ORIGINAL ARTICLE

2D-Span Resampling of Bi-RRT in Dynamic Path Planning

Hsien-I Lin[1,*] and Cheng-Sun Yang[2]

[1,2]*National Taipei University of Technology*

(Received 31 July 2014; Accepted 7 November 2014; Published on line 1 March 2015)

*Corresponding author email: sofin@ntut.edu.tw

DOI: 10.5875/ausmt.v5i1.837

Abstract: Path planning is an essential task in robot soccer to enable the robot to quickly arrive at a desired location from which it can shoot or dribble the ball to a goal. Previous work in path planning used sonar or laser-based sensors to obtain local information for avoiding obstacles and reaching the goal. In the process, the robot may move slowly and collide easily with other robots using similar obstacle-avoidance algorithms. This work proposes a 2D-span resampling method and post processing including pruning and smoothening of bi-directional rapidly-exploring random trees (Bi-RRT) to improve the path route and computational time of path planning. To avoid obstacles, the path is re-planned using a novel 2D-span resampling method in Bi-RRT. The post processing of pruning unnecessary Bi-RRT nodes and smoothing the path route enables a robot to reach the goal via a shorter path. Simulations showed the proposed method outperformed several common path-planning methods, generally resulting in a shorter route distance and less computational time.

Keywords: Path Planning; Bi-Directional Rapidly-Exploring Random Tree (Bi-RRT); 2D-Span Resampling

Introduction

RoboCup soccer games feature various types of soccer robots such as wheel-type robots and biped robots. Though the appearances of these robots may differ, they all require certain fundamental functions to play a soccer game such as vision, decision and task planning, obstacle avoidance, path planning, and communication. Path planning determines how quickly a robot can reach a destination without colliding with obstacles. Poor path planning normally results in two outcomes. First, it may take more time for a robot to reach a given destination using its self-embedded sensors because collision with one obstacle may lead to further collisions with other obstacles. Second, the robot may head toward the same direction as another robot executing similar obstacle-avoidance algorithms.

Path planning seeks to discover a collision-free path from a start point to an end point in the workspace. The found path must not only collision-free, but as short as possible. Depending on the amount of information available about a given environment, path planning is broken into two sub-problems: static path planning and dynamic path planning. In static path planning, the locations of potential obstacles are already known, but not in dynamic path planning.

Many approaches have been proposed to solve the problem of static path planning, including roadmap, cell decomposition, potential field, and evolutionary methods. (A) **Roadmap**: This method is preferred when many paths must be planned between different start and end points. This method requires the construction of a map topology to discover an optimal path from a graph as produced by the Visibility graph method [1] or the Voronoi Diagram method [2]. Instead of constructing a map graph, probabilistic roadmap (PRM) [2-4] adopts random samples to represent workspace configurations and build a collision-free path. To improve the computational time of PRM, [5] introduced Visib-PRM to reduce the number of random samples. Rapidly-exploring random tree (RRT) [6-8] was proposed to improve the computational time of PRM through minimizing collision detection instances.

(B) **Cell decomposition**: this method decomposes a map into several small cells [9] to determine which cell is occupied by an obstacle. Connected cells are then

searched to determine the shortest path. Map decomposition is achieved using two approaches: Exact cell decomposition and approximate cell decomposition. (C) **Potential field**: this method treats a robot as a point influenced by an artificial potential field [10,11]. A goal exerts an attractive force on the robot but an obstacle exerts a repulsive force. The total force drives the robot toward the goal and away from obstacles. (D) **Evolutionary method**: this uses iterative algorithms such as particle swarm optimization [12,13), neural networks [14], genetic algorithms [15,16] to identify the suboptimal path according to a given objective function.

From these four static path planning methods, we conclude that (i) visibility graphs and Voronoi diagrams achieve desirable results when obstacle positions are exactly known but the PRM methods produce similar results without knowing exact obstacle positions; (ii) cell-decomposition methods employ complex methods to find cell boundaries; (iii) potential-field methods result in a local solution, which causes a robot to stop in front of an obstacle; (iv) evolutionary method outcomes are easily affected by data representation and the size of the search space.

Dynamic path planning is more difficult than static path planning because obstacle locations are unknown. For example, when the location of a moving obstacle changes, computational time becomes important to path planning to avoid the moving obstacle. Thus, the methods used in static path planning are improved for dynamic obstacles. [17] reduced the number of configurations from two to one for a genetic algorithm (GA) to speed up the robot response time for dynamic obstacles. [18] introduced the concept of re-planning into a genetic algorithm to find a detour to avoid obstacles, starting from a non-colliding point before the obstacle. [19] proposed a new artificial potential field considering an obstacle's position and velocity in a dynamic environment. [20] used a PRM roadmap to generate a path. When an obstacle is on the path, the collided PRM samples were deleted and a new path was created based on the remaining PRM samples. [21] deleted the branches of RRT featuring obstacles and generated new branches until they reached the goal. [22] used the roadmap which was created around an obstacle to update the whole roadmap for dynamic planning. [23] proposed a method to deform planned paths featuring an obstacle.

These previous studies indicate that reducing computational time and shortening path routes are two major problems in path planning. To address these problems, this work proposes an enhanced RRT algorithm to improve the computational time and path route in a dynamic environment. The proposed algorithm deforms and resamples the RRT to avoid dynamic obstacles using a novel 2D-span resampling method, and applies post-processing to prune unnecessary random samples in the RRT to shorten the path and smooth it by a b-spline function. The proposed 2D-span resampling method limits the computational cost and the resampling nodes are close enough to dynamic obstacles to result in a small detour from the planned path. By the abovementioned steps, this work provides a fast algorithm to plan a short and smooth path to a goal in a dynamic environment.

The remainder of this paper is organized as follows. Section II introduces path planning in previous RoboCup soccer games and the proposed solution. Section III presents the system flowchart and the proposed enhanced RRT algorithm including the node deletion algorithm and 2Dspan resampling method. Section IV compares the performance of the proposed method and previous approaches on an Aldebaran Robotics Nao robot. Conclusions are summarized in Section V.

Proposed ideas in path planning in RoboCup Soccer

RoboCup soccer games are grouped in five leagues according to robot size: simulation, small-size, middle-size, standard platform, and humanoid. Each league has its game rules and field size. The Aldebaran Robotics Nao robots shown in Fig. 1 are used in the standard platform league (SPL), and all teams in this league are expected to use Nao robots. The SPL field size measures 9m by 6m, with two teams of five Nao robots playing autonomously.

B-human, the 2011 RoboCup championship team, executed path planning using sonar sensors. Figure 2 shows the zig-zag path, which caused the robot to spend more time avoiding obstacles [24]. In addition, many robots might converge to the same point when they used the similar strategy. B-human later used RRT to plan the path, with results in Fig. 3 showing a much smoother path over that shown in Fig. 2.

According to the results in [24], the RRT algorithm adopted by B-human responded poorly to dynamic obstacles. Also, the RRT algorithm was not designed specifically to minimize computational time. Thus, this work enhances the RRT algorithm to shorten and smoothen the path. Figures 4(a)-(d) show four phases of the proposed idea. Figure 4(a) shows the branches of the

Hsien-I Lin is an Associative Professor of Graduate Institute of Automation Technology at National Taipei University of Technology. He received the Ph.D. degree in Electrical and Computer Engineering from Purdue University, West Lafayette, Indiana in 2009. Dr. Lin's research focuses on humanoid robotics, mobile robotics, rehabilitation robotics, social robotics, and machine learning.

Cheng-Hsun Yang was a graduate student when the manuscript was undertook. He received his master degree in Graduate Institute of Automation Technology at National Taipei University of Technology in2012.

tree where the red circle represents the robot and the blue circle represents the goal. Figure 4(b) shows the obstacle represented by a rectangular box occupying the path to the goal. Figures 4(c) and (d) show the deletion of branches occupied by the obstacle. After branch deletion, it is necessary to resample new nodes to create a new path to avoid the obstacle. In this work, we propose a 2D-span resampling method to generate possible nodes to avoid obstacles. New nodes are generated around the obstacle to ensure a non-colliding detour. Thus, the **RRT** nodes are resampled around the obstacle and the new path is re-routed around the obstacle. Figure 5 shows the new branches generated by the resampling method.

In addition, some of the branches of the RRT algorithm result in unnecessarily long paths. Thus, to reduce path length, the nodes that cause the unnecessary branches are deleted. Figure 6 shows the deletion of the unnecessary nodes. After the node deletion, the red line is clearly much shorter than the blue one. After node deletion, we smoothen the path by a b-spline function. Figure 7 shows the path becomes much smoother after the approximation of the b-spline function.

Figure 1. Nao robot in SPL.

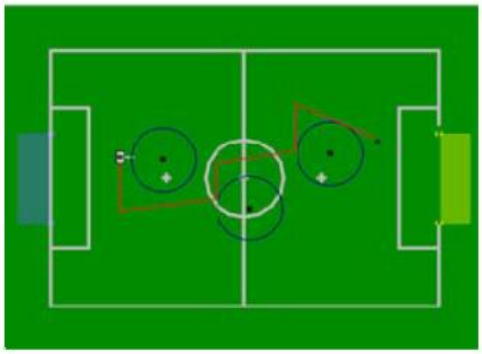

Figure 2. Path planning using sonar sensors. Picture is from [24].

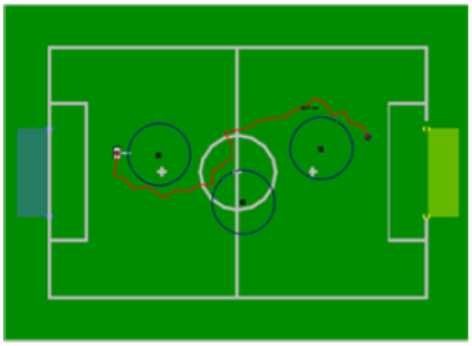

Figure 3. Path planning using RRT algorithm. Picture is from [24].

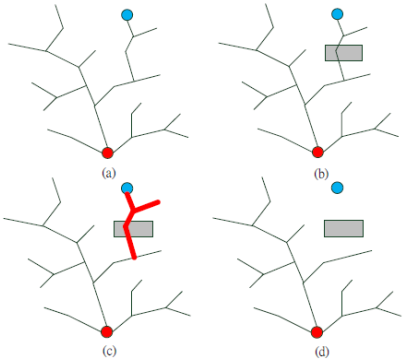

Figure 4. The proposed idea. The red circle represents the robot; the blue circle represents the goal; the rectangular box represents the obstacle.

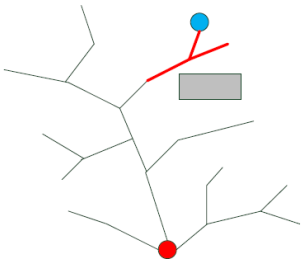

Figure 5. Generation of new tree branches by resample.

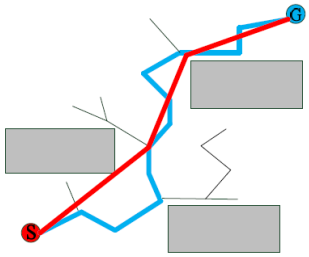

Figure 6. Deletion of the unnecessary nodes. S represents the robot and G represents the goal. The blue path is pre-node deletion, while the red path is post-node deletion.

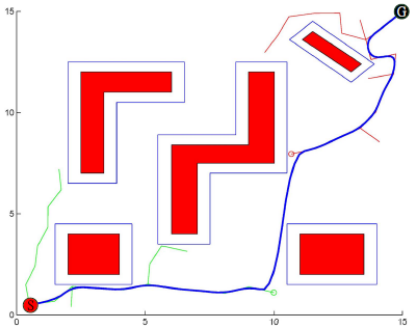

Figure 7. Path smoothening after node deletion.

In sum, we propose a novel idea to shorten the path generated by the basic RRT algorithm and reduce computational time. The contributions of this work are summarized as follows:

(a) Deform the tree generated by the basic RRT algorithm for dynamic obstacles;
(b) Delete unnecessary nodes to shorten the path;
(c) Confine the resampling region to generate RRT nodes and branches by a 2D-span resampling method;
(d) Smoothen the path after node deletion where (a) and (c) save the computational time, and (b) and (d) shorten the path.

Proposed enhanced RRT algorithm

System flowchart

Figure 8 shows the flowchart of the proposed method, with each step described below:

(a) **Bi-RRT for path planning:** we adopt the Bi-RRT algorithm [25] instead of the RRT algorithm because the bi-RRT is faster than the RRT.
(b) **Path:** represents the path found by the Bi-RRT algorithm.
(c) **Collision detection:** used to detect any collision of the path found in (b).
(d) **Collided-branch deletion:** deletes the collided branches detected by (c).
(e) **Deformation:** includes two steps: resampling and extension. Resampling represents that new-nodes are generated in the region where the colliding branches are deleted. Extension generates new branches among the new nodes and finds the new collision-free path.
(f) **Node deletion:** deletes the unnecessary nodes in the path in (e) to shorten the path.
(g) **Smoothening:** smooths the path in (f) by a b-spline function. Since a b-spline function uses knots to control local curves, it is suitable for path planning in this work.

Collision detection

In collision detection (c), we adopt the subdivision algorithm (1) which is a fast method to detect any collision.

Node deletion

Figure 9 shows the procedure for node deletion (f). Nodes 1-5 are the nodes on the path, some of which are unnecessary. The procedure starts with the first step from node 1 to node 5. If any collision occurs between nodes 1 and 5, then go to step 2. Otherwise, there is a shortcut from node 1 to node 5. Step 2 connects node 1 and node 4 and detects any collision between these two nodes. If yes, go to step 3. Otherwise, there is a shortcut. The procedure repeats until the last step from node 1 to node 3. When any shortcut occurs, the procedure repeats the above steps but the start node is changed to the first node of the shortcut path. For example, the procedure checks whether a direct path exists from node 3 to node 5, as shown in Fig. 10. The pseudo codes of the procedure are described as follows.

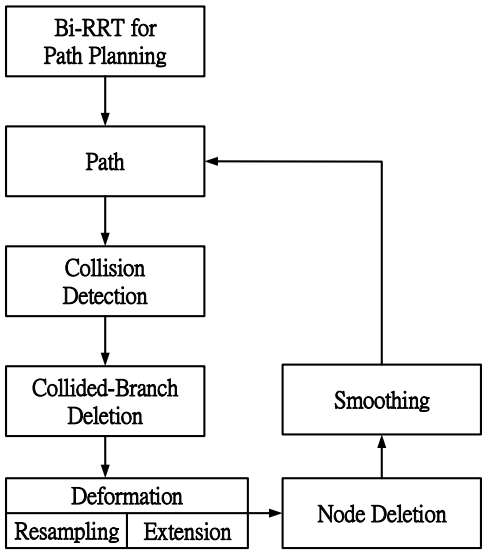

Figure 8. Flowchart of the proposed method.

Node deletion algorithm:

Input: Old Bi-RRT path

Output: New Bi-RRT path with node deletion

---

1. Set $q = q_i$ and i= 1 to max, i is the number of nodes in the path; q is the node in the path.
2. Set $q' = q_j$ and j from max to i + 2
3. **repeat**
4. **while** j≥i + 2 **DO**
5. **if** collision (q; q0) **then**
6. j = j - 1;
7. $q' = q_j$;
8. **else**
9. path (q; q′);
10. $q \leftarrow q_j$;
11. $q' \leftarrow q_{max}$;
12. **BREAK**;
13. **end if**;
14. **end while**;

15. **until** path←q_{max};

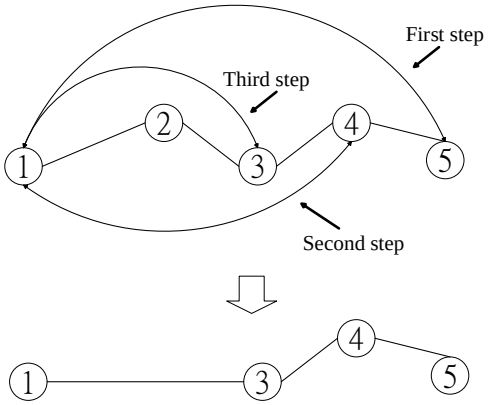

Figure 9. Procedure of node deletion.

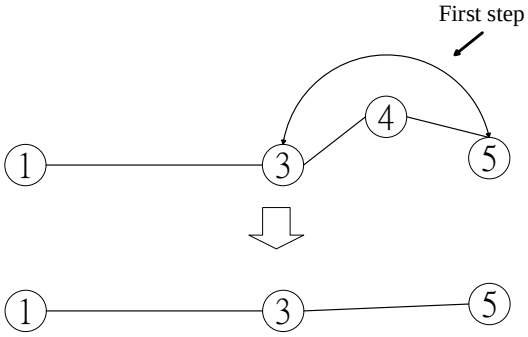

Figure 10. Procedure of node deletion (continued).

2D-span resampling method

In deformation, the branches of an RRT colliding with obstacles should be deleted. Then new nodes and branches are created to generate a new path to avoid the obstacles. With the only information about the collided nodes in the RRT, it is important to resample nodes to speed up the computation in path planning. Thus, we propose a 2D-span resampling method to generate nodes for a new path according to the locations of the collided nodes in the RRT. The generated nodes are located on two 1D grids along the horizontal and vertical axes.

Figures 11(a)-(c) show the collided nodes as black circles and the resampling nodes as red and blue circles. For each colliding node, the proposed method generates two new nodes on the horizontal and vertical axes. Specifically, each two nodes are resampled on an equally-spaced 1D grid spanned over the horizontal or vertical axis and on the opposite sides of the colliding node. In addition, these two nodes are chosen as the two closest to the colliding node and do not collide with the obstacle. Figure 11(d) shows the equally-spaced 1D grids over the horizontal and vertical axes. From Fig. 11(a)-(c), it is obvious that the resampling nodes are generated around the obstacle and can be used to generate a new path. The size of the 1D grid is adjustable, depending on how close to the obstacle the resampling nodes are. In other words, a suitable distance between the new nodes and the original colliding nodes is configurable according to the grid sizes of the horizontal and vertical axes. Figures 12(a) and (b) respectively show small and large grid sizes, and the resampling nodes in Fig. 12(b) are farther from the obstacle than those in Fig. 12(a).

The computational cost of the 2D-span resampling method is O(k) where k is the number of the colliding nodes of the RRT. In fact, there are twice as many newly generated nodes as there are colliding nodes. If it is assumed that the RRT nodes are uniformly distributed over the workspace, the number of colliding nodes and the computational cost are both proportional to the area of the obstacle. Thus, the computational cost of the 2D-span resampling method is limited. Another advantage is that the 2D-span resampling method works for obstacles which are convex (Figs. 11(a) and (b)) and non-convex (Fig. 11(c)). Regardless of the obstacle shapes, new nodes can be generated around the obstacles because the proposed method uses the orthogonal (horizontal and vertical) axes of the colliding nodes to create new nodes.

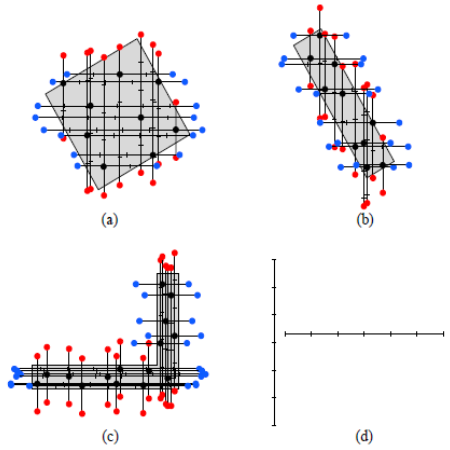

Figure 11. 2D-span resampling method on (a) and (b) convex obstacles and (c) a non-convex obstacle where the black circles are the colliding nodes of the **RRT** and the red and blue circles respectively represent the resampling nodes on the vertical and horizontal axes. (d) Equally-spaced 1D grids over the horizontal and vertical axes.

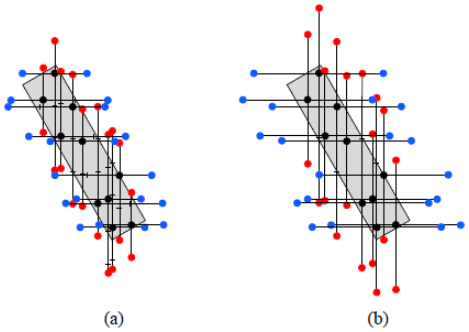

Figure 12. 2D-span resampling method with (a) small and (b) large grid sizes.

Performance comparison among various algorithms

Static path planning

In this section, we evaluate the performance of various algorithms including the proposed method in terms of path length and computation time. We simulate five obstacles in a 15 (m) x 15 (m) area. The red area is the obstacle and the outer boundary represents the safe margin. We evaluate the performance in five cases: (i) PRM vs. proposed method, (ii) Visib-PRM vs. proposed method, (iii) Artificial potential field (APF) vs. proposed method, (iv) RRT vs. proposed method, and (v) Bi-RRT vs. proposed method. Figures 13(a)-(d) show the results of the steps of the proposed method. Figure 13(a) is the result of the pure Bi-RRT algorithm. Figure 13(b) represents the result of node deletion; (c) represents the result of smoothing; (d) is the final result by node deletion and smoothing. Figures 14, 15, 16, and 17 respectively show the comparison results among PRM, Visib-PRM, artificial potential field (APF), and Bi-RRT.

We tested the various methods in MATLAB fifty times each on a computer using an Intel Core i3 processor. Table 1 shows the average and standard deviation of the elapsed time and path distance. Compared to the other methods, the proposed method dramatically reduces the path distance and the computation time is only slightly greater than that achieved using the Bi-RRT algorithm.

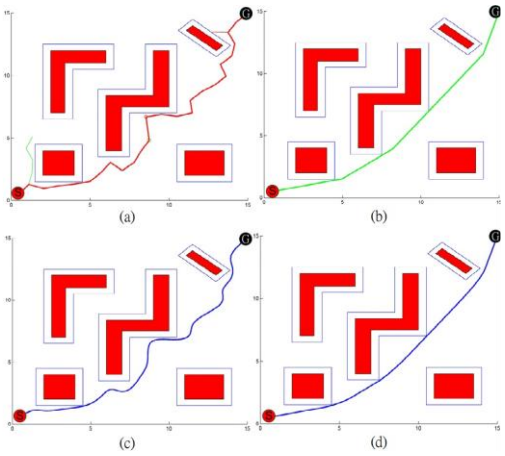

Figure 13. Results of the proposed method. (a)Bi-BRT; (b) Node deletion; (c) Smoothing; (d) Node deletion and smoothing.

We also evaluate these methods using different start and goal points. For example, Fig. 18 compares the results of the Bi-RRT algorithm and the proposed method. Figures (a), (c), (e), and (g) show the Bi-RRT algorithm results for different start and goal points, while Figs. (b), (d), (f), and (h) show the results for the proposed method, which clearly produces a much smoother and shorter path. Table 2 summarizes the performances of the Bi-RRT algorithm and the proposed method, and indicates that a shorter path length corresponds with a greater path distance savings.

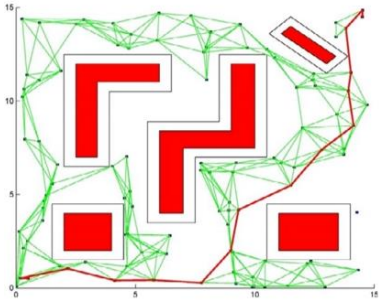

Figure 14. PRM method.

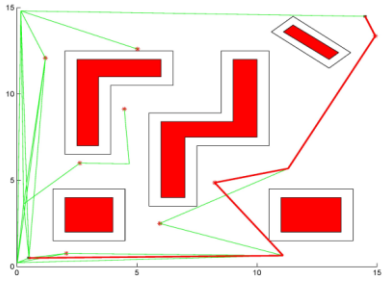

Figure 15. Visib method.

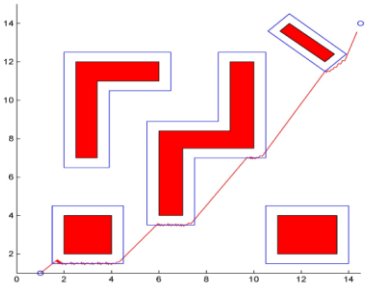

Figure 16. APF method.

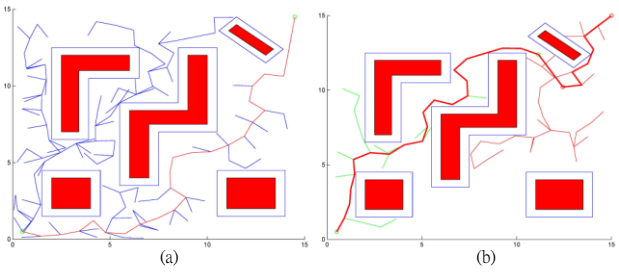

Figure 17. (a)RRT; (b) Bi-RRT methods.

Table 1. Comparison of the various methods in terms of elapsed time and path distance.

	Time avg. (s)	***Time std.*** (s)	***Dis. Avg.*** (m)	***Dis. Std.*** (m)
PRM	1.7271	0.8596	25.9114	2.3204
Visib-PRM	0.1918	0.03	24.7461	3.6822
APF	0.5920	0.0118	25.5	0
RRT	0.1773	0.1164	26.3579	3.3091
Bi-RRT	0.1121	0.0829	26.4011	2.29225
Proposed	0.1305	0.0650	22.4072	1.0268

Table 2. Comparison between the Bi-RRT algorithm and proposed method in terms of path distance savings percentage.

Start	***Goal***	***Bi-RRT***(m)	***Proposed***(m)	***Saving***(%)
(2,0.5)	(15,15)	28.1193	23.7999	15
(0,15)	(15,0)	26.9467	24.3597	10
(15,15)	(5,6)	23.5735	17.8451	24
(0,15)	(8,6)	27.9142	22.2582	20
(8,9)	(8,6)	20.5086	14.8266	28
(0,7.5)	(15,5)	21.2492	17.3337	18

Dynamic path planning

We tested the proposed method on dynamic obstacles in the same area in static path planning. The procedure for dynamic path planning is implemented by deformation (resampling and extension), node deletion and smoothing as described in Section III. Figures 19(a)-(d) show the results for dynamic path planning. The blue circles are two moving obstacles. Figure 19(a) shows the path found by the Bi-RRT algorithm. Once the moving obstacle is on the path, Fig. 19(b) shows the path deformation used to avoid the obstacle. To shorten the path, Fig. 19(c) shows the result of node deletion. Finally, after smoothing, Fig. 19(d) shows the path obtained by the proposed method.

Soccer simulation

We simulated a soccer game with two teams (blue and red), each with four robots. The field measures 6m by 4m. Figures 20(a)-(c) show the path from the blue striker to the ball. Figure 20(a) shows the result by the Bi-RRT algorithm. Figure 20(b) shows that the red players tried to block the path because they want to reach to the ball as well. However, Fig. 20(c) shows that the blue team dynamically re-planned a new path from the striker to the ball using the proposed method. Table 3 summaries the elapsed time and path distance in each step of the proposed method. The results shows that the proposed method shortens the path obtained by the Bi-RRT algorithm by 20(cm) only using 0.008(s). Instead, if the striker takes another single shot to re-plan the path using the Bi-RRT algorithm, it takes another 0.0040(s) to obtain a long and non-smooth path to the ball. However, from Table 3, the proposed method takes 0.0040(s) from the result of the Bi-RRT algorithm to obtain a smooth and short path. Thus the proposed method including the Bi-RRT algorithm, deformation, and node deletion is superior to the Bi-RRT algorithm for dynamic path planning.

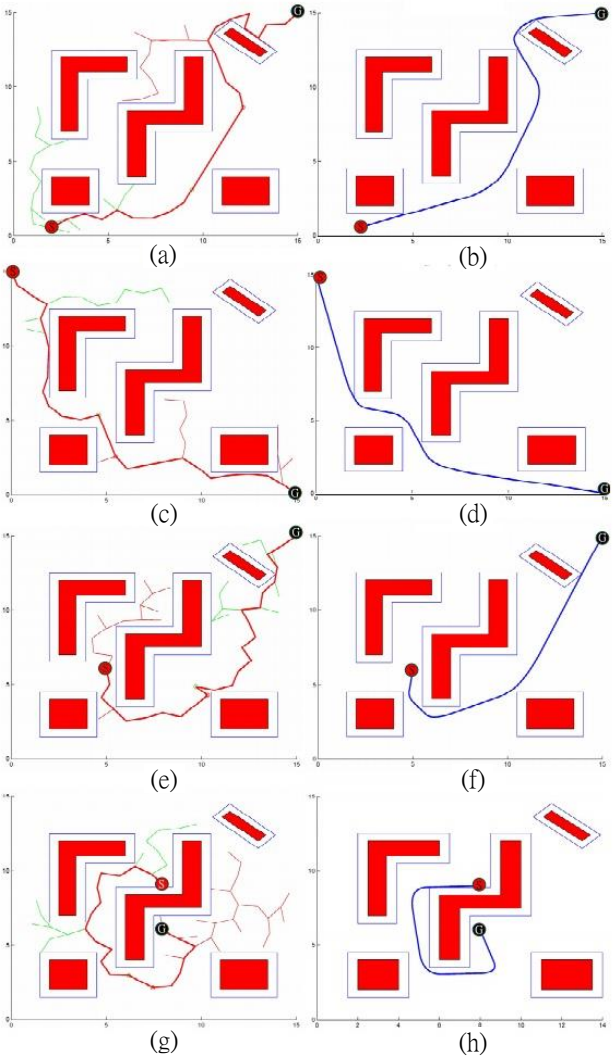

Figure 18. Different start and goal points. (a), (c), (e),and (g) are the results obtained by the Bi-RRT algorithm; (b), (d), and (h) are the results obtained by the proposed method.

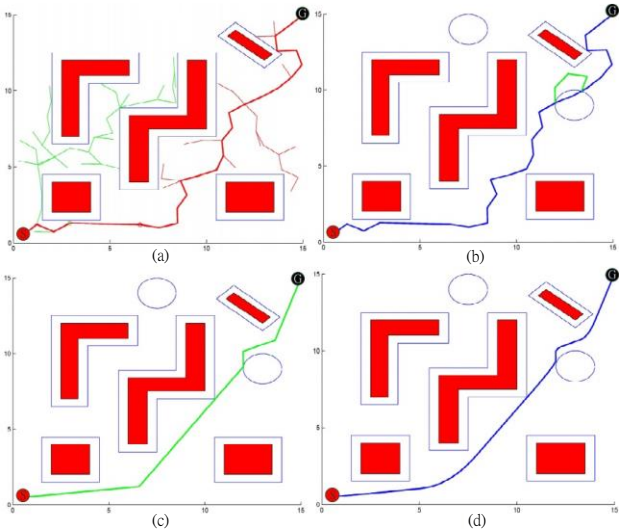

Figure 19. Dynamic path planning. (a) Result by the Bi-RRT algorithm; (b) Path deformation; (c)Node deletion; (d)Smoothing.

Table 3. Elapsed time and path distance in each step of the proposed method.

	Bi-RRT	***Bi-RRT*** + Deformation	***Bi-RRT*** + Node deletion	Proposed (***Bi-RRT***+ Deformation+ Node deletion)
Time(s)	0.0040	0.0068	0.0057	0.0080
Dis. (m)	3.2098	5.4179	3.0667	3.0206

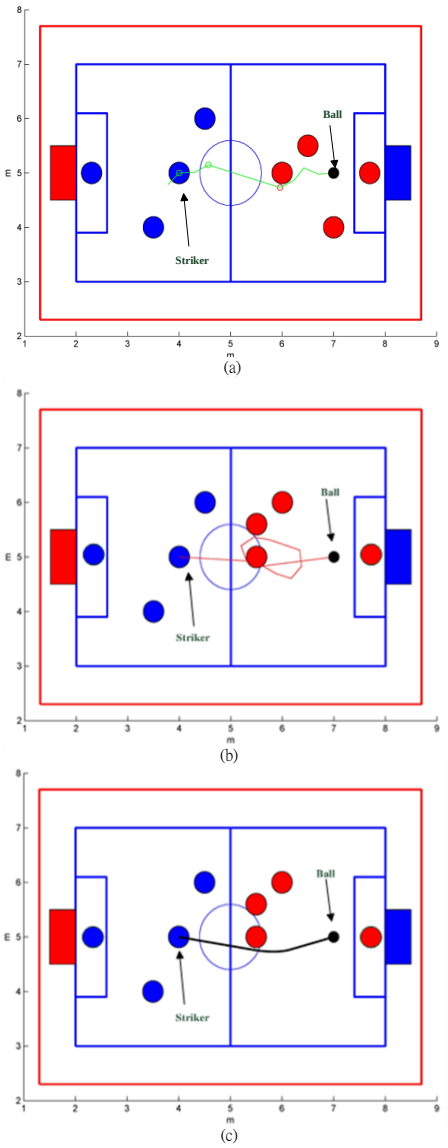

Figure 20. Soccer game simulation. (a) A planned path derived by the Bi-RRT algorithm; (b) the red team tries to block the path; (c) a new path is re-planned using the proposed method.

Conclusions

We propose an enhanced version of the RRT method to improve path planning in a robot soccer game. To reduce computational time and shorten path routes, we propose an enhanced RRT algorithm using a novel 2D-span resampling method and a pruning technique based on a b-spline function. The 2D-span resampling method enables the proposed algorithm to deform RRT to avoid dynamic obstacles. The density of the resampling nodes is configurable by the grid size of the 2D-span resampling method. Generating new nodes using the 2D-span resampling method entails a low computational cost, and the number of new nodes produced is twice that of the samples colliding with the obstacles. As a new path is generated, a pruning technique based on a b-spline function is applied to trim unnecessary random samples of RRT to shorten and smooth the path. The results show that the proposed method dramatically reduces path distance and computational time. The proposed method was implemented in the 2012 RoboCup Japan Open Standard Platform League.

In this work, the obstacle positions were assumed and used to compute the optimal router; however, the proposed method did not consider any discrepancy between the actual and assumptive obstacle positions. In addition, when the robot passed between two obstacles, router planning did not consider the safe margin. In the future, the robustness of the proposed method will be validated using various obstacle conditions.

References

[1] H. M. Choset, *Principles of Robot Motion: Theory, Algorithms, and Implementation (5th Ed.)*. Cambridge, Massachusetts: MIT, 2005.

[2] L. E. Kavraki, M. N. Kolountzakis, and J. C. Latombe, "Analysis of probabilistic roadmaps for path planning," *IEEE Transactions on Robotics and Automation,* vol. 14, no. 1, pp. 166–171, Feb 1998. doi: 10.1109/70.660866

[3] M. Morales, S. Rodriguez, and N. M. Amato, "Autonomous reaching and obstacle avoidance with the anthropomorphic arm of a robotic assistant using the attractor dynamics approach," in Proceeding of *IEEE Int. Conf. on Robotics and Automation,* New Orleans, LA, USA, April 26 - May 1, 2004, vol. 3, pp. 4427–4432.

[4] G. Song, S. Thomas, and N. M. Amato, "Autonomous reaching and obstacle avoidance with the anthropomorphic arm of a robotic assistant using the attractor dynamics approach," in Proceeding of *IEEE Int. Conf. on Robotics and Automation*, New Orleans, LA, USA, April 26 - May 1, 2004, vol. 3, pp. 4445–4450.

[5] C. Nissoux, T. Simeon, and J. P. Laumond, "Visibility based probabilistic roadmaps," in Proceeding of *IEEE Intelligent Robots and Systems Conf.*, Kyongju, South Korea, Oct. 17-21, 1999.

doi: 10.1109/IROS.1999.811662

[6] S. M. Lavalle, "Rapidly-exploring random trees: a new tool for path planning," Research Report, *Department of Computer Science,* Iowa State University, 1998.

[7] S. Rodriguez, X. Tang, J.-M. Lien, and N. M. Amato, "An obstacle-based rapidly-exploring random tree," in proceeding of *IEEE Int. Conf. on Robotics and Automation*, Orlando, FL, USA, May 15-19, 2006, pp.895-900.
doi: 10.1109/ROBOT.2006.1641823

[8] J. Nieto, E. Slawinski, V. Mut, and B.Wagner, "Online path planning based on rapidly-exploring random trees," in proceeding of *IEEE Int. Conf. on Industrial Technology (ICIT)*, Vina del Mar, Chile, March 14-17, 2010, pp. 1451-1456.
doi: 10.1109/ICIT.2010.5472492

[9] J. C. Latombe, *Robot motion planning*, Norwell, MA: Kluwer Academic Publishers, 1991.

[10] Y.-K. Hwang and N. Ahuja, "A potential field approach to path planning," *IEEE Transactions on Robotics and Automation*, vol. 8, no. 1, pp. 23-32, Feb 1992.
doi: 10.1109/70.127236

[11] L. Tang, S. Dian, G. Gu, K. Zhou, S. Wang, and X. Feng, "A novel potential field method for obstacle avoidance and path planning of mobile robot," in proceeding of *3rd IEEE Int. Conf. Computer Science and Information Technology (ICCSIT)*, Chengdu, China, July 9-11, 2010.
doi: 10.1109/ICCSIT.2010.5565069

[12] Y.-Q. Qin, D.-B. Sun, N. Li, and Y.-G. Cen, "Path planning for mobile robot using the particle swarm optimization with mutation operator," in proceeding of Int. Conf. on Machine Learning and Cybernetics, Shanghai, China, Aug. 26-29, 2004, pp. 2473–2478.
doi: 10.1109/ICMLC.2004.1382219

[13] E. Masehian and D. Sedighizadeh, "A multi-objective pso-based algorithm for robot path planning," in proceeding of IEEE Int. *Conf.* Industrial Technology, Vina del Mar, Chile, March 14-17, 2010, pp. 465-470.
doi: 10.1109/ICIT.2010.5472755

[14] I. K. Jung, K. B. Hong, S. K. Hong, and S. C. Hong, "Path planning of mobile robot using neural network," in proceeding of IEEE Int. Symposium on Industrial Electronics, Bled, Slovenia, July 12-16, 1999, pp. 979-983.
doi: 10.1109/ISIE.1999.796750

[15] G. Nagib and W. Gharieb, "Path planning for a mobile robot using genetic algorithms," *IEEE Proc. Robotics,* pp. 185-189, 2004.

[16] M.-Y. Ju and C.-W. Cheng, "Smooth path planning using genetic algorithms," in proceeding of *World Congress Intel. Control and Automat*, Taipei, June 21-25, 2011, pp. 1103–1107.
doi: 10.1109/WCICA.2011.5970687

[17] W.-G. Han, S.-M. Baek, and T.-Y. Kuc, "Genetic algorithm based path planning and dynamic obstacle avoidance of mobile robots," in proceeding of *IEEE Int. Conf. on Systems, Man, and Cybernetics*, Orlando, FL, USA, Oct. 12-15, 1997, pp. 2747-2751.
doi: 10.1109/ICSMC.1997.635354

[18] S.-C. Yun, S. Parasuraman, and V. Ganapathy "Dynamic path planning algorithm in mobile robot navigation," in proceeding of *IEEE Symposium on Industrial Electronics and Applications,* Langkawi, Malaysia, Sept. 25-28, 2011, pp. 364-369.
doi: 10.1109/ISIEA.2011.6108732

[19] Q. Cao, Y.-W. Huang, and J.-L. Zhou, "An evolutionary artificial potential field algorithm for dynamic path planning of mobile robot," in proceeding of *IEEE Int. Conf. on Intelligent Robots and Systems*, Beijing, Oct. 9-15, 2006, pp. 3331-3336, 2006.
doi: 10.1109/IROS.2006.282508

[20] L. Jaillet and T. Simeon, "A prm-based motion planner for dynamically changing environments," in proceeding of *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Sendai, Japan, Sept. 28-Oct. 2, 2004, vol. 2, pp. 1606–1611.
doi: 10.1109/IROS.2004.1389625

[21] D. Ferguson, N. Kalra, and A. Stentz, "Replanning with rrts," in proceeding of IEEE Int. Conf. on Robotics and Automation, Orlando, FL, USA, May 15-19, 2006, pp. 1243-1248.
doi: 10.1109/ROBOT.2006.1641879

[22] E. Yoshida, K. Yokoi, and P. Gergondet, "Online replanning for reactive robot motion: Practical aspects," in proceeding of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, Taipei, Taiwan, Oct. 18-22, 2010, vol. 2, pp. 5927-5933.
doi: 10.1109/IROS.2010.5649645

[23] E. Yoshida and F. Kanehiro, "Reactive robot motion using path replanning and deformation," in proceeding of IEEE Int. Conf. on Robotics and Automation, Shanghai, China, May 9-13, 2011, vol. 2, pp. 5456-5462.
doi: 10.1109/ICRA.2011.5980361

[24] T. Rofer, T. Laue, J. Muller, A. Fabisch, F. Feldpausch, K. Gillmann, C. Graf, T. J. de Haas, A. Hartl, A. Humann, D. Honsel, P. Kastner, T. Kastner, C. Konemann, B. Markowsky, O. J. L. Riemann, and F. Wenk, "B-human team report and code release," Nov. 2011.

[25] J. J. Kuffner and S. M. LaValle, "Rrt-connect: An efficient approach to single-query path planning," in proceeding of IEEE Int. Conf. on Robotics and Automation, San Francisco, CA, USA, April 24-28,

2000, pp. 995-1001.
doi: 10.1109/ROBOT.2000.844730