J Intell Robot Syst
DOI 10.1007/s10846-013-9963-y

Spline-Based RRT Path Planner for Non-Holonomic Robots

Kwangjin Yang · Sangwoo Moon · Seunghoon Yoo · Jaehyeon Kang · Nakju Lett Doh · Hong Bong Kim · Sanghyun Joo

Received: 28 August 2013 / Accepted: 16 September 2013

Abstract Planning in a cluttered environment under differential constraints is a difficult problem because the planner must satisfy the external constraints that arise from obstacles in the environment and the internal constraints due to the kinematic/dynamic limitations of the robot. This paper proposes a novel Spline-based Rapidly-exploring Random Tree (SRRT) algorithm which treats both the external and internal constraints simultaneously and efficiently. The computationally expensive numerical integration of the system dynamics is replaced by an efficient spline curve parameterization. In addition, the SRRT guarantees continuity of curvature along the path satisfying any upper-bounded curvature constraints. This paper presents the underlying theory to the SRRT algorithm and presents simulation and experiment results of a mobile robot efficiently navigating through cluttered environments.

Keywords Differential constraints · Rapidly-exploring random tree · Spline curve parameterization · Mobile robot

This work was supported in part by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MEST) (No. 2012R1A1A4A01005563).

K. Yang (✉) · S. Moon
Department of Aerospace and Mechanical Engineering, Korea Air Force Academy, Cheongwon, Republic of Korea
e-mail: ykj4957@gmail.com

S. Yoo
Department of Electrical Engineering and Computer Science, Korea Air Force Academy, Cheongwon, Republic of Korea

J. Kang · N. L. Doh
School of Electrical Engineering , Korea University, Seoul, Republic of Korea

H. B. Kim
Future Man Electronics, Daejon, Republic of Korea

S. Joo
Agency for Defense Development, Daejon, Republic of Korea

1 Introduction

Autonomous navigation in unknown cluttered environments is a challenging problem since both the obstacles and the robot's differential constraints have to be taken into account. Various path planning techniques have been proposed for computing a collision free path, predominately by dividing the problem into a global planning stage which aims to define a path that avoids obstacles, which is then refined by a local planner that takes care of the differential constraints of the robot. Recent research has focused on dealing with differential constraints directly in the global planner removing the refinement process. This problem can be considered as a class of two point boundary value problems, which compute a path that connects initial and goal states while satisfying differential constraints [1]. Before detailing the proposed algorithm, previous work on path planning dealing with differential constraints will be described.

1.1 Related Work

Understanding the maneuverability limitations of a vehicle is important in order to generate a kinematically feasible path. Research in this area can be categorized into two approaches: hierarchical and global planning. The global approach is further divided into deterministic and probabilistic search.

1.1.1 Hierarchical Planning

Hierarchical planning is divided into two steps. In the first step, the motion planner computes a collision free path from the initial to the goal state without considering differential constraints, usually as a selection of linear paths. In the second step, differential constraints are considered in order to refine the path locally. The refinement algorithms can be sorted into two groups depending on their methodology in tackling the nonholonomic constraints: path smoothing and local planning. The objective of *path smoothing* is to smooth sharp cusps of the originally generated linear path. Dubins curves are the most popular method used for path smoothing [2–4]. Clothoids [5, 6] and splines [7] have also been applied for path smoothing. The disadvantage of such approaches is that the resulting path is generally inefficient (or may not even be executable) since the differential constraints are not considered in the global path generation step.

The *local planning* method computes a feasible path to the local goal point which exists on the initial global path instead of applying a refinement algorithm to the whole path. In [8] a model-predictive high-fidelity vehicle model is used to generate the feasible local paths. This local planning strategy was adopted not only in the DARPA Grand Challenge [9] but also in the DARPA Urban Challenge [10, 11]. In addition, this method was used on an unmanned aerial vehicle in [12] which applied machine learning technology to generate a dynamically feasible path given unanticipated obstacles. Even though local planning methods can compute precise local maneuvers, they are susceptible to local minima [13].

1.1.2 Deterministic Search

Hierarchical planning has difficulty in satisfying differential constraints due to the decoupled planning and smoothing processes. Discrete representation of environmental and differential constraints is a well-recognized method to generate feasible paths with efficient computation. Deterministic Search uses this method to take differential constraints into account in the planning process. Deterministic search is further classified into control and state space discretization methods.

The control space sampling method discretizes the control space into a finite subset and applies a control input from this set during fixed times [14]. Therefore, the control space sampling method satisfies the differential constraints by construction because all control inputs are realizable. However, this algorithm is still computationally intensive. Furthermore, time and input discretizations are required in advance which can be difficult to determine.

State space sampling is a discretization of the configuration space into a set of states, representing configurations and connections between these states, where every connection represents a feasible path. State lattices provide a means for motion planning problems to be formulated as graph search problems [13]. The drawback of using graph-search techniques is the resolution lost due to discretization [15]. The only boundary states that can be reached are those that already exist in the network. Even though it has a powerful ability to generate a feasible path under differential constraints, it requires a high fidelity dynamic model of the system. Moreover, efficient state space discretization is non-trivial task. That is, the computational benefit of the discretization comes at the cost of sacrificing optimality and completeness.

1.1.3 Probabilistic Search

Instead of maintaining a fixed structure of lattices, the probabilistic search approach builds a feasible tree on the fly. Two traditional probabilistic search methods are Probabilistic Roadmaps (PRM) and Rapidly-exploring Random Trees (RRT) [16]. The PRM planner consists of an off-line roadmap construction step and an on-line graph search step for path generation. It constructs a roadmap of the configuration space by sampling a random point and trying to link it from each nearby sample point using local planning. After constructing a roadmap network, graph search is applied online once the initial point and goal points are assigned. However, it is difficult to apply PRM to a dynamic and rapidly changing environment as building a roadmap a priori may not be feasible [16]. Furthermore, the PRM algorithm is not well suited to nonholonomic path planning since it is difficult to design a good local planner which connects pairs of nearby configurations [16]. It is generally recognized that the complexity of designing a good local planner is comparable to the original nonholonomic planning problem.

The RRT planner incrementally builds a tree on-line from the starting point to the goal point by extending towards randomly selected points. The RRT algorithm tends to achieve efficient single-query planning by exploring the environment as little as possible [16]. A good trait of RRTs is that the tree grows towards unvisited regions of the space and never regresses into already explored areas which results in a rapid exploration of the space (Voronoi bias). After RRT was proposed in [17], it gains great popularity in motion planning community. The performance of RRT is enhanced by simply biased toward the goal state [16]. The RRT-connect extending tree until collision is detected and bi-directional RRT growing trees from both the start and goal states were also devised in [18]. In [19], the dynamic-domain RRTs was proposed to solve the narrow passage. An utility-guided RRT which directs tree expansion towards maximum utility regions was proposed in [20]. The most promising regions for exploration are obtained from previous explorations and the parameters that control tree expansion are updated during the planning process. Another approach which combines machine learning techniques such as gaussian process are proposed in [21–23]. In [24], an optimization heuristics which include path cost for tree expansion is suggested. This cost based approach is further developed in [25, 26]. Recently, optimal RRT path planning algorithm was proposed in [27] which converges to the optimal solution as the increase of the number of samples and incorporation of [28–30] by using this method.

However, probabilistic search methods require discretization of the input space and so if the set of inputs is small the resulting path will exhibit poor performance, and yet if it is too dense, it will increase planning time. In addition, if the time interval is too small, then significant tree expansion is required to compute a path and if it is too large, then it can fall into the same problem as sparse input discretization.

1.2 Approach

The efficiency of RRT stems from the Voronoi bias property which promotes tree growth towards unexplored regions. In other words, nodes closest to larger unexplored regions are more likely to grow the tree towards these regions. Therefore the key is determining the distance metric which computes the nearest-neighbor in the RRT algorithm.

In holonomic planning, the Euclidean distance is an ideal metric to generate a Voronoi bias because any node which is the closest from the sampled points can be expanded. If there exists differential constraints, however, which limit the evolution of the system states, the Euclidean distance measure fails to capture the true distance. Figure 1 exhibits this problem. A state x_r is drawn randomly in Fig. 1a. Here, x_s is an initial state of the system and $x_1 \cdots x_8$ are existing nodes in the current tree. If Euclidean distance is used for the distance metric, x_2 is chosen as the closest node from x_r as shown in Fig. 1b. However, this is not true for the system which has differential constraints. Instead, x_4 is the closest node from x_r

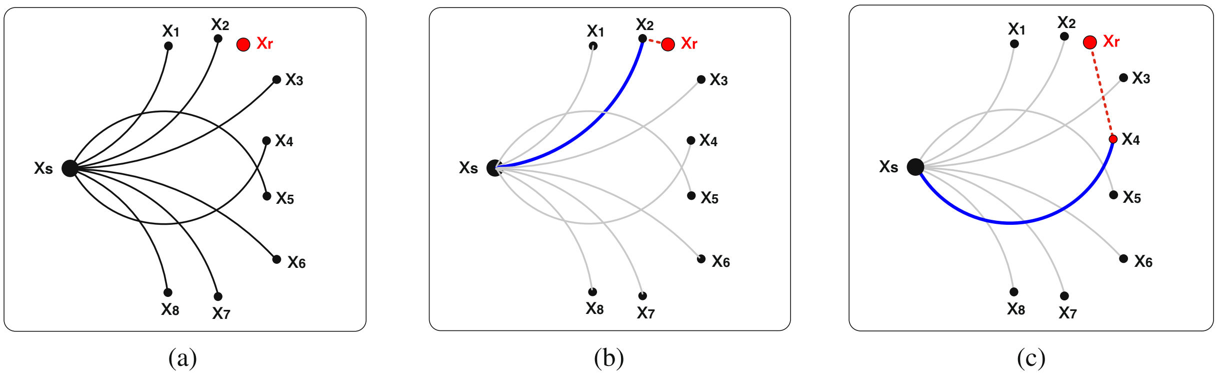

Fig. 1 Nearest neighbor calculation: **a** A state x_r is drawn randomly where x_s is an initial state of the system and $x_1 \cdots x_8$ are existing nodes in the current tree. **b** The Euclidean distance is a ideal metric to generate Voronoi bias in holonomic planning but it does not incorporate differential constraints of the system. **c** In nonholonomic planning, a new distance metric which incorporates limitations of the system is needed to compute the real distance

if the nonholonomic constraints are considered as in Fig. 1c. From this example, it can be seen that the true distance metric is extremely important to the RRT planner under differential constraints.

Conventional RRT chooses an input $(u(t))$ to apply to the system, and determines the duration time (Δt) of application of the selected input. Using this input and time interval, a forward simulation will obtain the state at time $t + \Delta t$. In this way, RRT designs an open-loop control law for the system to plan a feasible path under differential constraints.

Though RRT satisfies the differential constraints of the system by choosing the allowable input and applying forward simulation using this input, there are several problems. First, the performance of RRT is sensitive to the size of the input set and the time interval for the state evolution. Secondly, it is difficult to generate a good sequence of control inputs by applying constant control inputs with fixed time [31]. Thirdly, there is no proper distance measure for the decision of the nearest-neighbor node selection under differential constraints. Finally, there is a discontinuity of input because it is selected randomly among the input set. This piecewise action is problematic for the application of the mobile robot. The discontinuity of the action increases the possibility of slippage and results in a poor localization due to the sharp change of the motion [32].

1.2.1 Spline-Based Rapidly-Exploring Random Trees

Conventional RRT algorithms satisfy the differential constraints by sampling the space of action trajectories [1]. If there is an efficient boundary value problem solver, the problems associated with discretization of the action and time interval can be avoided. This is the motivation of our Spline-based Rapidly-exploring Random Tree (SRRT) algorithm. We utilize cubic Bézier splines as a local planner which connect two states. The time and input discretization are no longer needed and the forward simulation of RRT is replaced by the cubic Bézier splines parameterization. This is achieved by using splines as a local planner which solves the two-point boundary value problem. The local planner connects two points guaranteeing the continuity of curvature and satisfying the upper-bounded curvature constraints.

In addition to the spline based local planner algorithm, several improvements are made to solve the planning problem under differential constraints. First, a new distance measure is applied. A feasibility checking algorithm is devised to satisfy the upper-bounded curvature constraint. This algorithm is combined with the Euclidean distance metric to select a nearest-neighbor node. The Euclidean distance metric encourages RRT to preserve the Voronoi bias property, and the efficient feasibility checking algorithm ensures that the nonholonomic constraints are observed without involving expensive numerical integration. Secondly, a two-phase sampling approach is applied to draw samples. Before the frontier tree reaches a region within a specified distance from the goal point, the exploration sampling strategy is applied to grow trees quickly. After coming close to the goal point, the sampling strategy will shift from this exploration strategy to a concentration strategy which draws samples more near the goal point. This two-phase sampling approach greatly reduces the planning time. Thirdly, an efficient path pruning algorithm is proposed where redundant nodes are easily selected and removed by applying the feasibility checking algorithm.

1.3 Outline

The paper is organized as follows: Section 2 addresses a maximum curvature constraint checking algorithm which examines the feasibility of a node before adding it to the tree. Section 3 presents a spline-based RRT algorithm which incorporates a nonholonomic constraint checking algorithm. Section 4 describes path pruning algorithm which eliminates nodes that cause wavy motions. Section 5 presents simulation results and then Section 6 exhibits experimental results, and conclusions are presented in Section 7.

2 Feasible Path Generation Under Curvature Constraints

Holonomic RRT can add any nodes to the tree if there are no collisions between the node in the tree and the candidate random point. In a SRRT, however, a node can be added to the tree if it satisfies the feasibility condition arising from the differential constraints.

Figure 2 shows the difference between the holonomic RRT and the SRRT. P_v represents the current vehicle position, P_t is a node in the tree and P_1, P_2, P_3, P_4 are candidate points. In a holonomic RRT (Fig. 2a), these four candidate points can be added to the tree but it is not the case for the SRRT.

Figure 2b shows the allowable region of random points which satisfies the feasibility constraint. The feasibility constraint defines an allowable region bordered by the maximum turning angles ($\mathrm{a}_{\max}$ and $\mathrm{a}_{\min}$ respectively). B_u represents a point which stipulates an upper bound of the feasibility constraints, and B_l represents a lower bound of the feasibility constraints. Here $\mathrm{a}_{\max}$ is the supplementary angle of $\angle P_v P_t B_u$ and $\mathrm{a}_{\min}$ is the supplementary angle of $\angle P_v P_t B_l$ respectively. Among the four candidate points, P_1, P_2, P_3 fall within this region but P_4 does not. Therefore, only P_1, P_2, P_3 can be accepted as feasible points in the SRRT. The question then is how to determine the allowable angles $\mathrm{a}_{\max}$ and $\mathrm{a}_{\min}$.

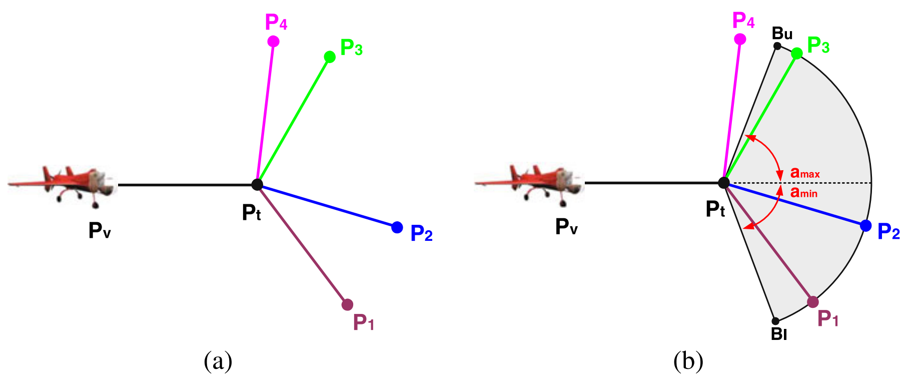

Fig. 2 Feasibility condition for the SRRT: **a** P_v represents the current vehicle position, P_t is a nodes in the tree and P_1, P_2, P_3, P_4 are candidate points. All candidate points can be added to the tree in the holonomic RRT. **b** Among four candidate points, P_4 cannot be added to the tree because it violates the feasibility condition of the SRRT

2.1 Feasibility Condition for the Maximum Curvature Constraints

An analytical algorithm for generating a continuous curvature path which satisfies an upper-bound curvature constraint is suggested in [33, 34]. In that work, the path smoothing algorithm was applied only after the RRT planner generate a path. This decoupled approach cannot guarantee the generation of a feasible path. On the contrary, the path smoothing algorithm is incorporated to the RRT planner in this research which always satisfies the maximum curvature constraints of the system. A Bézier curve is used to generate a path which is defined as:

$$P(s) = \sum_{i=0}^{n} P_i B_{n,i}(s), \quad B_{n,i}(s) = \binom{n}{i} s^i (1-s)^{n-i} \tag{1}$$

where n is a degree of the curve, s is a parameter such that $0 \leq s \leq 1$, P_i are control points, and $B_{n,i}(s)$ are Berstein polynomials.

The eight control points to generate a continuous curvature path among three points (W_1, W_2, W_3) in Fig. 3 are

$$\begin{aligned} B_0 &= W_2 + d \cdot u_1, & B_1 &= B_0 - g_b \cdot u_1, \\ B_2 &= B_1 - h_b \cdot u_1, & B_3 &= B_2 + k_b \cdot u_d, \\ E_0 &= W_2 + d \cdot u_2, & E_1 &= E_0 - g_e \cdot u_2, \\ E_2 &= E_1 - h_e \cdot u_2, & E_3 &= E_2 - k_e \cdot u_d \end{aligned} \tag{2}$$

where u_d is a unit vector along the line $\overline{B_2 E_2}$ and

$$\begin{aligned} h_b &= h_e = c_3 d, \\ g_b &= g_e = c_2 c_3 d, \\ k_b &= k_e = \frac{6 c_3 \cos\beta}{c_2 + 4} d \end{aligned} \tag{3}$$

The two variables β and d are determined as

$$\begin{aligned} \beta &= \frac{\gamma}{2} \\ d &= \frac{c_4 \cdot \sin\beta}{\kappa_{\max} \cdot \cos^2\beta} \end{aligned} \tag{4}$$

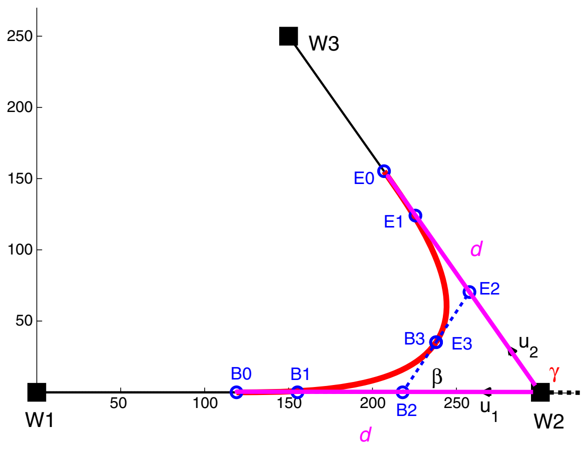

Fig. 3 Upper bounded continuous curvature path smoothing algorithm: the eight control points which generates an upper bounded continuous curvature path are determined once the maximum curvature ($\kappa_{\max}$) and γ are specified

where $\kappa_{\max}$ denotes the maximum curvature and

$$c_1 = 7.2364, \quad c_2 = \frac{2}{5}(\sqrt{6} - 1),$$
$$c_3 = \frac{c_2 + 4}{c_1 + 6}, \quad c_4 = \frac{(c_2 + 4)^2}{54 c_3}$$

2.2 Spline-Based RRT Tree Generation

The allowable angle which needs to be decided is γ in Fig. 3. The maximum curvature ($\kappa_{\max}$) is specified based on the kinematic constraints of the vehicle, thus γ is the unique variable for checking feasibility. Once γ is decided, the length of tree (d) is obtained using Eq. 4

To investigate this property, different values of γ are chosen with corresponding values of d that the length of the linear path similar. Table 1 shows γ, d and the length of linear path according to the tree depth when the maximum curvature is $\kappa_{\max} = 0.1$.

Table 1 Different tree depth with similar path length

	Tree depth = 1	Tree depth = 2	Tree depth = 3
γ	0.40π	0.25π	0.18π
d	10.08	5.03	3.40
Linear path length	20.16	20.13	20.38

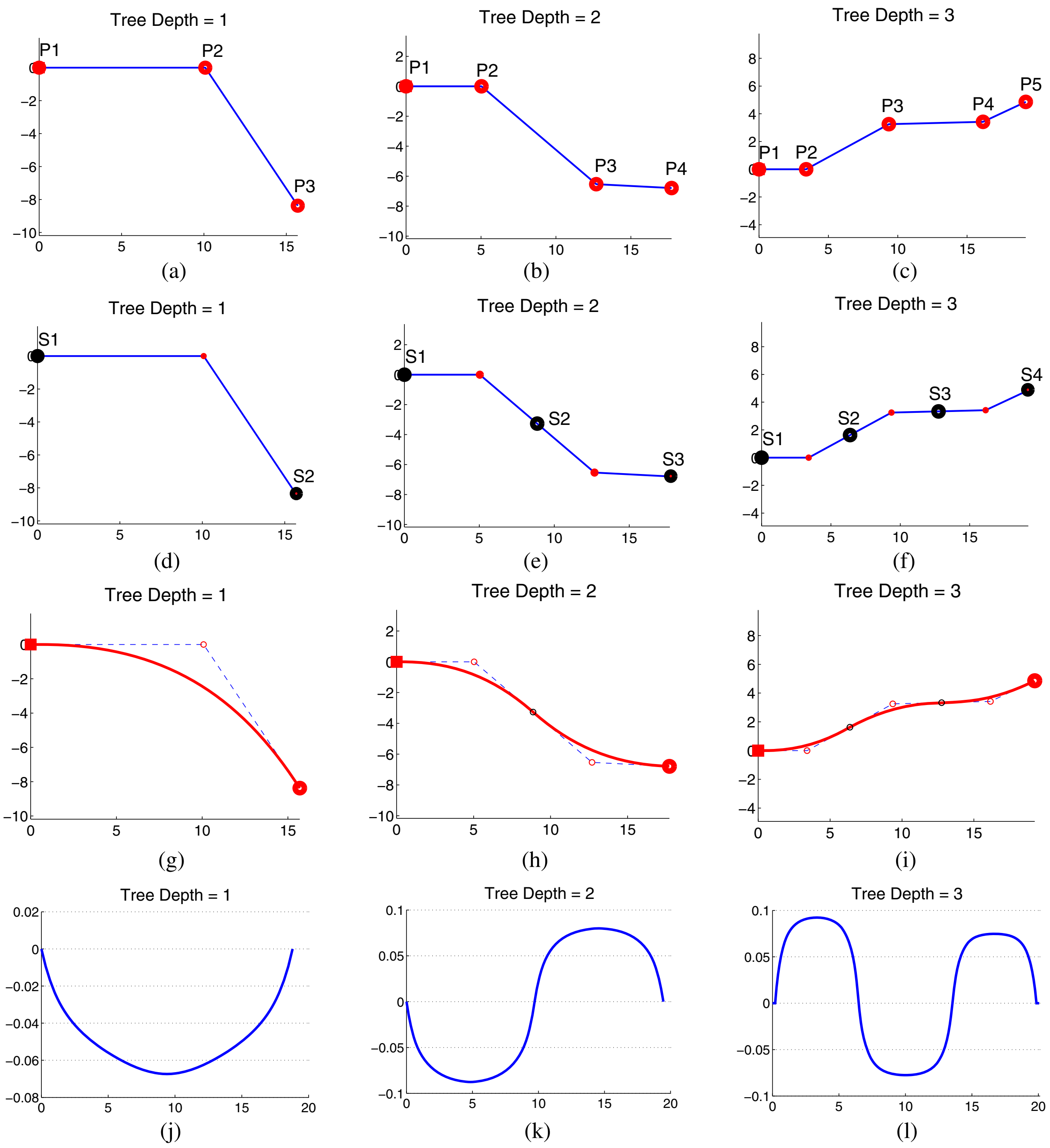

Fig. 4 SRRT path planning according to the tree depths: **a** Nodes when tree depth is one (P_1, P_2 are fixed, and P_3 is chosen randomly). **b** Nodes when tree depth is two (P_1, P_2 are fixed, and P_3, P_4 are chosen randomly). **c** Nodes when tree depth is three (P_1, P_2 are fixed, and P_3, P_4, P_5 are chosen randomly). **d** Smoothing points when tree depth is one (S_1, S_2). **e** Smoothing points when tree depth is two (S_1, S_2, S_3). **f** Smoothing points when tree depth is three (S_1, S_2, S_3, S_4). **g** Spline path when tree depth is one. **h** Spline path when tree depth is two. **i** Spline path when tree depth is three. **j** Curvature of the path when tree depth is one. **k** Curvature of the path when tree depth is two. **l** Curvature of the path when tree depth is three

Figure 4a shows the SRRT tree generation result when the tree depth is one. The first point (P_1) and the second point (P_2) are fixed, and P_3 is chosen randomly within the allowable angle ($\gamma \leqq 0.40\pi$). Figure 4d shows the start and end points used for path smoothing. S_1 corresponds to B_0 and S_2 corresponds to E_0 in Fig. 3. Figure 4g shows the smooth path generated by the cubic Bézier spiral curves and Fig. 4j shows the curvature of the path. Because P_3 is selected as the supplementary angle of $\angle P_1 P_2 P_3$ which is smaller than 0.40π, the maximum curvature of this path is smaller than $\kappa_{\max} = 0.1$.

Figure 4b shows the SRRT tree generation result when the tree depth is two. The first point (P_1) and the second point (P_2) are fixed as the tree depth one case, and P_3 and P_4 are chosen randomly within the allowable angle ($\gamma \leqq 0.25\pi$). There is one more consideration to determine the extension length of the SRRT tree when the tree depth is larger than two. The lines $\overline{P_1 P_2}$ and $\overline{P_2 P_3}$ are used just once when tree depth is one as can be seen in Fig. 4a. If the tree depth increases to two, however, the second line $\overline{P_2 P_3}$ must be used twice for path smoothing as illustrated in Fig. 4e. If the tree depth is three, the second line $\overline{P_2 P_3}$ and the third line $\overline{P_3 P_4}$ must be used twice for path smoothing as in Fig. 4f. Because of this property, the extension length of the SRRT tree must be two times larger than d in Eq. 4 if the tree depth is larger than two. Therefore, the extension length of the RRT for the both end points is

$$d_e = \frac{c_4 \cdot \sin\beta}{\kappa_{\max} \cdot \cos^2\beta} \tag{5}$$

and all other points is

$$d_o = 2\frac{c_4 \cdot \sin\beta}{\kappa_{\max} \cdot \cos^2\beta} \tag{6}$$

Figure 4e shows three points (S_1, S_2, S_3) used for path smoothing. Here S_2 used twice as a end point of the first curve and the starting point of the second curve. The smooth path and the curvature of the path are shown in Fig. 4h and k respectively. The curvature of the path is continuous over the whole path.

Figure 4c shows the SRRT tree generation result when the tree depth is three. Here, P_3, P_4 and P_5 are chosen randomly within the allowable angle ($\gamma \leqq 0.18\pi$). Figure 4i shows the smooth path and Fig. 4l shows the curvature of the path. From Fig. 4, it is shown that the SRRT generates an upper-bounded continuous curvature path.

2.3 Reachability Tree of the SRRT

The maximum curvature constraint restricts the arbitrary evolution of the tree. In the previous section, only one path is considered. In this section, one hundred candidate paths with different tree

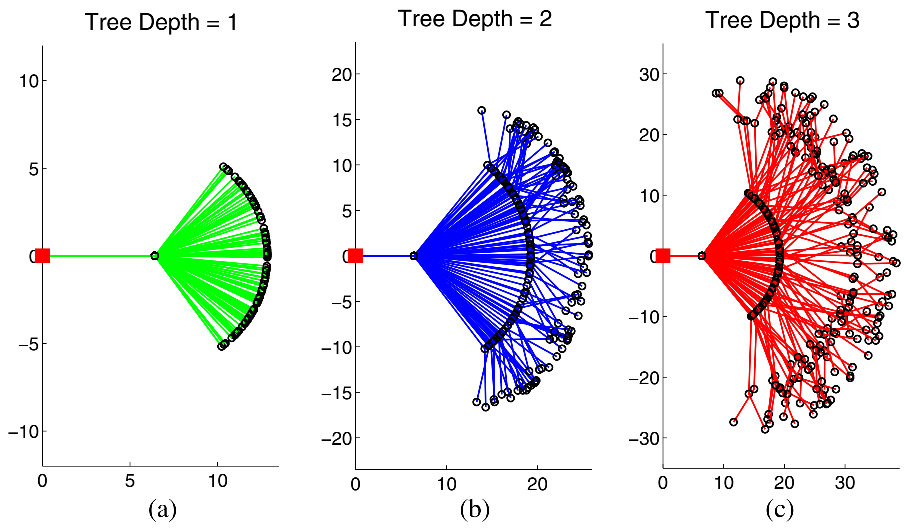

Fig. 5 Reachability tree of the SRRT according to the tree depths: **a** *Green paths* represent the reachability tree when the tree depth is one. **b** *Blue paths* represent the reachability tree when the tree depth is two. **c** *Red paths* represent the reachability tree when the tree depth is three

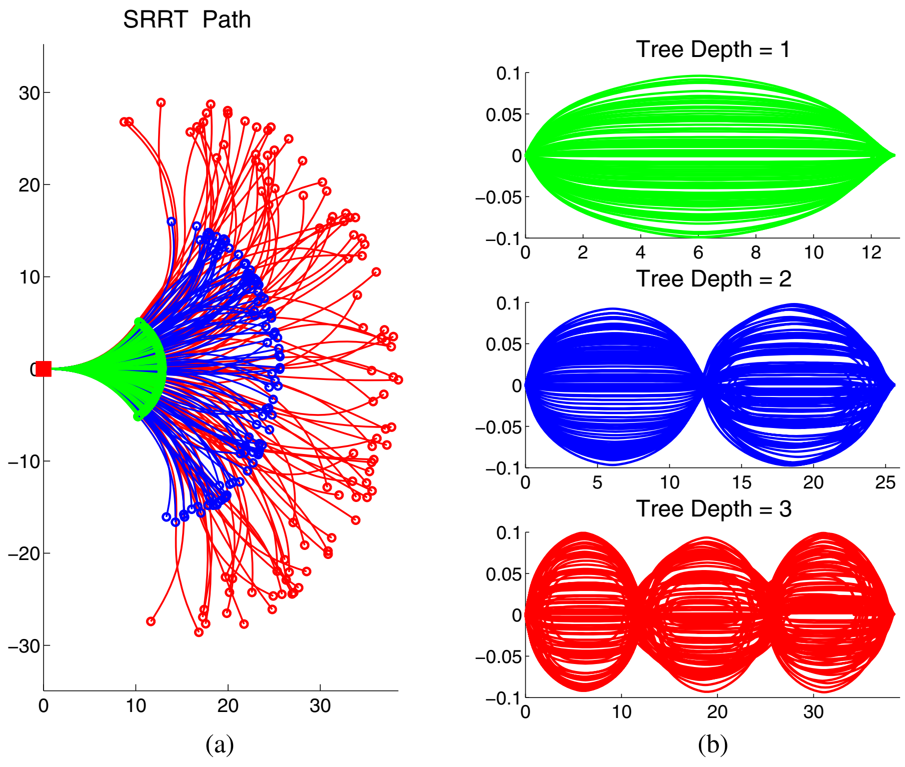

Fig. 6 Comparison of the SRRT smooth paths and their curvatures: **a** Smooth paths. **b** Curvature of the paths. *Green paths* are for tree depth of one, *blue paths* are tree depth two, and *red paths* are tree depth three

depths are generated to examine the reachable region of the SRRT. Here the maximum curvature is set as $\kappa_{\max} = 0.1$ and $\gamma = 0.3\pi$, which results in $d = 6.42$ from Eq. 4.

Figure 5 shows the reachability tree of the SRRT with increase in the tree depth. As explained in the previous Section, the tree length is d_e when the tree depth is one. If the tree depth is more than two, the first and the last tree lengths are d_e and all other trees except the first and the last tree are d_o. Therefore the length of the linear path is two times of nd_e when the tree depth is n.

Figure 6 illustrates smooth paths and curvatures of three different tree depths. Note that the allowable region for a tree of depth n is always a subset of the region for a tree with depth larger than $n+1$. Regardless of the tree depths, the SRRT satisfies both the curvature continuity and the maximum curvature constraints. The green color represents the allowable region of the tree when the tree depth is one. If the tree depth is two, the reachable region is extended to the blue color region and it will be further extended to the red color region if the tree depth is three.

3 Path Planning Using Spline-Based RRT

In this section, the Spline-based RRT algorithm which incorporates the feasibility condition in Section 2, is described.

3.1 Algorithm

There are three main functions in the SRRT algorithm as shown in Algorithm 1: drawing a sample (**StateSampling**); determining a node to extend (**NodeSelection**); and extending the selected node (**NodeExpansion**).

Algorithm 1 Spline-based RRT Algorithm

```
1: BUILD_SRRT($x_{\text{init}}$)
2:     $\mathcal{T}$.init($x_{\text{init}}$)
3: while Distance($x_{\text{goal}}, x_{\text{new}}$) $> d_{\text{lim}}$ do
4:     $x_{\text{rand}} \leftarrow$ StateSampling()
5:     $x_{\text{near}} \leftarrow$ NodeSelection($\mathcal{T}, x_{\text{rand}}$)
6:     $x_{\text{new}} \leftarrow$ NodeExpansion($\mathcal{T}, x_{\text{rand}}, x_{\text{near}}$)
7: end while
```

While basic roles of each function remains the same as conventional RRT, these functions are adjusted to improve the performance of SRRT. The salient difference between the holonomic RRT and SRRT is the ability to judge the feasibility of the random point before adding it to the tree. This ability is achieved in line 5 in Algorithm 1, which investigates the supplementary angle of the successive three points. In the following subsections, each function will be described in detail.

3.2 State Sampling

Shaping the sampling set for developing trees is one of the crucial factors governing the efficiency of the RRT algorithm. The focus of state sampling is to form the probability distribution of the sampling set in promising directions and areas. By applying biased goal point sampling, the tree growth can be focused in the direction to the goal point. The most difficult task in SRRT is to meet the nonholonomic constraints at the goal point. There exists only a small set of solutions at the goal point because nonholonomic constraints restrict the admissible area for developing trees. The SRRT tree can quickly get close to the goal point but it spends a lot of time trying to find a path that satisfies the feasibility condition near the goal point. It is a waste of computing resources to grow trees to cover the entire environment after the frontier tree is already close to the goal point. The ideal sampling set does not grow unnecessary trees which do not contribute to the path generation but grow more promising trees. Motivated by this reasoning, a new sampling algorithm is devised which applies different sampling strategies for growing trees. One is *sampling for exploration*, which grows trees to near the goal point, and the other is *sampling for concentration*, which draws samples to the neighborhood of the goal point. Therefore, the probability distribution of the sampling will be shifted from the broad set (entire environment) to the compact set (near the area of goal point) after the frontier node approaches near the goal point.

3.2.1 Sampling for Exploration

Goal biased sampling is a well-recognized method to enhance the convergence of the RRT algorithm. This method pulls the tree to the goal point. In SRRT, the same goal biased sampling strategy is used to grow trees towards the goal point. This sampling method is classified as *sampling for exploration*.

3.2.2 Sampling for Concentration

The big bottleneck of the SRRT, as mentioned earlier, is to satisfy the feasibility condition at the goal point. It is not effective to draw samples evenly from the entire environment if the SRRT tree has already approached the goal point. This sampling methodology unnecessarily grows all existing trees until the feasibility condition at the goal point is satisfied. It is a better strategy to focus the sampling within the promising area.

After the frontier tree reaches a region within a specified distance from the goal point, the sampling strategy will be shifted from the exploration to concentration. The concentration sampling strategy is to draw samples more near the goal point.

The random points (x_r, y_r) are determined by the following equation:

$$\begin{bmatrix} x_r \\ y_r \end{bmatrix} = \begin{bmatrix} P_g(x) \\ P_g(y) \end{bmatrix} + \begin{bmatrix} (L_d + L_h)\cos(\psi_0 + \psi_d \cdot \xi_\psi)|\xi_h| \\ (L_d + L_h)\sin(\psi_0 + \psi_d \cdot \xi_\psi)|\xi_h| \end{bmatrix} \times \begin{bmatrix} \cos(\pi) & \sin(\pi) \\ -\sin(\pi) & \cos(\pi) \end{bmatrix} \tag{7}$$

$$L_h = |P_c - P_g|$$

$$\psi_0 = \text{atan}\left(\frac{P_g(y) - P_c(y)}{P_g(x) - P_c(x)}\right)$$

where P_c is a position where the concentration is triggered, P_g is a goal position, L_h is a distance between the concentration point and the goal point, ξ_h and ξ_ψ are random variables where $(-1 \le \xi_h, \xi_\psi \le 1)$, ψ_0 is an angle between the concentration point and the goal point. Here, L_h, L_d and ψ_d are design variables which determine the size of the sampling cloud.

Figure 7 shows the state sampling strategy. In this example, L_c, L_d, ψ_d are chosen so that $L_h = 6d$, $L_d = 1.5d$ and $\psi_d = 1\pi$. Before the tree reaching near the goal point, only exploration sampling is applied. Once tree approaching near the goal

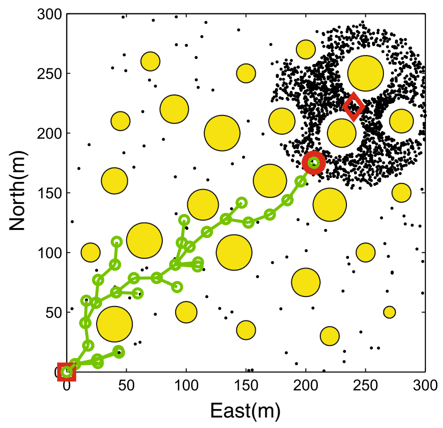

Fig. 7 State sampling strategy: sampling points are drawn from *exploration sampling* until trees grow towards the goal point. Then *concentration sampling* are applied to make tree grow near the goal point

point, the sampling strategy is shift to concentration sampling. In the second phase, 80 % samples are drawn from concentration sampling and 20 % samples are drawn from *exploration sampling*. If samples are drawn only from concentration sampling, it might be trapped in local minima. By using the exploration sampling together, however, SRRT tree can escape from local minima if it exists.

3.3 Node Selection

After drawing a sample, the next task is to determine the node to extend. In holonomic planning, the Euclidean distance is an ideal metric to generate a Voronoi bias because there is no restriction to the expansion of the tree. However, if there exists a differential constraints which limit the evolution of the system states, the Euclidean distance measure fails to capture the true distance.

A new nearest-neighbor selection method is proposed considering differential constraints. First, the Euclidean distance is calculated between the sample point and all existing nodes. Then, they are sorted in ascending order from the closest node to the farthest node. Finally, the feasibility condition is checked from the closest node until a node, which satisfies the feasibility condition, is found. The selected node is the closest node from the sample point which satisfies the nonholonomic constraint. The Euclidean distance metric encourages the RRT preserves the Voronoi bias property, and the efficient feasibility checking algorithm ensures the non-trivial maximum curvature constraints are adhered to without involving a expensive process. This method is simple but quite efficient. The combination of the Euclidean distance and feasibility checking algorithm is used as a *distance metric* in the SRRT algorithm.

3.4 Node Expansion

The last step of the SRRT algorithm is to extend the selected node towards the random point. The conventional RRT uses forward simulation to obtain a state by applying the input chosen from the action space for the fixed time interval. This expensive process is replaced with the spline curve parameterization in SRRT which is described in Fig. 4g–i using the algorithm in [34].

4 Extraneous Node Pruning

There is no straightforward manner by which to trim the control inputs to remove redundant nodes because the control input is used to grow tree and the resulting tree is related to the vehicle dynamics. In the SRRT algorithm however, the redundant nodes can be easily selected and

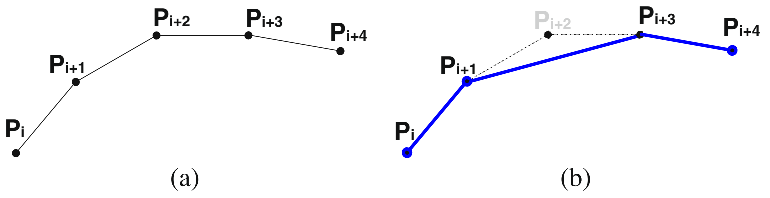

Fig. 8 Extraneous node pruning of the SRRT: **a** There are five nodes in the tree. **b** The goal of the path pruning is to remove redundant node P_{i+2}

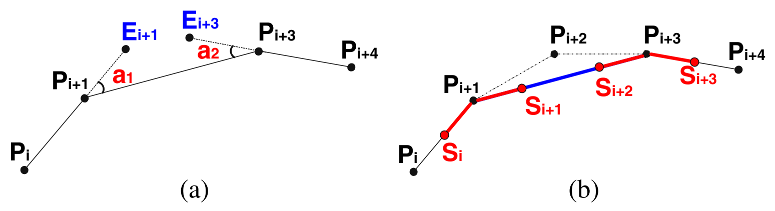

Fig. 9 Conditions for the extraneous node pruning of the SRRT: **a** Feasibility condition of nonholonomic constraints. **b** Collision checking along the spline path

removed because the node expansions do not involve system dynamics but are achieved by spline parameterization.

Figure 8 shows the extraneous node pruning process. There are five nodes in Fig. 8a. The goal of the path pruning is to remove redundant nodes as in Fig. 8b. Here P_{i+2} can be regarded as redundant if $P(i+1)$ and $P(i+3)$ are connected without collision. This method can get rid of unnecessary waypoints very quickly but it cannot be used in SRRT because the SRRT path must satisfy not only the collision free requirement but also the nonholonomic constraints.

A node $P(i+2)$ in the SRRT path is redundant if it satisfies the conditions in Eq. 8.

$$\begin{aligned}
\angle E(i+1)\,P(i+1)\,P(i+3) &\le \gamma \\
\angle E(i+3)\,P(i+3)\,P(i+1) &\le \gamma \\
BZ\{S(i),\ S(i+1)\} \cap Obs &= \varnothing \\
L\{S(i+1),\ S(i+2)\} \cap Obs &= \varnothing \\
BZ\{S(i+2),\ S(i+3)\} \cap Obs &= \varnothing
\end{aligned} \tag{8}$$

The first two conditions are for the feasibility condition of upper-bounded curvature constraints and the others are to ensure the collision-free path.

Figure 9a illustrates the feasibility condition checking process. Firstly, the supplementary angle of the first three nodes, which is $\angle P(i)\,P(i+1)\,P(i+3)$, is examined. Then, the supplementary angle of the last three nodes ($\angle P(i+1)\,P(i+3)\,P(i+4)$) is investigated. If both supplementary angles satisfy the feasibility condition, the collision checking algorithm is applied as shown in Fig. 9b. There are two spline sections(solid line in red) and one linear section(solid line in blue). The collision detection is tested along the spline between S_i and S_{i+1}, linear line between S_{i+1} and S_{i+2} and finally spline between S_{i+2} and S_{i+3}. If these five conditions are satisfied, then $P(i+2)$ can be removed.

5 Simulations

We now conduct the simulation experiments to investigate the performance of the spline-based RRT. First, we expand 500 nodes of the SRRT in both an obstacle-free environment and a cluttered environment to examine whether the SRRT has the Voronoi bias property or not. Secondly, the performance between the normal sampling method and the two-phase sampling method are compared. Finally, the path pruning algorithm in Section 4 is investigated.

5.1 Voronoi Bias Property of the SRRT

To investigate the performance of the combined distance metric 500 nodes are expanded in both an obstacle-free environment and cluttered environments. The environment size is set to 300 m × 300 m and the starting point is assigned to $x_{\text{init}} = (150, 150)$. The maximum curvature and angle is set as $\kappa_{\max} = 0.05$ and $\gamma = 0.23\pi$ which results in $d = 9.07$ in Eq. 4.

Figure 10 illustrates the SRRT exploration results in an obstacle-free environment. Initially, the SRRT tree rapidly grows in the diagonal direction from the center, and then it incrementally splits Voronoi regions of the visited region into smaller ones. After 500 iterations, the SRRT tree covers most of the area but none of the trees regress to already visited areas.

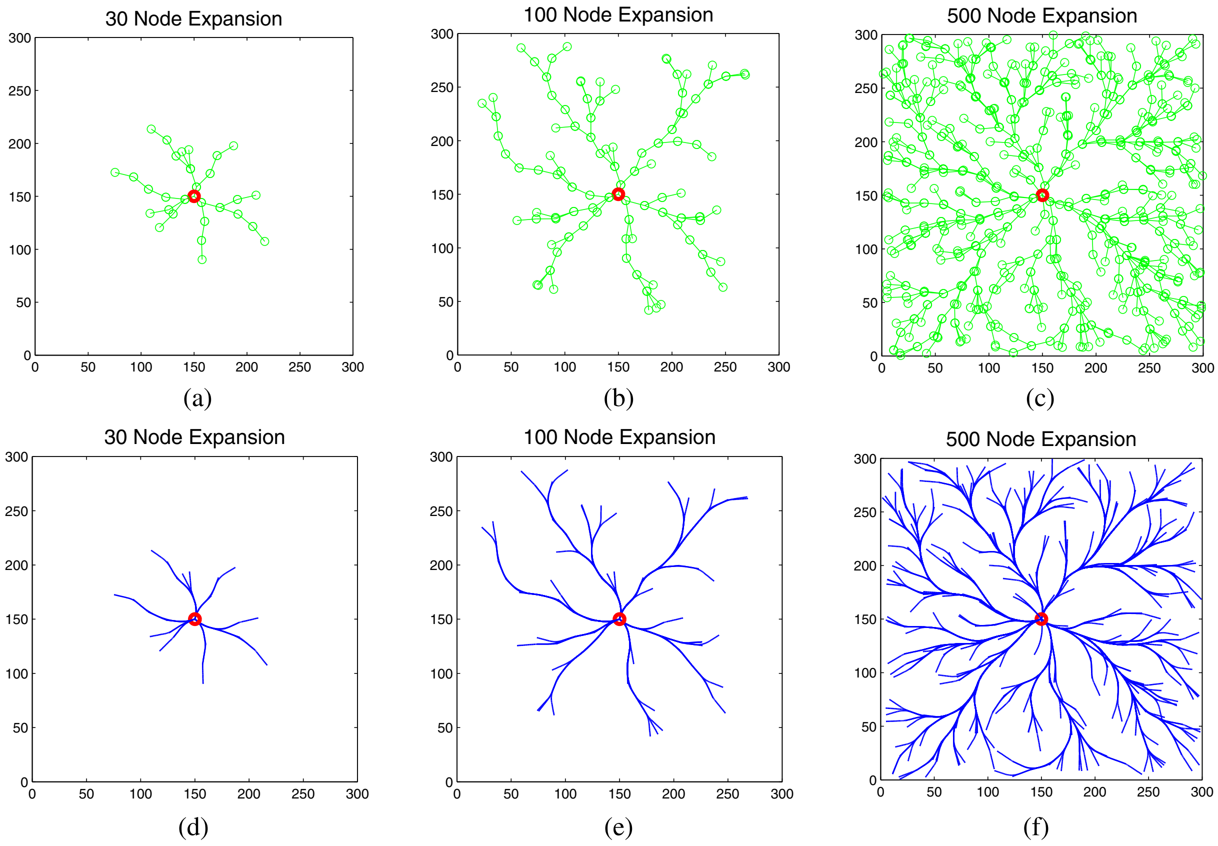

Fig. 10 SRRT expansion property in a free environment: **a** Tree expansion results when tree nodes is 30. **b** Tree expansion results when tree nodes is 100. **c** Tree expansion results when tree nodes is 500. **d** SRRT paths when tree nodes is 30. **e** SRRT paths when tree nodes is 100. **f** SRRT paths when tree nodes is 500. The SRRT tree rapidly grows to the diagonal direction from the center and then it incrementally splits Voronoi regions of the visited region into smaller Voronoi ones

Figure 11 shows the SRRT exploration results in an obstacle-ridden environment. In a similar fashion to the free environment case, the trees grow quickly in the diagonal direction from the center although the tree growth is frequently blocked by obstacles. After 500 iterations, the SRRT tree covers most of the unoccupied area of the environment.

From this result, it can be seen that the combined distance metric preserves the Voronoi bias property of RRT and satisfies the nonholonomic constraints at the same time. Note that all nodes in Figs. 10a–c and 11a–c are only used for the feasibility check of nonholonomic constraints. The SRRT generates a upper-bounded continuous curvature path such as in Figs. 10d–f and 11d–f.

5.2 Performance Comparison Between the Pure Exploration Sampling and the Two-Phase Sampling

The performance difference between the pure exploration sampling (PES) and the two-phase sampling (TPS) strategy is compared. Figure 12 shows the planning result of PES method and TPS method. The starting point is (0,0) and the goal point is (240,222), and the goal point is located in the narrow passage between obstacles. This condition severely restricts the allowable sampling points to satisfy the feasibility condition of the goal point. Figure 12a and b are the planning results of using the PES method. The safety boundary which is the cyan colored circle is overlaid

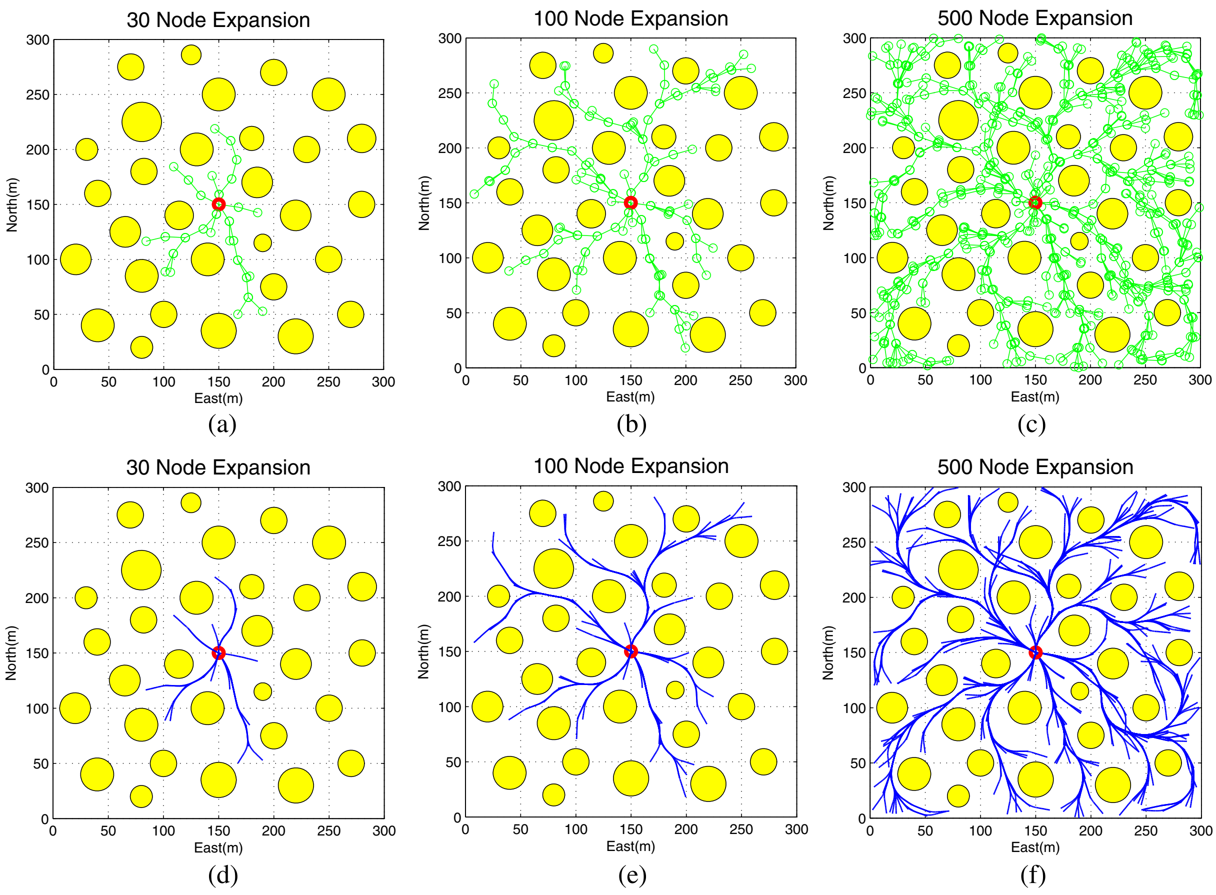

Fig. 11 SRRT expansion property in a cluttered environment: **a** Tree expansion results when tree nodes is 30. **b** Tree expansion results when tree nodes is 100. **c** Tree expansion results when tree nodes is 500. **d** SRRT paths when tree nodes is 30. **e** SRRT paths when tree nodes is 100. **f** SRRT paths when tree nodes is 500. In a similar fashion to the free environment case, the trees grow quickly in the diagonal direction from the center although the tree growth is frequently blocked by obstacles

on the obstacles to show the goal point feasibility condition clearly. Figure 12a shows the SRRT tree when it reaches the concentration point. Because the PES method does not use the two-phase sampling strategy, the SRRT grows trees uniformly in the environment as in Fig. 12b. The green line represents the trees grown before reaching the concentration point and the blue line represents the trees grown after reaching the concentration point. Although trees have come close to the goal point, the planner grows trees evenly in the existing trees. The dotted red line is the final path generated by the SRRT.

Figure 12c and d are the planning results of the TPS method. After reaching the concentration point, the sampling strategy is shifted from the exploration to the concentration. With the focused sampling near the goal point, the TPS method can generate a path with few iterations as shown in Fig. 12d.

Table 2 shows the average over 1000 runs of the SRRT applying the pure exploration sampling (PES) and the two-phase sampling (TPS) methods respectively. Q_{exp} is an exploration sampling query which is used before reaching a concentration point; Q_{con} is a concentration sampling query which is used after reaching a concentration point; Q_{tot} is the sum of the exploration sampling query and the concentration sampling query; TREE is the total number of trees added

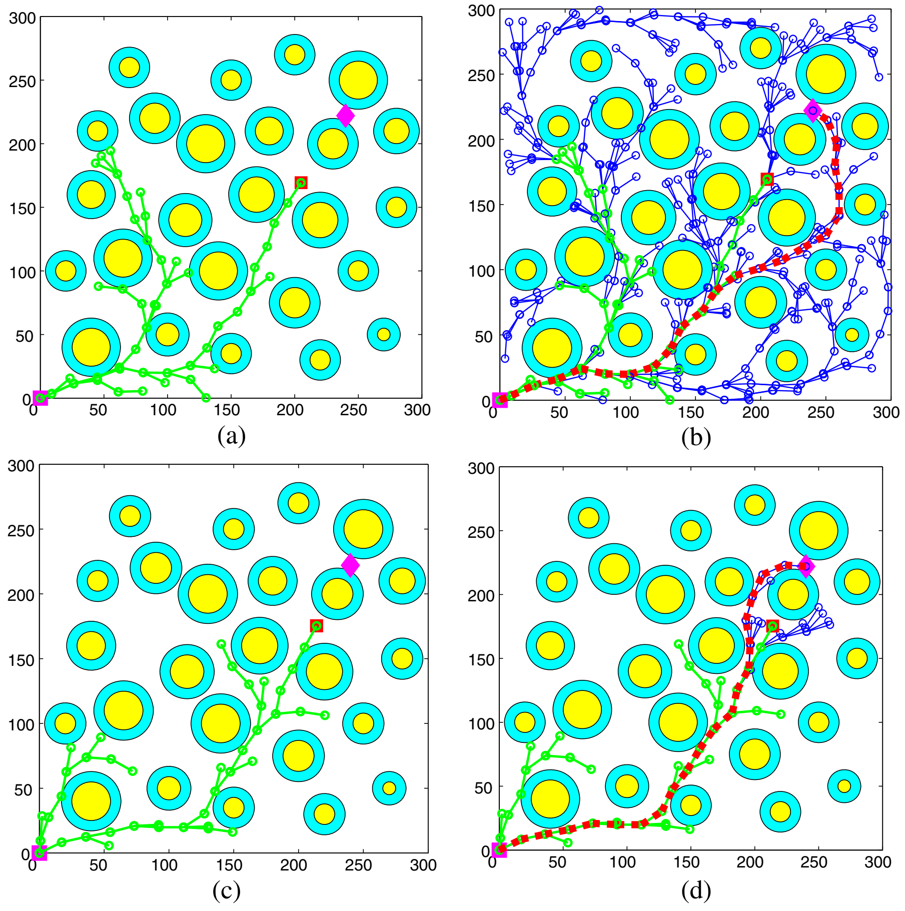

Fig. 12 Performance between the pure exploration sampling (PES) and the two-phase sampling (TPS) I: **a** PES case before reaching a concentration point. **b** PES case after reaching a concentration point. **c** TPS case before reaching a concentration point. **d** TPS case after reaching a concentration point

to the SRRT until a path is generated; TIME is the total planning time. The exploration queries between two methods are similar but there is a huge difference in the concentration queries. The number of concentration queries of the TPS method is about thirty but with the PES method, it is more than 1300. This difference results in an increase in the number of sampling queries and the planning time. From Table 2, it can be seen that the two-phase sampling method decreases the planning time by more than one order of magnitude compared with the pure exploration method.

Table 2 Performance comparison with different sampling strategies

	Q_{exp}	Q_{con}	Q_{tot}	TREE	TIME (s)
PES	112.1	1359.1	1471.2	283.5	10.06
TPS	110.9	28.5	139.4	58.7	0.64

5.3 Performance of the Path Pruning Algorithm

Figure 13 shows the path pruning result applied to the SRRT path. The maximum curvature and angle is set as $\kappa_{\max} = 0.05$ and $\gamma = 0.24\pi$. Figure 13a shows the SRRT path and its pruned path. Figure 13b shows the final smooth path of the original SRRT path and the pruned path from that path. The safety boundaries are plotted in Fig. 13c and d as enclosed by cyan colored circles to see whether the path pruning algorithm satisfies the safety condition. Figure 13c is a smooth path of the original SRRT path. There are a lot of wavy motions in the original path. These motions, however, are removed by the path pruning

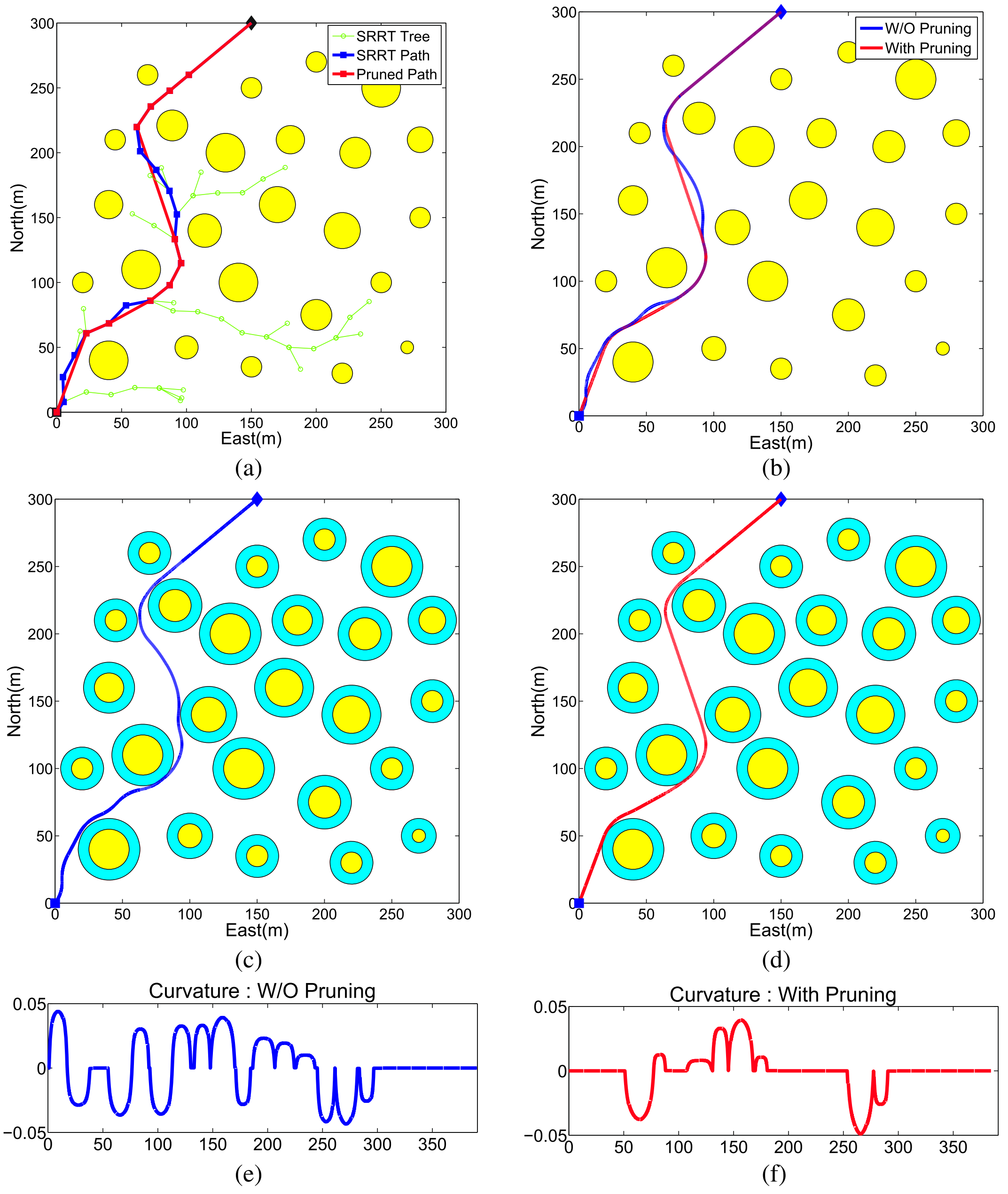

Fig. 13 SRRT path pruning I: **a** *Green line* denotes the SRRT trees and the *blue line* represents a path generated by the SRRT. The *red color* represents a path applying the path pruning algorithm to the original path. **b** The *blue curve* represents a smooth path without applying the pruning algorithm and the *red curve* is a smooth path after applying the pruning algorithm. **c** Smooth path without applying the pruning algorithm overlaid with a safety boundary. **d** Smooth path applying the pruning algorithm overlaid with a safety boundary. **e** Curvature of the path without applying the pruning algorithm. **f** Curvature of the path applying the pruning algorithm

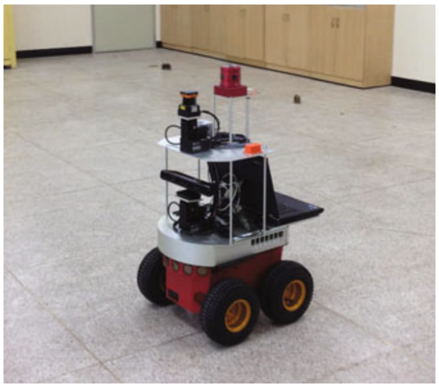

Fig. 14 The Pioneer 3-AT used for experiment. The mobile robot is a skid steer four wheel drive robot and Hokuyo UTM-30LX scanning laser is used to sense the obstacles

algorithm, as can be seen in Fig. 13d. Figure 13e and f show curvatures of the original and pruned paths respectively. Both methods satisfy the maximum curvature constraints. There are a lot of motion changes in the original path. The pruning process removes the swaying motions which result in a flattened path and reduces the path length.

6 Experiments

The performance of the spline-based RRT path planner was demonstrated in real experiment on a ground mobile vehicle. The platform is a Pioneer 3-AT (Fig. 14) which is a skid steer four wheel drive robot. The experiment was conducted in an cluttered indoor environment. The task was to navigate from a starting point to a goal point while avoiding the obstacles without any prior knowledge about obstacles. The Hokuyo UTM-30LX scanning laser range finder was used to sense obstacles.

Figure 15 presents the experimental result of navigation mission. The starting point is (0,0) and the goal point is (0,3.5). The green-colored lines denote the SRRT trees and the orange-colored lines denote the final path generated by the SRRT. The cyan-colored lines in Fig. 15d

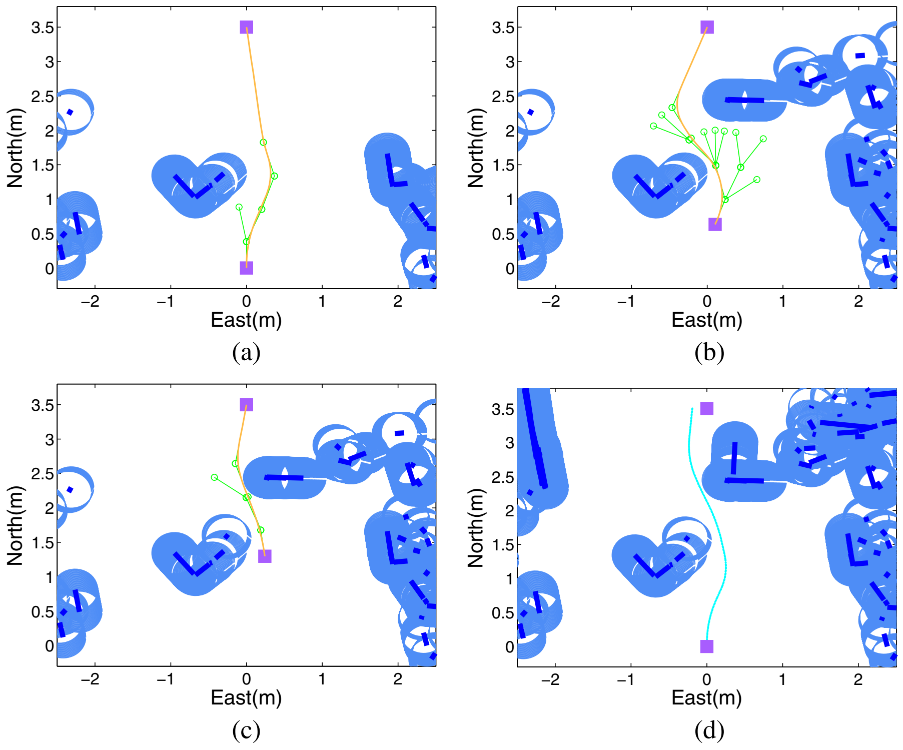

Fig. 15 Navigation experiment I: The mobile robot senses the obstacles near the beginning point at the start of the mission. As the robot moves toward the goal point, it gradually perceives the obstacles and replans the path to avoid obstacles

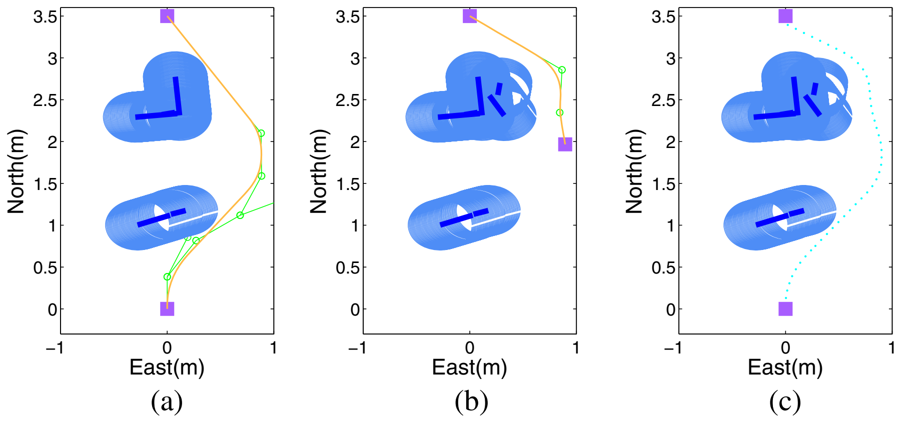

Fig. 16 Navigation experiment II: the starting point and the goal point are the same as the navigation I case. The SRRT planner completes the mission by replan the path once

represent real odometry data of the mobile robot. The dark blue line denotes the obstacles sensed by the laser scanner and the set of light blue circles denote the safety zone considering the size of the robot. Initially, the robot senses the obstacles near the starting point. As the robot moves toward the goal point, it gradually perceives the obstacles. When the prior generated path causes a collision with the obstacles, the SRRT planner replans this path to avoid obstacles as shown in Fig. 15b and c. In this way, the robot can safely achieve the mission. The tracking performance of the path following controller is not complete yet and need to be improved as shown in Fig. 15d.

Figure 16 depicts the second navigation experiment. The starting point and the goal point are the same as the navigation I case. Path replanning occurs only once after generating the initial path since there are just two obstacles between starting point and goal point. The SRRT planner generates a safe path and the mobile robot tracks the path more accurately than the first case as shown in Fig. 16c.

Figure 17 shows the third navigation experiment. The starting point and the goal point are the same as the previous two cases. The number and the location of obstacles, however, are changed. Similar to the experiment I and II, the SRRT planner generates a collision free path in the cluttered environment.

7 Conclusions

This paper has presented a spline-based Rapidly-exploring Random Tree path planning algorithm

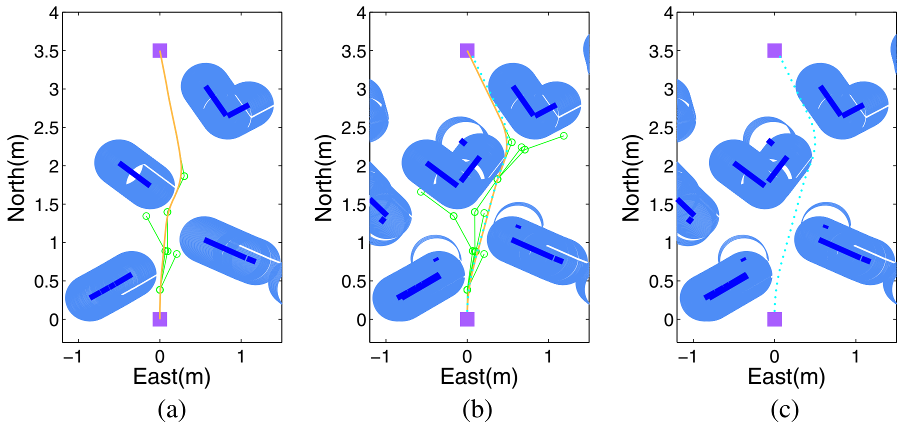

Fig. 17 Navigation experiment III: the SRRT planner generates a collision free path in the cluttered environment

for a differentially constrained mobile robot. Conventional RRT satisfies the differential constraints of the system by choosing the allowable input and applying forward simulation using this input. However, the performance of this method is sensitive to the size of the input set and the time interval for the state evolution. In addition, there is a discontinuity of input because it is selected randomly among the input set. Finally, there is no proper distance metric for the decision of the nearest-neighbor node selection under differential constraints. The SRRT framework overcomes these limitations by using splines as a local planner which solves the two-point boundary value problem. The local planner connects two points guaranteeing the continuity of curvature and satisfying the upper-bounded curvature constraints.

In addition to the novel spline based local planner algorithm, several improvements are made to solve the planning problem under differential constraints. First, a new distance measure is applied, which combines the feasibility checking algorithm with the Euclidean distance metric, to select a nearest-neighbor node. The Euclidean distance metric encourages the RRT to preserve the Voronoi bias property, and the feasibility checking algorithm ensures the non-trivial nonholonomic constraints are observed without involving expensive numerical integration. Secondly, a two-phase sampling approach is applied to draw samples. The exploration sampling strategy is applied to grow trees quickly towards the goal point and the concentration sampling strategy is used after the frontier tree reaches a region within a specified distance from the goal point. This two-phase sampling approach greatly reduces the planning time. Thirdly, an efficient path pruning algorithm is proposed. The redundant nodes are easily selected and removed by applying the feasibility checking algorithm. Contrary to the typical path pruning algorithm, the proposed method keep the maximum curvature constraint satisfying upper-bounded curvature limits.

The proposed spline-based RRT has several advantages. First, the parameterized spline path guarantees the continuity of curvature along the path which removes the sharp change of the controls in conventional RRTs. Secondly, control space discretization is unnecessary. Thirdly, the efficient distance metric preserves the Voronoi bias property. Finally, expensive numerical integration is not necessary because this process is replaced with the spline curves.

References

1. LaValle, S.M.: Planning Algorithms. Cambridge University Press, Cambridge (2006)
2. Jacobs, P., Canny J.: Planning smooth paths for mobile robots. In: IEEE Int. Conf. Robotics and Automation, Scottsdale (1989)
3. Fraichard, T.: Smooth trajectory planning for a car in a structured world. In: IEEE Int. Conf. Robotics and Automation, Sacramento (1991)
4. Anderson, E., Beard, R., McLain, T.: Real-time dynamic trajectory smoothing for unmanned air vehicles. IEEE Trans. Control. Syst. Technol. **13**(3), 471–477 (2005)
5. Fraichard, T., Scheuer, A.: From Reeds and Shepp's to continuous-curvature paths. IEEE Trans. Robot. **20**(6), 1025–1035 (2004)
6. Kito, T., Ota, J., Katsuiu, R., Mizuta, T., Arai, T., Ueyama, T., Nishiyama, T.: Smooth path planning by using visibility graph-like method. In: IEEE International Conference on Robotics and Automation, Taipei, Taiwan (2003)
7. Connors, J., Elkaim, G.: Analysis of a spline based, obstacle avoiding path planning algorithm. In: IEEE 65th Vehicular Technology Conference (2007)
8. Howard, T., Green, C., Kelly, A., Ferguson, D.: State space sampling of feasible motions for high performance mobile robot navigation in complex environments. J. Field Robot. **25**(6–7), 325–345 (2008)
9. Thrun, S., Ragusa, C., Ray, D., et al.: Stanley: the robot that won the DARPA grand challenge. J. Field Robot. **23**(9), 661–692 (2006)
10. Urmson, C., et al.: Autonomous driving in urban environments: boss and the urban challenge. J. Field Robot. **25**(8), 425–466 (2008)
11. Hundelshausen, F., et al.: Driving with tentacles: integral structures for sensing and motion. J. Field Robot. **25**(9), 640–673 (2008)
12. Scherer, S., Singh, S., Chamberlain, L., Elgersma, M.: Flying fast and low among obstacles: methodology and experiments. Int. J. Robot. Res. **27**(5), 549–574 (2008)
13. Likhachev, M., Ferguson, D.: Planning long dynamically feasible maneuvers for autonomous vehicles. Int. J. Robot. Res. **28**(8), 933–945 (2009)
14. Barraquand, J., Latombe, J.C.: Nonholonomic multibody mobile robots: controllability and motion planning in the presence of obstacles. Algorithmica **10**(2–4), 121–155 (1993)
15. Howard, T., Kelly, A.: Optimal rough terrain trajectory generation for wheeled mobile robots. Int. J. Robot. Res. **26**(2), 141–166 (2007)

16. LaValle, S.M., Kuffner, J.: Randomized kinodynamic planning. Int. J. Robot. Res. **20**(50), 378–400 (2001)
17. LaValle, S.M.: Rapidly-Exploring Random Trees: A New Tool for Path Planning. TR 98-11, Computer Science Dept., Iowa State University (1998)
18. Kuffner, J., LaValle, S.: RRT-connect: an efficient approach to single-query path planning. In: IEEE International Conference on Robotics and Automation, San Francisco (2000)
19. Yershova, A., Jaillet, L., Simeon, T., LaValle, S.: Dynamic-domain rrts: efficient exploration by controlling the sampling domain. In: IEEE International Conference on Robotics and Automation, Barcelona, Spain (2005)
20. Burns, B., Brock, O.: Single query motion planning with utilityguided random trees. In: IEEE International Conference on Robotics and Automation, Rome, Italy (2007)
21. Fulgenzi, C., Tay, C., Spalanzani, A., Laugier C.: Probabilistic navigation in dynamic environment using rapidly-exploring random trees and Gaussian processes. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, Nice, France (2008)
22. Yang, K., Gan, S., Sukkarieh, S.: A Gaussian process-based RRT planner for the exploration of an unknown and cluttered environment with a UAV. Adv. Robot. **27**(6), 431–443 (2013)
23. Aoude, G., Luders, B., Joseph, J., Roy, N., How, J.: Probabilistically safe motion planning to avoid dynamic obstacles with uncertain motion patterns. Auton. Robot. **35**(1), 51–76 (2013)
24. Frazzoli, E., Dahleh, M., Feron, E.: Real-time motion planning for agile autonomous vehicles. J. Guid. Control. Dyn. **25**(1), 116–129 (2002)
25. Ferguson, D., Stentz, A.: Anytime RRTs. In: International Conference on Intelligent Robots and Systems. Beijing, China (2006)
26. Yang, K.: Anytime synchronized-biased-greedy rapidly-exploring random tree path planning in two dimensional complex environments. Int. J. Control. Autom. Syst. **9**(4), 750–758 (2011)
27. Karaman, S., Frazzoli, E.: Sampling-based algorithms for optimal motion planning. Int. J. Robot. Res. **30**(7), 846–894 (2011)
28. Perez, A., Platt, R., Konidaris, G., Kaelbling, L., Lozano-Perez, T.: LQR-RRT*: optimal sampling-based motion planning with automatically derived extension heuristics. In: IEEE International Conference on Robotics and Automation, St. Paul (2012)
29. Jaillet, L., Porta, J.: Asymptotically-optimal path planning on manifolds. In: Robotics: Science and Systems, Sydney, Australia (2012)
30. Choudhury, S., Scherer, S., Singh, S.: RRT*-AR: sampling-based alternate routes planning with applications to autonomous emergency landing of a helicopter. In: IEEE International Conference on Robotics and Automation, Karlsruhe, Germany (2013)
31. Chakraborty, N., Akella, S., Trinkle, J.: Complementarity- based dynamic simulation for kinodynamic motion planning. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, St. Louis, USA (2009)
32. Kanayama, Y., Hartman, B.: Smooth local-path planning for autonomous vehicles. Int. J. Robot. Res. **16**(3), 263–283 (1997)
33. Yang, K., Sukkarieh, S.: 3D smooth path planning for a UAV in cluttered natural environments. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, Nice, France (2008)
34. Yang, K., Sukkarieh, S.: An analytical continuous-curvature path-smoothing algorithm. IEEE Trans. Robot. **26**(3), 561–568 (2010)