INTERNATIONAL JOURNAL OF PRECISION ENGINEERING AND MANUFACTURING Vol. 17, No. 5, pp. 591-596 MAY 2016 / 591
DOI: 10.1007/s12541-016-0071-3 ISSN 2234-7593 (Print) / ISSN 2005-4602 (Online)

Practical Probabilistic Trajectory Planning Scheme based on the Rapidly-Exploring Random Trees for Two-Wheeled Mobile Robots

Chang-bae Moon[1] and Woojin Chung[2,#]

1 School of Mechanical Engineering, Chonnam National University, 77 Yongbong-ro, Buk-gu, Gwangju, 61186, South Korea
2 Department of Mechanical Engineering, Korea University, 145, Anam-ro, Seongbuk-gu, Seoul, 02841, South Korea
Corresponding Author / E-mail: smartrobot@korea.ac.kr, TEL: +82-2-3290-3856, FAX: +82-2-3290-3856

KEYWORDS: Mobile robot, Path planning, Motion planning, Trajectory planning

The RRT (Rapidly Exploring Random Tree) based planners using probabilistic sampling approaches have been receiving significant attention because of their ability to deal with high-dimensional planning problems efficiently. However, it is still a challenge to generate trajectories for a mobile robot, given the kinematic and dynamic constraints. In this paper, we present an RRT node extension scheme using an asymptotically stable controller for a two-wheeled mobile robot. The proposed algorithm can generate dynamically feasible trajectories. The simulation results show that the proposed scheme can deal with the narrow regions efficiently. The computational time of the simulation results shows that the proposed scheme is twice as fast as the conventional approach.

Manuscript received: November 30, 2014 / Revised: January 8, 2016 / Accepted: January 19, 2016

1. Introduction

Recently, mobile service robots have become widely developed. A trajectory generation technique, which includes path planning and control input generation, is an essential requirement for a mobile robot to move around a given target space. The wide variety of path planning and motion control schemes are presented in Refs. 1-3. The path planning schemes can be classified into deterministic A* search-based methods[4] that use a grid-map, and probabilistic sampling-based schemes such as RRT (Rapidly-Exploring Random Tree).[5] It is difficult to apply the A* search-based method on high-dimensional planning problems. The RRT-based planners have received significant attention because these planners can solve high-dimensional planning problems in a continuous domain. However, it is still a challenge to generate a trajectory for a mobile robot considering the kinematic and dynamic constraints.

The deterministic nonholonomic path planning is a well-known NP-Hard problem.[6] In contrast with the deterministic planning schemes, the RRT based schemes solve path planning problems using a random sampling in configuration space. The RRT based schemes have a probabilistic completeness property that the planning problem can be solved if at least one solution exists. The basic RRT schemes exploit uniform random sampling strategies. The sampling methods considering stochastic environments for RRT-based schemes are widely developed. The proposed path planning schemes in Ref. 7-9 assumed the stochastic process transition models to deal with the motion uncertainty.

For two-wheeled mobile robots, holonomic path planning schemes are widely used, as proposed in Ref. 4. The approach presented in Ref. 4 exploits the dynamic programming method using a grid map. However, approaches similar to that proposed in Ref. 4 are suffered from curvature discontinuity of the generated path, in addition to the grid resolution having to deal with narrow areas. To address the curvature discontinuity problem, a continuous curvature path generation scheme is proposed in Ref. 10. The proposed scheme in Ref. 10 exploits holonomic path generation algorithm with curve-fitting using the Cubic B-Spline. To solve the grid resolution problem, a Local Multi Resolution Path Planning scheme is proposed in Ref. 11. The proposed scheme in Ref. 11 relaxes the resolution problem. However, the increase in complexity is inevitable and the discretization problem of environments remains unsolved.

Reactive schemes, such as Refs. 12-15, have been proposed to deal with the obstacle avoidance problem. These approaches compute optimal control input using sensor data. Generally, the reactive schemes are combined with A* search-based holonomic path planners to generate via-points. However, the local minimum problem and the problem of

© KSPE and Springer 2016

fluctuation in the heading direction of the mobile robot persist in most of these reactive schemes. Moreover, these approaches find difficulty in dealing with narrow passage problems, as shown in Ref. 16.

The sampling-based scheme RRT has been proposed to address the high-dimensional planning problems. The kinodynamic planning algorithm, proposed by LaValle and Kuffner, is given in Ref. 5. One of the significant advantages of the RRT path planners over the grid-based search schemes is that discretization of the environments is unnecessary. The RRT-based planners perform sampling in a continuous domain. Although the RRT planners can deal with the high-dimensional planning problem, it is hard to design an appropriate trajectory generation algorithm based on the RRT. For example, random sampling in 5-dimensions (x, y, θ, v, w) can be undertaken naively. Path planning in a 5-dimensional space requires an enormous amount of computational time. In general, RRT-based schemes are required to exploit advanced nearest search schemes such as K-d trees[17] or the Approximated Nearest Search.[18] However, these advanced search schemes cannot be exploited for a distance metric that includes the orientation. The computation of the nearest metric for control input still remains a problem for practical applications for two-wheeled mobile robots.

In order to use sampling-based planning schemes in practical environments, the following problems need to be resolved, as shown in Ref. 19. First, an appropriate design of the motion primitives is required in consideration of the robot configuration. For a two-wheeled mobile robot, it is desirable to include both of holonomic and nonholonomic motion primitives. For example, spot-turn and move motion can be advantageous in narrow and cluttered environments. Continuous-curvature trajectories are appropriate for high-speed navigation. Second, the metric function should be carefully designed. Finally, the Region of Inevitable Collision sets (RICs)[20] should be computed efficiently to guarantee safety. For computation of the RICs, a trajectory planning scheme that plans the path as well as the velocity is required. The computation of exact RICs involves high computational cost. Hence, it is recommended to reduce the computational costs by simplification.

The extension of the nodes can be achieved using local planners, which, as the name suggests, plan the local path. Local planning schemes that consider differential constraints are proposed for a car-like vehicle using pre-planned motion primitives in Refs. 21 and 22. A car-like vehicle has a finite curvature constraint. Several pre-planned local paths that are generated off-line are required to avoid resolution completeness problems and compute the exact velocity constraints for the RICs. In other methods, adjusting the resolution of the pre-planned local path is required to guarantee resolution completeness as shown in Ref. 22. However, a two-wheeled mobile robot is free from the curvature constraint. Several pre-planned local trajectories are required to deal with all of the available motions of a two-wheeled mobile robot.

A node extension scheme using the forward motion simulation was proposed in Ref. 23 for a car-like mobile robot. The proposed scheme in Ref. 19 was verified in the DARPA Urban Challenge.[24,25] In this paper, we present a trajectory generation scheme using the RRT that is based on the proposed scheme in Ref. 23, considering the kinematic and dynamic constraints of a two wheeled mobile robot. In this paper, the presented algorithm samples in a two dimensional space $\boldsymbol{x} \subset R^2$ without considering the orientation, to exploit the K-d trees. This approach is advantageous, as it can be adapted to many of the nearest neighbor searches, in between the previously generated nodes. We exploit the asymptotically stable motion controller proposed in Ref. 26 to extend the RRT nodes. Using the controller proposed in Ref. 26, the proposed scheme in this paper can be employed to generate dynamically feasible trajectories. The proposed scheme was compared with the steering based extension approach used in Refs. 27 and 28. The proposed scheme is advantageous since the proposed scheme does not require exhaustive search algorithms to compute the optimal velocity output between the admissible velocity sets. A comparison of the results proves that the proposed algorithm is faster than that of the compared extension approach.

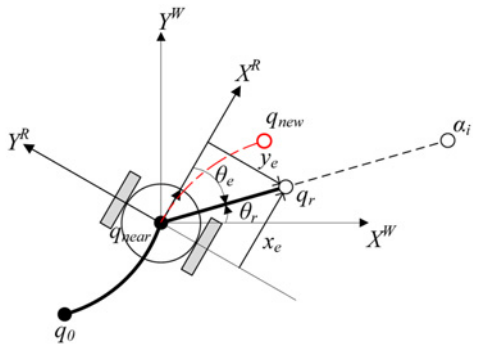


Fig. 1 CB-RRT: Error posture for computing output velocity

The remaining sections of this paper are organized as follows. In section II, we present our trajectory generation scheme. In section III, the simulation results and analysis will be presented. Finally, the concluding remarks are presented in section IV.

2. Trajectory Generation Scheme based on the RRT

2.1 Trajectory generation method based on the RRT

Fig. 2 illustrates the proposed Controller Based-RRT extension scheme, or simply CB-RRT. As shown in Fig. 1, random sampling and reference point computation are the same as the S-RRT (Steering RRT).[27,28] We exploit the control scheme proposed by Kanayama et al., in Ref. 26. The error posture is computed in a robot's local coordinate by computing the following equations.

$$q_e = \begin{bmatrix} x_e \\ y_e \\ \theta_e \end{bmatrix} = \begin{bmatrix} cos\theta_n & sin\theta_n & 0 \\ -sin\theta_n & cos\theta_n & 0 \\ 0 & 0 & 1 \end{bmatrix} (q_r - q_n) \tag{1}$$

Using Eq. (1), the error posture is computed using the nearest neighbor.

$$u = \begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} v(q_e, q_r) \\ \omega(q_e, q_r) \end{bmatrix} = \begin{bmatrix} v_r cos\theta_e + K_x x_e \\ \omega_r + v_r (K_y y_e + K_\theta sin\theta_e) \end{bmatrix} \tag{2}$$

The velocity output is computed using Eq. (2). The control parameters k_x, k_y and k_θ are positive real numbers. The convergence properties can be affected by the k_x, k_y and k_θ control parameters. For example, if a controller exploits a large k_θ, the mobile robot concentrates on minimizing the heading error. The velocity outputs, after considering

the acceleration limits, are computed using the following equations.

$$v_{new} = \begin{cases} v & ;|v - v_c| \le a \\ v_n + sign(v - v_c) \cdot a & ;otherwise \end{cases} \quad (3)$$

$$\omega_{new} = \begin{cases} \omega & ;|\omega - \omega_c| \le \alpha \\ \omega_n + sign(\omega - \omega_c) \cdot \alpha & ;otherwise \end{cases} \quad (4)$$

The parameters a and α are translational and rotational acceleration limits, respectively. The difference between the velocities is computed using Eqs. (3) and (4) and the current velocity is computed for determining the acceleration limits.

Algorithm I. Build_Tree(q_{init})

```
1  T.init(q_init);
2  X ← NULL; //State List
3  U ← NULL; //Control Input List
4  for k = 1 to K do
5    α_i ← GENERATERANDOMSTATE();
6    q_new ← EXTEND_TRAJECTORY(T, α_i);
7    if IS_GOAL_REACHED(q_new, ε_g) then
8      <X, U> ← GENERATETRAJECTORY(T, q_new);
9      break;
10   end if
11 end for
12 return <T, X, U>;
```

Algorithm II. EXTEND_TRAJECTORY(T, α_i)

```
1  q_n ← NEAREST_ NEIGHBOR (α_i, T);
2  x_ref ← COMPUTE_REFERENCE_POINT(α_i);
3  u_0 ← GET_VELOCITY(q_n); //CB-RRT only
4  u ← COMPUTE_CONTROL_INPUT(q_n, x_ref);
5  if EXIST_CONTROL_INPUT(q_near, u, ε_u) then
6      return false;
7  while(t ≤ t_h)
8      u_c ← u_0 + (u − u_0) · t_h //CB-RRT only
9      x ← x + ∫_t^{t+Δt} f(x(t), u_c)dt
10     t ← t + Δt
11     if FEASIBLE_STATE( x ) = false then break;
12   end while;
13 q_new=<x, t, u>;
14 T.add_vertex(q_new);
15 T.add_edge(q_near, q_new, u);
16 return true;
```

Algorithm I and II show the trajectory generation algorithm by computing the control input. The basic Algorithm I is based on the conventional RRT algorithm proposed in Ref. 5. The EXTEND_TRAJECTORY Algorithm II is completely redesigned for computing the control input and for trajectory generation. The state of a mobile robot, while considering the dynamics, is $\boldsymbol{x} \in SE(2) \times R^2$ where $\boldsymbol{x} = (x, y, \theta, v, \omega)^T$. The control input $\mathbf{u}$ is $(v, \omega)^T$. The S-RRT does not consider dynamic constraints. The third and eight lines of Algorithm II that considers the dynamic constraints are only applied in the CB-RRT.

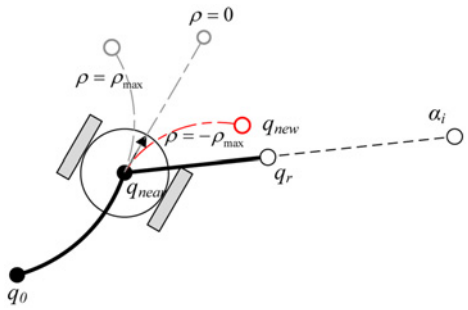


Fig. 2 S-RRT: Control input selection

We implemented a node extension scheme using the extreme steering angle presented in Refs. 27 and 28 for comparing the computational results quantitatively. We will refer to the steering-based RRT as S-RRT for simplicity.

Fig. 2 illustrates the control input selection method of the S-RRT. As shown in Fig. 2, q_r is computed by finding the nearest neighbor q_{near} from the randomly sampled point α_i. By computing the resultant trajectory candidates using the three radii of curvature $\{\rho_{max}, 0, -\rho_{max}\}$, a rotational velocity ω is selected. The S-RRT computes the control input considering only the kinematic constraints, without the dynamic constraints. The generated path using the $-\rho_{max}$ is nearest to $\boldsymbol{q_r}$ for the case shown in Fig. 2. As a result, $-\omega_{max}$ is selected corresponding to the $-\rho_{max}$ in Fig. 2.

3. Simulation Results

3.1 Simulation settings

Simulations are carried out for the two cases shown in Fig. 3. The dimensions of the simulation environments are 10 m × 10 m. As shown in Fig. 3, the passage distance between the start and the goal is 0.2 m. By employing the commonly used grid sizes have a range of 0.1~0.2 m, it is difficult to apply the grid-based scheme in these simulation environments. Because of the round-off errors of range sensors, such as the laser range finders, the narrow gaps between the start and the goal may be misinterpreted as blocked regions. Therefore, RRT-based planning schemes that use the continuous domain sampling are required to deal with the simulation environments shown in Fig. 3. The planning complexity of Case 2 is higher than that of Case 1 form the view point of the available number of solutions.

Path planning is considered a failure when the path planning is not completed within 50,000 sampling trials. Simulations are performed using an Intel I7 CPU with 2.6GHz. The simulations are executed using *K-d* trees; the observed computational time in this method proves that this method is almost ten times as fast as the brute force search method.

The initial configuration q_{start} (x, y, θ) is (-4, -4, 0) and the goal configuration q_{goal} is (4, 4, 0). The parameters of the S-RRT are set to v_{max} = 1.0 m/s and ω_{max} = {-30, 0, 30} deg/s. The following parameters are used for the CB-RRT. The control parameters k_x, k_y and k_θ are set to 1.5, 0.5 and 0.1, respectively. These parameters can be modified to adjust the convergence properties. The translational and rotational velocity ranges are set to v = [0.0, 1.0] m/s and ω = [-30, 30] deg/s,

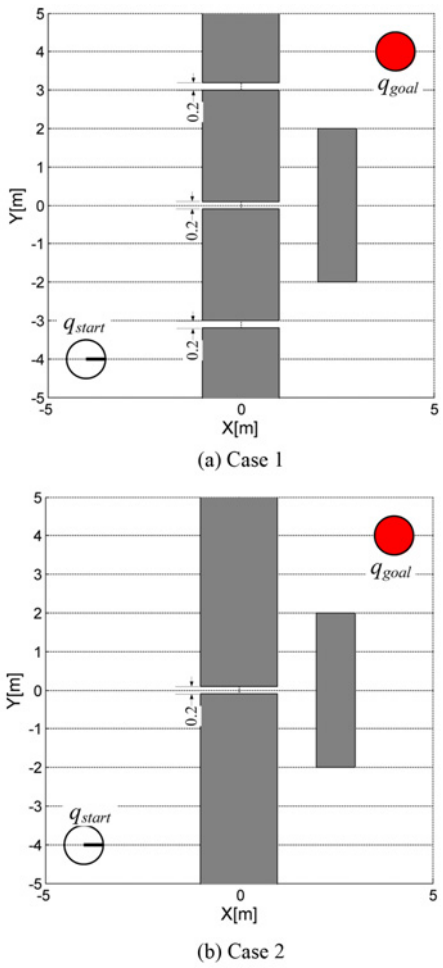


Fig. 3 Target C-Space: start (q_{start}) and goal (q_{goal}) configuration

respectively. The maximum accelerations for the translational velocity and rotational velocity are 0.6 m/s^2 and 30 deg/s^2, respectively. These parameters are based on our mobile robot Tetra-DS II.[29] The translational velocity range implies that the mobile robot does not realize backward motion. These assumptions are practically acceptable for a mobile robot lacks rear side sensors.

3.2 Simulation results

Fig. 4 shows the path generation results using the S-RRT and trajectory generation results using the CB-RRT. In Fig. 4, the red dots represent the generated node q and the thin red lines represent edges of the RRT. Green dots and thick green lines represent the generated path from the start position to the goal position. The detailed results are listed in Table 1. Fig. 4 demonstrates that the proposed algorithm is more efficient than the S-RRT since the computational time is lower than that of the S-RRT and the quality of the generated path is more applicable intuitively. The calculation time is twice as fast as that of the S-RRT. Despite the gap between the start and the goal node, the S-RRT and CB-RRT schemes successfully accomplished the path planning. Moreover, the proposed scheme completed the trajectory generation within 7 ms. This result shows that the proposed scheme is applicable to real-time planning.

Fig. 5 shows the trajectory generation results using the CB-RRT for

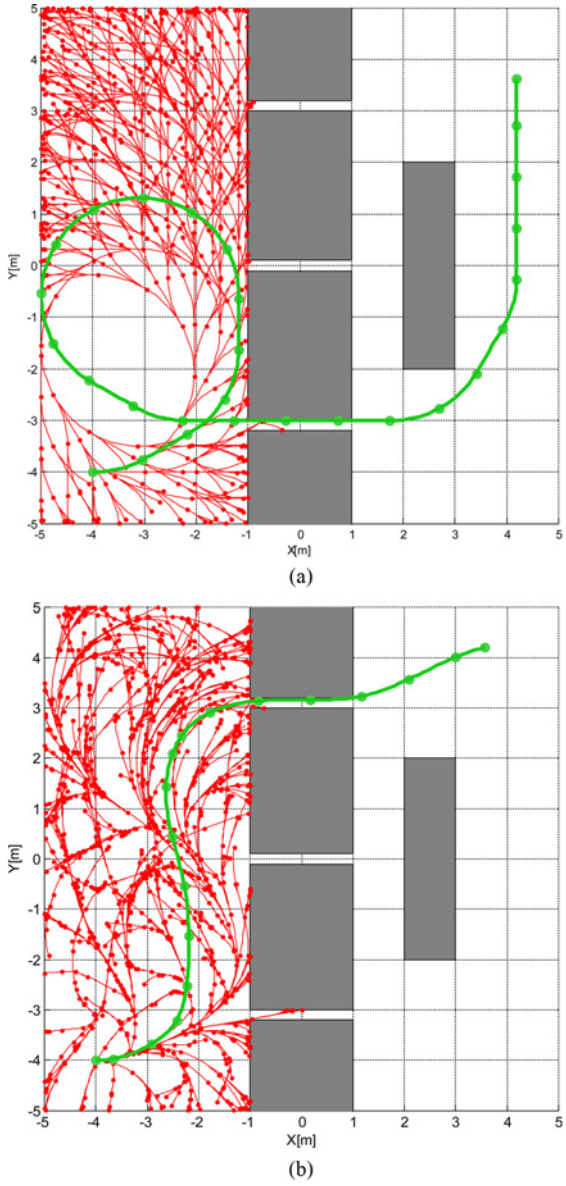


Fig. 4 Planning results using (a) S-RRT, (b) CB-RRT

Table 1 Simulation results for case 1

	Nodes	Time(ms)
S-RRT	658	15
CB-RRT	228	7.75

Case 2. The point turn positions (v = 0 m/s) are denoted by the blue squares in Fig. 5. We simulated the S-RRT scheme and the detailed simulation results are listed in Table 2. However, path generation failed using the S-RRT. The reason for this result is the motion limitations of the S-RRT upon encounters with sufficiently narrow areas. The CB-RRT scheme generates point turn control input automatically. This result implies that the full mobility including point turn is useful in handling narrow region passing problems. The bidirectional RRT that uses two RRT simultaneously, one from the start node and another from the goal node,[3] can be exploited to deal with these problems. However, in practical environments the current target goal may not be

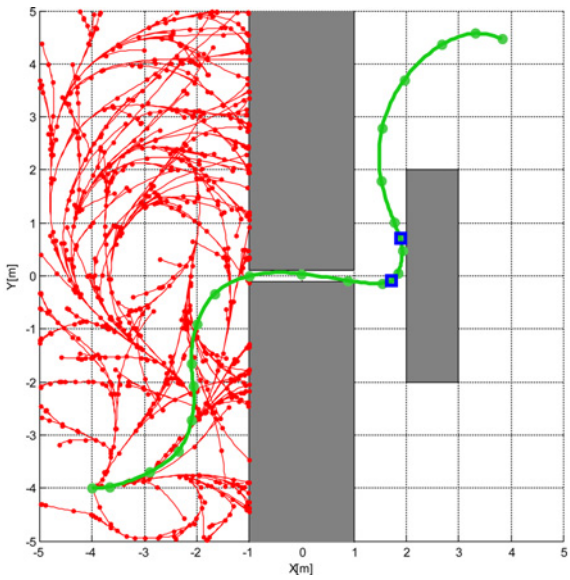


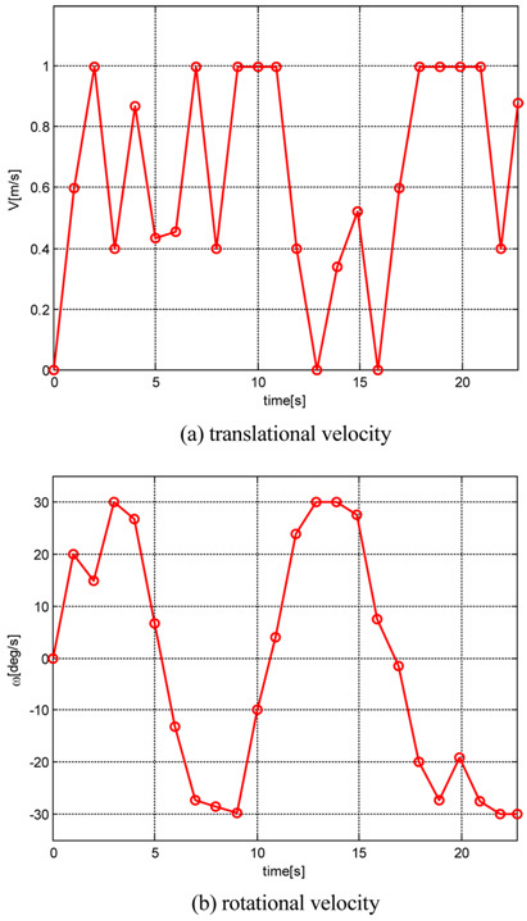
Fig. 5 Trajectory generation result using CB-RRT for the Case 2

Table 2 Simulation results for case 2

	Nodes	Time(ms)
S-RRT	failure	
CB-RRT	1,391	16

the final goal to complete the given tasks. Target environments are often partially unknown in real environments. The proposed scheme demonstrates efficient results without the use of a bidirectional search.

Fig. 6 shows the generated control inputs of both translational and rotational velocity commands. The x-axes unit of measurement for Fig. 6 is seconds. The red-dots represent time steps of every second for visual simplicity in Fig. 6. As shown in Fig. 6, the generated trajectory (including time information with pose) does not violate the dynamic constraints. These results show that the proposed CB-RRT scheme is advantageous because it does not require a path smoothing algorithm. Moreover, the computational time of the CB-RRT is shorter since the S-RRT requires additional trajectory generation algorithms using the planned path.



(a) translational velocity

(b) rotational velocity

Fig. 6 Generated control inputs for Case 2 by using the CB-RRT

4. Conclusions

In this research, we presented the RRT extension scheme using an asymptotic stable controller for a two wheeled mobile robot. The proposed algorithm can generate dynamically feasible trajectories. The simulation environments have narrow regions, and the simulation results show that the proposed scheme can deal with these narrow regions efficiently. The computational time of the simulation results proves that the proposed approach is twice as fast as the conventional approach.

ACKNOWLEDGEMENT

This Research was supported by the NRF, MSIP (NRF-2014R1A2A1A10049634), also supported by the Agriculture, Food and Rural Affairs Research Center Support Program (714002-07), MAFRA, also supported by the COMPA, MISP (2015K000122), also supported by the Industrial Convergence Core Technology Development Program, MOTIE (10048474), also supported by the Korea University Grant.

REFERENCES

1. Latombe, J.-C., “Robot Motion Planning,” Springer Science & Business Media, pp. 153-355, 1991.
2. Choset, H., Lynch, K. M., Hutchinson, S., Kantor, G., Burgard, W., et al., “Principles of Robot Motion: Theory,” MIT Press, pp. 197-268, 2005.
3. LaValle, S. M., “Planning Algorithms,” Cambridge University Press, pp. 651-711, 2006.
4. Konolige, K., “A Gradient Method for Realtime Robot Control,” Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems, Vol. 1, pp. 639-646, 2000.
5. LaValle, S. M. and Kuffner, J. J., “Randomized Kinodynamic Planning,” The International Journal of Robotics Research, Vol. 20, No. 5, pp. 378-400, 2001.

6. Canny, J., and Reif, J., "New Lower Bound Techniques for Robot Motion Planning Problems," Proc. of IEEE 28th Annual Symposium on Foundations of Computer Science, pp. 49-60, 1987.

7. Melchior, N. A. and Simmons, R., "Particle RRT for Path Planning with Uncertainty," Proc. of IEEE International Conference on Robotics and Automation, pp. 1617-1624, 2007.

8. Blackmore, L., Li, H., and Williams, B., "A Probabilistic Approach to Optimal Robust Path Planning with Obstacles," Proc. of IEEE American Control Conference, 2006.

9. Burns, B. and Brock, O., "Sampling-based Motion Planning with Sensing Uncertainty," Proc. of IEEE International Conference on Robotics and Automation, pp. 3313-3318, 2007.

10. Quinlan, S., "Real-Time Modification of Collision-Free Paths," Ph.D. Thesis, Department of Computer Science, Stanford University, 1994.

11. Behnke, S., "Local Multiresolution Path Planning," Robocup 2003: Robot Soccer World Cup VII, pp. 332-343, 2003.

12. Brock, O. and Khatib, O., "High-Speed Navigation using the Global Dynamic Window Approach," Proc. of IEEE International Conference on Robotics and Automation, Vol. 1, pp. 341-346, 1999.

13. Fox, D., Burgard, W., and Thrun, S., "The Dynamic Window Approach to Collision Avoidance," IEEE Robotics & Automation Magazine, Vol. 4, No. 1, pp. 23-33, 1997.

14. Ulrich, I. and Borenstein, J., "VFH^*: Local Obstacle Avoidance with Look-Ahead Verification," Proc. of IEEE International Conference on Robotics, and Automation, pp. 2505-2511, 2000.

15. Borenstein, J. and Koren, Y., "The Vector Field Histogram-Fast Obstacle Avoidance for Mobile Robots," IEEE Transactions on Robotics and Automation, Vol. 7, No. 3, pp. 278-288, 1991.

16. Moon, C.-b. and Chung, W., "Design of Navigation Behaviors and the Selection Framework with Generalized Stochastic Petri Nets Toward Dependable Navigation of A Mobile Robot," Proc. of IEEE International Conference on Robotics and Automation (ICRA), 2010, pp. 2989-2994, 2010.

17. Bentley, J. L., "K-d Trees for Semidynamic Point Sets," Proc. of the 6th Annual Symposium on Computational Geometry, pp. 187-197, 1990.

18. Arya, S., Mount, D. M., Netanyahu, N. S., Silverman, R., and Wu, A. Y., "An Optimal Algorithm for Approximate Nearest Neighbor Searching Fixed Dimensions," Journal of the ACM (JACM), Vol. 45, No. 6, pp. 891-923, 1998.

19. LaValle, S. M., "Motion Planning," IEEE Robotics and Automation Magazine, Vol. 18, No. 2, pp. 108-118, 2011.

20. Fraichard, T. and Asama, H., "Inevitable Collision States-A Step Towards Safer Robots?" Advanced Robotics, Vol. 18, No. 10, pp. 1001-1024, 2004.

21. Lacaze, A., Moscovitz, Y., DeClaris, N., and Murphy, K., "Path Planning for Autonomous Vehicles driving Over Rough Terrain," Proc. of IEEE ISIC/CIRA/ISAS Joint Conference, pp. 50-55, 1998.

22. Pivtoraiko, M., Knepper, R. A., and Kelly, A., "Differentially Constrained Mobile Robot Motion Planning in State Lattices," Journal of Field Robotics, Vol. 26, No. 3, pp. 308-333, 2009.

23. Kuwata, Y., Karaman, S., Teo, J., Frazzoli, E., How, J. P., and Fiore, G., "Real-Time Motion Planning with Applications to Autonomous Urban Driving," IEEE Transactions on Control Systems Technology, Vol. 17, No. 5, pp. 1105-1118, 2009.

24. Iagnemma, K. and Buehler, M., "Special Issue on the DARPA Grand Challenge, Part 1," Journal of Field Robotics, Vol. 23, No. 8, pp. 461-652, 2006.

25. Iagnemma, K., Buehler, M., and Singh, S., "Special Issue on the 2007 DARPA Urban Challenge," Journal of Field Robotics, Vol. 25, No. 8, pp. 423-860, 2008.

26. Kanayama, Y., Kimura, Y., Miyazaki, F., and Noguchi, T., "A Stable Tracking Control Method for an Autonomous Mobile Robot," Proc. of IEEE International Conference on Robotics and Automation, Vol. 1, pp. 384-389, 1990.

27. Kim, D. and Chung, W., "Motion Planning for Car-Parking using the Slice Projection Technique," Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 1050-1055, 2008.

28. Kim, D., Chung, W.-H., and Park, S., "Practical Motion Planning for Car-Parking Control in Narrow Environment," IET Control Theory & Applications, Vol. 4, No. 1, pp. 129-139, 2010.

29. DST Robot Co., Ltd., "Mobile Platform," http://www.dongburobot.com/jsp/cms/view.jsp?code=100783 (Accessed 17 MAR 2016)