J Intell Robot Syst (2018) 89:387–401
DOI 10.1007/s10846-017-0567-9

Faster RRT-based Nonholonomic Path Planning in 2D Building Environments Using Skeleton-constrained Path Biasing

Yiqun Dong · Efe Camci · Erdal Kayacan

Received: 23 January 2017 / Accepted: 2 May 2017 / Published online: 27 May 2017

Abstract This paper presents a faster RRT-based path planning approach for regular 2-dimensional (2D) building environments. To minimize the planning time, we adopt the idea of biasing the RRT tree-growth in more focused ways. We propose to calculate the skeleton of the 2D environment first, then connect a geometrical path on the skeleton, and grow the RRT tree via the seeds generated locally along this path. We conduct batched simulations to find the universal parameters in manipulating the seeds generation. We show that the proposed skeleton-biased locally-seeded RRT (skilled-RRT) is faster than the other baseline planners (RRT, RRT*, A*-RRT, Theta*-RRT, and MARRT) through experimental tests using different vehicles in different 2D building environments. Given mild assumptions of the 2D environments, we prove that the proposed approach is probabilistically complete. We also present an application of the skilled-RRT for unmanned ground vehicle. Compared to the other baseline algorithms (Theta*-RRT and MARRT), we show the applicability and fast planning of the skilled-RRT in real environment.

Keywords Path planning · Rapidly-exploring random tree (RRT) · Skilled-RRT · Unmanned ground vehicle

1 Introduction

Navigating through 2-dimensional (2D) building environments is a challenging task for ground vehicles. Prior work advocates to use vision or simultaneous localization and mapping (SLAM) algorithms. In our work, we propose to stop the vehicle in front of the 2D building map (i.e., fire-escape plan), use scanning devices (i.e., camera) to extract the specifics of the environment, and conduct the path planning on the 2D map. To complete the overall process as quick as possible, fast path planning is our key concern. Besides, since we aim for the real time applicat

Electronic supplementary material The online version of this article (doi:10.1007/s10846-017-0567-9) contains supplementary material, which is available to authorized users.

Y. Dong · E. Camci · E. Kayacan (✉)
School of Mechanical and Aerospace Engineering, Nanyang Technological University, Singapore 639798, Singapore
e-mail: erdal@ntu.edu.sg

Y. Dong
e-mail: dongyq@ntu.edu.sg

E. Camci
e-mail: efe001@e.ntu.edu.sg

Y. Dong
ST Engineering-NTU Corporate Laboratory, School of Electrical and Electronic Engineering, Nanyang Technological University, 50 Nanyang Avenue, Singapore 639798, Singapore

ions, we adopt the Rapidly-exploring Random Tree (RRT) based planning approach in order to account for nonholonomic constraints from vehicle geometries/dynamics.

Firstly proposed in [23], RRT is a sampling-based motion planning algorithm which finds a path by growing a tree from the root (q_{root}) towards unexplored regions. The key advantage of RRT-based planning is that it can incorporate the nonholonomic constraints. The planning time, path length and clearance, however, hinge on the randomized nature of the seeds generation. Although, there are a number of works in literature to circumvent this issue, including human-robot interaction concept for biasing the planning algorithm as in [10, 33], the authors advocate to address the issue in a more integrated way comprehending path length, path clearance, and planning time individually. For path length, RRT* in [20] rewires the tree based on the length cost, and [2, 6, 13, 14] propose to refine the path after a feasible one is found. The returned paths from these algorithms are shorter. For path clearance, [17] proposes Transition-based RRT which considers a user-defined cost function as an additional input to the standard path planning problem, which can be defined for the distance to be maximized between the vehicle and obstacles in order to find high-clearance solution paths. As another approach, [9] proposes MARRT which pushes all configurations on the RRT tree towards the medial-axis. This work compares MARRT with other baseline planners including RRT, RRT*, OBRRT in [32] and RRTObst (see [9]), and shows that MARRT has the maximum path clearance. Specifically for planning time, [21] proposes growing two different trees from start and goal points, while continuously trying to connect them with a greedy heuristic called *Connect*. They determine that *Connect* improves the running time especially for uncluttered environments. [19] proposes to filter non-viability nodes on the tree. [1] proposes Targetless-RRT to discard the need of target configuration; thus, improving the planning time for disassembly planning. [3] advocates to grow the tree into more promising regions. In [27], Theta*-RRT utilizes Theta* in [8] to find a geometrical path first, and then biases the RRT tree-growth along this path. Compared to the other baseline planners (RRT, RRT*, and A*-RRT in [5]), Theta*-RRT yields faster planning time.

In our work, to minimize the planning time, we follow Theta*-RRT idea; i.e., we find a geometrical path on the map first, then we use this path to bias the RRT tree-growth. Literature review indicates that similar approaches have appeared; apart from Theta*-RRT and A*-RRT, readers might refer to [4, 7, 28–31] for more details. The geometrical-path finding in most of these algorithms, however, yields a path that grazes the obstacles, wherein the RRT tree-growth can be hindered. Overall performance (planning time, path clearance) of these algorithms are therefore degraded.

In order to grow the RRT tree rapidly, the tree should keep away from the obstacles. For such purpose, we refer to the *skeleton* concept for 2D environments, which represents a homography class that keeps as far away as possible from the obstacles. Some skeleton-based path planning approaches such as [9, 15, 16, 18, 24, 26, 34] have appeared in literature, but there is no analysis on their performance for different agent vehicles in complex environments, especially regarding the path planning time.

In our work, particularly in order to address the path planning time issue, we combine the skeleton with RRT-based path planning: we extract the skeleton of the 2D building environment first, then connect the skeleton-constrained path. We generate sampling configurations for the RRT tree-growth locally within a strip along the path. We name this approach as *sk*eleton-b*i*ased *l*oca*l*ly-seed*ed* RRT (*skilled-RRT*). To be specific, the contributions of this work are:

- We use the skeleton-constrained path to bias the RRT tree growth. Under mild assumptions, we prove that the proposed algorithm is probabilistically complete. Via experimental tests, we also show that the planning time, path length and clearance are superior compared to other baseline algorithms.
- Compared with other path-biasing approaches (A*-RRT, Theta*-RRT, etc.) we propose to use skeleton to bias the tree growth; i.e., we find the skeleton-constrained path first, then the RRT tree growth is biased locally along this path.
- Compared with other skeleton-based approaches, we also combine the RRT framework into the algorithm. In this manner, we take the advantage of the RRT as it returns a feasible path regarding the nonholonomic constraints from the vehicles.

We note that our work is intended for real applications. Different (realistic) vehicles are used in *regular* 2D building environments in the simulation tests. We also note that we intend to minimize the planning time using all possible approaches, should they have been theoretically proved or experimentally justified.

2 Related Work

Previous research has combined path-biasing with RRT. In [28, 29], decomposition is adopted to find a sequence of regions to guide the RRT tree-growth. The informed subdivision in [4] utilizes an adaptive subdivision that updates the algorithm in exploring the entire environment. [7] proposes to implement a graph search algorithm first, and then use a low-level planner is used the nonholonomic constraints. In [30, 31], the exploring/exploiting tree is proposed, which trades the probabilistic completeness for computational efficiency. Different from all these planners, the skilled-RRT explores only the neighbouring regions of the skeleton-path. We use a continuous controller to satisfy nonholonomic constraints, and we prove (under mild assumptions) that the algorithm is probabilistic complete in our application scenarios.

A*-RRT in [5] and Theta*-RRT in [27] are similar to the skilled-RRT proposed herein: both approaches utilize the discrete search algorithm to find a geometrical path first, the tree growth is then biased by this path. However, the path generated by A*/Theta* locates close to the obstacles, which might hinder the RRT tree expansion. In contrast to these two planners, the tree growth in the skilled-RRT is faster as the skeleton-path stays away from the obstacles (see Fig. 1 [1]). Also, in both A*-RRT and Theta*-RRT, seeds to bias the tree growth are generated globally in the geometrical path neighbourhood. In the skilled-RRT, the seeds are generated locally in the nearest configuration (q_{near}) neighbourhood.

Several researchers have implemented the skeleton (or medial-axis) in path planning [9, 15, 16, 18, 24, 26, 34]. However, all of the aforementioned papers except the second one consider only (unrealistic) simple environments or simplified vehicles without aiming to minimize the planning time. In [9] (which can be the most similar work to our method), path planning for a point robot is conducted by the generation of sampling configurations globally, while the primary concern being path clearance. In contrast to all these works, the skilled-RRT is utilized for the path planning of three different vehicles in three different (2D regular building) environments by locally seeding. An extensive experimental report of our algorithm is presented regarding planning time, path length and clearance.

[1] For all the environment plots in this paper, black regions represent the obstacles and white regions the free space. Blue cross denotes the start, and red circle the goal.

3 Combining Skeleton with RRT

3.1 Problem Statement

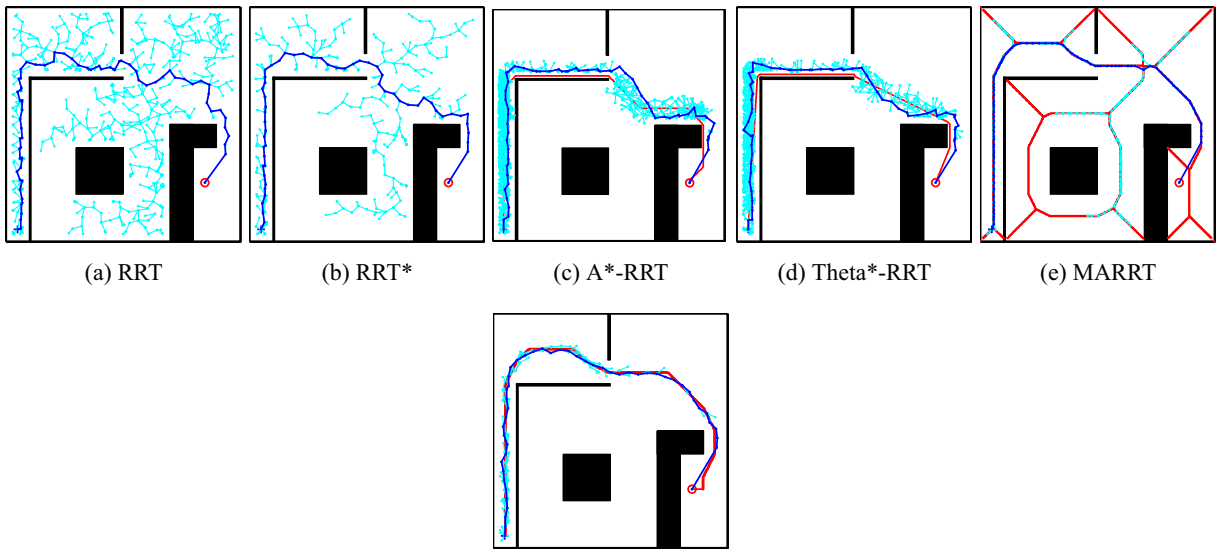
Let $\chi \subset R^d$ be the environment, $\mu \subset R^m$ the control space, $\chi_{obs} \subset \chi$ the obstacles, and $\chi_{free} = \chi \setminus \chi_{obs}$ the free space. The dynamics of the agent vehicle Σ are:

$$\dot{x}(t) = f(x(t))u(t) + e(t), \qquad x(0) = x_{init} \tag{1}$$

wherein $x(t) \in \chi$ is the state, and $u(t) \in \mu$ is the control input; f denotes the system dynamics, and e is the error in dynamics. In this work, we follow the definition of small-time controllable system in [27]. The dynamical system is small-time controllable from χ if, for any time T, the set of states reachable from χ before the moment T contains a neighbourhood of χ. For the skilled-RRT proposed in this paper, it needs to plan a path for small-time controllable nonholonomic systems; i.e., it finds a trajectory that connects a feasible path from $x_{init} \in \chi_{free}$ to goal $x(T) \in \chi_{free}$, wherein the dynamics subject to (1) are satisfied.

3.2 Algorithm Flow of the Skilled-RRT

See Algorithm 1 for the pseudo-code of the skilled-RRT. It first calculates the skeleton S from the workspace (line 1), then it connects the skeleton-constrained path P from the start to the goal (line 2). After this, the skilled-RRT finds a feasible path by growing a tree locally seeded along P (lines 6-17). In the beginning, the root configuration q_{root} is added onto the tree (line 6). For every new configuration (q_{new} in lines 6, 12), the algorithm examines whether



(a) RRT (b) RRT* (c) A*-RRT (d) Theta*-RRT (e) MARRT

(f) S killed – *RRT*

Fig. 1 *Office* environment: Path planning results using different planners. Cyan dot lines represent the RRT tree, red lines indicate the geometrical path used for tree-growth biasing, and blue lines are the planned path. We note that A* and Theta* paths locate close to the obstacles, MARRT grows the tree on the entire skeleton, and $skilled - RRT$ expands along the skeleton-constrained path only

or not it is located in an obstacle-free region of the goal configuration q_{goal} (line 7). If yes, q_{goal} is added to the tree (line 15), and the path is generated (line 17). If not, the RRT tree growth is performed locally along P (lines 7-14). This tree-growth will be repeated till q_{new} falls into an obstacle-free region of the goal (line 7). Then, q_{goal} and the vehicle response history are added onto the tree (lines 15-16) as edges, and the final path is connected (line 17). In following, we detail each of the functions in Algorithm 1.

FindSkeleton(χ) calculates the skeleton of the workspace. The flowchart of this function is briefly illustrated in Function 1. The skeletonization in this paper follows the wavefront algorithm; readers might find details and more explanations in [22].

SkeletonPath(S, q_{root}, q_{goal}) finds the skeleton-constrained path on S connecting from q_{root} to q_{goal}. See the flow chart illustrated in Function 2. For all the points on S, we first find the closest ones (q_r, q_g) to q_{root} and q_{goal} (line 3). Given the total distance between each point on S to q_{root} and q_{goal}, starting from the q_{root} (q_r), we perform the path expansion to the direction which corresponds to the shortest distance (lines 10-11), till the final goal configuration (q_g) is reached. We then add the final connection from q_g to q_{goal}, and the skeleton-constrained path is returned.

Algorithm 1 skilled-RRT$(\chi, q_{root}, q_{goal})$

1: $S \leftarrow$ **FindSkeleton**(χ)
2: $P \leftarrow$ **SkeletonPath**(S, q_{root}, q_{goal})
3: **if** $P = \emptyset$ **then**
4: **return:** *failure*
5: **end if**
6: τ.AddNode(q_{root})
7: **while** τ.AddNode $\notin$ **ObstacleFree**(q_{goal}) **do**
8: $Seed_L \leftarrow$ **LocationSeed**(P, X_{goal}, W)
9: $q_{near} \leftarrow$ **NearestNeighbor**$(\tau, Seed_L)$
10: $Seed_N \leftarrow$ **NavigationSeed**$(q_{near}, W, \Omega_{SR}, \Psi_{HB})$
11: $q_{new} \leftarrow$ **Navigate**$(\Sigma, q_{new}, Seed_N, \Delta q)$
12: τ.AddNode(q_{new})
13: τ.AddEdge(q_{new})
14: **end while**
15: τ.AddNode(q_{goal})
16: τ.AddBranch(q_{goal})
17: **return: ConnectFeasibleTrajectory**(τ)

Function 1 FindSkeleton(χ)

1: $S \leftarrow \emptyset, \mathcal{D}\chi \leftarrow \chi, \mathcal{L} \leftarrow \emptyset$
2: **for** q in $\mathcal{D}\chi_{free}$, **do**
3: $U(q) \leftarrow inf$
4: **end for**
5: **for** q in $\mathcal{D}\chi_{obs}$, **do**
6: $q' \leftarrow \mathcal{N}(q) \bigcap \mathcal{D}\chi_{free}$
7: **if** $q' \neq \emptyset$ **then**
8: $d_1(q) \leftarrow 0, O(q) = q$
9: $\mathcal{L}$.Add(q)
10: **end if**
11: **end for**
12: **while** $\mathcal{L} \neq \emptyset$ **do**
13: **for** q in $\mathcal{L}$ **do**
14: $q' \leftarrow \mathcal{N}(q) \bigcap \mathcal{D}\chi_{free}$
15: **if** $d_1(q') = inf$ **then**
16: $d_1(q') \leftarrow \dim\mathcal{L}, O(q') \leftarrow O(q)$
17: $\mathcal{L}$.Add(q')
18: **elseif** $D_1(O(q'), O(q) > \alpha)$ **then**
19: **if** $q \notin S$ **then** S.Add(q)
20: **end if**
21: **end for**
22: **end while**
23: **return:** S

ObstacleFree(q_{goal}) represents the set of state $x \in \chi_{free}$ wherein an obstacle-free path can be connected from x to q_{goal} in the configuration space. In the beginning of **while** loop (line 6), the vehicle is driven by the controller from q_{new} to q_{goal}. If a collision-free path exists, q_{goal} is added to the tree, and the path is connected. As discussed in the authors' previous work [11, 12], this "navigate-to-connect" strategy significantly expedites the RRT tree growth.

LocationSeed(P, q_{goal}, W) generates the location seed which is used to find the q_{near}. This seed generation follows the original RRT; i.e., 7.5% of the generation uses q_{goal} as the seed directly, while in other cases we follow [5, 27]; seeds are generated randomly from a strip around the skeleton-path P, and the width of the strip is W (see the left plot on Fig. 3).

NearestNeighbor$(\tau, Seed_L)$ searches for the nearest configuration (q_{near}) in the tree (τ). We follow the original RRT; i.e., all the nodes on the tree are inspected, and the one closest to $Seed_L$ (Euclidean distance) is returned as q_{near}.

Function 2 SkeletonPath(S, q_{root}, q_{goal})

1: $P \leftarrow \emptyset$
2: **for** q in S **do**
3: $q_r \leftarrow min\_dis(q_{root}, q)$, $q_g \leftarrow min\_dis(q_{goal}, q)$
4: $U_S(q) = U_S(q, q_r) + U_S(q, q_g)$
5: **end for**
6: P.Add(q_{root}), P.Add(q_r)
7: $q \leftarrow q_r$
8: **while** $q \neq q_g$ **do**
9: **for** $q' \in \mathcal{N}(q)$ **do**
10: $q \leftarrow min(U_S(q'))$
11: P.Add(q)
12: **end for**
13: **end while**
14: P.Add(q_g)
15: **return:** P

NavigationSeed$(q_{near}, W, \Omega_{SR}, \Psi_{HB})$ generates the sampling configuration ($Seed_N$) that is used to guide the RRT tree growth. Different from [5, 27], this configuration is generated from the neighbouring region (Ω_{SR}) of q_{near}. Normally, this strategy might result in a local-minimum problem. However, we propose to sample this configuration along the skeleton-path (strip width: W). Since the skeleton-path connects a global geometrically-feasible trajectory to the goal, local minimum problem can be solved.[2] Also to avoid the turning-back growth of the RRT tree (see Fig. 1), only the configuration within a heading bias Ψ_{HB} is adopted (see Fig. 3).

Navigate$(\Sigma, q_{near}, Seed_N, \Delta q)$ drives the vehicle (dynamics: Σ) from q_{near} to $Seed_N$ directly without using any smoothing method such as Bézier curve as in [25] or spline curve as in [35]. The controller will be detailed in following, here we define the stop conditions. Similar to [11, 12], the vehicle will stop if a) $Seed_N$ is approached, b) collision is detected, or c) maximum step (Δq) is reached. The vehicle stop configuration is designated as q_{new}, and is added onto the tree.

ConnectFeasibleTrajectory(τ) returns the planned path from the grown RRT tree (τ). When the tree growth is completed, q_{goal} is added to the tree as the

[2] Some assumptions need to be addressed to make this claim, see the following subsection.

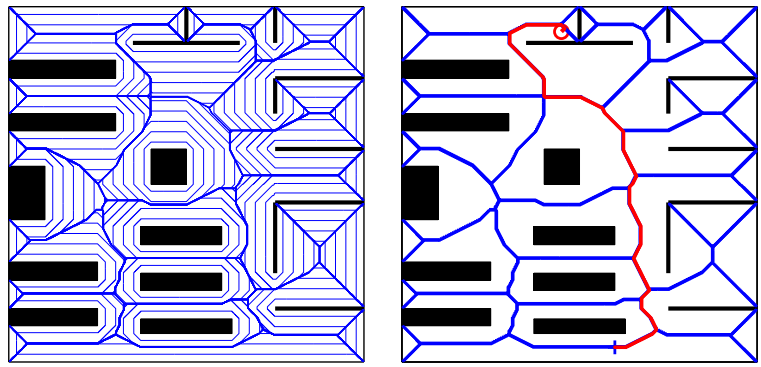


Fig. 2 *Lab* environment: Generation of the skeleton-constrained path. **Left:** Wavefront expansion process, the blue lines indicate the image obtained at each step of the expansion. **Right:** Blue lines represent the skeleton, and the skeleton-constrained path is connected in solid red lines

last node. Starting from q_{goal}, we connect q_{goal} to its parent, and then we repeat this process to connect to the parent nodes till the start of the tree (q_{root}). The planned path from the algorithm is then returned.

3.3 Analysis of the Skilled-RRT

We note that from lines 1-5 in Algorithm 1, a skeleton-constrained path is extracted in the environment (work space). However, for the ground vehicle we are using in real applications, two problems might arise: a) the skeleton-constrained path might go through narrow entrances, which is too tight for the vehicles to pass through; b) the skeleton-constrained path might bias the tree into narrow passages, wherein a feasible solution in the vehicle configuration space does not exist. For a), we advocate to follow [27]; i.e., the obstacles are inflated first to take into account the vehicle geometrical dimensions, path planning is then conducted (as if for a 2D point). As for the second problem, however, considering the regular 2D building environments of interest, we make the assumption that a feasible solution exists in the configuration space[3]; i.e., for any feasible path that can be connected in the work space, a close-enough path also exists in the vehicle configuration space. Given this assumption, below we prove that the skilled-RRT is probabilistically complete.

Theorem 1 *For the workspace χ, via mild assumptions, a geometrically-feasible skeleton-constrained path χ_{path} can be connected in χ_{free}. After the root q_{root} is added as the first node on the RRT tree, we will grow the RRT tree via both location and navigation seeds ($Seed_L$, $Seed_N$). Given the sampling strategies illustrated in above contents, we first prove that the generation of $Seed_L$ and $Seed_N$ is complete.*

Proof For $\forall x_p \in \chi_{path}$, let $\epsilon(x_p, \rho)$ be the ball with radius $\rho > 0$ centered on x_p. For all x_p, since the volume $\Omega = \epsilon(x_p, \rho)$ is non-zero in the Lebesgue metric (2-dimensional area in our case), the event of generating a location seed $Seed_L$ in Ω will happen with probability 1 asymptotically; i.e., we can find a $Seed_L$ which locates in the strip (width W) along the skeleton-constrained path χ_{path}. After $Seed_L$ is found, given the Euclidean distance metric we adopted in our work, the node to be expanded (nearest configuration q_{near}) can be found. Given the sampling strategy for $Seed_N$ in Fig. 3, we note that the valid area Ω_N (green lines) for $Seed_N$ is again non-zero by Lebesgue metric, the event of generating a navigation seed $Seed_N$ in Ω_N will also happen with probability 1 asymptotically. To summarize, given the sampling strategies in above contents, the generation of both $Seed_L$ and $Seed_N$ is complete. □

Given the generated seeds $Seed_L$ and $Seed_N$, the RRT tree is expanded gradually via agent vehicle navigation. We then prove that this tree growth is probabilistically complete; i.e., a trajectory from q_{root} to q_{goal} can be connected via this tree expansion.

Theorem 2 *Consider the (small-time controllable) agent vehicle with dynamics Σ; we have proved that both location and navigation seeds ($Seed_L$ and $Seed_N$) exist following the sampling strategy illustrated in above. Biased by these seeds, the skilled-RRT is probabilistically complete since the probability of*

[3] In real applications, it is either feasibility in both work space and configuration space, or mission failure.

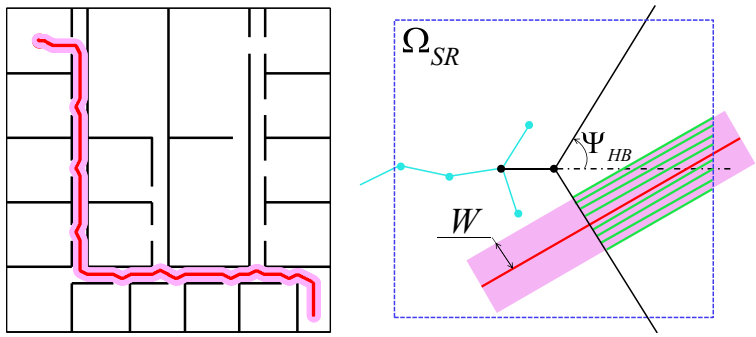


Fig. 3 *Floor* environment: **Left:** Strip for *Floor* environment. **Right:** Generation of the navigation seed. On the right plot, cyan dot lines indicate RRT tree, black dot to the right is q_{near}. Navigation seeds are generated from the region Ω_{SR}. The red line is the skeleton-path. The seed should fall in the strip (magenta, width: W), and the heading bias within Ψ_{HB}. The area for valid generation is marked in green lines

connecting the root $q_{root} \in \chi_{path}$ *to goal* $q_{goal} \in \chi_{path}$ *approaches 1 asymptotically.*

Proof Given q_{near} and $Seed_N$, the agent vehicle is navigated from q_{near} to $Seed_N$ to construct the RRT tree branch. Provided that the dynamical system (agent vehicle) is small-time controllable, the connection from q_{near} to $Seed_N$ can be found in the vehicle configuration space. The vehicle response history will be added onto the RRT tree (τ) as branch, and the vehicle stopped point is designated as node. The set $\bar{\chi}_{path} = \{x \in \chi_{path} | \forall x \notin \tau\}$ represents the uncovered part of the χ_{path} by τ. The volume of $\bar{\chi}_{path}$ will approach 0 when the RRT tree growth iterations approaches infinity. Since the skeleton-path χ_{path} is connected from q_{root} to q_{goal}, the goal configuration q_{goal} will fall into a close-enough neighbourhood of a certain node on the RRT tree (the last added node on the tree), and q_{goal} will be added onto the tree with probability 1. Given the trajectory path connection function illustrated above, the path from q_{root} to q_{goal} can be connected; i.e., the skilled-RRT is probabilistically complete. □

4 Experimental Setup

We compare the performance of the skilled-RRT with five other baseline planners via simulation. All planners use the same collision-detection or any other functions applicable. All the simulations are carried out in Matlab (2015a) environment on a computer with Ubuntu system. This computer is equipped with a 2.60 GHz Intel i7 quad-core processor and 8 GB RAM.

4.1 Nonholonomic Agent Vehicles

Three agent vehicles (see Fig. 4) are used in the simulations: an omnidirectional round car, a nonholonomic car, and a truck-tractor system. We detail each in the following.

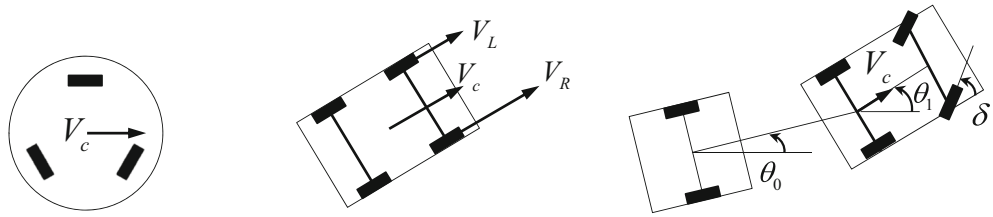
Omnidirectional car This vehicle is driven by omnidirectional wheels. The diameter of the vehicle is 1.5.[4] Given the vehicle state (x, y, θ), wherein $(x, y) \in R^2$ is Cartesian coordinates and $\theta \in [0, 2\pi)$ is the orientation, the equations of motion are:

$$\begin{aligned} \dot{x}(t) &= V_c cos(\theta) \\ \dot{y}(t) &= V_c sin(\theta) \end{aligned} \tag{2}$$

where V_c is the vehicle speed as 1.5, and Δq for the RRT is 2.0.

Nonholonomic car This vehicle has been used in the real applications in [11, 12]. Differential driving wheels are fixed to the front axle to control orientation, while rear wheels are used only for balance support. The length and width of the car are 1.5 and

[4] In this paper we omit the specification of units, all the values are normalized.



(a) Omnidirectional Car (b) Nonhonolomic Car (c) Truck-tractor system

Fig. 4 Agent vehicles used in the simulation tests

1.0. Given the vehicle state (x, y, θ), wherein $(x, y) \in R^2$ is Cartesian coordinates and $\theta \in [0, 2\pi)$ is the orientation, the equations of motion are:

$$\begin{aligned} \dot{x}(t) &= 0.5(V_L + V_R)cos(\theta) \\ \dot{y}(t) &= 0.5(V_L + V_R)sin(\theta) \\ \dot{\theta}(t) &= (V_R - V_L)/W \end{aligned} \tag{3}$$

wherein V_L and V_R are the speeds of the car left and right wheels, and W width of the car. The controller follows [11, 12], and in the RRT tree growth, Δq is 2.0.

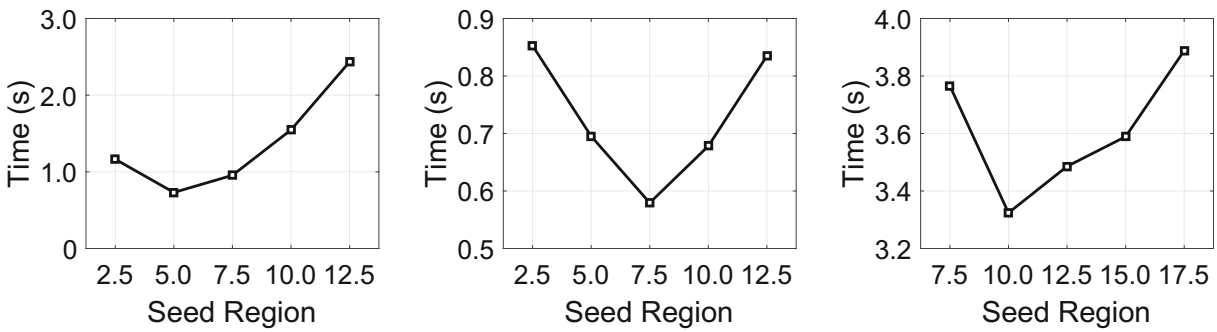
Truck-tractor system We adopt this vehicle from [27]. Coordinates of the truck's midpoint is (x, y), θ_0 and θ_1 is the heading of truck and tractor, δ is the tractor steering angle, d_0 is the distance between truck and hitch joint on tractor rear axle and d_1 is the distance between front and rear axles of the tractor. The equations of motion are:

$$\begin{aligned} \dot{\theta_1}(t) &= V_c tan(\delta)/d_1 \\ \dot{\theta_0}(t) &= V_c sin(\theta_1 - \theta_0)/d_0 \\ \dot{x}(t) &= V_c cos(\theta_1 - \theta_0)cos(\theta_0) \\ \dot{y}(t) &= V_c cos(\theta_1 - \theta_0)sin(\theta_0) \end{aligned} \tag{4}$$

wherein V_c is the vehicle speed, and Δq for the RRT is 2.0.

4.2 Test Environments

Three environments are included in the tests (see Figs. 1–3). We note that this work is intended for real applications; thus, we use environments that are similar to regular 2D building maps. The *Office* environment has a narrow passage in the entrance, and several obstacles are laid between the start and goal (see Fig. 1). In the *Lab* environment, the vehicle needs to find a path



(a) *Office* (b) *Lab* (c) *Floor*

Fig. 5 Planning time of the skilled-RRT for different environments using different seed regions. All the simulation runs are conducted using omnidirectional car ($W = 1.0$ and $\Psi_{HB} = 50°$). Results are averaged over 100 runs

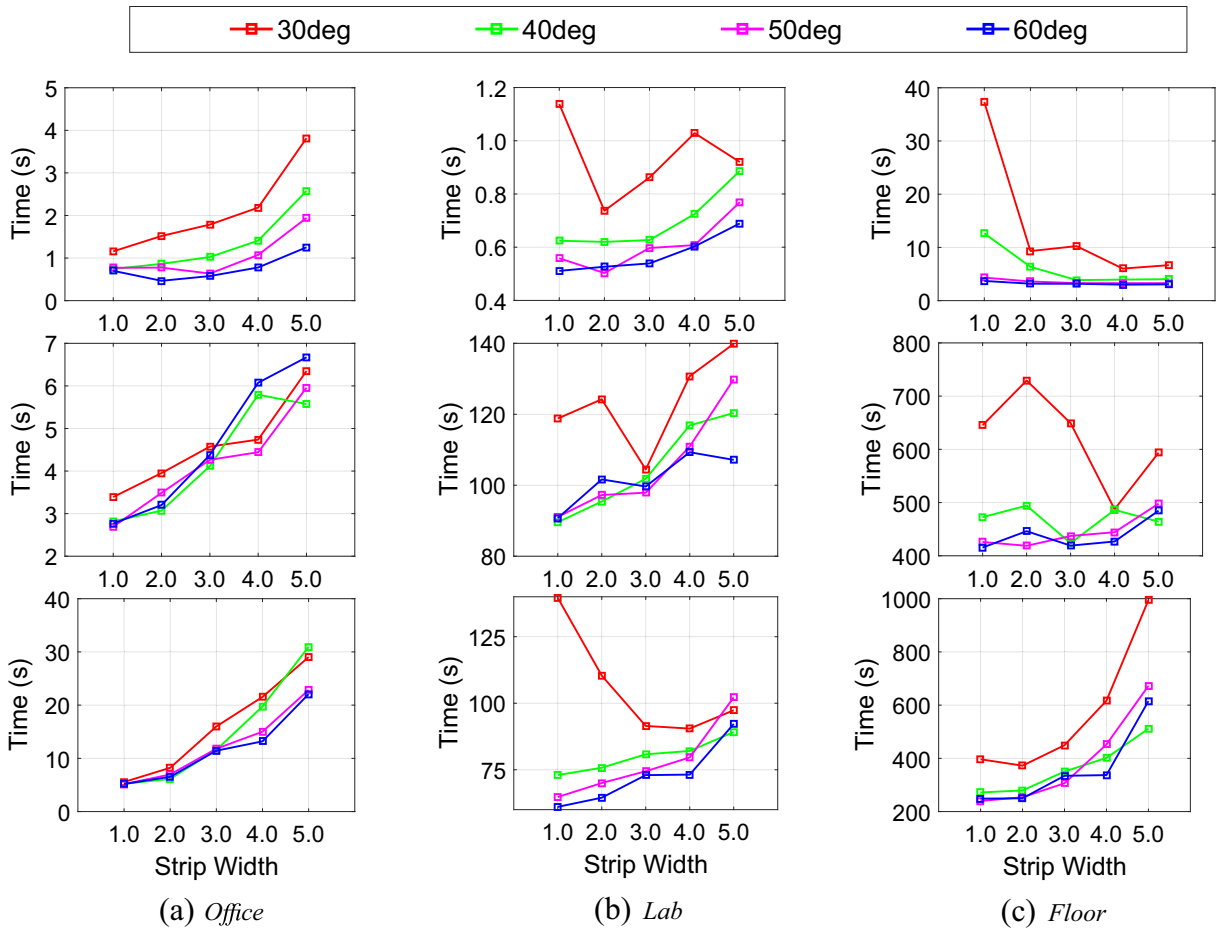


Fig. 6 Planning time of the skilled-RRT with different strip width and heading bias. **Top row:** Omnidirectional car. **Middle row:** Nonholonomic car. **Bottom row:** Truck-tractor system. Results are averaged over 100 simulation runs

around multiple obstacles, and pass through a narrow entrance to the goal (see Fig. 2). The *Floor* environment is the most complex one, which consists of many narrow passages and deadlocks (see Fig. 3). The sizes of *Office*, *Lab*, and *Floor* are 50×50, 100×100, and 200×200, respectively.

4.3 Baseline Algorithms Specifics

All the planners in this paper use the same functions wherever applicable. For A*-RRT and Theta*-RRT, we need to find the A* and Theta* paths first. In both, the environment needs to be discretized, and the grid size to be specified first. In literature, there is not a universal standard for this discretization. In this paper, following [27], we use the grid size of 10.0 for all environments. For MARRT it is crucial to decide the parameters including Δq, ΔI, and ϵ (see [9]). Some rules are mentioned in [9] for point robot, but it is not clear how these rules are decided, and whether or not they are adaptive to other environments. In this work, we apply MARRT to the OmniCar only (which is dynamically similar to the point robot). Also via extensive experimental test, we use Δq as 5% of the environment diagonal, $\Delta I = \Delta q/10$, and $\epsilon = 0.01$ for all simulations.

4.4 Parameters of the Skilled-RRT

Some manipulation parameters are defined for the skilled-RRT: Ω_{SR}, W, and Ψ_{HB} (see Fig. 3). In real work, it is not practical to tune these parameters on

Table 1 Performance of different planners

	Planner	Planning Time [s]	Maximum Clearance	Average Clearance	Path Length	Success Rate [%]
Environment: *Office*						
Omnidirectional Car	RRT	3.5043	9.0190	3.0141	114.6482	100
	RRT*	12.7975	8.6180	3.0534	108.5156	100
	A*–RRT	14.2492	7.2440	2.3743	107.3675	100
	Theta*–RRT	17.7873	7.1400	2.3525	104.8204	100
	MARRT	1.4902	11.9160	4.2441	103.3086	100
	skilled–RRT	0.5162	11.6030	4.2036	102.9865	100
Nonholonomic Car	RRT	44.4661	9.7450	3.0650	127.1183	100
	RRT*	404.4246	9.2383	3.1933	130.6678	47
	A*–RRT	85.0271	7.2430	2.1245	103.9476	100
	Theta*–RRT	63.7884	7.3510	2.1424	103.8430	100
	skilled–RRT	2.5950	11.8840	4.3201	100.2080	100
Truck-tractor System	RRT	182.0177	8.6143	3.1259	102.9232	7
	RRT*	307.4787	8.0800	3.3063	105.2596	10
	A*–RRT	213.9533	7.6211	2.5534	94.4167	19
	Theta*–RRT	379.9821	7.8667	2.5817	95.1164	15
	skilled–RRT	5.2325	11.9820	4.4681	100.4303	100
Environment: *Lab*						
Omnidirectional Car	RRT	3.6788	13.6110	5.7871	149.0386	100
	RRT*	9.9743	13.7070	6.0123	147.2040	100
	A*–RRT	3.7124	13.4750	4.2712	127.7310	100
	Theta*–RRT	3.7837	14.0290	4.4892	125.4271	100
	MARRT	1.2342	14.4960	8.2802	139.0930	100
	skilled–RRT	0.4901	14.3800	8.4311	124.6266	100
Nonholonomic Car	RRT	254.3144	14.0042	5.8927	149.7167	96
	RRT*	401.8987	14.0038	6.0220	152.0703	78
	A*–RRT	188.0022	13.6760	4.3108	123.1798	100
	Theta*–RRT	236.8468	14.4731	4.4935	119.9984	100
	skilled–RRT	77.3323	14.5610	8.4557	124.2744	100
Truck-tractor System	RRT	201.5992	14.0598	6.2296	126.6071	92
	RRT*	211.4062	14.0011	6.3443	125.1061	95
	A*–RRT	235.0171	13.1753	4.7741	111.4925	73
	Theta*–RRT	120.9394	14.4250	5.0578	107.7357	96
	skilled–RRT	72.0536	14.5500	8.8520	117.8359	100
Environment: *Floor*						
Omnidirectional Car	RRT	162.4284	8.1083	2.0876	373.2479	24
	RRT*	–	–	–	–	0
	A*–RRT	8.3006	7.9330	2.2126	350.8917	100
	Theta*–RRT	10.4602	7.9330	1.7261	342.9732	100
	MARRT	123.4594	8.5000	4.0191	339.7822	100
	skilled–RRT	3.8091	8.4640	3.6469	326.6068	100

Table 1 (continued)

	Planner	Planning Time [s]	Maximum Clearance	Average Clearance	Path Length	Success Rate [%]
Nonholonomic Car	RRT	–	–	–	–	0
	RRT*	–	–	–	–	0
	A*–RRT	871.5369	5.9344	2.5388	355.5575	32
	Theta*–RRT	785.1456	4.6017	2.1932	348.4687	26
	skilled–RRT	381.0321	8.6001	3.6901	324.0090	100
Truck-tractor System	RRT	–	–	–	–	0
	RRT*	–	–	–	–	0
	A*–RRT	878.7168	5.5743	2.5436	329.8238	6
	Theta*–RRT	–	–	–	–	0
	skilled–RRT	324.7729	8.5970	3.8173	324.7729	100

the field, we hope to find "universal" parameters/rules which are adaptive to all vehicles/(regular 2D building) environments. In this work, we first fix W and Ψ_{HB} at 1.0 and 50°, and measure the time of the skilled-RRT for the three environments using omnidir ectional car (see Fig. 5[5]). This plot corresponds to the physical intrinsics of Ω_{SR}. If the region is small, it is less probable that the seed is laid within the strip and heading bias. Or if the region is large, the seed may lead the vehicle into a collision. In either case, planning time will increase. We select the seed region that corresponds to the fastest solution; i.e., for *Office* 5.0, *Lab* 7.5, and *Floor* 10.0. Note that the dimensions of *Office*, *Lab*, and *Floor* are 50, 100, and 200, a mathematical rule to decide the seed region can be regressed as:

$$\Omega_{SR} = 5.0 + 2.5 \times \log_2^{D/50} \tag{5}$$

where D represents the dimension of the environment.[6]

Strip width and heading bias are more closely related. As in Fig. 3, if the strip width is narrow, large heading bias is needed to generate the seed to track the sharp turning on the skeleton path. Or if the heading bias is small, wide strip is needed for the vehicle to manoeuvre around the sharp turning. In order to find the optimal value, we compute the average time over multiple runs (see Fig. 6). Using the seed region determined from Fig. 5, strip widths from 1.0 to 5.0 and heading bias angles from 30^o to 60^o are examined. Based on Fig. 6, $W = 1.0$ and $\Psi_{HB} = 50^o$ are considered to be the best.

5 Experimental Tests

5.1 Planner Performance Metrics

For each environment, we perform 100 runs using different planners via different vehicles. Final results are obtained based on an averaged value from the 100 runs, and we are concerned with the planning time and success rate (which represents the reliability in real applications), maximum and average clearance (following [9]), and length of the path. For A*-RRT, Theta*-RRT, MARRT, and skilled-RRT, we need to extract the geometrical path first, the time consumed for which is then added to tree-growth to account for the total time. Moreover, we implement a time limit of 1,000 seconds; i.e., planning time below 1,000 seconds is considered as success.

See Table 1 for the final simulation results. For all performance metrics, smaller values are preferred for planning time and path length, and larger values are better for maximum and average clearance.

5.2 Simulation Test Results and Discussion

We note again that we intend to minimize the planning time. As in Table 1, for all cases, skilled-RRT

[5]The SeedRegion represents half of the Ω_{SR} square width in Fig. 3.

[6]Although in real work the rule could be different, this equation proves that it is possible to find one.

(in shadow) yields the fastest planning time. For the success rate, in all cases skilled-RRT successfully finds the path within the time-limit. Based on Table 1, with respect to planning time and success rate, skilled-RRT is considered to be the best. Additionally, it is also to be noted that skilled-RRT claims favourable results on path length and clearance. Below we discuss these results via the analysis on skilled-RRT and each baseline planners.

For RRT and RRT*, no geometrical path is involved; sampling configurations are generated randomly. It is to be noted that theoretically RRT* yields the shortest path if the nodes number is high enough. In our simulation tests, however, the RRT tree size is not large, and the "rewire" cannot be fully achieved. The path returned from RRT*, therefore, can be longer. Also, as RRT* needs to rearrange the tree edges, it is computationally heavier than RRT. The planning time of RRT*, therefore, is longer which can even go beyond 1,000 seconds which means failure decreasing the success rate.

A*-RRT and Theta*-RRT utilize the A* and Theta* path to bias the RRT tree growth. As can be seen in Fig. 1, the returned path from A* and Theta* *must* graze the obstacles, which might hinder the RRT tree growth. Planning time of A*-RRT and Theta*-RRT, therefore, can be longer. In some extreme cases (*Floor*), however, the purely-random RRT and RRT* fails to find a feasible solution (due to too many deadlocks in this environment), A*-RRT and Theta*-RRT is more applicable: success rate is higher, while planning time and clearance being fairly reasonable. This actually illustrates the advantage of biasing the RRT tree growth—the randomized nature can be attenuated and better performance can be yielded.

Both MARRT and skilled-RRT bias the RRT tree growth using skeleton. However, in contrast to the skeleton-constrained path in skilled-RRT, MARRT uses the *entire* skeleton. Some additional efforts need to be allocated for exploring ineffective regions. Planning time of MARRT, therefore, can be longer. Also note that in MARRT, q_{rand} is randomly generated, and is used also for vehicle navigation. In skilled-RRT, however, we use $Seed_L$ to decide q_{near}, and we generate the navigation seed $Seed_N$ to the front of the RRT tree. Tree growth in skilled-RRT, therefore, is smoother and hence faster. With respect to planning time and reliability, as can be seen in Table 1, although MARRT is also reliable (100% success rate for all cases), skilled-RRT is comparably faster.

The MARRT claims the best results of clearance due to its tree being located at ϵ close to the skeleton. In skilled-RRT, however, the sampling configurations are generated at the neighbourhood of q_{near}, and the tree is restricted within a W width along the skeleton. Path clearance is therefore also comparable (second place after MARRT in all cases).

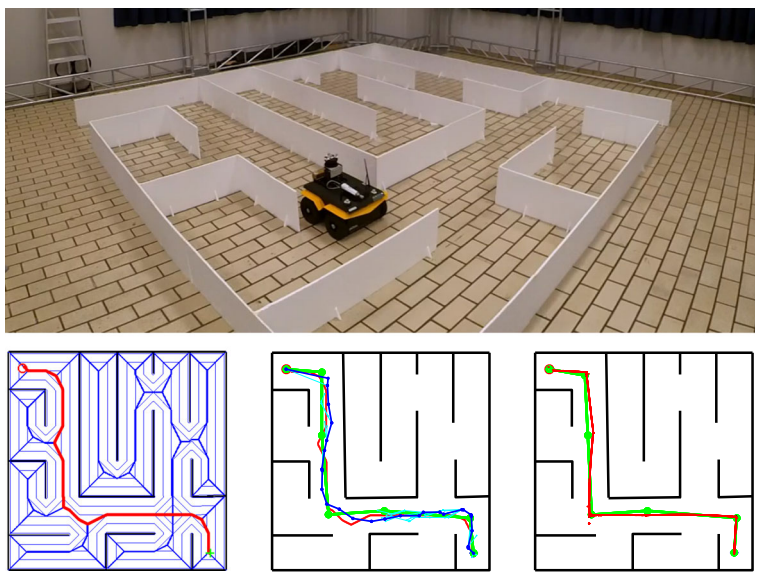


Fig. 7 Real test results. **Top:** Test environment and the UGV. **Bottom left:** Wave-front algorithm (blue) and the skeleton-constrained path (red). **Bottom middle:** Growth of the skeleton path-biased RRT tree (cyan dot), feasible path (blue dot) and optimized path (green circle) generation. **Bottom right:** Green circle lines denote desired path, and red lines represents vehicle tracking results

Table 2 Planning time for three algorithms

Planner	Theta*–RRT	MARRT	$skilled-RRT$
Planning time [s]	0.8561	0.7492	**0.6136**

In skilled-RRT, although no manipulation with respect to path length is deployed, we note that the length is also creditable (especially when compared to RRT* and RRT). This is mainly because the skilled-RRT biases the tree growth along the skeleton ($W = 1.0$); on one hand the zigzag from the original RRT/RRT* can be avoided, and on the other hand the path is allowed to take more short-cuts compared with MARRT ($\epsilon = 0.01$).

To summarize, the exclusive features of skilled-RRT which make the method superior present themselves as:

1) Compared to RRT and RRT*, the random nature of the seeds generation is attenuated. The skilled-RRT tree-growth is more focused.
2) Compared to A*-RRT and Theta*-RRT, the skeleton-path locates away from the obstacles, and the tree expands to the front only. The tree-growth is therefore smoother.
3) Compared to MARRT, the skilled-RRT propagates along the skeleton path, while MARRT inspects the entire skeleton and some ineffective explorations might be involved. Moreover, skilled-RRT explores only ahead of the RRT tree, which also expedites the tree growth.

6 Test in Real Environment

In this section, we present and discuss our experimental tests using an unmanned ground vehicle (UGV) in an indoor environment. See the general work space in Fig. 7; we build up a 2D obstacles maze first, then measure the position of the obstacles using the OptiTrack system. The main objective of this test is to find a path which can guide the vehicle from the start point (bottom right green cross) to the goal (top left red circle). In our test, we deploy the skilled-RRT for this navigation task, and we also include Theta*-RRT and MARRT as baseline planners for the comparison and discussion on overall planning performance.

A video file recording the final experimental test has been posted online.[7] For both baseline planners and the skilled-RRT, we need to extract a geometrical path first, then the RRT tree is grown along this path. We compute the overall planning time of each algorithm by adding the time needed for both the search of the geometrical path and the RRT tree growth. In the posted video and also in Table 2 above, the overall planning time which is needed for the baseline planners (Theta*-RRT, MARRT) and skilled-RRT are characterized. As shown by the results, skilled-RRT is faster than both Theta*-RRT (claimed as the fastest so far) and MARRT (the most similar to skilled-RRT).

We note that from the skilled-RRT, we have the feasible path only. In our test, we adopt the "off-line" path-optimization approach; i.e., we firstly find the feasible path using skilled-RRT, then we run the optimization to shorten the overall path length; see [11, 12] for more details. Generally, by randomly picking up two points on the feasible path by skilled-RRT, we strive to connect them directly using the vehicle navigation controller. If the navigation is successful (collision-free), and the returned path length is shorter than the original (skilled-RRT) path, the original path will be replaced. In this manner, we are trying to tune the overall path in a non-increasing manner, and after sufficient tuning trials, the resultant path is returned as the optimal one. As in Fig. 7, given the optimized path (green circle lines), ROS (Python)-based communication is used to control the vehicle to track the waypoints (green circles) on the path till the final goal point.

7 Conclusions

In this work, for faster path planning in 2D building environment, we propose to combine the skeleton-constrained path biasing with RRT-based nonholonomic motion planning, and we introduce the skilled-RRT algorithm. Compared to the other five baseline (RRT-based) planners (RRT, RRT*, A*-RRT, Theta*-RRT, and MARRT), the skilled-RRT yields faster planning time and higher path planning success rate, while the path length and path clearance also being fairly reasonable. Under mild assumptions, we prove

[7] Please see this video at: https://www.youtube.com/watch?v=-gvrtqVmQJk.

that the skilled-RRT retains the probabilistic completeness of the original RRT for small-time controllable systems. As a continuation of this work, we plan to examine the reliability of the skilled-RRT algorithm more extensively. Following the experiments in this paper, more real-vehicle tests will be conducted using different vehicles in various environments.

Acknowledgements The research was partially supported by the ST Engineering-NTU Corporate Lab through the NRF corporate lab@university scheme. The authors are also indebted to Mr. Mohhamadali Askari Hemmat from Department of Mechanical and Industrial Engineering in Concordia University Canada for the discussion on this idea, and Mr. Reinaldo Maslim from School of Mechanical and Aerospace Engineering in Nanyang Technological University for the real test.

References

1. Aguinaga, I., Borro, D., Matey, L.: Parallel rrt-based path planning for selective disassembly planning. Int. J. Adv. Manuf. Technol. **36**(11-12), 1221–1233 (2008)
2. Alterovitz, R., Patil, S., Derbakova, A.: Rapidly-exploring roadmaps: Weighing exploration vs. refinement in optimal motion planning. In: 2011 IEEE International Conference on Robotics and Automation (ICRA), pp. 3706–3712 (2011)
3. Arslan, O., Tsiotras, P.: Dynamic programming guided exploration for sampling-based motion planning algorithms. In: 2015 IEEE International Conference on Robotics and Automation (ICRA), pp. 4819–4826 (2015)
4. Bekris, K., Kavraki, L.: Informed and probabilistically complete search for motion planning under differential constraints. In: First International Symposium on Search Techniques in Artificial Intelligence and Robotics (STAIR), Chicago, IL (2008)
5. Brunner, M., Brüggemann, B., Schulz, D.: Hierarchical rough terrain motion planning using an optimal sampling-based method. In: 2013 IEEE International Conference on Robotics and Automation (ICRA), pp. 5539–5544 (2013)
6. Choudhury, S., Gammell, J.D., Barfoot, T.D., Srinivasa, S.S., Scherer, S.: Regionally accelerated batch informed trees (rabit*): A framework to integrate local information into optimal path planning. In: 2016 IEEE International Conference on Robotics and Automation (ICRA), pp. 4207–4214 (2016)
7. Cowlagi, R.V., Tsiotras, P.: Hierarchical motion planning with dynamical feasibility guarantees for mobile robotic vehicles. IEEE Trans. Robot. **28**(2), 379–395 (2012)
8. Daniel, K., Nash, A., Koenig, S., Felner, A.: Theta*: Any-angle path planning on grids. J. Artif. Intell. Res. **39**, 533–579 (2010)
9. Denny, J., Greco, E., Thomas, S., Amato, N.M.: Marrt: Medial axis biased rapidly-exploring random trees. In: 2014 IEEE International Conference on Robotics and Automation (ICRA), pp. 90–97 (2014). doi:10.1109/ICRA.2014.6906594
10. Denny, J., Colbert, J., Qin, H., Amato, N.M.: On the theory of user-guided planning. In: 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 4794–4801. IEEE (2016)
11. Dong, Y., Zhang, Y.: Application of rrt algorithm to unmanned ground vehicle motion planning and obstacle avoidance. In: Proceedings of International Conference on Intelligent Unmanned Systems, vol. 11 (2015)
12. Dong, Y., Zhang, Y., Ai, J.: Experimental test of unmanned ground vehicle delivering goods using RRT path planning algorithm. Unmanned Syst. **5**(1), 45–57 (2017). doi:10.1142/S2301385017500042
13. Gammell, J.D., Srinivasa, S.S., Barfoot, T.D.: Informed rrt*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic. In: 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 2997–3004 (2014)
14. Gammell, J.D., Srinivasa, S.S., Barfoot, T.D.: Batch informed trees (bit*): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs. In: 2015 IEEE International Conference on Robotics and Automation (ICRA), pp. 3067–3074 (2015)
15. Garrido, S., Moreno, L., Abderrahim, M., Martin, F.: Path planning for mobile robot navigation using voronoi diagram and fast marching. In: 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 2376–2381 (2006). doi:10.1109/IROS.2006.282649
16. Geraerts, R.: Planning short paths with clearance using explicit corridors. In: 2010 IEEE International Conference on Robotics and Automation (ICRA), pp. 1997–2004. IEEE (2010)
17. Jaillet, L., Cortés, J., Siméon, T.: Sampling-based path planning on configuration-space costmaps. IEEE Trans. Robot. **26**(4), 635–646 (2010)
18. Jalel, S., Marthon, P., Hamouda, A.: A new path generation algorithm based on accurate nurbs curves. International Journal of Advanced Robotic Systems 13, doi:10.5772/63072 (2016)
19. Kalisiak, M., van de Panne, M.: Faster motion planning using learned local viability models. In: Proceedings 2007 IEEE International Conference on Robotics and Automation, pp. 2700–2705 (2007)
20. Karaman, S., Frazzoli, E.: Sampling-based algorithms for optimal motion planning. Int. J. Robot. Res. **30**(7), 846–894 (2011)
21. Kuffner, J.J., LaValle, S.M.: Rrt-Connect: an efficient approach to single-query path planning. In: Proceedings. ICRA'00. IEEE International Conference On Robotics and Automation, 2000, vol. 2, pp. 995–1001. IEEE (2000)
22. Latombe, J.: Robot Motion Planning. The Springer International Series in Engineering and Computer Science. Springer, USA (2012)
23. LaValle, S.M., Kuffner, J.J.: Randomized kinodynamic planning. Int. J. Robot. Res. **20**(5), 378–400 (2001)
24. Mirtich, B., Canny, J.: Using skeletons for nonholonomic path planning among obstacles. In: Proceedings

1992 IEEE International Conference on Robotics and Automation, vol. 3, pp. 2533–2540 (1992). doi:10.1109/ROBOT.1992.220060
25. Neto, A.A., Macharet, D.G., Campos, M.F.: On the generation of trajectories for multiple uavs in environments with obstacles. In: Selected papers from the 2nd International Symposium on UAVs, Reno, Nevada, USA June 8–10, 2009, pp. 123–141. Springer (2009)
26. Oriolo, G., Vendittelli, M., Ulivi, G.: Path planning for mobile robots via skeletons on fuzzy maps. Intell. Autom. Soft Comput. **2**(4), 355–374 (1996)
27. Palmieri, L., Koenig, S., Arras, K.O.: Rrt-based nonholonomic motion planning using any-angle path biasing. In: 2016 IEEE International Conference on Robotics and Automation (ICRA), pp. 2775–2781 (2016)
28. 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)
29. Plaku, K.L.E.E., Vardi, M.Y.: Motion planning with dynamics by a synergistic combination of layers of planning. IEEE Trans. Robot. **26**(3), 469–482 (2010)
30. Rickert, M., Brock, O., Knoll, A.: Balancing exploration and exploitation in motion planning. In: IEEE International Conference on Robotics and Automation, 2008. ICRA, pp. 2812–2817 (2008)
31. Rickert, M., Sieverling, A., Brock, O.: Balancing exploration and exploitation in sampling-based motion planning. IEEE Trans. Robot. **30**(6), 1305–1317 (2014)
32. Rodriguez, T.X., Lien, J.M., Amato, N.M.: An obstacle-based rapidly-exploring random tree. In: Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006, pp. 895–900 (2006). doi:10.1109/ROBOT.2006.1641823
33. Taïx, M., Flavigné, D., Ferré, E.: Human interaction with motion planning algorithm. J. Intell. Robot. Syst. **67**(3), 285–306 (2012)
34. Yang, D.H., Hong, S.K.: A roadmap construction algorithm for mobile robot path planning using skeleton maps. Adv. Robot. **21**(1-2), 51–63 (2007)
35. Yang, K., Moon, S., Yoo, S., Kang, J., Doh, N.L., Kim, H.B., Joo, S.: Spline-based rrt path planner for nonholonomic robots. J. Intell. Robot. Syst. **73**(1-4), 763 (2014)

Yiqun Dong Born in 1990, Dr. Dong completed his Bachelor in Fudan University Shanghai China in 2010, he then moved on to the graduate school in the same place till 2016 with a Ph.D. in Aerospace Control. From 2014 October to 2016 March, Dr. Dong was an exchange Ph.D. student in Concordia University, Montreal, Quebec Canada. Dr. Dong's research is focused on path planning, bottom-level control, and image-based positioning/localization, etc. Now he is also working on the Deep Learning-based applications. For more information please see www.yiqundong.com.

Efe Camci was born in Izmir, Turkey. He went to Izmir Ataturk High School, Izmir, Turkey. He received his B.Sc. degree in Aerospace Engineering at Middle East Technical University, Ankara, Turkey. He is currently pursuing his research as a Ph.D. candidate in School of Mechanical and Aerospace Engineering at Nanyang Technological University, Singapore. His research areas are flight control, reinforcement learning, and unmanned aerial vehicles.

Dr. Erdal Kayacan holds a PhD in Electrical and Electronic Engineering from Bogazici University (2011). He was a visiting scholar in University of Oslo in 2009 with the research fellowship of Norway Research Council. After his post-doctoral research in KU Leuven in the Division of Mechatronics, Biostatistics and Sensors (MeBioS), Dr. Kayacan went on to pursue his research in Nanyang Technological University at the School of Mechanical and Aerospace Engineering as assistant professor (2014–current).

His current research projects focus on the design and development of ground and aerial robotic systems, vision-based control techniques and artificial intelligence. Dr. Kayacan is co-writer of a course book "Fuzzy Neural Networks for Real Time Control Applications, 1st Edition Concepts, Modeling and Algorithms for Fast Learning", Butterworth-Heinemann, Print Book ISBN:9780128026878. (17 Sept 2015). He is a Senior Member of Institute of Electrical and Electronics Engineers (IEEE). From 1st Jan 2017, he is an Associate Editor of IEEE Transactions on Fuzzy Systems.