Journal of Intelligent & Robotic Systems (2020) 100:1527–1543
https://doi.org/10.1007/s10846-020-01247-4

Searching Multiple Approximate Solutions in Configuration Space to Guide Sampling-Based Motion Planning

Vojtěch Vonásek[1] · Robert Pěnička[1] · Barbora Kozlíková[2]

Received: 18 May 2020 / Accepted: 12 August 2020 / Published online: 31 August 2020

Abstract
High-dimensional configuration space is usually searched using sampling-based motion planning methods. The well-known issue of sampling-based planners is the narrow passage problem caused by small regions of the configuration space that are difficult to cover by random samples. Practically, the presence of narrow passages decreases the probability of finding a solution, and to cope with it, the number of random samples has to be significantly increased, which also increases the planning time. By dilating the free space, e.g., by scaling-down or thinning the robot (or obstacles), narrow passages become wider, which allows us to compute an approximate solution. Then, the configuration space can be sampled densely around the approximate solution to find the solution of the original problem. However, this process may fail if the final solution is too far from the approximate one. In this paper, we propose a method to find multiple approximate solutions in the configuration space to increase the chance of finding the final solution. The approximate solutions are computed by repeated search of the configuration space while avoiding, if possible, the already discovered solutions. This enables us to search for distinct solutions leading through different parts of the configuration space. The number of approximate solutions is automatically determined based on their similarity. All approximate solutions are then used to guide the sampling of the configuration space. The performance of the proposed approach is verified in scenarios with multiple narrow passages and the benefits of the method are demonstrated by comparing the results with the state-of-the-art planners.

Keywords Motion planning · Rapidly-exploring random tree

The presented work has been supported by the Czech Science Foundation (GAČR) under research project No. 19-22555Y. Computational resources were supplied by the project "e-Infrastruktura CZ" (e-INFRA LM2018140) provided within the program Projects of Large Research, Development and Innovations Infrastructures.

✉ Vojtěch Vonásek
vonasek@labe.felk.cvut.cz

Robert Pěnička
penicrob@fel.cvut.cz

Barbora Kozlíková
kozlikova@fi.muni.cz

[1] Faculty of Electrical Engineering, Czech Technical University in Prague, Technická 2, Prague, 166 27, Czech Republic

[2] Faculty of Informatics, Masaryk University, Botanická 68a, Brno, 602 00, Czech Republic

1 Introduction

The task of motion planning is to find a collision-free path of an object between two configurations in a given environment. In this paper, we consider the planning for 3D solid objects. Besides robotics, applications of this task can be found in computer-aided design for assembly/disassembly solvers [12], in computational biology [1], for example, to study ligand unbinding pathways [23] and for protein folding [3].

Finding paths for 3D solid objects requires to search the six-dimensional configuration space. High-dimensional configuration spaces are usually searched by sampling-based planning that explores the space using randomly generated samples [21]. The narrow passage problem is the well-known issue of the sampling-based planners [21, 22]. Collision-free regions with a low volume have a low probability of being sampled. Consequently, significantly more random samples need to be generated to construct a path through the passage, which also increases the planning time.

A possible approach to cope with the narrow passage problem is to utilize the guided sampling, where the random samples are generated along a guiding path. Computing the guiding paths in the workspace is suitable only for low-dimensional configuration spaces, where the path in the workspace correlates with the path in the configuration space. Guiding the sampling in the high-dimensional configuration space requires to compute the guiding path also in the configuration space.

However, computing a guiding path in the configuration space requires solving the same planning problem and it would be time-consuming. Instead, the guiding path can be computed as a solution of a similar, yet simpler problem. For example, the robot or obstacles can be scaled-down [4, 15, 31], which effectively widens the narrow passages and, in consequence, makes the sampling-based search easier. The result of this search is an approximate solution that is valid for the reduced robot, but possibly invalid for the original problem. The solution of the original problem is then computed by dense search in the vicinity of this approximate solution. Using only one guiding path may, however, fail, if the desired solution, i.e., the path for the non-scaled object, cannot be found in the vicinity of the approximate solution.

In this paper, we propose a method for motion planning in the six-dimensional configuration space using multiple approximate solutions. Ideally, the approximate solutions should belong to different homotopy classes to increase the probability of finding the desired solution. The computation of homotopy classes in configuration space is still a challenging problem [6]. Therefore, we propose a novel heuristic approach to obtain diverse approximate solutions in the configuration space.

The principle of our approach is illustrated on the problem of finding a path for an L-shaped robot (Fig. 1a). First, several approximate solutions are constructed considering a smaller version of the robot (Fig. 1b). Then, the configuration space is sampled densely along these approximate solutions to find a solution for the original robot (Fig. 1c). The approximate solutions are found iteratively. After an approximate solution (path) is found (Fig. 1d), it defines *the inhibited regions* that are temporarily prohibited from random sampling (Fig. 1e, f). This ensures that the subsequent search prefers to explore a different part of the configuration space. The sampling of the inhibited regions may be allowed if it would block the finding of new solutions. This typically happens when multiple solutions lead through the same narrow passage (Fig. 1e, f).

The idea of inhibited regions was introduced in our previous work [32] where the number of approximate solutions and the probability of penetrating the inhibited regions were defined by the user. Therefore, the work [32] required some a priori knowledge of the configuration space to correctly estimate the number of approximate solutions and the probability of sampling the inhibited

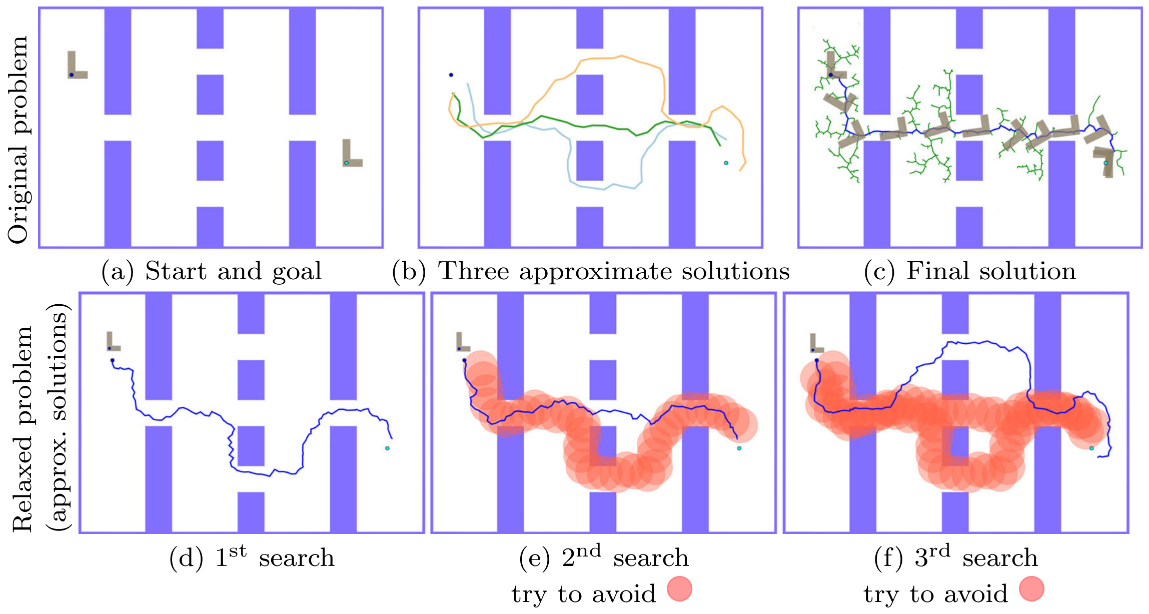


Fig. 1 Illustrated overview of the proposed method in case of motion planning of the L-shaped robot (gray). To solve the planning problem (**a**), several approximate solutions are found first (**b**) and the configuration space is sampled more densely around them (**c**). The approximate solutions are found using a scaled-down robot (**d**, **e**, **f**); after an approximate solution is found, it defines an inhibited region (red) that the next search tries to avoid if possible (**e**, **f**)

regions. This paper is motivated by the practical usage where such a priori knowledge is not always known and where methods requiring minimal user input are preferred. The contribution of this paper is a) the extension of [32] to automatically determine the number of approximate solutions based on their similarity; b) the method to estimate the speed of exploration of the configuration space; c) the novel approach for enabling/disabling sampling of inhibited regions based on the expected speed of exploration; and d) extended experiments on challenging scenarios.

2 Related Work

The Probabilistic Roadmaps (PRM) [19] and Rapidly-exploring Random Trees (RRT) [20] are the most used sampling-based planners. RRT incrementally builds a tree of collision-free configurations. In each iteration, a random sample is generated and the nearest node in the tree is found. The node is then expanded towards the random sample. PRM first generates the random samples and stores only the collision-free ones. Then, the close free samples are connected to make the roadmap.

The original PRM and RRT assume the uniform distribution of random samples, which leads to the well-known narrow passage problem [11, 17, 22]. The narrow passages have a low volume (in comparison to the volume of the whole configuration space), and therefore, the probability of placing uniformly-distributed samples into them is low. As the removal of narrow passages leads to changes of connectivity of the configuration space, the inability to sample the narrow passages consequently prevents the planners from finding a solution.

PRM can cope with the narrow passage problem e.g. by generating more random samples near obstacles [2, 16, 24]. Alternatively, medial axis of the workspace can be used to generate the random samples [33]. Increasing the probability of sampling does not necessarily help in the case of RRT-based planners, as they extend the tree iteratively by small steps, and the expansion can be blocked by obstacles. The RRT-based planners need to improve also the expansion step to cope with the narrow passage problem. In [35], the collision-free random samples are retracted towards a more desirable region, usually towards the boundary of the obstacles, which increases the probability of expanding the tree. The method [34] limits the selection of nodes for the expansion by maintaining a distance threshold for each node. Random sample can select a node for expansion only if the distance towards the node is less than the threshold. Initially, the threshold is set to infinity and it is decreased to a finite value after the expansion fails.

The growth of the tree can be "steered" by changing the distribution of the random samples. This is used in the well-known goal-bias modification, where the random sample is generated around the goal configuration with a predefined probability. In [28], the tree is attracted towards multiple key-configurations that are supposed to be placed near the narrow passages. The work [28], however, does not specify how to find these key-configurations.

The generalization of the goal-bias is the guided sampling, where a sequence of waypoints (a guiding path) is used to generate the random samples [9, 10, 29, 30]. They are generated with the increased probability around a given waypoint of the guiding path until the tree approaches the waypoint close enough. The sampling then continues around the next waypoint [10, 30, 31]. In [30], the guiding path is computed using Voronoi diagram or Visibility graph. The work [10] utilize Reeb graph. Alternatively, the workspace can be discretized, and the guiding path is found using a graph-search method [5, 25] In Exploration/Exploitation Tree [26], the workspace connectivity is estimated using a sphere-based wavefront expansion [7]. The space is then sampled along the wavefront decomposition of the workspace. The guiding principle can be further augmented by using local information about the guiding path, e.g., the density of obstacles [29].

Generating the random samples along paths computed in workspace is not sufficient for searching high-dimensional configuration spaces due to the low correlation between a low-dimensional motion in the workspace and a high-dimensional motion in the configuration space. To make the construction of high-dimensional guiding paths practically possible, we can compute them as approximate solutions of the original problem by relaxing some constraints [4, 15, 31]. For example, the collision constraints can be relaxed by reducing the volume of the robot (or obstacles), e.g., by scaling-down the geometry of the robot [4, 31] or by its thinning [15]. The reduction of the robot volume increases the relative volume of the narrow passages, which also increases the probability of sampling the passage. A similar effect can be achieved by allowing a little penetration between the robot and the obstacles.

The first idea of searching the configuration space along a solution of the relaxed problem was proposed form PRM planner in [4]. First, the PRM builds the roadmap considering the relaxed problem and the solution is iteratively repaired until it is valid also the original problem. In [15], the geometry of the robot and obstacles are adaptively thinned, and PRM is used to find the approximate solutions. In [31], RRT-based planner utilizes a single approximate solution to guide the search in the configuration space. The robot volume is reduced by an iterative removal of the surface triangles which allows us to find the approximate solutions.

Sampling of the configuration space along only a single approximate solution is proposed in several works [4, 15, 31]. The performance of these planners is sensitive to the computation of the approximate solutions. As only the vicinity of the approximate solution is sampled to find the desired solution, this approach may fail if the desired solution is located far away from the approximate one. To increase the probability of finding the desired solutions, multiple approximate solutions are needed. Ideally, the should be located in different homotopy classes. The homotopy classes has been studied only for low-dimensional spaces [13]. The structure of $\mathbb{R}^2$ environment is captured using the Constrained Delaunay Triangulation in [8]. The path finding is then simplified to a graph search over the triangles such that the homotopy class of each path is different. Authors of [14] propose an algorithm for finding the shortest path with a certain homotopy class for $\mathbb{R}^2$ workspace with polygonal obstacles. For semi-algebraic obstacles in the $\mathbb{R}^2$ workspace, [13] proposes an algorithm for the shortest path in a given homotopy class. The PRM-based method for 2D workspace is used in [27] to find the homotopy classes. This method creates roadmap vertices that cover the free part of the configuration space such that each point of the free part can be connected to the roadmap without collision. The roadmap nodes are then connected in such a way that only one path exists in each homotopy. The authors of [6] propose a discretization-based method for computing trajectories in different homotopy class in two, three, and four dimensional Euclidean space. It uses the H-signature augmented graph that stores homotopy class information for a trajectory leading from start to each node. This is based on the discretization of the configuration space, which practically limits the usage to low-dimensional spaces.

The method proposed in this paper copes with the narrow passage problem by guided-sampling along approximate solutions. In comparison with the most relevant works [4, 15, 31], we propose to utilize several approximate solutions. The approximate solutions should be mutually different, i.e., leading through various parts of the configuration space and ideally belonging to different homotopy classes. However, the determination of homotopy classes in the high-dimensional configuration space with narrow passages is very challenging. Therefore, we propose a method to compute an approximate solution iteratively, ensuring that a newly computed solution is different from the previously detected ones.

3 Algorithm Overview

We consider the classical motion planning problem where the task is to find a collision-free path for a robot starting at $q_{\text{start}} \in \mathcal{C}_{free}$ and ending at $q_{\text{goal}} \in \mathcal{C}_{free}$, where $\mathcal{C}_{free} \subseteq \mathcal{C}$ is the collision-free region of the configuration space $\mathcal{C}$.

The proposed planning method, referred as RRT-IR (RRT with Inhibited Regions) in the rest of the paper, consists of two procedures: a) sampling of the configuration space (Algorithm 1); and b) maintenance of the approximate solutions (Algorithm 2). The workflow of the RRT-IR planner is depicted in Fig. 2. First, Algorithm 2 repeatedly employs the sampling procedure (Algorithm 1) to find approximate solutions. After a new approximate solution is found, it extends the set of inhibited regions, so it is avoided in the next search realized again by Algorithm 1. The number of approximate solutions to be found is determined based on their similarity. After enough approximate solutions are found, the configuration space is densely sampled around these approximate solutions.

As already stated, the sampling of $\mathcal{C}$ is realized by Algorithm 1. This algorithm is presented in a versatile form implementing the sampling that avoids the inhibited regions, and also the guided sampling along approximate solutions. The particular behavior of Algorithm 1 depends on the parameter settings, which will be described in the following text.

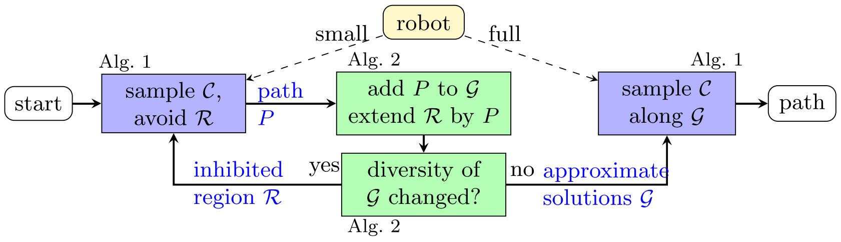

robot
small
full
Alg. 1
Alg. 2
Alg. 1
start
sample $\mathcal{C}$, avoid $\mathcal{R}$
path P
add P to $\mathcal{G}$ extend $\mathcal{R}$ by P
sample $\mathcal{C}$ along $\mathcal{G}$
path
inhibited region $\mathcal{R}$
yes
diversity of $\mathcal{G}$ changed?
no
approximate solutions $\mathcal{G}$
Alg. 2

Fig. 2 Overview of the RRT-IR planner. First, Algorithm 1 samples the space to find an approximate solution (path) P. The path P then extends the inhibited region $\mathcal{R}$, so the next search by Algorithm 1 will avoid it. After enough approximate solutions $\mathcal{G}$ are found, the configuration space is sampled again around $\mathcal{G}$

```
Algorithm 1: findPath($q_{\text{start}}, q_{\text{goal}}, \mathcal{G}, \mathcal{R}, p_{bias}$).
Input: start/goal: $q_{\text{start}}, q_{\text{goal}}$, guiding paths
       $\mathcal{G} = \{P_1, \dots, P_m\}$, inhibited region $\mathcal{R}$, guiding bias
       $p_{bias}$
Data:  size of inhibited region $d_{\text{inh}}$, distance to goal $d_{\text{goal}}$,
       maximal number of samples $N_{\max}$, blocking level $B$
Output: path from $q_{\text{start}}$ to $q_{\text{goal}}$ or $\emptyset$ (Failure)
Result: behavior as basic RRT: $\mathcal{G} = \emptyset$ or $p_{bias} = 0$;
        sampling while avoiding inhibited regions: $p_{bias} = 0$
        and $\mathcal{R} \neq \emptyset$; guided sampling: $\mathcal{G} \neq \emptyset$ and $p_{bias} > 0$
1  $\mathcal{T}$.addNode(($q_{\text{start}}$, 0))  // tree is rooted at $q_{\text{start}}$
2  $v_i = 1, i = 1, \dots, m$  // active waypoints for guided sampling
3  for $n_{\text{samples}} = 1, \dots, N_{\max}$ do
4      if random(0,1) $< p_{bias}$ and $|\mathcal{G}| > 0$ then  // guided sampling along paths $\mathcal{G}$
5          $i$ = random $1 \leq i \leq m$  // select random path
6          $q_{v_i}$ = $v_i$-th point of path $P_i$
7          $q_{\text{rand}}$ = random configuration around $q_{v_i}$
8      else
9          $q_{\text{rand}}$ = random configuration from $\mathcal{C}$  // uniform sampling
10     $q_{\text{near}} = \mathcal{T}$.nearestNeighbor($q_{\text{rand}}$)  // KD-tree search
11     $q_{\text{new}}$ = straightLineExpansion($q_{\text{near}}, q_{\text{rand}}$)
12     if isFree($q_{\text{new}}$) then
13         $enterInhibited = True$
14         if $|\mathcal{R}| > 0$ then  // can $q_{\text{new}}$ enter inhibited region $\mathcal{R}$?
15             $enterInhibited = False$
16             $q' = \mathcal{R}$.nearestWaypoint($q_{\text{new}}$)  // KD-tree search
17             $d = \varrho(q_{\text{new}}, q')$  // distance towards the region $\mathcal{R}$
18             if $d > d_{\text{inh}}$ or $n_{\text{samples}} > n_s(q') \cdot B$ then  // attempt to enter $\mathcal{R}$
19                 $enterInhibited = True$
20         if $|\mathcal{R}| = 0$ or $enterInhibited = True$ then
21             $\mathcal{T}$.add(($q_{\text{new}}, n_{\text{samples}}$))  // $n_s(q_{\text{new}}) = n_{\text{samples}}$
22             $\mathcal{T}$.addEdge($q_{\text{near}}, q_{\text{new}}$)
23             for $i = 1, \dots, m$ do  // update active waypoints
24                 if $\varrho(q_{v_i}, q_{\text{new}}) \leq d_{\text{goal}}$ then  // is active waypoint reached?
25                     $v_i = v_i + 1$  // next guiding waypoint
26             if $\varrho(q_{\text{new}}, q_{\text{goal}}) \leq d_{\text{goal}}$ then  // is goal reached?
27                 return path in $\mathcal{T}$ from $q_{\text{start}}$ to $q_{\text{goal}}$
28 return $\emptyset$
```

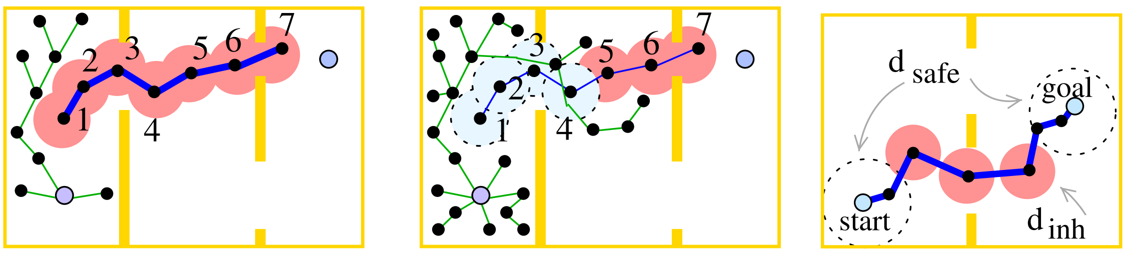
3.1 Construction of Multiple Approximate Solutions

The configuration space is sampled in Algorithm 1 using the RRT principle. The tree $\mathcal{T}$ of collision-free configurations rooted at $q_{\text{start}} \in \mathcal{C}_{free}$ is iteratively built. In each iteration, a random sample q_{rand} is generated and its nearest neighbor $q_{\text{near}} \in \mathcal{T}$ is found. The tree is expanded from q_{near} towards q_{rand} using the straight-line expansion [20], which results in a new configuration q_{new} that can be eventually added to the tree if it is collision-free. The sampling process is repeated until the goal is reached to the predefined distance d_{goal}, or until the number of planning iterations $N_{\max}$ is reached.

Each node in the tree contains, besides a configuration q, also the value $n_s(q)$ which denotes the number of random samples (variable n_{samples} in Algorithm 1, line 21) that were generated before q was added to the tree. If $n_s(q) < n_s(q')$ we consider the configuration q to be reached "before" the configuration q'. The property $n_s()$ estimates how fast the tree explores the configuration space.

Finding multiple diverse approximate solutions is achieved using the concept of inhibited regions. The inhibited regions are defined around the waypoints of already known solutions (paths). Let $q'_1, \dots, q'_k \in \mathcal{C}$ be k waypoints and let $\mathcal{R} \subseteq \mathcal{C}$ denote the inhibited region: $\mathcal{R} = \{q \in \mathcal{C} | \varrho(q, q') \leq d_{\text{inh}}, q' \in q'_1, \dots, q'_k\}$, where $\varrho(\cdot, \cdot)$ is the distance between two configurations and d_{inh} determines the size of the inhibited region. We denote the waypoints $q'_1, \dots, q'_k$ as the waypoints forming $\mathcal{R}$, and by notation $q' \in \mathcal{R}$, we refer to these waypoints. The waypoints forming $\mathcal{R}$ are made of nodes of the trees built in the previous runs of Algorithm 1. The property $n_s(q'), q' \in \mathcal{R}$ is therefore known also for the waypoints of the inhibited region and it estimates the expected number of samples required to reach the waypoint q'.

Let $q' \in q'_1, \dots, q'_k$ be the nearest point towards the configuration $q_{\text{new}} \in \mathcal{C}$ that is the candidate for expanding the tree. The candidate q_{new} is added to the tree if it is outside the inhibited region, i.e., if $\varrho(q_{\text{new}}, q') > d_{\text{inh}}$ (lines 18–19 in Algorithm 1). If q_{new} is inside the inhibited region, it can be added to the tree only if the tree is blocked too much by the inhibited regions. The level of blocking is determined based on the actual number of samples $n_s(q_{\text{new}})$ required to reach q_{new} and the number of samples $n_s(q')$ required to reach the waypoint of the inhibited region. If the two values are similar, we can consider the actual tree not to be blocked by the inhibited region and therefore, q_{new} is not added to the tree. The candidate q_{new} is added to the tree only if $n_s(q_{\text{new}}) > B \cdot n_s(q'), B > 0$, i.e., when the tree grows B times slower than is expected by the waypoints of $\mathcal{R}$ (lines 18–19 in Algorithm 1). The parameter B controls the ratio of random samples (relatively to $n_s(q')$)



(a) 2nd search, inhibited reg. 🟥 $n_{\text{samples}} \leq D \cdot n_s(q'_1)$ (b) 2nd search, inhibited reg. 🟥 $n_{\text{samples}} > D \cdot n_s(q'_4)$ enter to ◌ allowed (c) parameter d_{safe}

Fig. 3 Sampling of $\mathcal{C}$ while avoiding the inhibited region $\mathcal{R}$ located around waypoints $q'_1, \ldots q'_7 \in \mathcal{R}$. Let assume $n_s(q'_1) < \ldots < n_s(q'_7)$. The inhibited region $\mathcal{R}$ is avoided by the tree if the number of random samples $n_{\text{samples}} \leq D \cdot n_s(q'_1)$ (**a**). When $n_{\text{samples}} > D \cdot n_s(q'_4)$, the tree can expand towards the inhibited region around waypoints $q'_1, \ldots, q'_4$ (blue) (**b**). The inhibited regions are not defined around start and goal, respectively (**c**)

that have to be generated before the waypoint $q' \in \mathcal{R}$ can be approached by the tree.

The advantage of using $n_s()$ to determine if $\mathcal{R}$ can be entered is that it respects the speed of growth of the tree, because the property is derived from the number of samples required to reach the particular configuration in the previous runs of Algorithm 1. For example, quickly growing trees (i.e., when the goal is reached using a small number of samples), would result in low values of $n_s()$. Similarly, a slow growth (e.g., due to presence of narrow passages or because of a small expansion step) would result in large values of $n_s()$. By comparing the actual number of samples (n_{samples} in Algorithm 1) to $n_s()$, the subsequent attempt to find approximate solution is "calibrated" to the speed of growth in the given configuration space. The tree is therefore forced to explore other regions of $\mathcal{C}$ than $\mathcal{R}$, but allows the tree to enter $\mathcal{R}$ if needed, e.g., if two solutions have to lead through the same narrow passage (Fig. 3).

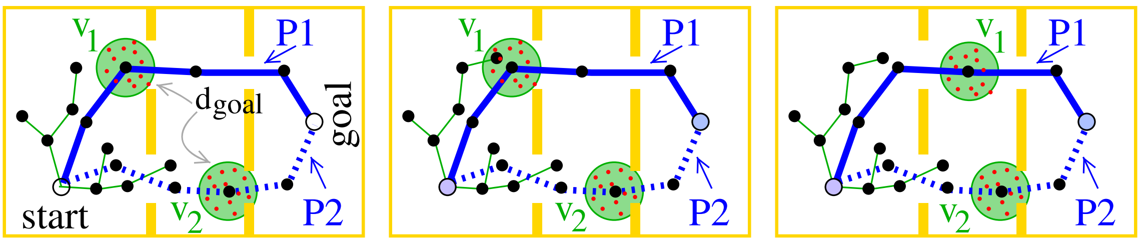
The search for approximate solution requires the generation of uniformly distributed random samples. Therefore, the approximate solutions are found using Algorithm 1 with $p_{bias} = 0$ and $\mathcal{G} = \emptyset$. The sampling along the approximate solutions is described in the next section.

3.2 Guided Sampling Along Multiple Guiding Paths

In the guided sampling, random samples q_{rand} are generated around the approximate solutions $\mathcal{G} = \{P_1, \ldots, P_m\}$ of m paths, where each path $P = (q_i), q_i \in \mathcal{C}_{free}$ is a sequence of waypoints. The guided sampling along m guiding paths is realized using *active waypoints* $v_i, i = 1, \ldots, m$, that denote the index of a waypoint of i-th guiding path, $1 \leq v_i \leq |P_i|$.

The random samples q_{rand} are generated from a randomly chosen path $1 \leq i \leq m$ around its active waypoint $q_{v_i} \in P_i$ (lines 4–7 in Algorithm 1) with the probability p_{bias}. Otherwise, q_{rand} is generated uniformly in $\mathcal{C}$. The active waypoints are shifted forward on the guiding path if the tree approaches them to a predefined distance d_{goal} (lines 23–25 in Algorithm 1).

By drawing the random samples q_{rand} around the active waypoint of a randomly chosen path, the tree attempts to grow along all guiding paths simultaneously. Yet, if some of the paths cannot be followed, e.g., due to an obstacle, the tree can still follow the other paths, because the active waypoints are adapted independently. The update of the active waypoints v_i is depicted in Fig. 4.



(a) (b) (c)

Fig. 4 Guided sampling along two guiding paths P_1 and P_2. The random samples (red) are generated around the active waypoints v_1 and v_2, respectively (green) (**a**). When the tree reaches an active waypoint in the distance d_{goal} (v_1 in this case) (**b**), the particular active waypoint is moved forward along the guiding path (**c**)

3.3 The Main Procedure of the RRT-IR Planner

The main loop of the planner is described in this section and listed in Algorithm 2. In the first phase, initial approximate solutions are found considering the scaled-down robot (lines 6–13 in Algorithm 2). These approximate solutions are stored in the set $\mathcal{G}$. For each found solution, its waypoints are added to the inhibited region $\mathcal{R}$ (line 10 in Algorithm 2), so the next sampling of the configuration space will avoid the previously found solutions. Waypoints in the distance d_{safe} from q_{start} and q_{goal}, respectively, are omitted when defining $\mathcal{R}$. This is necessary in order to enable the tree growth from q_{start} and also to enable the tree to reach q_{goal}. The effect of the parameter d_{safe} is illustrated in Fig. 3c.

The construction of approximate solutions is terminated based on their mutual diversity. The similarity between two paths is the average distance between waypoints of P_1 and their nearest waypoints in P_2:

$$s(P_1, P_2) = \frac{1}{|P_1|} \sum_{q \in P_1} \varrho(q, q^*),$$
$$q^* \in P_2 \text{ is the nearest waypoint towards } q, \quad (1)$$

and the diversity $d(\mathcal{G})$ of paths $\mathcal{G}$ is

$$d(\mathcal{G}) = \min_{P_1, P_2 \in \mathcal{G}, P_1 \neq P_2} \max\left(s(P_1, P_2), s(P_2, P_1)\right). \quad (2)$$

The search for approximate solutions continues while the diversity of the path is changing (Algorithm 2 line 6), which means, that each new approximate solution is different from the previously found ones. The progress of diversity with

Algorithm 2: Main loop of the RRT-IR planner.

```
Input: $q_{\text{start}}, q_{\text{goal}} \in \mathcal{C}_{free}$
Output: feasible solution (path) or $\emptyset$ if no solution is found
1  $\mathcal{R} = \emptyset$                              // inhibited regions
2  $\mathcal{G} = \emptyset$                                  // guiding paths
3  $robot$.scaleDown()
4  $d_{prev} = -1$
5  $d_{actual} = 0$
6  while $d_{prev} \neq d_{actual}$ do                   // search for
   approximate solutions
7      $P$ = findPath($q_{\text{start}}, q_{\text{goal}}, \emptyset, \mathcal{R}, p_{bias} = 0$) // Alg. 1
8      if $P \neq \emptyset$ then
9          $\mathcal{G} = \mathcal{G} \cup \{P\}$             // new approximate
           solution
10         $\mathcal{R} = \mathcal{R} \cup \{q_i \in P | \varrho(q_i, q_{\text{start}}) >$
           $d_{\text{safe}} \wedge \varrho(q_i, q_{\text{goal}}) > d_{\text{safe}}\}$        // extend $\mathcal{R}$
11     $d_{prev} = d_{actual}$
12     $d_{actual} = d(\mathcal{G})$            // path diversity (2)
13
14 $\mathcal{R} = \emptyset$    // inhibited regions are not used
   now
15 $robot$.originalScale()
16 $P$ = findPath($q_{\text{start}}, q_{\text{goal}}, \mathcal{G}, \mathcal{R}, p_{bias} \neq 0$)     // Alg. 1
17 return $P$                          // solution (path)
```

the increasing number of approximate solutions is depicted in Figs. 5 and 6.

The result of the first phase of the algorithm is the set $\mathcal{G}$ of guiding paths — each representing an approximate solution. In the second phase, the configuration space is sampled along the guiding paths $\mathcal{G}$ while considering the original size of the robot. The disabled regions are not considered in this case (line 14 in Algorithm 2). The result of the second

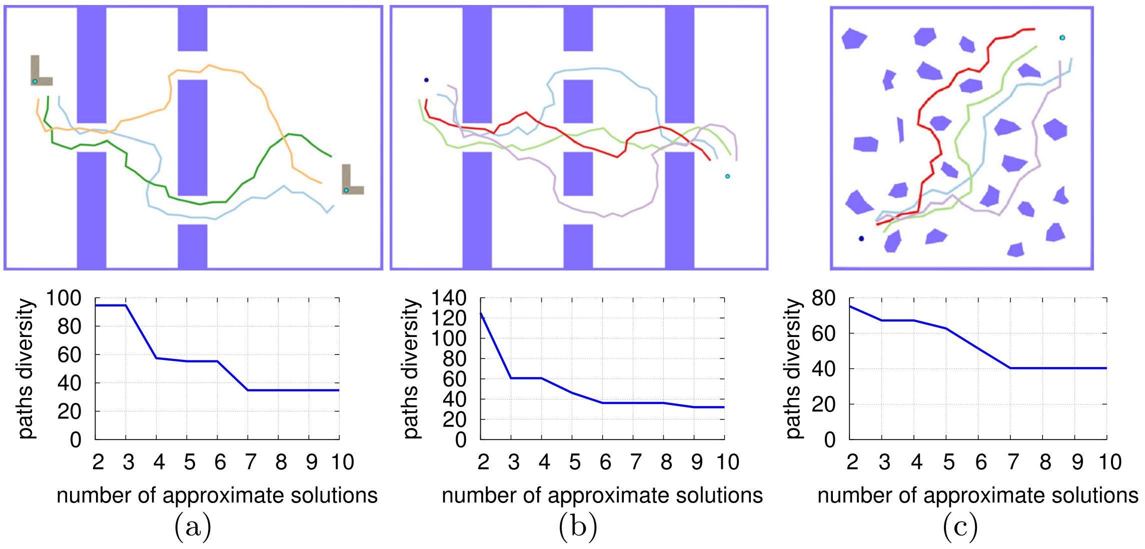


Fig. 5 The progress of diversity of approximate solutions with the increasing number of the solutions for an L-shaped robot. The lowest number of approximate solutions where the diversity does not change determines the number of approximate solutions that will be used for guiding. In (**a**), three solutions will be used, whereas in (**b**) and (**c**) we operate with four solutions

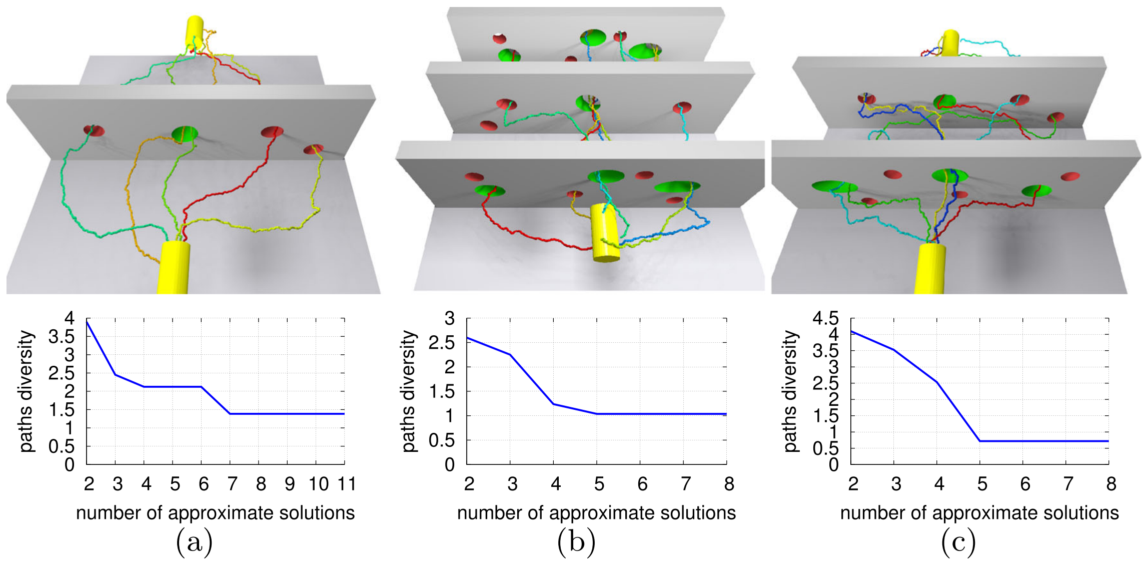


Fig. 6 The progress of the diversity of the approximate solutions in 3D workspace for the yellow cylinder object. The lowest number of approximate solutions where the diversity does not change determines the number of approximate solutions that will be used for guiding. The searched configuration space is six-dimensional. The approximate solutions are found for the scaled-down cylinder that can pass through red and green holes. The original robot can fit only into the green passages. Five approximate solutions are found in (**a**), two of them leading through the desired green narrow passage. In scenario (**b**), six approximate solutions are found through all feasible (green) passages, four of them lead through the shared narrow passage of the middle wall. In scenario (**c**), six approximate solutions are found through both shared and non-shared narrow passages

sampling is the final solution of the original, non-relaxed problem.

3.4 Discussion

The RRT-IR planner relies on the concept of inhibited regions to ensure the diversity of the approximate solutions. It is worth to note that the approximate solutions cannot be computed by a naïve repeated RRT search. The classic RRT tends to find similar paths in each trial. Due to the narrow passage problem, the classic RRT more likely reaches the goal configuration via the easily accessible regions (e.g., widely open regions) rather than through the narrow passages. In environments with several different narrow passages, the repeated RRT search likely finds paths through the most easiest one and therefore, the diversity of the found paths would be low. This effect is illustrated in Fig. 7.

The computationally most intensive part of the RRT-IR algorithm is the configuration space sampling (Algorithm 1) because of the frequent usage of collision detection and the nearest-neighbor search. Algorithm 1 performs two searches: sampling with inhibited regions and guided sampling along multiple approximate solutions.

In the former case, time complexity of one iteration of Algorithm 1 is $\mathcal{O}(\log n + \log c + \log |\mathcal{R}|)$, where the first term is the complexity of the nearest-neighbor search in the tree of n nodes realized by KD-tree, the second term represents hierarchical collision-detection between c objects, and $\log |\mathcal{R}|$ represents the nearest-neighbor search with inhibited region that are made of $|\mathcal{R}|$ waypoints. Practically, the number of waypoints forming $\mathcal{R}$ is significantly smaller than the number of nodes in the configuration tree, therefore, the term $\log |\mathcal{R}|$ is negligible to other two terms.

In the latter case, where Algorithm 1 performs guided sampling, the time complexity is $\mathcal{O}(\log n + \log c + m)$, where the first and the second terms represent the nearest-neighbor search and collision-detection, respectively, and m is the

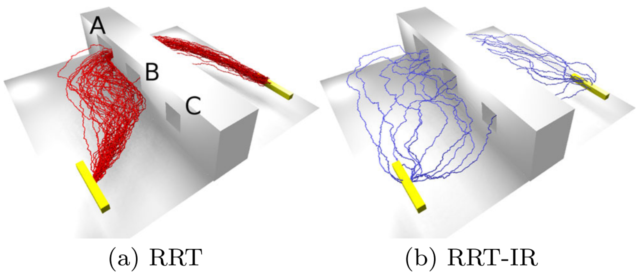


Fig. 7 Computing multiple paths in an environment with three narrow passages, where the passage A is the widest (i.e., easy to access by the robot) and the passage C is the smallest (i.e., difficult to traverse). The paths computed by repeated basic RRT search tends to lead through the easy narrow passage and the resulting paths are similar (**a**). Contrary, paths computed via the concept of inhibited regions are more diverse and lead through all narrow passages (**b**)

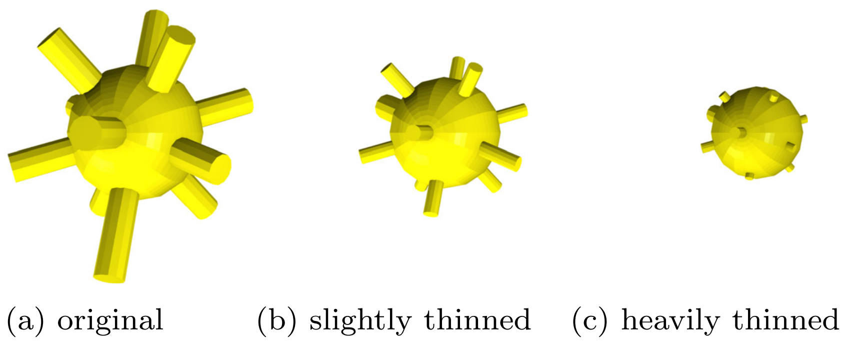


Fig. 8 Reducing the volume of the object (robot) for finding approximate solutions should preserve the necessary geometric details. Thinning applied to the Hedgehog puzzle (**a**) reduces the spikes more than the spherical core of the object (**b**, **c**). The approximate solutions should be computed using the slightly thinned version (**b**)

complexity of updating the active waypoints of m guiding paths. Practically, the number of approximate solutions is in order of units, therefore, the update of active waypoints is negligible with respect to other two procedures.

We can assume that the sampling process (Algorithm 1) is therefore comparable to the basic RRT which has time complexity $\mathcal{O}(\log n + \log c)$. Nevertheless, RRT-IR planner searches the configuration space several times, while RRT searches it only once. From this point of view, RRT-IR is $|\mathcal{G}|$ times slower, where $|\mathcal{G}|$ is the number of approximate solutions. However, as RRT-IR searches for the approximate solutions using the scaled-down robot, the number of iterations needed to find an approximate solution is significantly lower than the time needed to solve the problem with basic RRT, which makes the RRT-IR planner fast from the runtime point of view. This is shown in the experimental verification.

The strength of the guided sampling is controlled by the parameter p_{bias}, and we suggest to use values in the range $0.8 \leq p_{bias} < 1$. The parameter d_{inh} determines the volume of the inhibited region. More diverse paths can be achieved with higher value of d_{inh}. We suggest to set d_{inh} according to the average thickness of the robot. The parameter B controls how likely the inhibited regions can be entered by the tree. The low values $0 \leq B \leq 1$ suppress the effect of the inhibited regions which results in approximate solutions with low diversity. Higher values $B > 2$ make the inhibited regions less accessible, which results in paths with a higher diversity. However, too high B requires the tree to enter the inhibited region with a relatively high number of samples, which increases the runtime requires to find diverse approximate solutions. Our preliminary experiments have shown that value $B = 3$ is a good trade-off between the diversity of paths and planning time. The ability of the RRT-IR planner to find the approximate solutions depends on the amount of reducing the volume of the robot, which also depends on the technique used to reduce the volume. In [15], the robot is thinned even to 0.5 (50%) of its original size. Alternatively, the robot geometry can be scaled-down. We recommend to reduce the volume of the robot in such a way that the necessary geometric details are preserved. For example, the level of thinning for the Hedgehog object should preserve its spikes (Fig. 8).

4 Experiments

The performance of the RRT-IR planner was verified in scenarios with narrow passages and compared to the state-of-the-art RRT-based planners Retraction-RRT [35], ADD-RRT [18], and also the original RRT [20]. See the animations of the planning process and the results.[1]

All planners detect collisions with resolutions $\varepsilon = 0.2$ map units and expand the tree by the straight-line expansion with the same expansion step ε. The approximate solutions of RRT-IR are found for the scale 0.6 (60% of the original robot). ADD-RRT uses the initial radius of 1.0 map unit and the adaptation rate 0.1 (these parameters lead to the best performance of ADD-RRT in the tested scenarios). Retraction-RRT can utilize up to 10 retraction steps.

Two variants of RRT-IR were tested: RRT-IR where the number of approximate solutions is determined automatically, and RRT-IR^c where the number of approximate solutions is given by the user. All planners terminate their search if they approach the goal to the distance $d_{\text{goal}} = 1$ map unit. The other parameters of RRT-IR are $d_{\text{inh}} = d_{\text{safe}} = d_{\text{goal}}$, $B = 3$, and $p_{bias} = 0.8$. The guided sampling in RRT-IR (Algorithm 1) is realized by drawing the translation part of q_{rand} from $N(q_{v_i}, \Sigma)$, where q_{v_i} is the active waypoint of the guiding path i, Σ is the 3×3 matrix with diagonal entries d_{goal}. The rotational part of q_{rand} (yaw, pitch, roll) is generated from $U(r-\pi, r+\pi)$, where r is the corresponding yaw/pitch/roll of the q_{v_i}. All planners employ the 6D Euclidean metric to measure the distance ϱ in the configuration space.

All planners are run 100 times in each tested scenario and the cumulative probability of finding a solution is computed. The solution is found if the planners reaches the goal configuration to the distance $d_{\text{goal}} = 1$ map unit or less.

[1] https://www.youtube.com/watch?v=3VprfBISAX8&list=PLPjuFI-_2rxweQC6dUrmMvEEePo250-8h

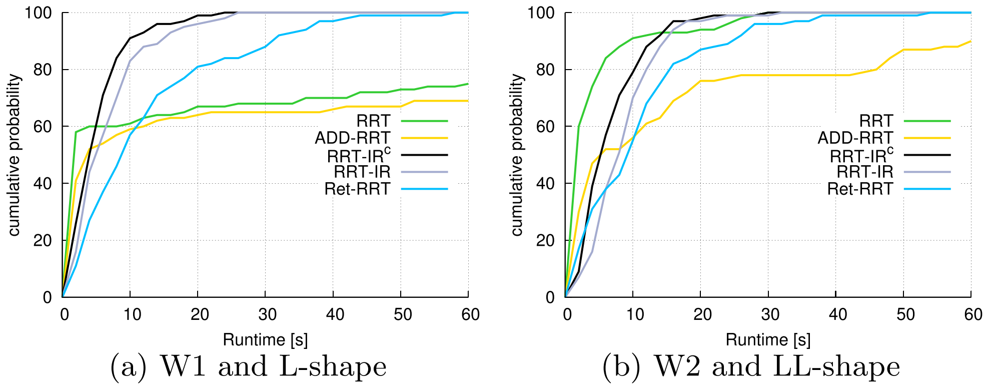


Fig. 9 Success rate in the Windows scenarios

We refer to this probability as the success rate in the rest of the paper. Example of the success rate is depicted in Fig. 9. The runtime at which each planner achieved its best success rate is identified and the related measures are computed. Besides the success rate and runtime, n_{samples} shows the average number of random samples required to find a solution. For the RRT-IR and RRT-IR^c planners, $n^{\text{1st}}_{\text{samples}}$ is the average number of random samples required to find an approximate solution, and $|\mathcal{G}|$ shows the number of discovered approximate solutions. Note that $|\mathcal{G}|$ is defined by the user in the case of RRT-IR^c.

4.1 Windows Scenario

The task is to find a collision-free path for the L-shape and LL-shape robots (Fig. 10a) in rooms separated by small windows (scenarios W1, W2 and W3 in Fig. 10a,b,c). The results are summarized in Table 1.

In all Windows scenarios, both RRT-IR and RRT-IR^c planners achieve 100% success rate in the shortest time. The variant RRT-IR^c is faster than RRT-IR, which is due to the lower number of found approximate solutions (RRT-IR^c uses only three approximate solutions as was predefined by the user, while RRT-IR automatically finds between four and five approximate solutions). Planners Ret-RRT, RRT-IR, and RRT-IR^c use tens of thousands of random samples to find a solution, while RRT and ADD-RRT need hundreds of samples.

Comparison of n_{samples} of RRT and $n^{\text{1st}}_{\text{samples}}$ of RRT-IR shows that finding an approximate solution requires significantly less random samples than finding the full solution. For example, in W3, RRT needs in average 144×10^3 samples to find a solution for the L-shape robot, while one approximate solution is found by RRT-IR with 18×10^3 samples.

4.2 Cylindrical Scenario

In the second experiment, a cylindrical robot had to pass several cylindrical holes in walls (scenarios C1–C4 in Fig. 11). The radius of the cylindrical robot is 95% of the radius of the wall holes, which form the narrow passages. The maximal allowed runtime is 800 seconds. The results are summarized in Table 2.

Except for Ret-RRT in C2, the RRT, Ret-RRT, and ADD-RRT fail to deliver a solution within the given time limit, which is indicated by their low success rate. Contrary, RRT-IR and RRT-IR^c deliver plans in all cases. RRT-IR automatically detects between 4 and 5 approximate solutions, while RRT-IR^c was run with three approximate solutions.

Due to the similar size of the robots and holes to be passed, the resulting narrow passages are more challenging than in the Windows scenario. The RRT-IR planner has a significant advantage over other planners. For example, the scenario C4 is solved by RRT with 1% success rate, i.e., only one case out of tested 100 trials. That case required $n_{\text{samples}} = 398 \times 10^3$ random samples, while RRT-IR finds its first approximate solution with only $n^{\text{1st}}_{\text{samples}} = 13 \times 10^3$ samples. In average, RRT-IR needs in total 39×10^3

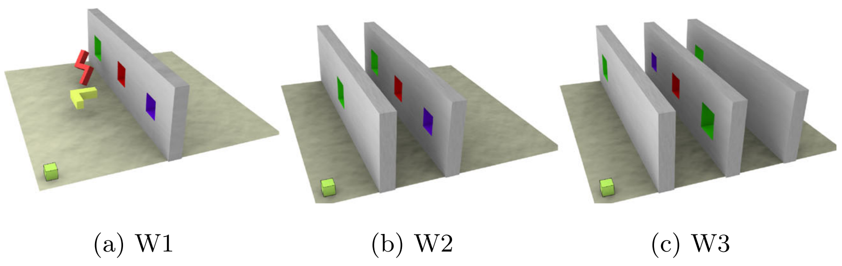


Fig. 10 Windows scenarios W1–W3 (**a**, **b**, **c**) and the L-shape (yellow) and LL-shape robots (red). The green cube in the bottom left corner illustrates the size of one map unit

Table 1 Comparison of the planners in the Windows scenario

	RRT	Ret-RRT	ADD-RRT	RRT-IR	RRT-IR^c
W1 and L-shape					
Runtime [s]	498	58	590	26	24
Cum. prob.	100%	100%	99%	100%	100%
$n_{\text{samples}} \times 10^3$	40	16	34	12	10
$\lvert\mathcal{G}\rvert \;\; \lvert n^{\text{1st}}_{\text{samples}} \times 10^3$				4.4 | 9	3.0 | 7
W1 and LL-shape					
Runtime [s]	52	54	214	28	26
Cum. prob.	100%	100%	100%	100%	100%
$n_{\text{samples}} \times 10^3$	5	16	9	11	8
$\lvert\mathcal{G}\rvert \;\; \lvert n^{\text{1st}}_{\text{samples}} \times 10^3$				4.2 | 8	3.0 | 6
W2 and L-shape					
Runtime [s]	560	68	718	64	46
Cum. prob.	99%	100%	88%	100%	100%
$n_{\text{samples}} \times 10^3$	89	17	69	31	24
$\lvert\mathcal{G}\rvert \;\; \lvert n^{\text{1st}}_{\text{samples}} \times 10^3$				4.3 | 13	3.0 | 9
W2 and LL-shape					
Runtime [s]	30	54	418	32	30
Cum. prob.	100%	100%	100%	100%	100%
$n_{\text{samples}} \times 10^3$	9	15	20	19	14
$\lvert\mathcal{G}\rvert \;\; \lvert n^{\text{1st}}_{\text{samples}} \times 10^3$				4.3 | 9	3.0 | 6
W3 and L-shape					
Runtime [s]	728	136	788	152	98
Cum. prob.	89%	100%	82%	100%	100%
$n_{\text{samples}} \times 10^3$	144	18	83	48	47
$\lvert\mathcal{G}\rvert \;\; \lvert n^{\text{1st}}_{\text{samples}} \times 10^3$				4.2 | 18	3.0 | 17
W3 and LL-shape					
Runtime [s]	206	68	278	48	46
Cum. prob.	100%	100%	100%	100%	100%
$n_{\text{samples}} \times 10^3$	23	16	31	28	29
$\lvert\mathcal{G}\rvert \;\; \lvert n^{\text{1st}}_{\text{samples}} \times 10^3$				4.2 | 13	3.0 | 13

samples to solve this problem. The process of computing the approximate solutions is illustrated in Fig. 12.

4.3 Puzzle Robots

The task is to find paths for the Cross, Gear, and Hedgehog robots (Fig. 13). The Cross and Gear robots are only 10% smaller than their matching windows which leads to long narrow passages. Similarly, the radius of the Hedgehog robot is 10% smaller than the width of each cage's window. The maximal allowed runtime is 1500 seconds. The comparison of the planners is summarized in Table 3 and the success rate is shown in Fig. 14.

While all planners provide solution for the Cross robot with almost 100% success rate, planning for the Gear robot is more challenging for the RRT, ADD-RRT, and Ret-RRT as their success rate is significantly smaller than the success rate of RRT-IR and RRT-IR^c. Finding one approximate solution requires in average a significantly less number of planning iterations than finding the solution for the original problem. For example, for the Gear robot, RRT needs up to $n_{\text{samples}} = 3400 \times 10^3$ iterations to find the solution, while RRT-IR needs in average $n^{\text{1st}}_{\text{samples}} = 79 \times 10^3$ iterations to find one approximate solution and the total number of planning iterations of RRT-IR is $n_{\text{samples}} = 221 \times 10^3$, which is still in order of magnitude less than RRT needs.

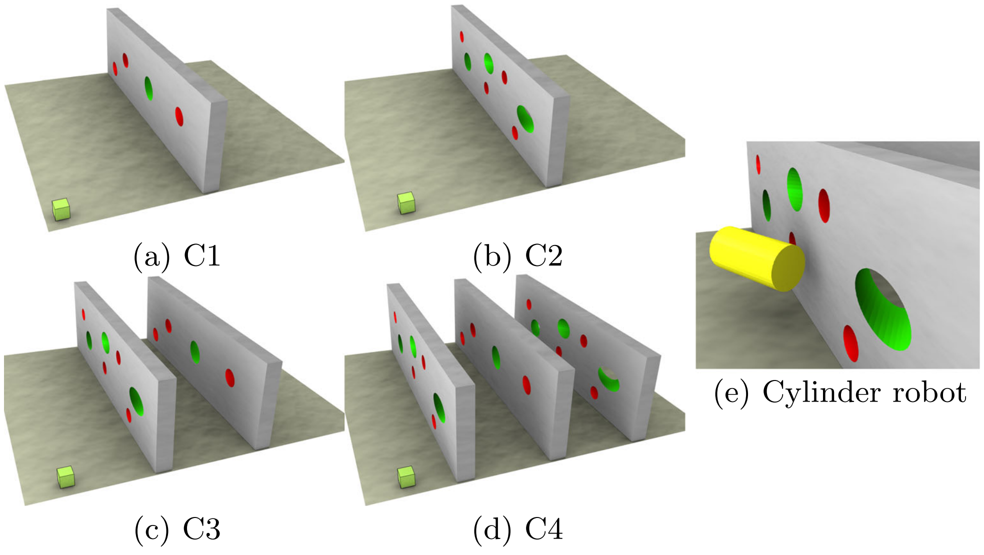


Fig. 11 Scenarios with multiple round passages C1–C4 (**a**–**d**) for the Cylinder robot (**e**). Robots reduced to 60% of their size can pass through all passages (green/red), so the approximate solutions can lead through all the holes. The original robot can fit only to the green passages

4.4 Tunnels

The performance in long narrow passages is investigated in the last experiment. Rooms with the start and goal are connected with one long (T1) or two long tunnels (T2). In the scenarios T1O and T2O, an additional window allows the robot to access the middle room, where neither start nor goal is located. The scenarios are depicted in Fig. 15.

The maximum allowed runtime was 800 seconds. The results are summarized in Table 4. In scenarios T1 and T2, all planners except Ret-RRT, found a solution in few tens of seconds with 100% success rate.

The performance of all planners is significantly worse in the scenarios with the accessible middle room (T1O and T2O). The success rate in these scenarios is shown in Fig. 16. While the success rate of RRT-IR, RRT-IR^c,

Table 2 Comparison of the planners in the Cylindrical scenarios

	RRT	Ret-RRT	ADD-RRT	RRT-IR	RRT-IR^c
C1					
Runtime [s]	693	800	731	462	482
Cum. prob.	14%	0%	18%	98%	100%
$n_{\text{samples}} \times 10^3$	517	296	263	28	26
$\lvert\mathcal{G}\rvert$ $\lvert n^{1\text{st}}_{\text{samples}} \times 10^3$				4.0 | 12	3.0 | 10
C2					
Runtime [s]	671	70	722	94	288
Cum. prob.	25%	100%	14%	100%	100%
$n_{\text{samples}} \times 10^3$	316	11	229	13	11
$\lvert\mathcal{G}\rvert$ $\lvert n^{1\text{st}}_{\text{samples}} \times 10^3$				4.3 | 8	3.0 | 6
C3					
Runtime [s]	800	800	800	288	196
Cum. prob.	3%	0%	3%	100%	100%
$n_{\text{samples}} \times 10^3$	435	365	227	31	21
$\lvert\mathcal{G}\rvert$ $\lvert n^{1\text{st}}_{\text{samples}} \times 10^3$				4.2 | 13	3.0 | 8
C4					
Runtime [s]	800	800	800	240	430
Cum. prob.	1%	1%	0%	100%	99%
$n_{\text{samples}} \times 10^3$	398	293	253	39	30
$\lvert\mathcal{G}\rvert$ $\lvert n^{1\text{st}}_{\text{samples}} \times 10^3$				4.5 | 13	3.0 | 8

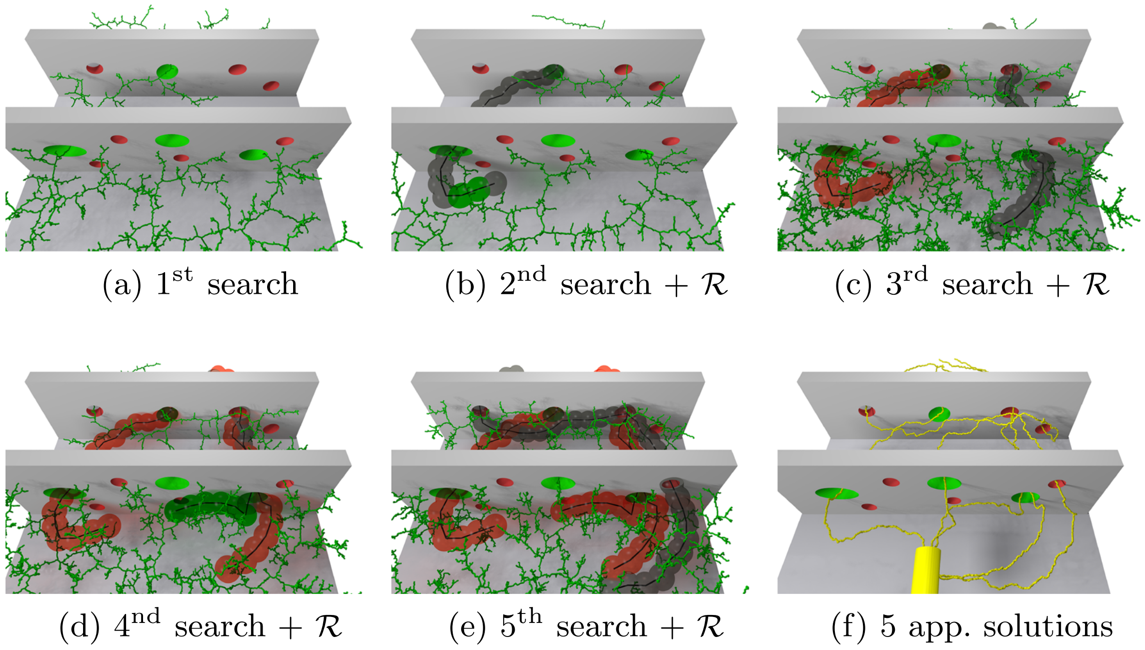

(a) 1st search
(b) 2nd search + $\mathcal{R}$
(c) 3rd search + $\mathcal{R}$
(d) 4nd search + $\mathcal{R}$
(e) 5th search + $\mathcal{R}$
(f) 5 app. solutions

Fig. 12 Example of computing the approximate solutions in scenario C3. The tree is depicted in green. Gray spheres highlight the inhibited region made of the previously found solution. Red spheres are the inhibited regions that cannot be entered (**c**, **d**, **e**), green spheres are inhibited regions that can be entered (**d**). Result of this process is a set of five approximate solutions (**f**). Animations are available at http://mrs.felk.cvut.cz/jint2020rrt-ir

and Ret-RRT increases with the allows computing time, the success rate of RRT and ADD-RRT does not increase. For example, in T1O, RRT and ADD-RRT finds solution relatively quickly in 50% and 32% cases, respectively, but this success rate is not increased even with larger allowed runtimes.

In the T1O and T2O scenarios, the robot can enter also the middle room. In the configuration space, the tree therefore also explores to the regions related to this middle room. However, to find a solution, the tree has to explore the regions of the configuration space related to the tunnels. This requires to generate the random samples inside these regions (i.e., inside the tunnel) and also to expand the tree towards them. Even though some random samples are generated inside the tunnels, the nearest-neighbor rules more likely selects to expand the tree from the nodes located inside the room, which would lead to collision with the walls of the tunnels. In cases where the most of the tree nodes are located in the middle room, the expansion of the tree inside the tunnel is significantly suppressed. The basic RRT and ADD-RRT can therefore find a solution only if the configuration tree fills the tunnel-related regions of the configuration space faster than filling the regions related to the middle room. In the case of RRT, this happens in the 50% of cases (in both scenarios).

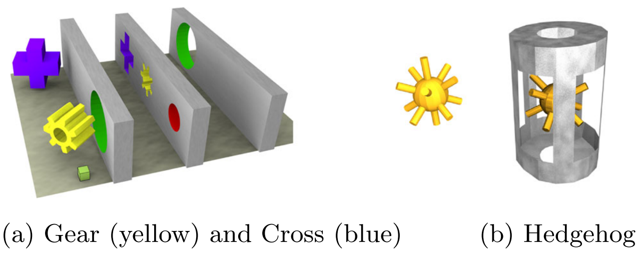


Fig. 13 Scenarios with puzzle robots

4.5 Discussion

The experiments have shown the advantage of RRT-IR planner in scenarios with narrow passages. Finding the approximate solutions is easier than finding the solutions of the original problem, which is supported by the lower values of $n^{1\text{st}}_{\text{samples}}$ of RRT-IR, and higher values of n_{samples} of other planners. Despite the repeated search for approximate solutions, RRT-IR is faster than other planners. In all scenarios, RRT-IR^c variant was forced to find three approximate solutions, while RRT-IR determined the number of approximate solutions automatically. The number of automatically detected approximate solutions was higher than three, therefore the RRT-IR was slower than RRT-IR^c. The exceptions are the Cylindrical scenario (Table 2), where RRT-IR^c is slower due to computing the approximate solutions through passages that are not traversable by the larger robot. In this case, the user incorrectly required to use three approximate solutions, while it would be better to discover more of them.

Table 3 Comparison of the planners in the Puzzle scenario

	RRT	Ret-RRT	ADD-RRT	RRT-IR	RRT-IR^c
Cross					
Runtime [s]	278	170	328	60	194
Cum. prob.	100%	99%	100%	99%	100%
$n_{\text{samples}} \times 10^3$	307	9	239	219	131
$\lvert\mathcal{G}\rvert \;\; \lvert n^{\text{1st}}_{\text{samples}} \times 10^3$				4.3 | 99	3.0 | 52
Gear					
Runtime [s]	1500	1500	1396	332	432
Cum. prob.	67%	6%	69%	100%	100%
$n_{\text{samples}} \times 10^3$	3406	24	1627	221	175
$\lvert\mathcal{G}\rvert \;\; \lvert n^{\text{1st}}_{\text{samples}} \times 10^3$				3.8 | 79	3.0 | 58
Hedgehog					
Runtime [s]	1500	488	1320	316	274
Cum. prob.	64%	82%	76%	96%	98%
$n_{\text{samples}} \times 10^4$	167	20	111	62	64
$\lvert\mathcal{G}\rvert \;\; \lvert n^{\text{1st}}_{\text{samples}} \times 10^4$				4.3 | 20	3.0 | 10

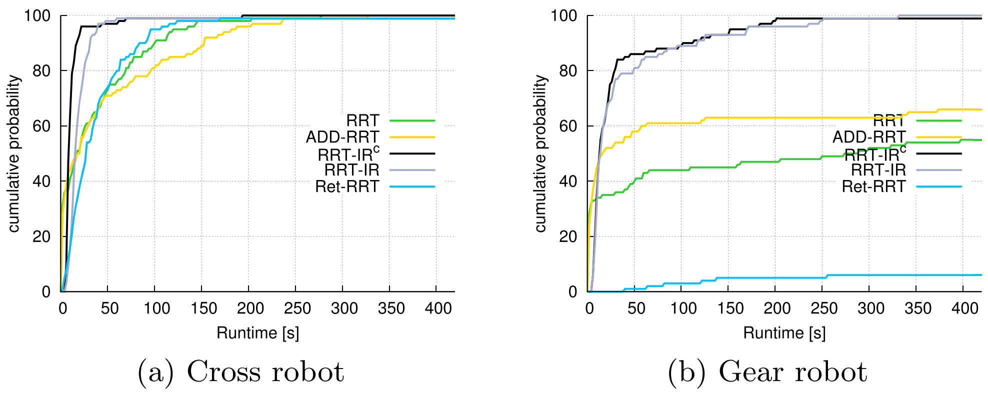


Fig. 14 The cumulative probability of finding a solution for the Cross and Gear robots

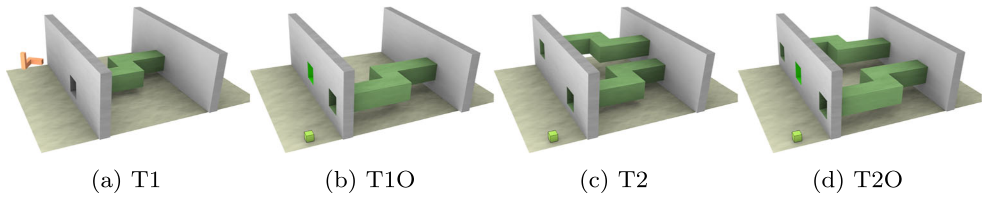


Fig. 15 The Tunnels scenarios with one (**a**, **b**) and two tunnels (**c**, **d**), respectively. The scenarios (**b**, **d**) also contains a small windows (green) that leads to the middle room. The S-robot is depicted in orange (**a**)

Table 4 Comparison of the planners in the Tunnels scenario for the S-robot

	RRT	Ret-RRT	ADD-RRT	RRT-IR	RRT-IR^c
T1					
Runtime [s]	36	122	54	26	12
Cum. prob.	100%	100%	100%	100%	100%
$n_{\text{samples}} \times 10^3$	106	8	125	213	94
$\|\mathcal{G}\|$ $\|n^{1\text{st}}_{\text{samples}} \times 10^3$				4.1 | 61	2.0 | 25
T1O					
Runtime [s]	270	676	40	612	238
Cum. prob.	50%	62%	32%	86%	94%
$n_{\text{samples}} \times 10^3$	2698	140	1186	2194	902
$\|\mathcal{G}\|$ $\|n^{1\text{st}}_{\text{samples}} \times 10^3$				3.7 | 1176	2.0 | 482
T2					
Runtime [s]	14	226	32	26	12
Cum. prob.	100%	100%	100%	100%	100%
$n_{\text{samples}} \times 10^3$	36	10	77	135	73
$\|\mathcal{G}\|$ $\|n^{1\text{st}}_{\text{samples}} \times 10^3$				4.5 | 50	2.0 | 23
T2O					
Runtime [s]	532	682	12	526	132
Cum. prob.	50%	42%	54%	98%	98%
$n_{\text{samples}} \times 10^3$	2488	106	1274	2331	748
$\|\mathcal{G}\|$ $\|n^{1\text{st}}_{\text{samples}} \times 10^3$				4.4 | 1164	2.0 | 372

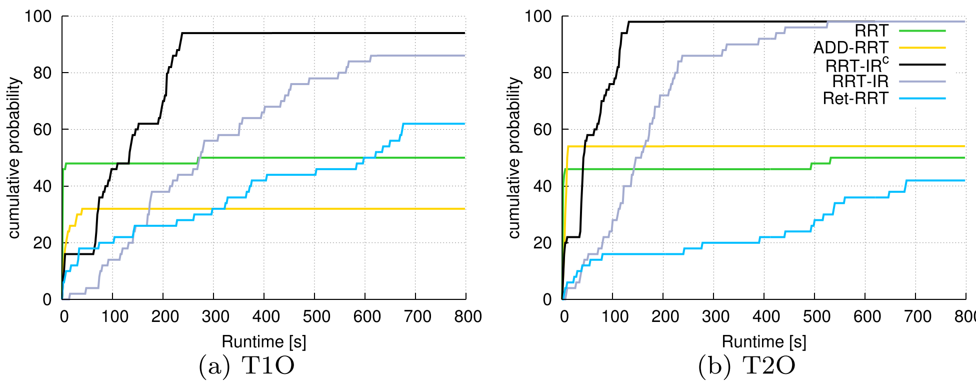


Fig. 16 The cumulative probability of finding solution in the T1O and T2O scenarios

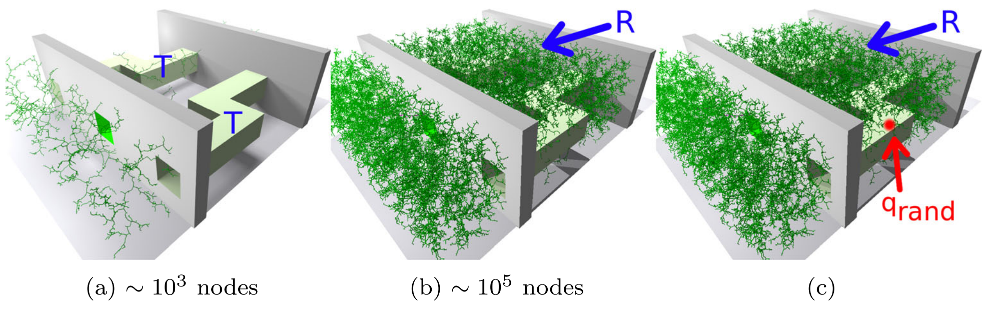


Fig. 17 Example of RRT growth in the T2O scenario. Finding a solution requires to connect the side rooms through the tunnels T (**a**). The configuration tree can grow to both tunnels and also to the middle room (R). When the tree enters the room, it quickly fills it by many nodes (**b**). In this situation, expanding towards the tunnels is more difficult. Even though same random samples q_{rand} (red) are generated inside the tunnels, the tree attempts to expand towards them more like from the nodes located in the rooms. As this leads to collisions, expansion of the tree inside the tunnel is significantly more difficult than expanding the tree inside the room (**c**)

This shows the advantage of computing the approximate solutions based on their diversity. Nevertheless, there is nothing like a universal planner suitable for each scenario. The RRT-IR planner is designed for scenarios with narrow passages, where the repeated search of approximate solutions is faster than solving the original problem. Contrary, utilizing RRT-IR in scenarios without a narrow passage is counterproductive as it would be slower than basic RRT due to repeated search for approximate solutions (Fig. 17).

5 Conclusion

Motion planning of 3D objects leads to the search of six-dimensional configuration space, which can be difficult due to the presence of narrow passages. The paper presents a method to solve the motion planning problem using approximate solutions. The configuration space is first searched considering a scaled-down version of the robot. This effectively widens the narrow passages, which makes the computation of the approximate solutions easier than solving the original problem. The approximate solutions are searched iteratively and, when a new solution is found, it defines an inhibited region that is preferably avoided by the subsequent search. This ensures the diversity of the approximate solutions. After the approximate solutions are found, the configuration space is sampled considering the non-scaled (full) robot. This sampling is guided by the approximate solutions. The comparison with state-of-the-art methods shows that the proposed method has higher success rate of finding a solution.

References

1. Al-Bluwi, I., Siméon, T., Cortés, J.: Motion planning algorithms for molecular simulations: a survey. Comput. Sci. Rev. **6**(4), 125–143 (2012)
2. Amato, N.M., Dale, L.K., Bayazit, O.B., Jones, C.H., Vallejo, D.: OBPRM: an obstacle-based PRM for 3D workspaces. In: Workshop on the Algorithmic Foundations of Robotics (WAFR), pp. 155–168. A. K. Peters, Ltd., Natick (1998)
3. Amato, N.M., Dill, K., Song, G.: Using motion planning to map protein folding landscapes and analyze folding kinetics of known native structures. J. Comput. Biol. **10**(3-4), 239–255 (2003)
4. Bayazit, O.B., Xie, D., Amato, N.M.: Iterative relaxation of constraints: a framework for improving automated motion planning. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 3433–3440 (2005)
5. Belter, D., Labecki, P., Skrzypczynski, P.: An exploration-based approach to terrain traversability assessment for a walking robot. In: IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR), pp. 1–6 (2013)
6. Bhattacharya, S., Likhachev, M., Kumar, V.: Topological constraints in search-based robot path planning. Auton. Robot. **33**(3), 273–290 (2012)
7. Brock, O., Kavraki, L.E.: Decomposition-based motion planning: a framework for real-time motion planning in high-dimensional configuration spaces. In: IEEE International Conference on Robotics and Automation (ICRA), vol. 2, pp. 469–1474 (2001)
8. Demyen, D., Buro, M.: Efficient triangulation-based pathfinding. In: AAAI: Proceedings of the 21st National Conference on Artificial Intelligence (2006)
9. Denny, J., Sandström, R., Amato, N.M.: A general region-based framework for collaborative planning. In: Robotics Research, pp. 563–579. Springer (2018)
10. Denny, J., Sandström, R., Bregger, A., Amato, N.M.: Dynamic region-biased rapidly-exploring random trees. In: Twelfth International Workshop on the Algorithmic Foundations of Robotics (WAFR) (2016)
11. Elbanhawi, M., Simic, M.: Sampling-based robot motion planning: a review. IEEE Access **2**, 56–77 (2014)
12. Ghandi, S., Masehian, E.: Review and taxonomies of assembly and disassembly path planning problems and approaches. Comput. Aided Des. **67-68**, 58–86 (2015)
13. Grigoriev, D., Slissenko, A.: Polytime algorithm for the shortest path in a homotopy class amidst semi-algebraic obstacles in the plane. In: ISSAC, vol. 98 (1998)
14. Hershberger, J., Snoeyink, J.: Computing minimum length paths of a given homotopy class. In: Workshop on Algorithms and Data Structures, pp. 331–342. Springer (1991)
15. Hsu, D., Cheng, H., Latombe, J.-C.: Multi-level free-space dilation for sampling narrow passages in PRM planning. In: IEEE International Conference on Robotics and Automation (ICRA) (2006)
16. Hsu, D., Jiang, T., Reif, J., Sun, Z.: The bridge test for sampling narrow passages with probabilistic roadmap planners. In: IEEE International Conference on Robotics and Automation (ICRA), pp. 4420–4426 (2003)
17. Hsu, D., Latombe, J.-C., Kurniawati, H.: On the probabilistic foundations of probabilistic roadmap planning. Int. J. Robot. Res. **25**(7), 627–643 (2006)
18. Jaillet, L., Yershova, A., LaValle, S.M., Simeon, T.: Adaptive tuning of the sampling domain for dynamic-domain RRTs. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 2851–2856 (2005)
19. Kavraki, L.E., Svestka, P., Latombe, J.-C., Overmars, M.H.: Probabilistic roadmaps for path planning in high-dimensional configuration spaces. IEEE Trans. Robot. Autom. **12**, 566–580 (1996)
20. LaValle, S.M.: Rapidly-exploring random trees: a new tool for path planning. Technical report 98-11 (1998)
21. LaValle, S.M.: Planning Algorithms. Cambridge University Press, Cambridge (2006)
22. Lindemann, S.R., LaValle, S.M.: Current issues in sampling-based motion planning. In: Robotics Research: The Eleventh International Symposium, pp. 36–54 (2005)
23. Nguyen, M.K., Jaillet, L., Redon, S.: ART-RRT as-rigid-as-possible exploration of ligand unbinding pathways. Journal of Computational Chemistry **39**(11), 665–678 (2018)
24. Overmars, M.H.: The gaussian sampling strategy for probabilistic roadmap planners. In: International Conference on Robotics and Automation (ICRA), pp. 1018–1023 (1999)
25. Plaku, E., Kavraki, L.E., Vardi, M.Y.: Discrete search leading continuous exploration for kinodynamic motion planning. In: Robotics: Science and Systems, pp. 326–333 (2007)
26. Rickert, M., Brock, O., Knoll, A.: Balancing exploration and exploitation in motion planning. In: IEEE International Conference on Robotics and Automation (ICRA), pp. 2812–2817 (2008)

27. Schmitzberger, E., Bouchet, J.-L., Dufaut, M., Wolf, D., Husson, R.: Capture of homotopy classes with probabilistic road map. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (2002)
28. Szadeczky-Kardoss, E., Kiss, B.: Extension of the rapidly exploring random tree algorithm with key configurations for nonholonomic motion planning. In: IEEE International Conference on Mechatronics, pp. 363–368 (2006)
29. Uwacu, D., Rex, R., Wang, B., Thomas, S.L., Amato, N.M.: Annotated-skeleton biased motion planning for faster relevant region discovery. arXiv:2003.02176 (2020)
30. Vonásek, V., Faigl, J., Krajník, T., Přeučil, L.: RRT-path — a guided rapidly exploring random tree. In: Robot Motion and Control (RoMoCo), pp. 307–316. Springer (2009)
31. Vonásek, V., Pěnička, R.: Path planning of 3d solid objects using approximate solutions. In: 2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), pp. 593–600 (2019)
32. Vonásek, V., Pěnička, R., Kozlíková, B.: Computing multiple guiding paths for sampling-based motion planning. In: 2019 19th International Conference on Advanced Robotics (ICAR), pp. 374–381 (2019)
33. Wilmarth, S.A., Amato, N.M., P. F. Stiller: MAPRM: a probabilistic roadmap planner with sampling on the medial axis of the free space. In: IEEE International Conference on Robotics and Automation (ICRA), pp. 1024–1031 (1999)
34. Yershova, A., Jaillet, L., Simeon, T., LaValle, S.M.: Dynamic-domain RRTs: efficient exploration by controlling the sampling domain. In: IEEE International Conference on Robotics and Automation (ICRA) (2005)
35. Zhang, L., Manocha, D.: An efficient retraction-based RRT planner. In: IEEE International Conference on Robotics and Automation (ICRA), pp. 3743–3750 (2008)

Vojtěch Vonásek is a post-doc researcher at the Department of Cybernetics, Faculty of Electrical engineering, Czech Technical University (FEE-CTU) in Prague. He received his Ph.D. degree in artificial intelligence and biocybernetics from FEE-CTU in 2016. His research interests include path and motion planning techniques and their applications, automatic learning of locomotion gaits and physical simulation.

Robert Pěnička received his Ph.D. degree at the CTU in 2020. His Ph.D. thesis was entitled Data Collection Planning for Aerial Vehicles. He is currently a postdoctoral researcher in the Multi-robot Systems lab (MRS). His research interests are in constrained multi-goal path planning, optimization, and collision-free motion planning for Unmanned Aerial Vehicles. He contributed to the successful participation of the MRS team in both the Mohamed Bin Zayed International Robotics Challenge 2017 and 2020.

Barbora Kozlíková is an Associate Professor at the Faculty of Informatics, Masaryk University in Brno, Czech Republic. Her research interests include visualization and visual analysis of biomolecules, with focus on comparative and explorative tasks in large molecular dynamics simulations. She is the head of the Visitlab at the Faculty of Informatics: the visualization group specializing on diverse topics in visualization and its connections to virtual reality.