Robotics and Autonomous Systems 157 (2022) 104238

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot

A rapidly-exploring random trees approach to combined task and motion planning

Riccardo Caccavale *, Alberto Finzi

Department of Electrical Engineering and Information Technologies (DIETI), Università degli Studi di Napoli Federico II, Via Claudio 21, Napoli, 80125, Italy

ARTICLE INFO

Article history:
Available online 17 August 2022

Keywords:
Robot planning
Sampling-based planning
Rapidly-exploring random trees
Mobile robotics

ABSTRACT

Task and motion planning in robotics are typically addressed by separated intertwined methods. Task planners generate abstract high-level actions to be executed, while motion planners provide the associated discrete movements in the configuration space satisfying kinodynamic constraints. However, these two planning processes are strictly dependent, therefore the problem of combining task and motion planning with a uniform approach is very relevant. In this work, we tackle this issue by proposing a RRT-based method that addresses combined task and motion planning. Our approach relies on a combined metric space where both symbolic (task) and sub-symbolic (motion) spaces are represented. The associated notion of distance is then exploited by a RRT-based planner to generate a plan that includes both symbolic actions and feasible movements in the configuration space. The proposed method is assessed in several case studies provided by a real-world hospital logistic scenario, where an omni-directional mobile robot is involved in navigation and transportation tasks.

1. Introduction

Combining task and motion planning is a crucial issue in robotics. These two planning problems are usually treated in a separated manner to exploit the complementary characteristics of the two planning methods. While task planners typically work with abstract specifications of the planning problem, searching for plans of symbolic actions, motion planners generates plans of discrete motions in the configuration space, satisfying the associated motion constraints. Since these algorithms work at different levels of abstraction, a typical approach is to first address the task planning problem and then to deploy the motion planner to find the actual implementation of the abstract plan in the configuration space. However, when the mission is complex and/or the environment is highly constrained (navigation in cluttered environments, manipulation problems, etc.), such decoupled approach is no longer effective. In these settings, since symbolic and motions constraints can be strictly intertwined, a symbolic plan generated by the task planner can be easily not executable in the configuration space. In these circumstances, a unified approach to task and motion planning is needed to address both symbolic and motion constraints [1].

Several approaches have been proposed in the literature to address combined task and motion planning [2–13]. Following [9], we can distinguish approaches where: motion planning is primary, but guided by the task planning algorithm [6,8,10,11]; task planning is primary, while motion planning is selectively invoked [7]; task planning and motion planning are interleaved and a generated task plan is incrementally expanded by the motion planner, which simultaneously checks whether each symbolic action is executable in the configuration space [2,3,12,13].

In this work, we explore a novel approach in which task and motion planning problems are handled in a uniform manner by a sampling-based planning algorithm. Specifically, we propose and investigate the effectiveness of a novel method which fully relies on an extension of Rapidly-exploring Random Trees (RRTs) that works on a combined symbolic and sub-symbolic space.

RRTs are extensively exploited algorithms for motion planning in mobile and manipulation robotics [14] since they permit an efficient search in non convex, high-dimensional spaces. On the other hand, the deployment of these methods for high-level task planning has been only partially investigated in the literature. RRTs algorithms applied to discrete spaces have been proposed and explored by [15], however the problem of symbolic task planning was not considered. As far as symbolic planning is concerned, an interesting approach is provided by [16], where RRTs are adapted to solve AI planning problems encoded in the STRIPS (Stanford Research Institute Problem Solver) representation [17]; on the other hand, this work does not address the combination of task and motion planning. An embedding of the symbolic state into the continuous space is proposed in [10] to enable sampling-based planning methods extended to symbolic, geometric, and

* Corresponding author.
E-mail addresses: riccardo.caccavale@unina.it (R. Caccavale), alberto.finzi@unina.it (A. Finzi).

https://doi.org/10.1016/j.robot.2022.104238
0921-8890/

kinematic constraints. Here, combined task and motion planning is addressed, but task planning is primary since a task planning heuristic is exploited as a long-horizon symbolic guidance through the search space. Another relevant approach is proposed by [11], where RRTs are deployed for motion planning in the configuration space, while symbolic task planning is exploited to support motion planning by providing actions and regions of the continuous space that the sampling-based motion planner can further explore to advance its search. In this case, symbolic and motion planning processes are associated with different search strategies and an external black-box symbolic planner is invoked to compute an action plan.

In this paper, we propose and investigate a different approach to combined task and motion planning, which directly applies the RRT-based algorithm to an extended search space which combines the configuration space and the symbolic state space. Our aim is to explore the extent to which a direct deployment of a basic RRT sampling-based search is feasible and effective in this extended space. Since the RRT needs a metric space to be sampled, we firstly introduce a distance measure suitable for an extended state space that combines configurations and symbolic states; we then deploy this distance to guide the expansion of the RRT in the extended metric state.

Notice that in this work, for the sake of simplicity, we deliberately deploy a vanilla version of the RRT search [14] to assess the sampling-based search mechanism, without the support of other optimizations (e.g., [18]). We detail the approach and discuss its feasibility and scalability in real-world robotic case studies provided by a hospital logistic scenario. We consider incrementally complex pick, carry, and place tasks in both uncluttered and cluttered environments. The results collected with the proposed pilot study suggest that the approach is feasible in realistic scenarios, while its effectiveness is more emphasized in complex and cluttered environments.

This work presents an extended and further detailed version of the framework introduced in [19] at the IEEE European Conference on Mobile Robots (ECMR 2021). In this version we provided a more comprehensive description and explanation of the overall framework; we extended the discussion of related literature; we extended the experimental evaluation and discussion considering additional scenarios where the performance of the proposed system is compared with respect to a baseline approach with decoupled task and motion planning.

2. Related works

Combined task and motion planning have been explored from different perspectives [9]. From the motion planning side, task planning has been deployed to support the motion planning search. For instance, the SamplSGD framework [11] exploits symbolic action planning to find discrete actions and regions of the continuous space that a sampling-based motion planner can selectively explore to generate feasible motion plans. A sampling-based approach is also used in ASyMov [6]. In this case, cost functions are deployed to decide whether to expand the roadmaps of already included actions or to call a task planner to add new actions starting new roadmaps. Following a different approach, in [5] a motion planner is deployed to generate a tree in the configuration space along with an associated task planning domain. The latter is exploited to generate a symbolic plan, which is then converted into the configuration space. In [8], the authors address multi-modal planning considering both configurations and modes. In this case, the authors propose a hierarchical algorithm along with a bidirectional RRT-based search for the combined space. A multi-modal sampling-based method for combined task and motion planning is also proposed by [20], where an extension of dRRT [21] is deployed to explore the composite space of multi-arm task and motion planning problem. In other works task planning is primary, while motion planning is selectively invoked to assess geometric and dynamic constraints. In this direction, in [7] the task planner operates in an abstracted state space, while a suitable interface is introduced to communicate to the task level the geometric constraints found in the continuous state. Geometric backtracking for task and motion planning are investigated by [9] comparing two approaches: heuristics based on geometric conditions to guide the search; geometric constraints to prune the search space. Along this lines, PDDL extensions with semantic attachments enabling specialized motion method have been proposed and investigated by [22]. Combined task and motion planning is also achieved through the interleaved task and motion planners [2,3,12,13]. For instance, in Erdem et al. [3] the task planner is used to guide a motion planner that attempts to generate a continuous motion for each robot. Another iterative approach is provided by [2], here incremental constraint solving is exploited to iteratively provide motion feasibility information into the task planner. In the framework by [12], an interface is proposed to enable an effective interaction between task and motion planners. An interface-based approach is also proposed by [13], in this case the aim is to provide a planner-independent layer to connect task and motion planning.

Differently from these approaches, our method relies on a unified sampling-based exploration of an extended space that combines continuous and abstract states. Closely related to our work, a sampling-based method to search in the symbolic space has been proposed by [16]. Analogously to our approach, here, a STRIPS-like planning domain representation is adopted, while a distance measure for the symbolic states is introduced and exploited by an RRT. However, this method is only applied at the task level, while combined task and motion planning has not been tackled. Sampling-based planning methods extended to symbolic representations and motion constraints are proposed in [10]. In this case, the symbolic planning problem is embedded into the continuous space, while suitable functions are introduced to guide the motion planner towards regions of the configuration space where symbolic actions can be executed. In this space a motion planner is deployed to generate a motion plan addressing all the embedded constraints. In contrast, we define a combined space associated with a novel measure of combined distance that enables us to generate both the symbolic and the motion plan. Related with our approach, multi-modal methods [8,20] propose sampling-based search in a combined task and motion space, however, symbolic representations of the planning domain are not employed.

3. RRT-based task and motion planning

In this section, we introduce our Task and Motion RRT planner (TM-RRT) obtained as an extended version of a simple RRT-based planner. First of all, we define our combined task and motion planning problem, which can be defined by the tuple $(C, S, M, A, q_{init}, s_{init}, s_{goal}, G)$, where $C \subseteq \mathbb{R}^n$ is the n-dimensional configuration space of the robot, S is the symbolic state space, M and A are sets of motions and symbolic actions respectively, $q_{init} \in C$ and $s_{init} \in S$ are the initial configuration and initial symbolic state, and $s_{goal} \in S$ is the symbolic goal state. Each state in the symbolic space S is implicitly represented by means of a finite set of ground predicates in a set P representing all the possible properties that can hold in a state (with closed-world assumption), i.e., $S = 2^P$. For instance, the symbolic state of a mobile robot involved in pick-and-place tasks can be symbolically described by the predicates $at(Pose)$, $on(Obj, Pose)$, $holding(Obj)$, representing, respectively, the pose of the robot, the disposition

of objects, the object held by the robot. In this setting, a symbolic state can be described as a list of fully instantiated predicates, e.g., $at(p_1)$, $holding(obj_1)$, $on(obj_2, p_2)$, $on(obj_3, p_3)$ describing a robot in p_1 holding obj_1, while other objects are available in p_2 and p_3. Notice that, following the closed-world assumption, a state is represented by the set of instantiated predicates that holds, while not mentioned proprieties are assumed as negated.

In this framework, we represent motions and actions. A motion m is a n-dimensional vector of movements that linearly drives the robot from a configuration to another. On the other hand, an action $a \in A$ is a STRIPS-like symbolic operator [17], which is associated with preconditions $pre(a)$, add-list and delete-list $eff(a)^+$, $eff(a)^-$, specifying the transition between two symbolic states (preconditions and effects are represented by sets of ground predicates in P). For instance, the robot action $pick(Obj, Pose)$ of picking an object Obj from $Pose$ can be executed (precondition) if and only if the robot is close the position of the object ($pre(a)$ holds iff $at(Pose)$ and $on(Obj, Pose)$); on the other hand, the action positive effect is that the agent holds the object ($eff(a)^+ = \{holding(Obj)\}$), while the negative effect is that the object is not anymore in the previous pose ($eff(a)^- = \{on(Obj, Pose)\}$).

Moreover, in order to assess action accomplishment in the configuration space, we introduce a function $G : A \times S \rightarrow C$ such that, given a symbolic action a applied in the symbolic state s, it provides a target configuration $q \in C$ representing the configuration to be reached by the robot in order to finalize a. Such function is part of the domain specification that can be defined using symbolic references to configurations [6,9,13], which provide an interface between the symbolic and the configuration space. Following this approach, target configurations can be obtained from referenced terms (e.g., poses) mentioned as arguments in actions and predicates of the state. For instance, the action $pick(o_1, p_1)$ applied in the symbolic state $s = \{on(o_1, p_1), at(p_1)\}$ can be associated with the target configuration q_{p_1} obtained from the geometric pose of a symbolic term p_1 representing the location for that object (e.g., $G(pick(o_1, p_1), s) = q_{p_1}$). Notice that this approach is similar to the one adopted by [13], where predicate arguments for real values range not over continuous values, but over finite sets of symbolic references to configurations.

In this context, we want to find a sequence of motions and actions that can be executed by the robot in order to reach the desired symbolic state s_{goal}, starting from the initial configuration q_{init} and the initial symbolic state s_{init}. To this end, we aim at producing a list of action and motion pairs $\pi = \langle (a_1, m_1), \ldots, (a_k, m_k) \rangle$, where each motion specifies a movement to be performed in the configuration space by the robot to accomplish the associated action. In this setting, a plan is a list of motions, each labeled with an action representing the symbolic motivation for that motion.

For instance, consider a person that aims at opening a bottle of water. At the task level, the action can be decomposed in two actions: *grasp-bottle* (a_1) and *open-bottle* (a_2). The first one is needed to hold the bottle in position, while the second one permits to remove the cap from it. On the other hand, at the motion level, the first action can be executed by moving the left arm towards the bottle (m_1) and closing the left hand to hold it (m_2), while the second action execution is composed of a first right hand motion towards the cap (m_3) followed by another one performed to remove it (m_4). In this case, the first two motions are performed in order to accomplish the *grasp-bottle* action, while the others are needed to accomplish the *open-bottle* action. Therefore, in the final plan π we will have four pairs, one for each motion, and two for each action, that is $\pi = \langle (a_1, m_1), (a_1, m_2), (a_2, m_3), (a_2, m_4) \rangle$. Notice that in this plan each pair represents a motion labeled by a symbolic action (e.g., $m1$ and $m2$ are labeled by a_1), hence each symbolic action is implemented by the associated motions (e.g., a_1 is executed by the motions m_1, m_2). It is also worth noticing that symbolic actions may also interleave at the motion level (e.g., an action a_1 with motions m_1, m_2 interleaved by the action a_2 with motions m_3, m_4 can be represented by a plan $\langle (a_1, m_1), (a_2, m_3), (a_2, m_4), (a_1, m_2) \rangle$).

3.1. An extended RRT

The RRT algorithm [14,23] is a sampling-based planning method that provides a motion plan by rapidly generating a tree rooted at the starting configuration until the goal configuration is reached. As previously mentioned, we introduce our approach considering a basic version of the RRT algorithm, which will be suitably adapted to handle both task and motion planning. Notice that more sophisticated algorithms based on RRTs can be similarly adapted and deployed [18,24–27].

Following the formulation by [23], the class of problems considered by RRTs can be defined by the following elements: the *state space* X; the *boundaries*, representing initial and goal configurations $x_{init} \in X$ and $X_{goal} \subset X$ respectively; a *collision detector* $D : X \rightarrow \{true, false\}$ checking whether or not a state is accessible; a set of *control inputs* U (motions); an *incremental simulator* which predicts the state transition over a time interval given the current state and an input; a *metric* $\rho : X \times X \rightarrow [0, \inf[$ specifying the distance between states.

In this section, we show how combined task and motion planning problems can be included in this class in order to be solved by RRT-based methods.

In order to adapt the above RRT problem formulation to our setting, we introduce a combined state space $X = C \times S$ (i.e., configurations and symbolic states) and a combined set of inputs $U = M \times A$ (i.e., motions and actions). Starting from these sets, we can define as boundary values $x_{init} = (q_{init}, s_{init})$ (i.e., the combined initial state) and $X_{goal} = (\cdot, s_{goal}) \subset C \times S$ (i.e., the combined goal state), and a collision detection function $D : C \times S \rightarrow \{true, false\}$ that checks for constraints in the combined state. In this formulation, inconsistent symbolic states can be treated analogously to obstacles to be avoided along the path.

As for the state transition, we define an incremental simulator such that, given a combined state $x(t) = (q(t), s(t))$ and a combined input $u(t') = (m(t'), a(t'))$, with $t' \in [t, t + \Delta t]$, the new state $x(t + \Delta t)$ obtained by applying u is computed as follows:

$$x(t + \Delta t) = \begin{cases} (q(t + \Delta t), s(t)) & \text{if } q(t') \neq G(a(t'), s(t)) \\ (q(t + \Delta t), s(t + \Delta t)) & \text{otherwise} \end{cases} \quad (1)$$

That is, if q is not the target configuration associated with the symbolic action a, only the configuration q is updated by the m motion in u, otherwise, if the target configuration has been reached, the symbolic state s is updated as well since a has been accomplished. In this case, the updated symbolic state is obtained exploiting $eff(a)^+$ and $eff(a)^-$, i.e., $s(t + \Delta t) = (s(t) \setminus eff(a)^-) \cup eff(a)^+$.

In order to deploy a RRT-based algorithm in the combined state, we have to introduce a suitable notion of distance, which enables us to obtain metric space. We recall that a *metric space* is defined by the couple (X, d), where X is a non empty set of elements and d is a metric on X, i.e., a distance function between any two elements in X. In our setting, X is the combined state, while as a metric we introduce a distance d_u on X obtained as a weighted sum of two distance functions: the distance d_c in the configuration space C and a distance d_s on the symbolic space S. As for d_c, we assume any metric in the configuration space C, while d_s requires a non-geometric distance associated with

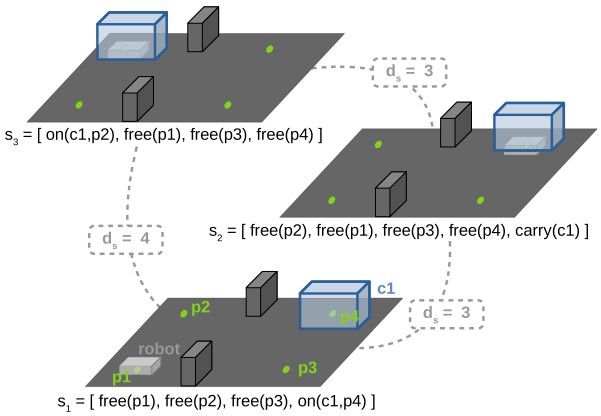

Fig. 1. Example of the symmetric distances between 3 symbolic states: s_1, s_2 and s_3. Here, a mobile robot can move towards different positions p_1, p_2, p_3, p_4 in order to pick, carry, and place the cart c_1.

sets of symbols. Typical symbolic distances used for sequences of symbols are Hamming, Levenshtein distances, etc. Since we aim at comparing symbolic states, we rely on the notion of *symmetric difference* of two sets. More specifically, the distance d_u is defined as follows:

$$d_u((q', s'), (q'', s'')) = w_c i + w_s j \tag{2}$$

where $w_c, w_s \in \mathbb{R}^+$ are two positive and non-zero constant values introduced to weight the configuration distance i and the symbolic distance j, which are defined as follows:

$$i = d_c(q', q'') \tag{3}$$

$$\begin{aligned} j &= d_s(s', s'') = \\ &= |s' \triangle s''| = |(s' \cup s'') \setminus (s' \cap s'')| \end{aligned} \tag{4}$$

Here, the distance d_c is for any distance between two configurations q' and q'' (i.e., the one used to expand the RRT in the configuration space), while the symbolic distance is defined as the cardinality of the *symmetric difference* of the two symbolic states s' and s''. Since symbolic states are represented by sets of ground predicates, this cardinality provides a notion of proximity among the states. An exemplification of this notion is provided in Fig. 1, where three symbolic states s_1, s_2, s_3 along with their distances are illustrated.

We now can show that d_u is a distance in the metric space (X, d_u).

Proposition 1. *The function d_u is a distance in the metric space (X, d_u).*

Proof. We recall that a function $d : X \times X \to \mathbb{R}$ is a distance in X iff (i) $d(x, y) \geq 0$; (ii) $d(x, y) = 0 \iff x = y$; (iii) $d(x, y) = d(y, x)$; (iv) $d(x, y) \leq d(x, z) + d(y, z)$.

First of all, we can observe that if both d_c and d_s are distances in C and S respectively, then (i), (ii), (iii), (iv) hold also for d_u in $C \times S$ since d_u is a weighted summation of d_c and d_s, with non-zero positive weights. Moreover, we have also that d_c is a distance on the configuration space by assumption. Finally, we can shown that the cardinality of the symmetric difference satisfies all the distance properties: (i), (ii), (iii) are trivial, while the triangular inequality follows from $s_x \triangle s_y \subseteq (s_x \triangle s_z) \cup (s_z \triangle s_y)$ along with $|s_x \triangle s_y| \leq |(s_x \triangle s_z) \cup (s_z \triangle s_y)|$ and $|(s_x \triangle s_z) \cup (s_z \triangle s_y)| \leq |(s_x \triangle s_z)| + |(s_z \triangle s_y)|$. □

3.2. RRT-based task and motion planner

Given the distance measure introduced in the previous section, we now aim at introducing a RRT algorithm that spans towards the unified metric space (configurations and symbolic) in search for a plan that is both collision-free and symbolically consistent with respect to the task planning representation.

Algorithm 1 TM-RRT algorithm

```
1: function BUILD_TMRRT(x_init, s_goal)
2:     T ← addRRT(T, x_init)
3:     while ¬ reached(T, s_goal) do
4:         x_rand ← random_state(T, s_goal)
5:         T ← extend(T, x_rand)
6:     end while
7:     return T
8: end function
```

The TM-RRT algorithm that combines task and motion planning is illustrated in Algorithm 1. It receives in input the initial state in the combined space $x_{init} = (q_{init}, s_{init})$ along with a symbolic goal state s_{goal} and generates a RRT whose branches are associated with task-level symbolic actions and free-obstacle motions. The output of Algorithm 1 is the tree structure itself, since the plan can be suitably retrieved by going backward from the solution vertex (leaf) to the initial vertex (root). The algorithm works as follows. The tree is firstly initialized to the initial configuration (line 2) then, until the goal state s_{goal} is reached (line 3), the algorithm randomly selects a combined state (line 4) and extends the tree in that direction (line 5). Finally, the tree containing the goal state is given as output (line 7). Notice that, in contrast with the basic version of the RRT by [14], this algorithm stops when a feasible path connecting the initial and the goal states is found. Moreover, our algorithm also incorporates a suitable *random_state* function in which symbolic actions are exploited to guide the sampling process towards the goal state.

Algorithm 2 TM-RRT random sampling

```
1: function RANDOM_STATE(T, s_goal)
2:     s_rand ← random_sym(p_s, s_goal)
3:     s_near ← nearest_sym(T, s_rand)
4:     a ← select_action(s_near, s_rand)
5:     q_sub ← G(a, s_near)
6:     q_rand ← random_conf(p_c, q_sub)
7:     return (q_rand, s_rand)
8: end function
```

The description of the *random_state* function is illustrated in Algorithm 2. In a first phase, a symbolic state s_{rand} is drawn from the space S (line 2). To bias this sampling towards the goal state, we adopt a randomized choice by selecting the goal state s_{goal} with probability p_s along with a randomly extracted symbolic state (i.e., a subset of P as in [16]) with probability $1-p_s$. The selected state s_{rand} should induce a tree expansion in its direction. At the symbolic level, this expansion is performed by a symbolic action. In this respect, given s_{rand}, the algorithm selects the nearest state s_{near} of the tree (line 3) and selects an action a from the state s_{near} towards s_{rand} (line 4).

In a second phase, we exploit the G function (line 5) to retrieve the target-configuration q_{sub} associated with the symbolic action a executed in s_{near}. Notice that, the q_{sub} is here exploited as a guidance that drives motion-level planning towards a trajectory that accomplishes a (sub-goal accomplishment). Also in this case, a randomized choice is introduced to obtain this drive: with probability p_c the drawn configuration q_{rand} equals the sub-goal q_{sub}, otherwise, with probability $1 - p_c$ a new random configuration is sampled, with a uniform distribution over C (line 6). Once we

Algorithm 3 TM-RRT extension

1: **function** EXTEND($\mathcal{T}$, x)
2: $x_{near} \leftarrow$ nearest_neighbor($\mathcal{T}$, x)
3: $(u_{new}, x_{new}) \leftarrow$ new_state(x_{near}, x, len)
4: **if** check(u_{new}, x_{new}) **then**
5: $\mathcal{T} \leftarrow$ add_vertex($\mathcal{T}$, x_{new})
6: $\mathcal{T} \leftarrow$ add_edge(x_{near}, x_{new}, u_{new})
7: **end if**
8: **return** $\mathcal{T}$
9: **end function**

have both q_{rand} and s_{rand}, a new combined sample (q_{rand}, s_{rand}) in the unified space is provided in output. Upon the random selection of the state, the tree is expanded in that direction. This process is described in Algorithm 3. Analogously to the basic version of the RRT [23], the *extend* function takes as input the tree $\mathcal{T}$ and the random state x. The function selects the nearest node of the tree from the random state x according to the combined distance d_u (line 2); then, a combined input $u_{new} = (m, a)$, with m of maximum length len, is selected and applied to the nearest state x_{near} in order to reach a new state x_{new} (line 3). This state generation process is managed by the *new_state* function that exploits Eq. (1) to manage the state transition, symbolic action preconditions and effects, and the collision function D. Specifically, in order to enable the expansion, the selected symbolic action a should be executable from the current state (i.e., $pre(a)$ satisfied in s_{near}), if this is the case, the new motion m with length len in the direction of x can be explored (while monitoring collisions with D). Notice that, if the new configuration state q_{new} is a target state for the action a (i.e., $q_{new} = G(a, s_{near})$) then a symbolic state transition is performed to generate a new symbolic state s_{new} by applying the effects $eff(a)^+$ and $eff(a)^-$ of the action a to the symbolic state s_{near} (i.e., $x_{new} = (q_{new}, s_{new})$ with $s_{new} = (s_{near} \setminus eff(a)^-) \cup eff(a)^+$). Otherwise, if the motion m is not colliding and the configuration state is not terminal (i.e., $q_{new} \neq G(a, s_{near})$), only the transition in the configuration space is performed (i.e., $x_{new} = (q_{new}, s_{near})$). On the other hand, if the motion m leads to a collision or the selected action a is not executable in s_{near}, then an empty new state is returned, which is eventually rejected by the following *check* function (line 4). Finally, when a new consistent state is found the tree is expanded by adding a new node x_{new} to $\mathcal{T}$ (line 5) and a new edge (line 6).

4. Case studies

In this section, we discuss our method in different case studies provided by a hospital-logistic scenario (see Fig. 2 (right)), where a mobile robot is involved in multiple pick, carry, and place tasks. In particular, three set-ups will be considered. Firstly, we will discuss the performance of the proposed algorithm with respect to different combinations of parameters regulating its functioning. We then propose two additional and incrementally complex set-ups. The first one is uncluttered, hence almost all symbolic actions are executable from a motion perspective. In this case, the combined task and motion planning problem is simplified. The second set-up is cluttered, therefore the environmental configuration often impairs the execution of the actions and different coherent symbolic plans must be explored to generate the associated executable motions. In the two latter cases, the performance of the proposed planner is also compared with respect to a baseline approach, where task and motion planners are decoupled. Specifically, the aim is to compare the performance of the proposed method with respect to a standard 2-layered approach to task and motion planning, where a symbolic task planner is firstly invoked to solve the high-level planning problem generating a plan of symbolic actions while, in the second layer, a motion planner provides collision-free paths to execute the primitive actions of the generated task plan. To adopt a general and basic baseline, the task plan is directly generated by a simple Breadth-First Search (BFS) in the action space, while a basic RRT algorithm [14] is exploited to expand primitive actions in the configuration space.

All the experiments have been carried out in a realistic *CoppeliaSim* simulated environment of 10 × 10 m, deploying a laptop *Intel i5-5200U 2.20* GHz, *8Gb ram*, with a single threaded implementation of the algorithm[1]. An example of the simulated environment is provided in Fig. 2, where we can find a mobile robot (gray, in simulation), two carts (cyan) and four target poses (green squares). The robotic system is an omni-directional mobile platform endowed with four mecanum wheels and an elevator that allows it to go under the carts and lift them from below. We assume the robot localized within a complete map of the environment. The configuration of the environment (rooms, passages, number of carts and poses) will be changed in the simulated experiments in order to generate incrementally complex settings.

In this scenario, we assume the robot configuration space $C \subseteq \mathbb{R}^2 \times SO(1)$, hence each $q \in C$ is a triple $q = [x, y, \theta]$, where x and y are coordinates and θ is the rotation. The set A includes two types of actions, $pick(Cart, Pose)$ and $place(Cart, Pose)$, while the symbolic state is described by the predicates: $free(Pose)$ that holds if the $Pose$ is free, $carry(Cart)$ which holds if the robot is carrying $Cart$, $on(Cart, Pose)$ stating that $Cart$ is in $Pose$, and $carrying$, that holds if the robot is moving a cart. Both predicates and actions can be instantiated with carts and target-poses, hence the number of ground actions and predicates can be defined by the number of carts/target-poses. As for the G function, it is defined exploiting the configurations associated with the symbolic poses. Specifically, we state that $G(a, s) = ref(target(a, s))$, where $target(a, s)$ provides the symbolic target p associated with the action a in s, while $ref(p)$ provides the configuration $q_p \in C$ associated with the symbolic term p (e.g., $G(pick(cart2, p_0), s)$ is defined by $target(pick(C, P), S) = P$ and $ref(p_0) = q_{p_0}$).

4.1. Case 1: Parameters setting

The TM-RRT set-up depends on 5 main parameters: the probabilities p_c and p_s (see lines 2 and 6 in Algorithm 2) that regulate the goal-oriented sampling of the RRT, the length of the motion segment towards the random sample len (see line 3 in Algorithm 3), and the weights w_c and w_s introduced to define the unified distance d_u (see Eq. (2)). As a preliminary step, our aim is to assess the performance of the planner by changing the parameters that structurally affect the underlying metric space and the way it is explored. In particular, as parameters we consider the ratio between the two weights w_c and w_s used to define the unified distance d_u along with the len expansion step of the tree. In contrast, the probabilities p_c and p_s are both fixed to 0.3 in order to compare the results with the same fixed search strategy.

In the following tests, we assume $w_c = 1$ and $w_s = kw_c$ and the ratio $k \in \{0.1, 0.5, 1, 2, 5, 10\}$, and len ranging in $[0.1 - 1]$ with an increasing step of 0.1. The environment used for testing is illustrated in Fig. 2 (left), where a mobile robot that can move, pick, carry and place carts in two areas: a small room with 2 entrances and a bigger outer area. In this setting, we deploy 2 carts (cyan objects c_1 and c_2) and introduce 4 target poses (green squares p_1 to p_4). As for the task, the goal for the robot is to place c_1 in p_3 and c_2 in p_4. Notice that one of the carts (c_1) is

[1] The open-source version of the code can be downloaded from https://github.com/ri-caccavale/tm_rrt

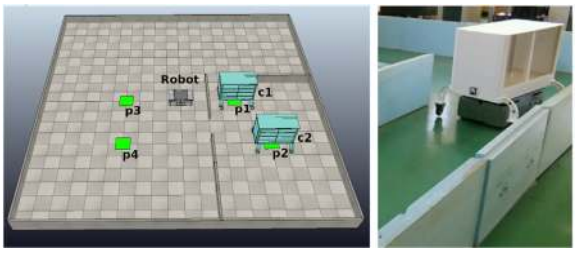

Fig. 2. Example of simulated (left) and real (right) hospital environment. In the simulation 2 carts are deployed ($c1$ and $c2$) and one of them ($c1$) is blocking the entrance (for interpretation of the references to color in this figure legend,the reader is referred to the web version of this article).

intentionally positioned close to the bigger entrance in order to obstruct the passage. This way, the mobile robot can still access the room through the small entrance, but a free passage is needed anyway to carry out the cart c_2. This environment represents a simple "trap" since all the plans starting with the $pick(c_2)$ action cannot be completed with *carry* and *place* because the only exit from the small room, available for c_2, is blocked. Instead, in order to solve the task, the robot firstly has to pick carry and place c_1 from pose p_1 to pose p_3, then cart c_2 can be moved from pose p_2 to pose p_4.

In this scenario, we tested the algorithm by performing 30 execution for each combination of the parameters. We also introduced a time-limit of 300 seconds considering failures all the executions exceeding this limit. For each couple of parameters (len, k) we measured the time needed to generate the plan along with its size (average and standard deviation) and the failure rate. The results are illustrated in Fig. 3. It is possible to notice that the failure rate decreases rapidly when the weight w_s of the symbolic distance becomes two or more times greater than w_c. In this respect, we can also observe that this regulation emphasizes the symbolic component along with the guidance of the symbolic actions towards the symbolic goal. Analogously, the planning time also decreases with the increment of both k and len. In contrast, the size of the plan increases with the increment of len. This results was expected since longer paths are generally associated to fast, but inefficient routes, while shorter paths are more detailed but computationally expensive. On the other hand, the plan size is still slightly decreasing with the increment of k. These collected results suggest that settings where $w_c > w_s$ are self-defeating (higher failure rate), while the len parameter can be regulated to trade-off between efficiency and quality of the solution.

This preliminary analysis enabled us to select a suitable parameter regulations for the other case studies. Specifically, we selected the set-up associated with no failures and minimal execution time (see green line in Fig. 3), namely $len = 0.9$ and $k = 5$.

4.2. Case 2: Straight uncluttered environments

We now consider the scalability of the proposed approach with respect to the number of actions and ground predicates (i.e., the size of the sets A and P). In this case study, the performance of our TM-RRT algorithm is also compared with respect to a 2-layered BFS+RRT baseline algorithm, where task and motion planning are decoupled. In order to compare the approach with a general baseline avoiding planner-specific heuristics, we deploy a straightforward BFS algorithm to generate the task plan, while a RRT-based algorithm generates a motion plan for each symbolic action in the task plan. To get a comparable method, such RRT search is the one of TM-RRT restricted to the configuration space.

Table 1
Results of Case 2 (60 runs for each setting).

Number of carts			1	2	3	4
TM-RRT	Tot. time (s)	avg	0.22	1.31	11.80	39.27
		std	0.32	1.46	10.61	48.85
	Length (m)	avg	12.66	32.77	55.26	80.61
		std	6.64	8.51	12.07	11.88
	Plan-size	avg	142	364	625	906
		std	77	97	143	151
	Success		100%	100%	100%	93%
BFS+RRT	Tot. time (s)	avg	0.43	2.06	7.15	29.30
		std	0.78	2.74	6.13	11.14
	RRT time (s)	avg	0.43	2.05	6.87	8.68
		std	0.78	2.74	5.85	6.77
	Length (m)	avg	11.36	38.80	62.76	89.17
		std	6.38	9.23	11.82	13.02
	Plan-size	avg	130	429	707	992
		std	78	106	124	146
	Success		100%	100%	100%	100%

As for the BFS, on the one hand, it provides symbolic plans with minimal length, on the other hand, no heuristic is exploited for the search, hence plan generation is not efficient. Notice, however, that the baseline is here exploited to obtain a sort of empirical upper-bound for the temporal performance of a task planner. Moreover, we can estimate an empirical lower-bound performance by disregarding task-planning time in so assuming instantaneous task planning.

Regarding the baseline, it is worth also noticing that the two-layered planning performance is affected by two factors: the order in which the BFS explores the possible symbolic plans and the timeout used by the lower-level RRT to estimate when a target pose is unreachable. In this respect, in our tests the RRT timeout is set to 2 seconds, while we shuffle randomly the order of the symbolic actions before every test.

We defined a simulated environment with 2 rooms and 3 passages between rooms that are large enough to allow cart transitions (see Fig. 4). In this scenario, the robot has to move carts from their positions in the right-room to the fixed position in the left-room (e.g., from p_1 to p_2 in Fig. 4, first on the left). We considered 4 configurations of this scenario with an increasing number of carts (from 1 to 4) and target-poses (from 2 to 8). The increment of tasks/poses induces an increment of the available ground actions and predicates as illustrated in Fig. 5.

For each setting, we performed 60 runs (240 tests in total), measuring planning-times (in seconds), size of the plans (number of steps), paths length (meters) and success rate (percentage of plans generated in less than 300 seconds).

In Table 1 a comparison between our TM-RRT approach and the BFS+RRT baseline is proposed. In particular we monitored planning times, length of the overall solution (in meters), the plan-size representing the number of RRT nodes in the solution and the success rate. Moreover we also report the RRT planning time for the baseline by disregarding the time spent by the BFS planner from the total planning time. This additional measure provides an insight about the baseline performance independently from the employed symbolic search strategy.

It is possible to notice that, as long as the number of carts increases, the performance of the baseline is slightly better in terms of success and execution time, while it is slightly worst in terms of length and plan-size. This is particularly relevant in the 4-carts setting, where the total time of the baseline is 25% shorter than TM-RRT (or about 80% shorter if we assume instantaneous task planning/replanning), with 7% increased success rate (always successful), while the TM-RRT provides more efficient solutions that are about 10% and 11% shorter in terms of plan-size and length respectively.

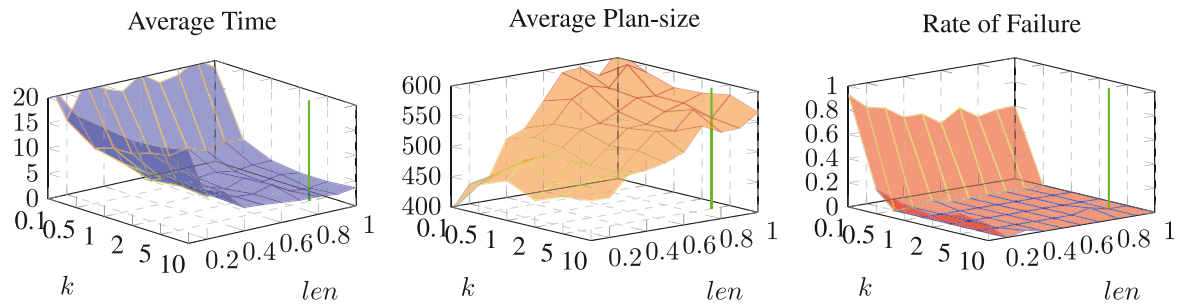

Fig. 3. Representation of the average times (seconds), plan-size (steps) and the rate of failures for each couple of parameters over 30 runs. For the next cases we select the parameters producing no failures and minor average time (green line) (for interpretation of the references to color in this figure legend,the reader is referred to the web version of this article).

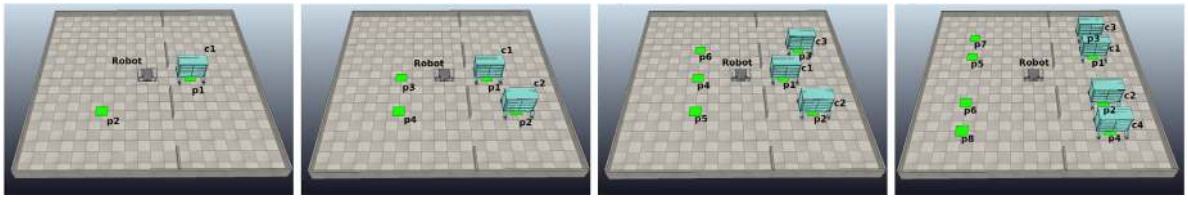

Fig. 4. Simulated environments for each setting from 1 (left) to 4 (right) carts. In all these settings the robot carries carts from the right room to the target-poses in the left room. There are three passages between rooms.

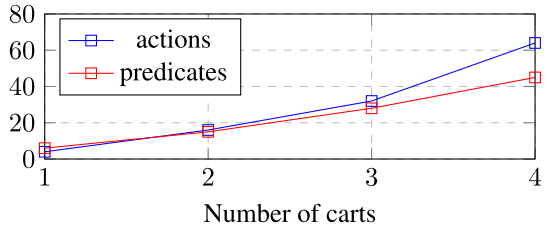

Fig. 5. Number of actions and ground predicates for each domain.

4.3. Case 3: Anomalous cluttered environments

In order to assess our approach in case of anomalies, we now propose variations of the above hospital logistic scenario where the carts configuration is intentionally designed to hinder task execution. More specifically, we assume that the simulated environment includes a small room, in which a variable number of carts is positioned, and an outer area with different target positions (see Fig. 6, up). The goal is to move specific carts from the room to the target poses in the outer area, leaving the uninvolved carts to their current positions (see Fig. 6, down). In particular, we propose the following 5 incrementally complex settings in this scenario.

- Setting 1: 2 carts and 4 poses, where both carts have to be moved from the room to the outer area.
- Setting 2: 2 carts and 4 poses, where only the inner cart c_2 have to be moved from the room to the outer area.
- Setting 3: 3 carts and 6 poses, where only the middle cart c_2 have to be moved from the room to the outer area, while the cart c_3 is used as a distraction to increment the size of the planning problem.
- Setting 4: 3 carts and 6 poses, where both the middle and the inner carts c_2 and c_3 must be moved from the room to the outer area.
- Setting 5: 3 carts and 6 poses, where only the inner cart c_3 must be moved from the room to the outer area.

Notice that, in this environment, the left room is designed to be narrow enough to allow the transportation of only one cart per time, forcing the robot to carry outside carts in a specific order: from the outer ones (close to the entrance of the room) to the inner ones (far from the entrance). As a result, in order to move outside one of the inner carts, all the outer blocking carts must be temporarily removed from their initial position. For example, let consider the setting 2 (Fig. 6, second from the left). Here the goal is to move the cart c_2 from the pose p_2 inside the room to the pose p_8 outside. In order to free the passage for the target cart, the robot has firstly to move the blocking cart c_1 to the neutral pose p_4 and then to move it back to p_1 once the target cart is placed. This configuration is designed to stress the combined task and motion planning approaches by providing coherent task plans that cannot be associated to any collision-free motion plan.

The collected empirical results are illustrated in Table 2. Differently from the previous case, as the problem complexity increases, the performance of the proposed TM-RRT framework strongly outperforms the baseline in terms of success rate and planning time even when the BFS time is neglected, as also illustrated in Fig. 7. For example, in Setting 4 the proposed approach has a 100% success rate while the baseline can rarely solve the problem within the deadline (17% of runs). Additionally, in Setting 5 the baseline is always unable to solve the problem within the deadline (0% success rate), while the proposed approach has a success rate of 83% with an average running time of 174 seconds. We can also observe that the baseline tends to generate shorter plans in cluttered environments. In these cases, once a geometrically feasible symbolic plan is obtained by the baseline, short-range RRT expansions can efficiently plan the motions of each action in the plan. In contrast, the TM-RRT planner exploits long-range RRT expansions in the combined space, hence the generated solutions can be less efficient when the combined planning problem is more complex.

4.4. Discussion

In the previous sections, two different incrementally complex cases studies have been proposed to illustrate the feasibility of our approach and compare its performance with respect to a

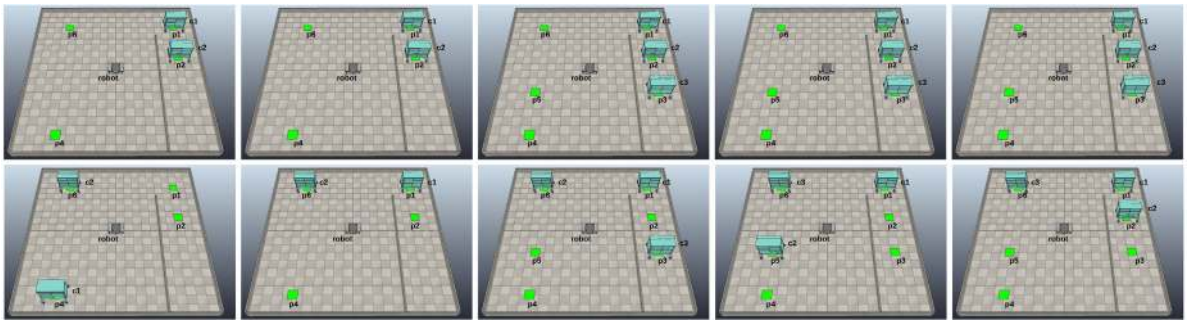

Fig. 6. Simulated environments for each setting from 1 (left) to 5 (right), showing the initial states (up) and the goal states (down). In all these settings the robot carries carts from the right room to the target-poses on the left. In order to carry the inner carts, the outer carts must be removed.

Table 2
Results of Case 3 (60 runs for each setting).

Test id			1	2	3	4	5
TM-RRT	Tot. time (s)	avg	11.45	24.29	17.95	87.60	174.09
		std	28.50	31.90	15.69	49.29	66.36
	Length (m)	avg	54.39	81.43	76.52	157.05	171.25
		std	5.16	7.18	12.85	24.50	21.88
	Plan-size	avg	662	1009	953	1959	2137
		std	97	137	167	295	291
	Success		100%	100%	100%	100%	83%
BFS+RRT	Tot. time (s)	avg	8.61	46.46	89.15	132.66	N/A
		std	10.41	35.09	76.25	7.07	N/A
	RRT time (s)	avg	8.61	46.43	88.62	130.64	N/A
		std	10.40	35.07	75.71	7.03	N/A
	Length (m)	avg	52.62	77.71	73.69	122.06	N/A
		std	3.37	4.41	8.08	8.51	N/A
	Plan-size	avg	615	919	874	1423	N/A
		std	56	75	108	70	N/A
	Success		100%	100%	80%	17%	0%

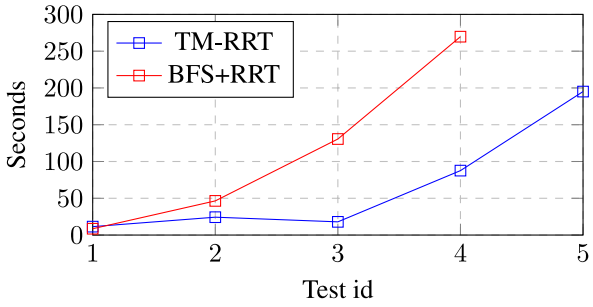

Fig. 7. Comparison between the TM-RRT total planning time and the planning time of the baseline excluding the BFS. Unsuccessful runs are included with a penalty of 300 seconds (time limit for each run).

simple baseline method where task and motion planning are decoupled. In this respect, the benefit of the proposed TM-RRT algorithm with respect to the 2-layered baseline emerges as the environment becomes more cluttered and anomalous. As expected, if the environment is mainly uncluttered (case 2), generated symbolic plans are hardly impaired by physical obstacles, hence, decoupled task and motion planning can be very efficient, while the advantage of our combined approach becomes less evident. Notice however that in this paper we proposed a basic version of the algorithm to show the feasibility of the approach, in order to improve its performance more sophisticated RRT-based search strategies can be deployed along with more refined implementation of the algorithm. On the other hand, in complex and cluttered environments (case 3) where coherent task plans frequently fail kinodynamic constraints check, the combined approach proposed in this paper starts to show its potential. This is particularly evident in Fig. 7, where the time performance of our approach outperforms the proposed baseline even when we assume an high-level task planner capable of instantaneous generation of minimal plans.

5. Conclusions

We presented a sampling-based approach to combined task and motion planning that relies on RRTs. While RRT-based methods are widely exploited to efficiently solve motion planning problems in high-dimensional spaces, their deployment for symbolic task planning problems has been partially investigated. In this work, we addressed this issue by proposing a RRT-based algorithm suitable for searching task and motion plans in an extended search space which combines configuration states and symbolic states. This extended state space is obtained by introducing an extended metric space associated with a distance measure that combines a distance in the configuration space and distance on the symbolic state. Such extended notion of distance enabled us to deploy a basic version of the RRT-based search on the extended metric space to generate an executable combined plan including both symbolic actions and the associated motions.

We detailed the overall approach and discussed its feasibility and scalability in incrementally complex case studies provided by a real-world hospital logistic scenario that involves an omni-directional mobile robot assigned to multiple pick, carry and place tasks. In these contexts, we deployed the proposed method in both clutter and uncluttered environments comparing the performance of the approach with respect to a baseline algorithm where task and motion planning are decoupled. Despite the simplicity of the proposed RRT algorithm, the collected empirical results show that the approach is feasible and effective in realistic scenarios with typical mobile robotics tasks. The comparison with respect to the proposed baseline – where task and motion planning are decoupled – suggests that the benefit of the combined approach is emphasized in complex and cluttered cases, where the configuration of the environment hinders coherent task plans from a motion perspective. On the other hand, it is worth noticing that in this work we deployed a basic version of the RRT algorithm. Our aim was to present the overall approach in clear and simple setting and show its feasibility and scalability in typical navigation and logistic tasks. In order to demonstrate the approach in its full potential more sophisticated versions of the RRT algorithm can be deployed, while different notions of symbolic distance may be tested and compared. In particular, as a future work, we plan to study the performance of refined RRT algorithms in more complex domains, where not only navigation and transportation activities are considered, but also manipulation and mobile-manipulation tasks.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The research leading to these results has been partially supported by the projects HARMONY (H2020-ICT-46-2020, grant agreement 101017008) and ICOSAF (PON R&I 2014–2020).

References

[1] F. Ingrand, M. Ghallab, Deliberation for autonomous robots: A survey, Artif. Intell. 247 (2017) 10–44.
[2] N.T. Dantam, Z. Kingston, S. Chaudhuri, L.E. Kavraki, Incremental task and motion planning: A constraint-based approach, in: Robotics: Science and Systems, 2016.
[3] E. Erdem, K. Haspalamutgil, C. Palaz, V. Patoglu, T. Uras, Combining high-level causal reasoning with low-level geometric reasoning and motion planning for robotic manipulation, in: IEEE International Conference on Robotics and Automation, ICRA 2011, Shanghai, China, IEEE, 2011, pp. 4575–4581.
[4] F. Lagriffoul, D. Dimitrov, J. Bidot, A. Saffiotti, L. Karlsson, Efficiently combining task and motion planning using geometric constraints, Int. J. Robot. Res. 33 (14) (2014) 1726–1747.
[5] J. Choi, E. Amir, Combining planning and motion planning, in: IEEE International Conference on Robotics and Automation, ICRA 2009, Kobe, Japan, 2009, pp. 238–244.
[6] S. Cambon, R. Alami, F. Gravot, A hybrid approach to intricate motion, manipulation and task planning, I. J. Robot. Res. 28 (1) (2009) 104–126.
[7] L.P. Kaelbling, T. Lozano-Pérez, Hierarchical task and motion planning in the now, in: IEEE International Conference on Robotics and Automation, ICRA 2011, Shanghai, China, 2011, pp. 1470–1477.
[8] J.L. Barry, L.P. Kaelbling, T. Lozano-Pérez, A hierarchical approach to manipulation with diverse actions, in: IEEE International Conference on Robotics and Automation, ICRA 2013, Karlsruhe, Germany, 2013, pp. 1799–1806.
[9] J. Bidot, L. Karlsson, F. Lagriffoul, A. Saffiotti, Geometric backtracking for combined task and motion planning in robotic systems, Artificial Intelligence 247 (2017) 229–265.
[10] W. Thomason, R.A. Knepper, A unified sampling-based approach to integrated task and motion planning, in: International Symposium on Robotics Research, ISRR 2019, Hanoi, Vietnam, 2019.
[11] E. Plaku, G.D. Hager, Sampling-based motion and symbolic action planning with geometric and differential constraints, in: IEEE International Conference on Robotics and Automation, ICRA 2010, Anchorage, Alaska, USA, 2010, pp. 5002–5008.
[12] L. de Silva, A.K. Pandey, R. Alami, An interface for interleaved symbolic-geometric planning and backtracking, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2013, Tokyo, Japan, 2013, pp. 232–239.
[13] S. Srivastava, E. Fang, L. Riano, R. Chitnis, S. Russell, P. Abbeel, Combined task and motion planning through an extensible planner-independent interface layer, in: IEEE International Conference on Robotics and Automation, ICRA 2014, Hong Kong, China, IEEE, 2014, pp. 639–646.
[14] S.M. Lavalle, J.J. Kuffner, Randomized kinodynamic planning, in: IEEE International Conference on Robotics and Automation, ICRA 1999, Detroit, Michigan, USA, 1999, pp. 473–489.
[15] S. Morgan, M. Branicky, Sampling-based planning for discrete spaces, in: IEEE International Conference on Intelligent Robots and Systems, IROS 2004, Sendai, Japan, 2004, pp. 1938–1945.
[16] D. Burfoot, J. Pineau, G. Dudek, RRT-plan: A randomized algorithm for STRIPS planning, in: International Conference on Automated Planning and Scheduling, ICAPS 2006, Cumbria, UK, 2006, pp. 362–365.
[17] R.E. Fikes, N.J. Nilsson, STRIPS: a new approach to the application of theorem proving to problem solving, Artificial Intelligence 2 (3) (1972) 189–208.
[18] J.J. Kuffner, S.M. Lavalle, RRT-connect: An efficient approach to single-query path planning, in: IEEE International Conference on Robotics and Automation, ICRA 2000, San Francisco, CA, USA, 2000, pp. 995–1001.
[19] R. Caccavale, A. Finzi, Combining task and motion planning through rapidly-exploring random trees, in: European Conference on Mobile Robots, ECMR 2021, Bonn, Germany, IEEE, 2021, pp. 1–6.
[20] R. Shome, K.E. Bekris, Anytime multi-arm task and motion planning for pick-and-place of individual objects via handoffs, in: International Symposium on Multi-Robot and Multi-Agent Systems, MRS 2019, New Brunswick, NJ, USA, 2019, pp. 37–43.
[21] K. Solovey, O. Salzman, D. Halperin, Finding a needle in an exponential haystack: Discrete RRT for exploration of implicit roadmaps in multi-robot motion planning, I. J. Robot. Res. 35 (5) (2016) 501–513.
[22] C. Dornhege, P. Eyerich, T. Keller, S. Trüg, M. Brenner, B. Nebel, Semantic attachments for domain-independent planning systems, in: International Conference on Automated Planning and Scheduling, in: ICAPS 2009, Thessaloniki, Greece, AAAI Press, 2009, pp. 114–121.
[23] S.M. Lavalle, J.J. Kuffner, Rapidly-exploring random trees: Progress and prospects, in: Algorithmic and Computational Robotics: New Directions, 2000, pp. 293–308.
[24] S. Karaman, E. Frazzoli, Sampling-based algorithms for optimal motion planning, Int. J. Robot. Res. 30 (7) (2011) 846–894.
[25] J.D. Gammell, S.S. Srinivasa, T.D. Barfoot, Informed RRT*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2014, Chicago, IL, USA, IEEE, 2014, pp. 2997–3004.
[26] S. Klemm, J. Oberländer, A. Hermann, A. Roennau, T. Schamm, J.M. Zollner, R. Dillmann, RRT-connect: Faster, asymptotically optimal motion planning, in: IEEE International Conference on Robotics and Biomimetics, ROBIO 2015, Zhuhai, China, IEEE, 2015, pp. 1670–1677.
[27] L. Chen, Y. Shan, W. Tian, B. Li, D. Cao, A fast and efficient double-tree RRT*-like sampling-based planner applying on mobile robotic systems, IEEE/ASME Trans. Mechatronics 23 (6) (2018) 2568–2578.

Riccardo Caccavale is currently a postdoctoral researcher at the Department of Electrical Engineering and Information Technology, University of Naples "Federico II" (Italy). His research interests are focused on Cognitive Robotics, Human–Robot Interaction, Cognitive Control. He is involved in research projects sponsored by the European Community

Alberto Finzi is Associate Professor at the Department of Electrical Engineering and Information Technology (DIETI), University of Naples "Federico II" (Italy). His research interests include Cognitive Robotics, Human–Robot Interaction, Executive and Cognitive Control, Autonomous and Adaptive Systems, Planning and Scheduling Systems, Multi-agent Systems, Formal Methods for Autonomous Systems. He has been recently involved in several research projects sponsored by the EC (European Community), NASA (National Aeronautics and Space Administration), ESA (European Space Agency), ASI (Italian Space Agency), FWF (Austrian Science Fund), MIUR (Italian Ministry for University and Research), and private industries.