Kinodynamic Planner Dual-Tree RRT (DT-RRT) for Two-Wheeled Mobile Robots Using the Rapidly Exploring Random Tree

Chang-bae Moon and Woojin Chung, *Member, IEEE*

***Abstract*—We propose a new trajectory generation scheme called dual-tree rapidly exploring random tree (DT-RRT), which is designed on the basis of a rapidly exploring random tree (RRT) method. The DT-RRT is suitable for high-speed navigation of a two-wheeled differential mobile robot. The proposed dual tree is composed of a workspace tree and a state tree. The workspace tree finds near sets in the target workspace without considering robot kinematics. Robot trajectories are generated by the extension of the state tree under the consideration of kinematic and dynamic constraints. The proposed scheme allows for different topological structures between the workspace tree and the state tree. Owing to the different structures between two trees, flexible node extensions can be achieved. As a result, the success rate of the node extension can be increased, while the computational cost can be saved. In order to improve the quality of the trajectory, we propose a *reconnect-tree* scheme that can modify the generated tree structure. The advantage of the *reconnect-tree* scheme is that the repropagation of the conventional RRT structure is not required. From simulations, the superior performance of DT-RRT was clarified in terms of computing time and success rate. The experimental result supported high quality of the generated trajectory with fast and smooth motion of the robot.**

***Index Terms*—Kinodynamic planning, mobile robots, nonholonomic motion planning, rapidly exploring random tree (RRT).**

I. Introduction

IN THIS paper, we address the real-time trajectory generation problem of a two-wheeled differential drive mobile robot considering nonholonomic and dynamic constraints. Many studies have examined path planning and motion control schemes [1]–[7]. Planning schemes based on rapidly exploring random trees (RRTs) [8] have received considerable attention owing to their superior planning capability in high-dimensional space. In recent days, the asymptotically optimal RRT-based path planning schemes were proposed in [23], [24], and [31]–[33]. Path smoothing schemes to improve the quality of the planned path using the RRTs are proposed in [34]–[36] using the Bezier curves. Devaurs *et al.* compared three parallel versions of RRT-based planning schemes in [37]. Elebanhawi and Simic presented a comprehensive survey of sampling-based planning schemes in [38]. The aim of this research is to develop a practical real-time trajectory planner for the high-speed navigation of a wheeled mobile robot. The proposed scheme in this paper is a sampling-based planning scheme. The proposed scheme in [34]–[36] extends node using the local path planning scheme using the Bezier curves to generate a smooth path. In this paper, we exploit motion controllers for node extension and direct generation of the robot trajectory.

In practical dynamic environments, incremental planning schemes are preferable over conventional optimal planning schemes [23], [24], [31]–[33] because the target environments contain unknown obstacles and pedestrians. The computation of the optimal path requires longer computational time for planning than the suboptimal one. Furthermore, the optimal path may become infeasible due to dynamic obstacles. Hence, multiple candidate trajectories are required when a preplanned trajectory becomes infeasible during the high-speed movement of a mobile robot. A mobile robot can select a different trajectory from the preplanned candidate trajectories. It is well known that RRTs are useful for incremental planning and multiple-candidate-path generation.

The nodes of the trees can be extended by using local planners that, as the name suggests, plan the local path. Local planning schemes for a car-like vehicle using preplanned motion primitives were proposed in [9] and [10]. A car-like vehicle has a bounded curvature constraint. A large number of preplanned local paths must be generated offline to avoid resolution completeness problems. The resolution of the preplanned local path must also be adjusted to guarantee resolution completeness, as shown in [10]. A two-wheeled mobile robot is free from the curvature constraint. Many preplanned local trajectories are also required to deal with all the available motions of a two-wheeled mobile robot considering the dynamic constraints.

RRT-based planning schemes were combined with trajectory controllers in [11] and [12]. A controller-based local planning scheme using model predictive control [13]–[15], [46] was proposed in [11]. In [11], an identical control scheme was used for the planner to generate a trajectory by forward

Manuscript received December 5, 2013; revised April 24, 2014 and July 1, 2014; accepted July 9, 2014. Date of publication August 6, 2014; date of current version January 7, 2015. This work was supported by the NRF grant funded by the MSIP (NRF-2014R1A2A1A10049634).

The authors are with the School of Mechanical Engineering, Korea University, Seoul 136-713, Korea (e-mail: changbae.moon@gmail.com; lunar97@korea.ac.kr; smartrobot@korea.ac.kr).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIE.2014.2345351

simulation and for robot tracking control. The scheme in [11] could achieve accurate tracking performances. The closed-loop RRT (CL-RRT) algorithm in [12] used a reference guideline for generating trajectories. The CL-RRT guarantees safety by maintaining the state of the terminal nodes in the stop condition. The distance metrics used in [11] and [12] are still required for an additional search scheme to find the nearest neighbor.

As shown in [16], the design or selection of a distance metric to find the nearest node significantly affects the performance of RRT-based algorithms. In [17], Cheng and Lavalle presented an improved RRT scheme that exploits the exploration information for searching the unexplored state space while using poor metrics. The results presented in [17] show that the use of additional information improves planning performance. The distance metrics include different physical values such as distance and angles. Hence, parameter tunings are required for adjusting the sensitivity of the combined physical values.

The proposed planners in [18]–[20] have to sample in space of reduced dimensions due to the presence of the constraints. A high-dimensional motion planning scheme for an underactuated manipulator was proposed in [18]. In [18], sampling was carried out directly in low-dimensional task space. Berenson and Srinivasa [19] presented a proof of the probabilistic completeness of RRT-based planning schemes when planning with constraints on end-effector poses. Porta *et al.* [20] proposed a randomized-path planning algorithm on manifolds for high-dimensional problems. The kinodynamic motion planning by interior–exterior cell exploration algorithm was presented in [21] to deal with motion planning problems using grid-based discretization of the state space.

In this paper, we present the *dual-tree RRT* (*DT-RRT*) trajectory planning scheme. DT-RRT consists of a workspace tree and a state tree. The workspace tree contains the position of a node, node type, and connections of nodes. The state tree contains the trajectories with control inputs. DT-RRT exploits space reduction sampling, similar to the methods presented in [18]–[21]. However, in DT-RRT, the workspace tree topology is different from the state tree topology. The workspace tree node is used for finding near sets in the target workspace, whereas the state tree is used for the trajectory generation considering kinematic and dynamic constraints using the workspace tree nodes. The conventional single-tree-based schemes extend a new node using a single nearest node. Then, a new node can be rejected when trajectory generation fails due to initial conditions. By using the proposed dual trees, the node extension can be carried out flexibly because the nearest node may not be a parent node of the newly extended node in the state tree. As a result, the success rate of the node extension can be increased. The workspace tree is advantageous because it can be used with advanced nearest-point search schemes such as *K-d trees* [22] with an appropriate space dimensionality reduction.

The remaining sections of this paper are organized as follows. In Section II, the state equations and problem statements are presented. The proposed trajectory planning and replanning schemes are presented in Sections III and IV, respectively. The simulation and experimental results are presented in Section V. Finally, the concluding remarks are presented in Section VI.

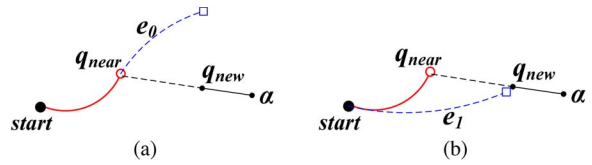


Fig. 1. Illustration of the conventional RRT extension from (a) a nearest node and (b) a different parent node.

II. State Equations and Problem Statements

A. State Equations and Control Inputs

The scope of this paper is limited to trajectory planning for two-wheeled differential mobile robots. The dynamic model is defined by the following equation:

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{v} \\ \dot{\omega} \end{bmatrix} = \begin{bmatrix} v\cos(\theta) \\ v\sin(\theta) \\ \omega \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} u_1 + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} u_2. \quad (1)$$

Equation (1) shows the state transition equation for two-wheeled mobile robots. The state vector $\mathbf{x} = (x, y, \theta, v, \omega)^T$ is in five-dimensional space that consists of the Cartesian coordinates (x, y), orientation θ, translational velocity v, and rotational velocity ω. The control inputs are bounded by dynamic constraints where $u_1 \in [-a_{\max}, a_{\max}]$ and $u_2 = [-\alpha_{\max}, \alpha_{\max}]$ are the bounded translational and rotational acceleration, respectively. Furthermore, the velocities are bounded as $v \in [0, v_{\max}]$ (without backward motion) and $\omega \in [-\omega_{\max}, \omega_{\max}]$. A two-wheeled mobile robot can turn on the spot with zero translational and nonzero rotational velocities.

B. Problem Statements

In this paper, we address the problem of kinodynamic motion planning for a circular two-wheeled mobile robot on the basis of sampling-based algorithms. The state of a mobile robot, while considering the dynamics, is $\mathbf{x} \in SE(2) \times \mathbb{R}^2$, where $\mathbf{x} = (x, y, \theta, v, \omega)^T$. The workspace of the target environment is $\mathbf{q} \in \mathbb{R}^2$, where $\mathbf{q} = (x, y)$. The target environments of this study are partially known environments with a given environmental map.

The goal of this research is to compute the admissible minimum time trajectories $\tau : [0, T] \to C_{\text{free}}$, such that the initial state is $\tau(0) = \mathbf{x}(0)$, and the goal state is $\tau(T) \in B_{\text{goal}}$, where B_{goal} defines the goal region around the target goal point.

III. DT-RRT Using Dual-Tree Structures

A. Finding a Parent State Node

Fig. 1 shows the node extension scheme using the conventional RRT. A mobile robot is located in the start position. Node q_{near} is previously generated using the node extension scheme using the conventional RRT. As shown in Fig. 1(a), position α is randomly sampled to generate trajectories toward the goal position. An intermediate position q_{new} is generated using an appropriate node extension length using random sampled position α. To complete node extension, nodes q_{near} and

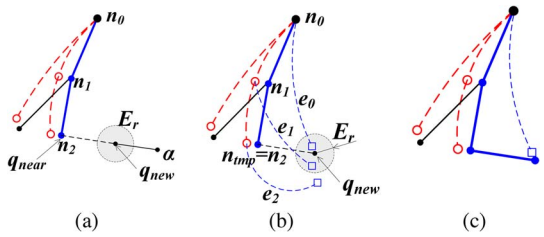


Fig. 2. Illustration of the proposed method DT-RRT to find a minimum-cost parent state node.

q_{new} should be connected by generating trajectories from q_{near} to q_{new}. However, generated edge e_0 cannot connect nodes q_{near} and q_{new} as shown in Fig. 1(a) due to the kinematic and dynamic constraints. Fig. 1(b) shows the successful node extension result using a different parent node. As shown in Fig. 1, the success rate of the node extension is significantly affected by the selection of an appropriate parent node.

The proposed DT-RRT scheme uses dual tree (solid-line and dashed-line trees) as shown in Fig. 2. The solid-line tree is the workspace tree that finds an obstacle-free movable area for a mobile robot. The workspace tree is similar to that of a conventional RRT. The attributes of a workspace tree node include position (x, y) and a node type. The positions of the workspace tree nodes are used to find the parent state. The dashed-line tree is the state tree that generates reference trajectories for a mobile robot considering kinematic and dynamic constraints using the workspace tree.

Fig. 2 shows the proposed method for finding the minimum-cost parent state node. As shown in Fig. 2(a), the nearest node q_{near} is selected using a randomly sampled workspace position α. Thick blue lines represent candidate workspace tree nodes (n_0, n_1, n_2) of a minimum-cost parent node. In this paper, the cost function is computed using the required time to reach the target points. A workspace tree node q_{new} is generated using q_{near}.

In Fig. 2(b), the candidate state nodes are extended toward the workspace node q_{new}. The path cost of e_1 is larger than that of e_0 in Fig. 2(b). The target position of the state tree edge candidate is q_{new}. The square-ended dashed blue lines represent the candidate edges of the state tree. The boundary $\boldsymbol{E}_r$ represents the maximum range between the workspace tree node and the state tree node. The state tree edge candidate e_2 using the workspace tree node $n_{\text{tmp}} = n_2$ toward q_{near} is generated using the local trajectory generators in the Appendix. However, edge e_2 cannot move into the boundary $\boldsymbol{E}_r$ due to the constraints. This result implies the node extension failure of the conventional RRT. The procedure to find an appropriate parent node is carried out using the bottom-up manner.

The state tree edge candidate e_1 is newly generated using the workspace tree node $n_{\text{tmp}} = n_1$. The state tree edge generation is carried out until the workspace node n_{tmp} becomes the workspace tree root node n_0. Finally, the minimum-cost edge, i.e., e_0, in Fig. 2(b) is selected within the range $\boldsymbol{E}_r$ as shown in Fig. 2(b) and (c).

To achieve high-speed navigation, generated trajectories should be guaranteed collision free. Additionally, a mobile robot can avoid obstacles at every state of the trajectories. The collision check is carried out in the configuration space for each individual edge of the state tree. The configuration space is updated in real time for experiments using the laser scanner. In this paper, we compute a sweeping region of a mobile robot while generating the trajectories in the configuration space. The collision check is carried out using the sweeping region for the intermediate trajectories of the edge. The two-wheeled mobile robot does not have a curvature limit. When translational velocity of a mobile robot is zero, the mobile robot is collision free and can avoid obstacles. The last state of the trajectories of the edge is used for checking the collision-free condition under the consideration of the dynamic constraints. The collision-free condition for the last state of the edge is checked by a safe stop condition.

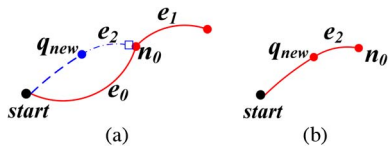


Fig. 3. Illustration of the conventional RRT rewire operation proposed in [23].

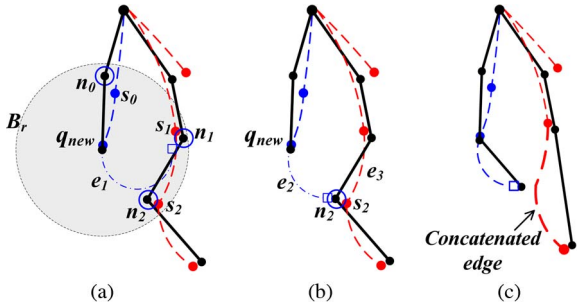


Fig. 4. Illustration of the *reconnect-tree* scheme.

B. Reconnecting the Nodes Using DT-RRT

The original RRT is almost suboptimal, as was shown in [23]. In other words, the probability that RRT constructs an optimal path in a finite number of iterations is zero. In order to guarantee asymptotic optimality, the *rewire* operation [23] reconstructs the edge structures of the RRT when the cost from a new node to the existing nodes is lower than the previous cost. This operation affects all child nodes of the rewired nodes based on the assumption that states from an optimal state also have optimal costs. However, if the sampling is carried out in the workspace $\mathbb{R}^2$, the optimal-cost assumption is not guaranteed. In addition, the trajectories of the rewired nodes may become infeasible. A child node may be repropagated or deleted to deal with the infeasibility of the child nodes, as proposed in [24]. It is preferable, however, to have as many trajectories as possible for real-time operations since the planned trajectories may be blocked by unmapped or moving obstacles.

Fig. 3 shows the rewire operation using the conventional single-tree RRT. Suppose that node q_{new} and edge e_2 in Fig. 3(a) are newly generated and that the cost of edge e_2 is smaller than the cost of e_0. The child edge e_1 from node n_0 becomes infeasible because e_2 and e_1 cannot be smoothly connected due to dynamic constraints. As a result, the child edge e_1 should be deleted as well as the old edge e_0, as shown in Fig. 3(b).

Fig. 4 illustrates the proposed *reconnect-tree* scheme. As shown in Fig. 4(a), the candidate workspace tree nodes are n_0,

n_1, and n_2 within reconnect boundary $\boldsymbol{B}_r$. s_0, s_1, and s_2 are state tree nodes that are located at the close neighborhood of n_0, n_1, and n_2, respectively. The lines terminated by squares represent trajectories generated to compute the cost from new node $\boldsymbol{q}_{\text{new}}$ to candidate nodes n_1 and n_2. Node n_0 is not reconnected because node n_0 is the parent node of $\boldsymbol{q}_{\text{new}}$. The original cost of the state tree node s_1 is smaller than that of the newly generated state tree edge e_1 in Fig. 4(a) from node $\boldsymbol{q}_{\text{new}}$. As a result, node n_1 is not reconnected by the use of e_1. The cost of the state tree edges are $e_1 > e_3 > e_2$.

The workspace tree node n_2 and the state tree node s_2 in Fig. 4(b) are reconnected since the cost of the state tree edge e_2 from node $\boldsymbol{q}_{\text{new}}$ is smaller than the original cost of node s_2. In Fig. 4(c), the workspace tree and state tree nodes are reconnected to node $\boldsymbol{q}_{\text{new}}$ by the use of e_2. The original state tree edge e_3 is concatenated to the child state nodes as shown in Fig. 4(c). The *reconnect-tree* scheme *locally* reconnects using the newly generated node and maintains the child nodes by replacing the parent of the child nodes using the workspace tree, as shown in Fig. 4. The *reconnect-tree* scheme for DT-RRT maintains the child nodes. The reuse of the child nodes contributes to the reduction in computational cost. The repropagation operation of the child nodes is not required.

C. Probabilistic Completeness

Probabilistic completeness means that if a path exists, the probability of finding the path is one, as the number of iterations goes to infinity. The RRT scheme is probabilistically complete, as shown in [8]. However, if the sampling is carried out in subspace $\mathbb{R}^2$ of the state space, the planner may not be probabilistically complete without careful consideration. This can be illustrated by a simple example. Assume that there exists a node moving in a corridor around a narrow area. The sample can be located around this narrow area. However, the node cannot steer to the narrow area due to dynamic constraints. As a result, the planner cannot be used to pass through the narrow area.

The DT-RRT should have the following properties in order to guarantee probabilistic completeness. First, every state generated by the DT-RRT must be in the set of feasible states. In other words, the states must be able to stop safely without collision. The second property is that there must exist two types of nodes, namely, the *stop* and *move* nodes. Finally, a *stop* node must be able to generate an arbitrary path around the *stop* node.

Lemma 1: If there exists a sequence $N = \{N_0, \ldots, N_i\}$ of workspace nodes, then the *stop* node N_s from N_i can be generated.

Proof: All nodes N_i obtained using the proposed planner is in the safe stop states. The trajectories can be generated by stopping around node N_i. Since the *stop* node can generate an arbitrary path around the nodes, the trajectories can be generated around N_i. □

Theorem 1: If an arbitrary *stop* node N_s can be generated, the planning scheme is probabilistically complete.

Proof: According to Lemma 1, N_s can be generated around an arbitrary node. A stop node can generate holonomic motion using appropriate edge generation. The holonomic motion guarantees connectivity in the target space $\mathbb{R}^2$. If there is connectivity in the target space, the completeness can be shown in the same manner as that proposed in [8]. Hence, the DT-RRT is probabilistically complete. □

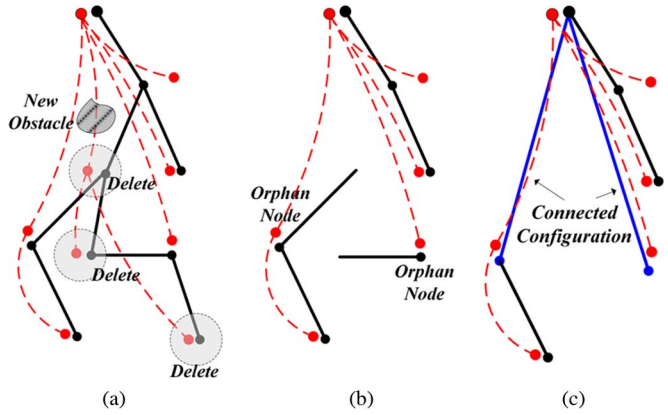


Fig. 5. Illustration of the deletion sequence for collision state nodes.

D. DT-RRT Algorithm Summary

Algorithm I. Build Tree$(q_{\text{init}}, \mathrm{x}_0, goal)$

1 $\boldsymbol{TW}$.init$(q_0, goal)$; // Workspace Tree
2 $\boldsymbol{TX}$.init$(x_0, goal)$; // State(Trajectory) Tree
3 **while**(t_{limit})
4 $q_{\text{rand}} \leftarrow$ RandomSample();
5 $[Q_{\text{new}}\ Q_{\text{near}}\ type] \leftarrow$ ExtendWorkspace$(\boldsymbol{TW}, q_{\text{rand}})$;
6 $[X_{\text{parent}}\ X_{\text{new}}] \leftarrow$ FindParentState $(Q_{\text{near}}, q_{\text{new}}, type)$;
7 **if** $X_{\text{parent}} = nil$ **then continue**;
8 $\boldsymbol{TW}$.Add$(Q_{\text{near}}, Q_{\text{new}})$;
9 $\boldsymbol{TX}$.Add$(X_{\text{parent}}, X_{\text{new}})$;
10 ReConnectTrees$(Q_{\text{new}}, X_{\text{new}})$;
11 **end while**;
12 **return**;

Algorithm I shows the overall computational procedure of the proposed DT-RRT. The notations used in the algorithms are as follows. The lowercase letters are numerical values or vectors, and the capital letters represent defined data types such as nodes that include node points and additional data. The capital letters in bold are data structures such as trees and lists. Finally, the capitalized words are subroutines. The planning time limit t_{limit} in Line 3 is typically set to be a cycle time of a robot controller. The subroutine ExtendWorkspace in Line 5 extends the workspace tree as for the original RRT and determines the node type.

IV. Replanning Algorithm for DT-RRTs

A. Updating Target Trajectory and Deleting Infeasible Nodes

A collision check for the trajectories generated in the previous planning stages is carried out when the target trajectory is selected in the subroutine FindTrajectory$(\boldsymbol{TX})$. Fig. 5 shows the deletion sequence for the collision state nodes. In Fig. 5(a), a new obstacle blocks the planned trajectories, and the

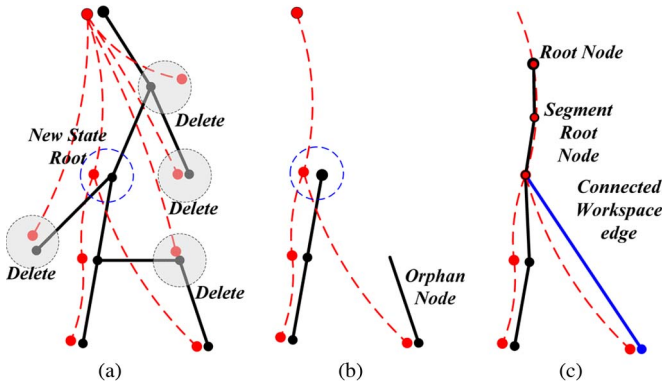


Fig. 6. Illustration of the branching and segmentation sequences using the new tree root.

child state nodes are marked to be deleted. The marked nodes are deleted in Fig. 5(b). Since the workspace and state trees may have different parents, the orphan nodes should be connected, as shown in Fig. 5(b) and (c). The parent of an orphan node is determined from the state tree.

B. Branching Trees

The infeasible nodes that do not start from the target state must be deleted. Fig. 6 shows the sequence for branching and segmentation using a new tree root node. In Fig. 6(a), a new state root node is selected since the target tracking trajectory is updated to the root state edges. The state nodes that do not stem from the new state root node are pruned. The resulting workspace and state trees are shown in Fig. 6(b). In Fig. 6(c), the edges of the orphan nodes are connected, and the root trajectories are segmented along a constant length.

The segmentation is carried out for the following purposes. First, the target trajectories must have sufficiently short execution times and lengths. For example, the turn-on-spot motion can take a significant amount of time; however, segmentation is not required. The second purpose of segmentation is to generate intermediate nodes if the trajectories are extremely long. New states can be produced using the segmented nodes.

C. Replanning Algorithm Summary

The executing trajectories constitute a list that contains the target states. The controller pops the front of the trajectory list for each of the control loops. When the list is empty, the list is updated by executing Algorithm II.

Algorithm II. UpdateTargetTrajectory $(q_{\rm cur}, \mathbf{x}_{\rm cur})$

1 TranslateTree($\boldsymbol{TW}, \boldsymbol{TX}, \mathbf{x}_{\rm cur}$);
2 $X_{\rm target} \leftarrow$ FindTrajectory($\boldsymbol{TX}$);
3 **if** $X_{\rm target} = nil$ **then return** empty-list;
4 BranchingTree($\boldsymbol{TW}, \boldsymbol{TX}$);
5 trajectory-list $\leftarrow$ GetTrajectory($X_{\rm target}$);
6 **return** trajectory-list;

The TRANSLATETREE subroutine shifts the planned trajectories to deal with the positioning error due to the localizer.

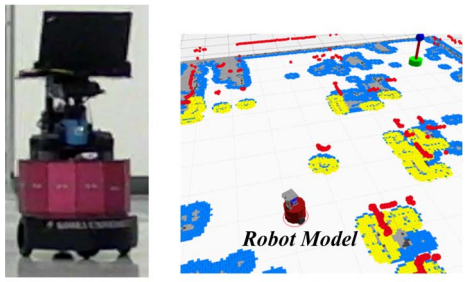


Fig. 7. Setups. (a) Experimental mobile robot. (b) Simulation robot model.

The FINDTRAJECTORY subroutine on Line 2 of Algorithm II finds a minimum-cost node if any nodes exist around the goal region. Otherwise, the FINDTRAJECTORY subroutine returns the node that is nearest to the goal point. The GETTRAJECTORY($X_{\rm target}$) subroutine returns a trajectory list that contains the target states from the root state to the $X_{\rm target}$ node.

V. SIMULATION AND EXPERIMENTAL RESULTS

A. Simulation and Experimental Settings

Fig. 7(a) shows a TetraDS-II robot. The diameters of the TetraDS-II and the simulation model were both 0.5 m. The collision check was carried out by a grid map with a resolution of 2.5 cm. Simulations were carried out using a laptop with an i7 mobile processor. The grid map was employed using the Costmap2d package [25] supported by the robot operating system (ROS) [26]. We exploit the adaptive Monte Carlo localization (AMCL) package in [39] that was presented in [40] using the KLD-sampling scheme. The performance evaluation of the particle filters was shown in [42]. The Kullback–Leibler distance (KLD)-sampling was widely used for navigation and collision avoidance [42]–[45]. The experiments were carried out using the Costmap2d for collision checking and the AMCL for localization of the mobile robot. For simulations, the maximum speeds of the simulated robot were set to 1.0 m/s and 30 deg/s. The maximum acceleration was set to 0.6 m/s^2 and 30 deg/s^2. The maximum distance $\boldsymbol{E}_r$ between the state and workspace nodes, and the reconnecting boundary $\boldsymbol{B}_r$ was set to 0.5 m. The conventional RRT was compared with the proposed DT-RRT using the multiple edge types. The exploited edge types and motion controllers are in the Appendix.

B. Simulation Results

Fig. 8 shows the results for path planning in an open environment without obstacles. The planning results are summarized in Fig. 9 and Table I. In Table I, the "*Plan Time*" rows mean the planning time until finding the path to the given goal region using the path planner. The "*Trajectory*" rows show the travel times required to reach the goal by executing the planned trajectories. The "*over time*" values mean that the trajectories were not produced within 10 s. The mean travel and planning times were computed by excluding the overtime cases to avoid statistical distortion due to large values. The results in Fig. 8(a) show that the conventional RRTs can be practical in obstacle-free space. The *t-test* results show that the differences between

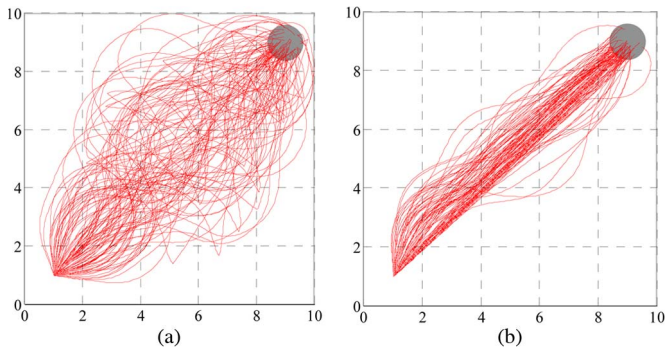


Fig. 8. Planning results for 100 trajectories. (a) RRT. (b) DT-RRT.

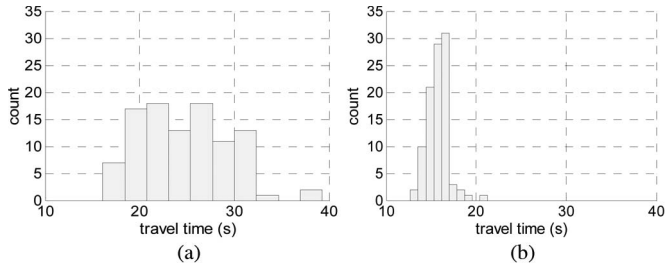


Fig. 9. Histograms of the navigation completion times for the results shown in Fig. 7. (a) RRT. (b) DT-RRT.

TABLE I
SUMMARY OF PLANNING RESULTS SHOWN IN FIG. 8

		mean (s)	std. dev.	min (s)	max (s)	over time
RRT	Plan Time	0.23	0.16	0.10	0.90	0
	Trajectory	24.90	4.79	16.10	39.30	
DT-RRT	Plan Time	0.49	0.26	0.20	1.60	0
	Trajectory	15.79	1.22	12.70	21.30	

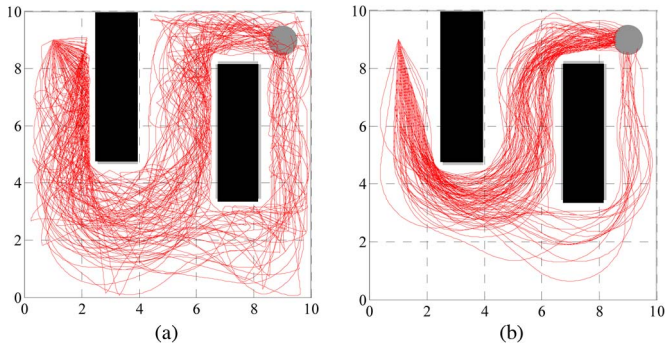


Fig. 10. Planning results for 100 trajectories. (a) RRT. (b) DT-RRT.

the mean travel times of the planned trajectories are significant at the level of $p > 0.05$, and the mean travel time was improved by about 10 s using the DT-RRTs. The mean planning times of the RRTs and DT-RRTs are not significantly different, at the level of $p < 0.05$. These results show that the use of DT-RRTs does not increase the planning times significantly.

Fig. 10 shows the planning results for an environment with obstacles. As shown in Table II, the mean travel time of trajectories generated by the conventional RRT was three times longer than that of the proposed scheme. DT-RRT required additional computation. However, the planning time did not significantly increase, as indicated in Table II. Moreover, the difference between the maximum and minimum planning times was smaller than that for the conventional RRT (see Fig. 11).

TABLE II
SUMMARY OF PLANNING RESULTS SHOWN IN FIG. 10

		mean (s)	std. dev.	min (s)	max (s)	over time
RRT	Plan Time	0.78	1.15	0.10	7.2	7
	Trajectory	67.28	14.96	35.1	103.1	
DT-RRT	Plan Time	1.04	0.38	0.4	1.8	0
	Trajectory	22.25	3.75	16.0	35.3	

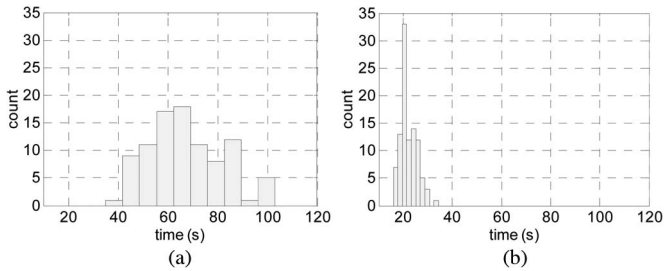


Fig. 11. Histograms of the navigation completion times for the results shown in Fig. 9. (a) RRT. (b) DT-RRT.

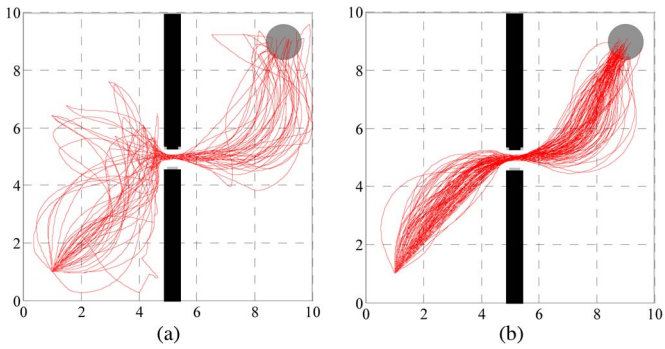


Fig. 12. Planning results for 100 trajectories with a 0.7-m narrow passage. (a) RRT. (b) DT-RRT.

TABLE III
SUMMARY OF PLANNING RESULTS SHOWN IN FIG. 12

		mean (s)	std. dev.	min (s)	max (s)	over time
RRT	Plan Time	1.04	1.26	0.10	6.00	58
	Trajectory	46.15	14.56	21.90	110.9	
DT-RRT	Plan Time	1.90	1.30	0.60	7.70	0
	Trajectory	18.84	2.71	15.00	28.40	

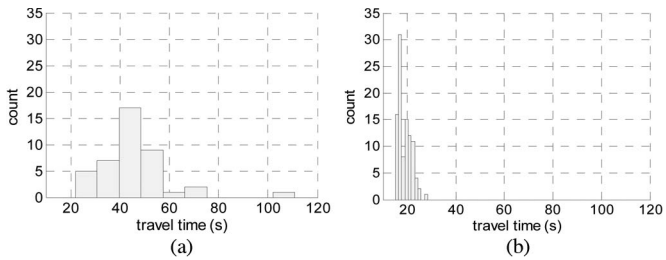


Fig. 13. Histograms of the navigation completion times for the results shown in Fig. 11. (a) RRT. (b) DT-RRT.

Fig. 12 shows the planning results for an environment with a narrow opening that is similar to a doorway. The mean travel time of the DT-RRT was almost half the mean travel time of the conventional RRT. The statistical analysis using the successful cases within the time limits is shown in Table IV. As shown

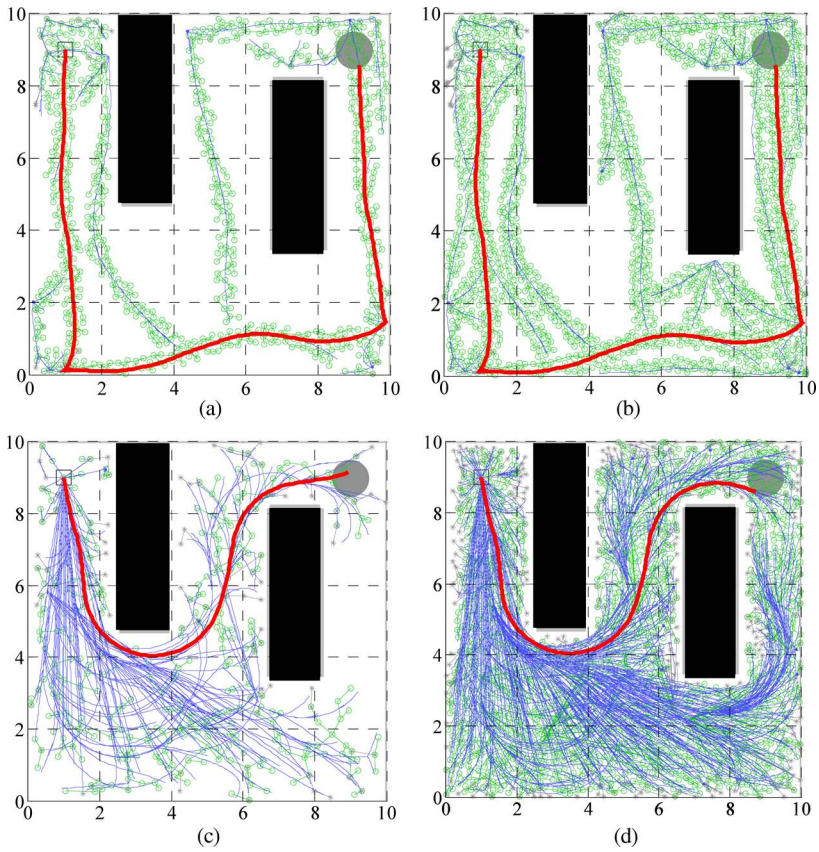


Fig. 14. Comparison results. (a) and (b) RRT at 1 and 30 s, respectively. (c) and (d) DT-RRT at 1 and 30 s, respectively. (The thick red line represents the selected trajectory. Blue and green lines represent workspace and state trees, respectively.)

TABLE IV
DESCRIPTIVE ANALYSIS RESULTS SHOWN IN FIG. 12

		N	95% confidence interval for mean	
			Lower Bound	Upper Bound
RRT	Plan Time	42	0.64	1.43
	Trajectory		41.61	50.69
DT-RRT	Plan Time	100	1.67	2.12
	Trajectory		18.30	19.37

TABLE V
SUMMARY OF PLANNING RESULTS SHOWN IN FIG. 14

		1*s*	5*s*	10*s*	30*s*
RRT	Node Size	761	1091	1352	1583
	Time	44.3*s*	44.3*s*	44.3s	44.3s
DT-RRT	Node Size	356	646	992	1905
	Time	17.5s	17.4s	17.2s	16.5s

in Table IV, the planning time using the conventional RRTs is lower than that of the DT-RRTs. These results imply that, when successful, the conventional RRTs can carry out the planning faster than the DT-RRTs owing to the algorithmic simplicity when carried out in a narrow-opening environment. However, the success rates using the conventional RRTs were significantly decreased, as shown in Table III. These simulated environments represent typical navigational situations, and the planning results show that the DT-RRTs can be applicable in practical environments (see Fig. 13).

Fig. 14(a) and (b) shows the results using RRT, whereas Fig. 14(c) and (d) shows the results using DT-RRT. The node sizes and trajectory results are summarized in Tables IV and V. In Fig. 14, from the viewpoint of the reachability of the target space, the path planning performance of the conventional RRT using multiple edges is significantly lower than that of DT-RRT. As shown in Table V, the node size using DT-RRT is smaller than that of RRT. However, the area covered using DT-RRT is significantly larger than the area covered by the conventional RRT, since the conventional RRT requires an additional steering input search scheme. The planned trajectories using DT-RRT are replaced by lower-cost trajectories as the iteration number. A parent node is found by using the list of parent nodes of the nearest node in the workspace tree. The search cost is affected by the depth of the workspace tree and not by its number of nodes. The proposed scheme calculates the candidate trajectories by additional computation; however, the simulation results in this section show that the computational costs are only slightly increased.

Fig. 15(a) and (b) shows resultant paths under the same number of nodes (1900 nodes). Fig. 15(a) shows the planning result without *reconnect trees*. Fig. 15(a) contains many trajectories whose qualities are not satisfactory. As shown in Fig. 15(a), the previously generated detouring trajectories still exist. The result using the *reconnect trees* is shown in Fig. 15(b). The remarkable differences are shown in the bottom regions. The trajectories are *reconnected*, and the resultant trajectories become almost straight lines as shown in the magnified region. The previously generated detouring trajectories are replaced by new trajectories

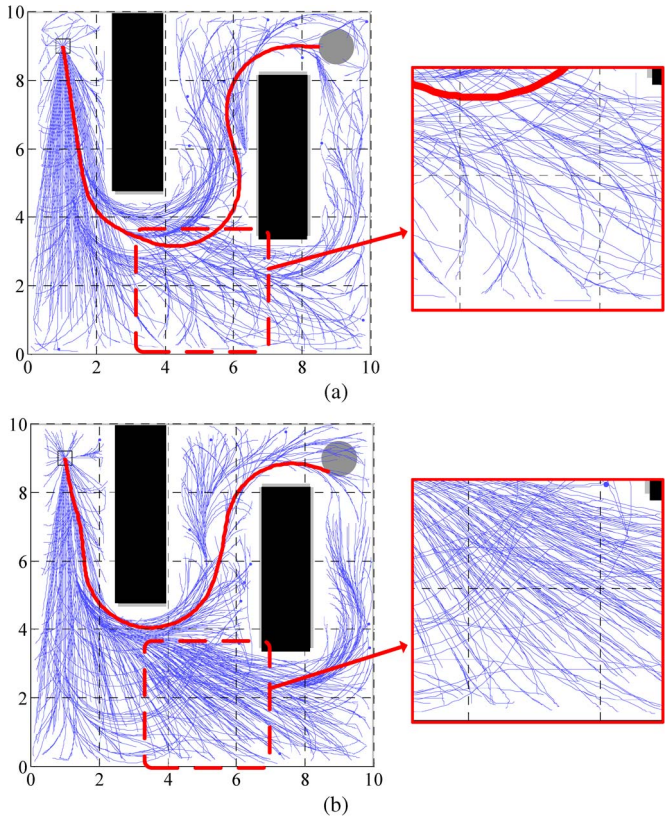


Fig. 15. Comparison results. (a) DT-RRT without *reconnect trees*. (b) DT-RRT with reconnect trees. (The thick red line represents the selected trajectory. Blue lines represent state trees.)

after finding the efficient parent nodes using the proposed *reconnect trees* scheme. The result shows that the proposed *reconnect trees* scheme improves the quality of the planned trajectories.

The experiments were carried out using the mobile robot Tetra DS-II shown in Fig. 7(a). The maximum translational speed was 1.5 m/s due to the physical limits of the mobile robot. The maximum rotational speed was 30 deg/s to generate stable cornering paths. The other parameters were identical to those of the simulations. Fig. 16 shows the tracked trajectories using the conventional RRT and the proposed DT-RRT, respectively. The path planner was planning the trajectories incrementally while the mobile robot moves to the goal region. The travel time results are summarized in Fig. 17 and Table VI. The incremental DT-RRT occasionally generates inefficient detouring trajectories initially. While the mobile robot moves along the trajectories, DT-RRT generates new trajectories and reconnects trees to generate more efficient trajectories. As shown in Fig. 16(b), DT-RRT generates efficient trajectories than that of the conventional RRT. The experimental results show that the travel times were decreased, and the path qualities were significantly improved.

C. Experimental Results

Fig. 18 shows the tracked trajectories using DT-RRT. The mobile robot started in the room at point (42, 5). As shown in Fig. 19(a), the mobile robot then moved out through the door and stopped once in the corridor. In the corridor, around

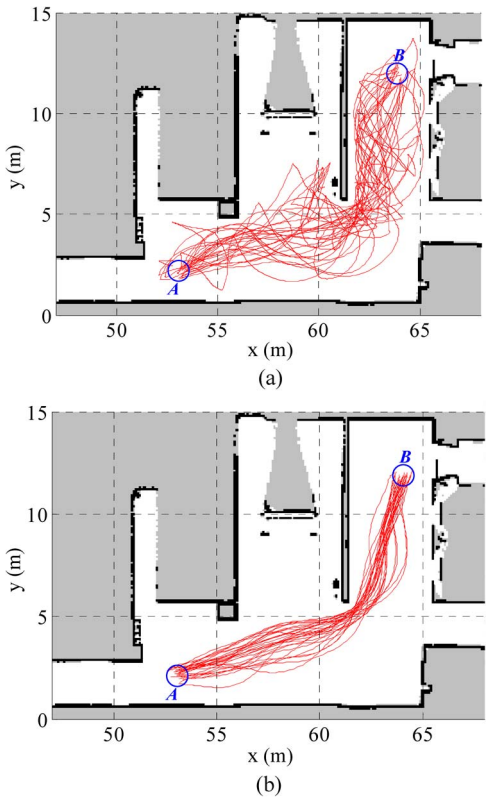


Fig. 16. Experimental results for 30 trajectories. (a) Conventional RRT. (b) DT-RRT.

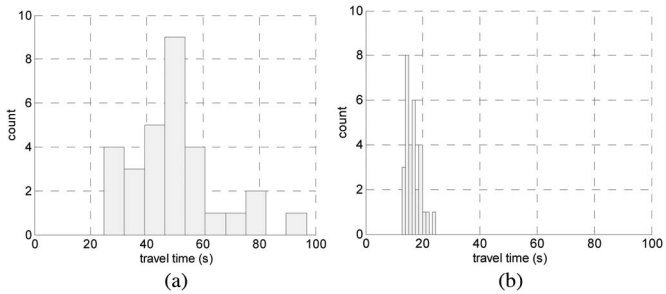


Fig. 17. Histograms of the navigation travel times for the results shown in Fig. 16. (a) Conventional RRT. (b) DT-RRT.

TABLE VI
SUMMARY OF EXPERIMENTAL TRAVEL TIME OF THE TRAJECTORIES SHOWN IN FIG. 16

	mean (s)	std. dev.	min (s)	max (s)
RRT	49.16	16.34	24.80	96.80
DT-RRT	16.57	2.61	12.80	24.50

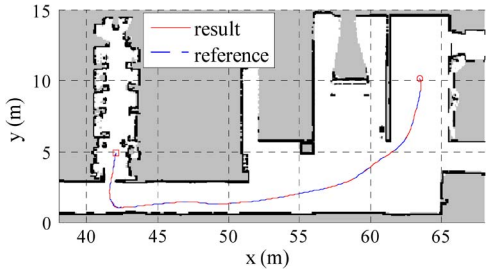


Fig. 18. Experimental results using DT-RRT. (Square) Start position. (Circle) Target goal position.

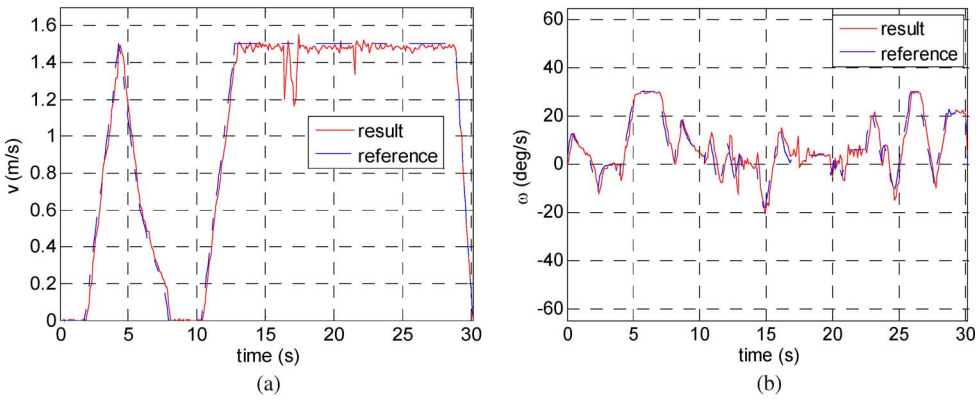


Fig. 19. Velocity profiles. (a) Translational velocity. (b) Rotational velocity.

point (42.2, 1.05), the mobile robot executed a *stop–move* trajectory from $t = 7$ to 10 s. At $t > 13$ s, the mobile robot moved at its maximum speed. The experimental maximum tracking error was 5 cm with a maximum speed of 1.5 m/s. The experimental results show that DT-RRTs can be applicable in practical environments. The video for the experimental result can be found in [20].

We implemented our system using the ROS [26]. The planner, robot hardware, and the simulation process were completely separated in order to support the component-based development of the planner and other navigation components. The simulation results of the proposed scheme showed strong similarity with the experimental results.

VI. Conclusion

In this paper, we have proposed a kinodynamic trajectory planning scheme, i.e., DT-RRT, which uses a dual-tree structure based on RRT. Our proposed algorithm is designed for kinodynamic trajectory planning for a two-wheeled mobile robot. The trajectory planning results showed that the computing costs of the planned trajectories can be reduced using DT-RRT. The *single-query* trajectory planning results showed that the costs of the planned trajectories can be reduced using DT-RRT. The trajectory planning results of *multiple-candidate-path generation* showed that DT-RRT can generate a variety of trajectories although the number of nodes is not sufficient for *rewiring* the nodes. While the iterations were carried out, the cost of the planned trajectories decreased when the *reconnect-tree* algorithm was used. The proposed *reconnect-tree* algorithm is advantageous for handling partially known environments. The experimental results showed that a mobile robot can accurately follow trajectories that were generated using DT-RRT.

Appendix
Edge Types and Motion Controllers for Trajectory Generation

A robot is either in a stop state with zero velocity or in a moving state with nonzero velocity. A stop state is useful because it allows the mobile robot to turn on the spot in narrow or cluttered environments. The edge types of RRT are generated by combining terminal conditions. As a result, four edge types

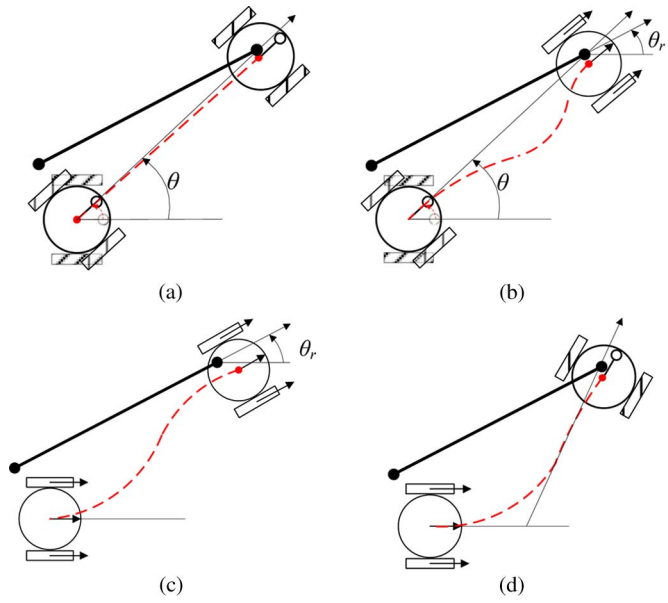


Fig. 20. Illustration of four edge types: (a) *stop–stop*, (b) *stop–move*, (c) *move–move*, and (d) *move–stop*. The thick black lines represent workspace extension edges, and the red dashed lines show the trajectories of the state tree.

are required, as shown in Fig. 20. The robot aligns toward the target workspace point first when the edges in Fig. 20(a) and (b) are selected. When the edges shown in Fig. 20(a) and (d) are selected, a robot stops at respective target points since the target states of the nodes are the *stop* nodes.

We use the asymptotically stable control schemes proposed by Aicardi *et al.* [28] and Kanayama *et al.* [30]. Fig. 21 shows the error postures of the motion controllers. The controller of Aicardi *et al.* computes outputs as per the following equations:

$$\rho = \sqrt{\Delta x^2 + \Delta y^2}$$
$$\gamma = \mathrm{a}\tan 2(\Delta y, \Delta x) - \theta + \pi$$
$$\delta = \gamma + \theta \tag{2}$$
$$\hat{\mathbf{u}} = \begin{bmatrix} \hat{v} \\ \hat{\omega} \end{bmatrix} = \begin{bmatrix} k_1 \rho \cos\gamma \\ k_2\gamma + k_1 \frac{\sin\gamma}{\gamma}(\gamma + k_3\delta) \end{bmatrix}. \tag{3}$$

The notations in (2) and (3) were presented by DeLuca *et al.* in [29] on the basis of the controller of Aicardi *et al.* The

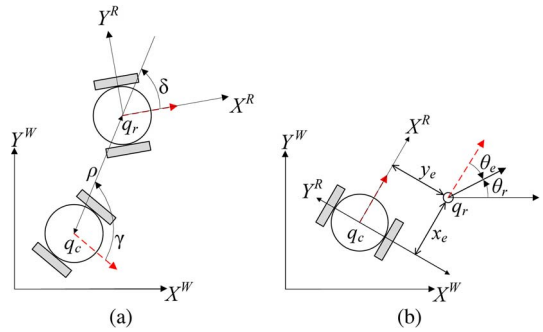


Fig. 21. Error postures of the motion controller. (a) Controller of Aicardi *et al.* for planning the trajectory with the final stop condition [28]. The error posture was presented in the work of DeLuca *et al.* [29]. (b) Controller of Kanayama *et al.* for planning the trajectory without final condition constraints [30].

variables k_1, k_2, and k_3 are positive control gains. The angles γ and δ are not defined at the target point. When $\rho \to 0$, a mobile robot must retain the values for γ and δ assumed in the final approaching phase, since the input vector field associated with v is singular for $\rho = 0$. These singularities can be easily avoided by setting the rotational velocity to zero when $\gamma \simeq 0$.

The error posture, as shown in Fig. 20(b), is computed using the robot's local coordinates in the following equations:

$$\mathbf{q_e} = \begin{bmatrix} x_e \\ y_e \\ \theta_e \end{bmatrix} = \begin{bmatrix} \cos\theta_c & \sin\theta_c & 0 \\ -\sin\theta_c & \cos\theta_c & 0 \\ 0 & 0 & 1 \end{bmatrix} (\mathbf{q_r} - \mathbf{q_n}) \quad (4)$$

$$\hat{\mathbf{u}} = \begin{bmatrix} \hat{v} \\ \hat{\omega} \end{bmatrix} = \begin{bmatrix} v_{\max}\cos\theta_e + k_x x_e \\ \theta_e + v_{\max}(k_y y_e + k_\theta \sin\theta_e) \end{bmatrix}. \quad (5)$$

In (4), the error posture is computed using the target workspace point, and the velocity output is calculated using (5). The reference velocities of (5) were changed to $v_r = v_{\max}$ and $\omega_r = \theta_e$ to generate the trajectory. Control gains k_x, k_y, and k_θ are positive real numbers and affect the convergence properties. For example, a large k_θ puts emphasis on heading errors. The output velocities computed by (3) and (5) are saturated by the dynamic and maximum velocity constraints. In (5), θ_e can be set to zero because it is used in the trigonometric function. In this paper, we use θ_e in order to share the same controller in (5) both for generating and tracking trajectories. The effect of the dual feedback part of θ_e is adjusted by using k_θ, i.e.,

$$\mathbf{u} = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} \min(\hat{v} - v_c, a_{\max} \cdot t_{\text{cycle}}) \\ \min(\hat{\omega} - \omega_c, sign(\hat{\omega} - \omega_c) \cdot \alpha_{\max} \cdot t_{\text{cycle}}) \end{bmatrix}. \quad (6)$$

In (6), the dynamically feasible control input is computed using either (3) or (5) depending on the edge type. The parameters v_c and ω_c are current translational and rotational velocities, respectively, and t_{cycle} is the control cycle time of the controller. The robot motion is simulated to obtain feasible trajectories while the distance is decreased from the robot and target reference position. The generated poses by using the simulation are added to the trajectory list. The stability and convergence of the controller of Aicardi *et al.* and Kanayama *et al.* were proved without an acceleration constraint. If the controllers are used with an acceleration constraint, the generated trajectories may not converge to the target location. However, the controllers can be used for generating the candidate trajectories of the proposed scheme with sufficient initial conditions. If the generated trajectories do not converge to the target location, the target sample point is rejected.

REFERENCES

[1] J. C. Latombe, *Robot Motion Planning*. Norwell, MA, USA: Kluwer, 1991.

[2] H. Choset *et al.*, *Principles of Robot Motion*. Cambridge, MA, USA: MIT Press, 2005.

[3] S. M. Lavalle, *Planning Algorithms*. Cambridge, U.K.: Cambridge Univ. Press, 2006.

[4] K.-S. Hwang and M.-Y. Ju, "A propagating interface model strategy for global trajectory planning among moving obstacles," *IEEE Trans. Ind. Electron.*, vol. 49, no. 6, pp. 1313–1322, Dec. 2002.

[5] C.-C. Tsai, H.-C. Huang, and C.-K. Chan, "Parallel elite genetic algorithm and its application to global path planning for autonomous robot navigation," *IEEE Trans. Ind. Electron.*, vol. 58, no. 10, pp. 4813–4821, Oct. 2011.

[6] H. Rezaee and F. Abdollahi, "A decentralized cooperative control scheme with obstacle avoidance for a team of mobile robots," *IEEE Trans. Ind. Electron.*, vol. 61, no. 1, pp. 347–354, Jan. 2014.

[7] B. Fidan, V. Gazi, S. Zhai, N. Cen, and E. Karatas, "Single-view distance-estimation-based formation control of robotic swarms," *IEEE Trans. Ind. Electron.*, vol. 60, no. 12, pp. 5781–5791, Dec. 2013.

[8] S. M. LaValle and J. J. Kuffner, Jr., "Randomized kinodynamic planning," *Int. J. Robot. Res.*, vol. 20, no. 5, pp. 378–400, May 2001.

[9] A. Lacaze, Y. Moscovitz, N. DeClaris, and K. Murphy, "Path planning for autonomous vehicles driving over rough terrain," in *Proc. IEEE ISIC/CIRA/ISAS Joint Conf.*, Gaithersburg, MD, USA, Sep. 14–17, 1998, pp. 50–55.

[10] M. Pivtoraiko, R. A. Knepper, and A. Kelly, "Differentially constrained mobile robot motion planning in state lattices," *J. Field Robot.*, vol. 26, no. 3, pp. 308–333, Mar. 2009.

[11] A. Brooks, T. Kaupp, and A. Makarenko, "Randomised MPC-based motion-planning for mobile robot obstacle avoidance," in *Proc. IEEE Int. Conf. Robot. Autom.*, Kobe, Japan, May 12–17, 2009, pp. 3962–3967.

[12] Y. Kuwata *et al.*, "Real-time motion planning with applications to autonomous urban driving," *IEEE Trans. Contr. Syst. Technol.*, vol. 17, no. 5, pp. 1105–1118, Sep. 2009.

[13] N. Nikdel, P. Nikdel, M. A. Badamchizadeh, and I. Hassanzadeh, "Using neural network model predictive control for controlling shape memory alloy-based manipulator," *IEEE Trans. Ind. Electron.*, vol. 61, no. 3, pp. 1394–1401, Mar. 2014.

[14] H. Li and Y. Shi, "Network-based predictive control for constrained nonlinear systems with two-channel packet dropouts," *IEEE Trans. Ind. Electron.*, vol. 61, no. 3, pp. 1574–1582, Mar. 2014.

[15] C. A. Rojas, J. I. Yuz, C. A. Silva, and J. Rodriguez, "Comments on 'Predictive Torque Control of Induction Machines Based on State-Space Models'," *IEEE Trans. Ind. Electron.*, vol. 61, no. 3, pp. 1635–1638, Mar. 2014.

[16] S. M. Lavalle, "From dynamic programming to RRTs: Algorithmic design of feasible trajectories," in *Control Problems in Robotics*. Berlin, Germany: Springer-Verlag, 2002.

[17] P. Cheng and S. M. Lavalle, "Reducing metric sensitivity in randomized trajectory design," in *Proc. IEEE Int. Conf. Intell. Robots Syst.*, Maui, HI, USA, Oct. 29–Nov. 3, 2001, pp. 43–48.

[18] A. Shkolnik and R. Tedrake, "High-dimensional underactuated motion planning via task space control," in *Proc. IEEE/RSJ IROS*, Nice, France, Sep. 22–26, 2008, pp. 3762–3768.

[19] D. Berenson and S. S. Srinivasa, "Probabilistically complete planning with end-effector pose constraints," in *Proc. IEEE ICRA*, Anchorage, AK, USA, May 3–8, 2010.

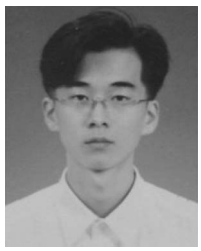
[20] J. M. Porta, L. Jaillet, and O. Bohigas, "Randomized path planning on manifolds based on higher-dimensional continuation," *Int. J. Robot. Res.*, vol. 31, no. 2, pp. 201–215, 2012.

[21] I. A. Sucan and L. E. Kavraki, "A sampling-based tree planner for systems with complex dynamics," *IEEE Trans. Robot.*, vol. 28, no. 1, pp. 116–131, Feb. 2012.

[22] J. L. Bentley, "K-d trees for semidynamic point sets," in *Proc. 6th SCG*, New York, NY, USA, 1990, pp. 187–197.

[23] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *Int. J. Robot. Res.*, vol. 30, no. 7, pp. 846–894, Jun. 2011.

[24] J. H. Jeon, S. Karaman, and E. Frazzoli, "Anytime computation of time-optimal off-road vehicle maneuvers using the RRT*," in *Proc. IEEE CDC*, Dec. 12–15, 2011, pp. 3276–3282.
[25] CostMap2D. [Online]. Available: http://www.ros.org/wiki/costmap_2d
[26] Robot Operating System (ROS). [Online]. Available: http://www.ros.org
[27] Accessed Jun. 2013. [Online]. Available: http://www.youtube.com/watch?v=sp96fSdcy-s
[28] M. Aicardi, G. Casalino, A. Bicchi, and A. Balestrino, "Closed loop steering of unicycle-like vehicles via Lyapunov techniques," *IEEE Robot. Automat. Mag.*, vol. 2, no. 1, pp. 27–35, Mar. 1995.
[29] A. DeLuca, G. Oriolo, and M. Vendittelli, "Control of wheeled mobile robots: An experimental overview," in *RAMSETE. Articulated and Mobile Robots for SErvices and TEchnology*. London, U.K.: Springer-Verlag, 2001.
[30] Y. Kanayama, Y. Kimura, F. Miyazaki, and T. Noguchi, "A stable tracking control method for an autonomous mobile robot," in *Proc. IEEE Int. Conf. Robot. Autom.*, Cincinnati, OH, USA, May 1990, pp. 384–389.
[31] G. Goretkin, A. Perez, R. Platt, and G. Konidaris, "Optimal sampling-based planning for linear-quadratic kinodynamic systems," in *Proc. IEEE ICRA*, Karlsruhe, Germany, May 6–10, 2013, pp. 2429–2436.
[32] D. J. Webb and J. van den Berg, "Kinodynamic RRT*: Asymptotically optimal motion planning for robots with linear dynamics," in *Proc. IEEE ICRA*, Karlsruhe, Germany, May 6–10, 2013, pp. 5054–5061.
[33] S. Karaman and E. Frazzoli, "Sampling-based optimal motion planning for non-holonomic dynamical systems," in *Proc. IEEE ICRA*, Karlsruhe, Germany, May 6–10, 2013, pp. 5041–5047.
[34] Y. Kwangjin and S. Sukkarieh, "An analytical continuous-curvature path-smoothing algorithm," *IEEE Trans. Robot.*, vol. 26, no. 3, pp. 561–568, Jun. 2010.
[35] Y. Kwangjin, "An efficient spline-based RRT path planner for non-holonomic robots in cluttered environments," in *Proc. ICUAS*, Atlanta, GA, USA, May 28–31, 2013, pp. 288–297.
[36] Y. Kwangjin, D. Jung, and S. Sukkarieh, "Continuous curvature path-smoothing algorithm using cubic Bézier spiral curves for non-holonomic robots," *Adv. Robot.*, vol. 27, no. 4, pp. 247–258, Mar. 2013.
[37] D. Devaurs, T. Simeon, and J. Cortes, "Parallelizing RRT on large-scale distributed-memory architectures," *IEEE Trans. Robot.*, vol. 29, no. 2, pp. 571–579, Apr. 2013.
[38] M. Elebanhawi and M. Simic, "Sampling-based robot motion planning: A review," *IEEE Access*, vol. 2, pp. 56–77, Jan. 2014.
[39] Adaptive Monte Carlo Localization (AMCL) package. [Online]. Available: http://wiki.ros.org/amcl
[40] D. Fox, "Adapting the sample size in particle filters through KLD-sampling," *Int. J. Robot. Res.*, vol. 22, no. 12, pp. 985–1003, Dec. 2003.
[41] R. Chou, Y. Boers, M. Podt, and M. Geist, "Performance evaluation for particle filters," in *Proc. 14th Int. Conf. FUSION*, Chicago, IL, USA, Jul. 5–8, 2011, pp. 985–1003.
[42] S. Zaman, W. Slany, and G. Steinbauer, "ROS-based mapping, localization and autonomous navigation using a Pioneer 3-DX robot and their relevant issues," in *Proc. SIECPC*, Riyadh, Saudi Arabia, Apr. 24–26, 2011, pp. 1–5.
[43] D. Claes, D. Hennes, K. Tuyls, and W. Meeussen, "Collision avoidance under bounded localization uncertainty," in *Proc. IEEE/RSJ IROS*, Algarve, Portugal, Oct. 7–12, 2012, pp. 1192–1198.
[44] L. F. Bertuccelli, A. Wu, and J. P. How, "Robust adaptive Markov decision processes: Planning with model uncertainty," *IEEE Trans. Contr. Syst. Technol.*, vol. 32, no. 5, pp. 96–109, Oct. 2012.
[45] T. Lee, S. Baek, and S. Oh, "Sector-based maximal online coverage of unknown environments for cleaning robots with limited sensing," *Robot. Autonom. Syst.*, vol. 59, no. 10, pp. 698–710, Oct. 2011.
[46] C.-K. Lin, T.-H. Liu, J.-t. Yu, L.-C. Fu, and C.-F. Hsiao, "Model-free predictive current control for interior permanent-magnet synchronous motor drives based on current difference detection technique," *IEEE Trans. Ind. Electron.*, vol. 61, no. 2, pp. 667–681, Feb. 2014.



Chang-bae Moon received the B.S., M.S., and Ph.D. degrees from the School of Mechanical Engineering, Korea University, Seoul, Korea, in 2006, 2008, and 2012, respectively.

He is currently a Research Professor with Korea University. His research interests include mobile robot motion control, path planning, localization, and selection of motion controllers using discrete-event systems.

Woojin Chung (M'05) received the B.S. degree from the Department of Mechanical Design and Production Engineering, Seoul National University, Seoul, Korea, in 1993 and the M.S. and Ph.D. degrees from the Department of Mechano-Informatics, The University of Tokyo, Tokyo, Japan, in 1995 and 1998, respectively.

From 1998 to 2005, he was a Senior Research Scientist with the Korea Institute of Science and Technology, Seoul. Since 2005, he has been with the Department of Mechanical Engineering, Korea University, Seoul. His research interests include the design and control of nonholonomic underactuated mechanical systems, trailer system design and control, and mobile robot navigation.

Dr. Chung received an Excellent Paper Award from the Robotics Society of Japan in 1996 and the King-sun Fu Memorial Best Transactions Paper Award from the IEEE Robotics and Automation Society in 2002.