, manipulation and task planning," *The International Journal of Robotics Research*, vol. 28, no. 1, 2009.

[24] L. Kaelbling and T. Lozano-Pérez, "Hierarchical task and motion planning in the now," in *Proc. IEEE ICRA*, 2011.

[25] S. Kiesel, E. Burns, C. Wilt, and W. Ruml, "Integrating vehicle routing and motion planning," in *Proc. ICAPS*, 2012.

[26] T. Siméon, J.-P. Laumond, J. Cortés, and A. Sahbani, "Manipulation planning with probabilistic roadmaps," *The International Journal of Robotics Research*, vol. 23, no. 7-8, 2004.

[27] D. Devaurs, T. Siméon, and J. Cortés, "Efficient sampling-based approaches to optimal path planning in complex cost spaces," in *Proc. WAFR*, 2014.

[28] T. Siméon, J.-P. Laumond, and F. Lamiraux, "Move3D: a generic platform for path planning," in *Proc. IEEE ISATP*, 2001.

[29] J. Bentley, "Experiments on traveling salesman heuristics," in *Proc. ACM-SIAM Symposium on Discrete Algorithms*, 1990.

[30] D. Nieuwenhuisen and M. Overmars, "Useful cycles in probabilistic roadmap graphs," in *Proc. IEEE ICRA*, 2004.

[31] J. Marble and K. Bekris, "Towards small asymptotically near-optimal roadmaps," in *Proc. IEEE ICRA*, 2012.