Fast Path Planning Algorithm for the RoboCup Small Size League

Saith Rodríguez[1], Eyberth Rojas[1], Katherín Pérez[1], Jorge López[1], Carlos Quintero[1], and Juan Calderón[1,2]

[1] Universidad Santo Tomás, Colombia
{saithrodriguez,eyberthrojas,andrea.perez,jorgelopez,
carlosquinterop,juancalderon}@usantotomas.edu.co
[2] University of South Florida, Tampa, FL, USA
juancalderon@mail.usf.edu

Abstract. Plenty of work based on the Rapidly-exploring Random Trees (RRT) algorithm for path planning in real time has been developed recently. This is the most used algorithm by the top research teams in the Small Size League of RoboCup. Nevertheless, we have concluded that other simpler alternatives show better results under these highly dynamic environments. In this work, we propose a new path planning algorithm that meets all the robotic soccer challenges requirements, which has already been implemented in the STOx's team for the RoboCup competition in 2013. We have evaluated the algorithm's performance using metrics such as the smoothness of the paths, the traveled distance and the processing time and compared it with the RRT algorithm's. The results showed improved performance over RRT when combined measures are used.

Keywords: Path Planning, mobile robots, real-time systems, RoboCup

1 Introduction

Collision-avoidance path planning has been a major challenge for robotics researchers. Different proposals have been made to address individual, but contrasting, requirements, such as following the shortest or smoothest trajectory or minimizing processing time. Our system has the goal to quickly find a smooth path with a low computational cost without ensuring that it is the shortest trajectory.

RoboCup has reveal a strong and rapid need to develop efficient path planning algorithms for complex environments. It is a global initiative in which researchers around the world present their best developments in the topics of robotics, artificial intelligence and related areas[1]. Based on the RoboCup initiative, every year many tournaments are held in different countries around the world, therefore teams participate in various disciplines [2].

For the case of the Small Size League (SSL) [3], the challenge is a soccer contest in which full autonomous robots are able to cooperate to score and win

a match. Its artificial vision system sends field images at a rate of 60 fps [4], so path planning and intelligence processing must be made within the span of $16ms$. Thus, the challenge involves a highly dynamic multi-agent environment which implies the need for obstacles avoidance and fast path planning algorithms.

Initially, we present a short review of related work, specifically describing the RRT [5] algorithm, some applications and results that this algorithm generates in situations with several obstacles. After that, we describe in detail the proposed algorithm and we suggest a set of benchmarking scenarios in the context of this league, including real game situations from the RoboCup 2013 competition.

Then, we show the results of our performance analysis for both algorithms over the proposed scenarios. We evaluated specific attributes such as path smoothness, distance traveled and the processing time. Finally, we include conclusions and future work sections.

2 Related Work

The problem of path planning under dynamic environments has been tackled by a variety of researchers. Most of them have focused on optimizing specific performance measures such as obtaining smooth paths in the less amount of time as possible. For example, Tsubouchi et. al. analyze the behavior of a single robot within a multi-obstacle dynamic environment [6]. The navigation scheme in this work is based on a heuristic and assumes that obstacles move with a piecewise constant velocity.

On the Small Size League challenge, the robotic players have omnidirectional traction and the dynamic characteristic of the environment have to be carefully taken into account for a proper navigation. Han et. al. studied the control of multiple non-holonomic robotic agents in which half the obstacles are non-controllable opponents whose dynamic patterns are unknown [7]. The algorithm in this work creates a set of halfway points between the initial robot position and its final destination. These points are calculated based on the evaluation of potential blockages in the route.

Kuffner et. al. presented an algorithm based on Rapidly-exploring Random Trees (RRT) [8]. This algorithm is specifically suited to overcome the constraints that arise in dynamic environments. RRT creates the path that should be followed by an agent from its initial position to a target point by iteratively building search trees that quickly explore the environment. The general procedure is divided in 5 main processes as shown in Fig. 1. However, we have found that this approach can still be improved with regards to computational cost and path smoothness, in contexts like the RoboCup SSL environment.

The result of applying the RRT algorithm to an environment with a set of obstacles is shown in Fig. 2. Where the green region represents the obstacles, the white region is the obstacle-free configuration space, blue branches are the RRT, and the black line is the solution path between the starting configuration (blue) and the goal configuration (red).

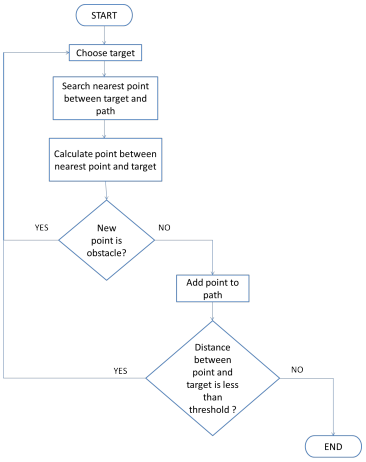


Fig. 1. Diagram RRT

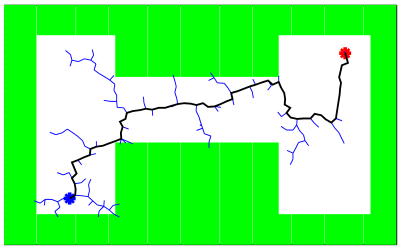


Fig. 2. Example RRT

The RRT algorithm has been widely embraced in a variety of applications in different environments. This has been especially true for the teams that participate in the small size league of the RoboCup initiative. For instance, Bruce et. al. showed a RRT-based planning system in simulation and its implementation on actual robots[9]. This algorithm is presented as ERRT and aims at reducing the cost of searching the nearest point to the path that is being build when compared to the original RRT. This feature increases the efficiency of the path planning procedure for real time applications even in dynamic and continuous environments.

Desaraju et. al. presented a Decentralized Multi-Agent Rapidly-exploring Random Tree (DMA-RRT) algorithm [10]. This approach allows to perform an efficient planning by considering complex environments. It uses a coordination strategy to dynamically update the order in which the robots carry out their individual planning.

3 Proposal

After having reviewed path planning algorithms for dynamic environments, we have implemented RRT and validated it in simulated game situations. After that, we have measured the time taken by the algorithm to generate paths in complex situations. Finally we have proposed a new planning algorithm that is capable of dramatically reducing the time required to create the path and compared it with the RRT algorithm. Both are validated in real game situations of the RoboCup 2013.

The proposed algorithm is based on generating straight trajectories between an initial state and a goal state. To accomplish this, an initial straight trajectory between these points is defined and checked for collisions against all obstacles. If there is no obstacle, the selected route is returned, as shown in Fig. 3. Otherwise, a subgoal state is generated to avoid the obstacle. As a consequence, the original trajectory is split in two: one between the initial state and the subgoal, and another one between the subgoal and the goal state. Then, these new trajectories are recursively evaluated until the algorithm finds an obstacle-free path. Below we show the developed algorithm:
Pseudocode of the proposed path planning

```
function FastPathPlanning (environment,trajectory,depth)

  obstacle = trajectory.Collides(environment)

  if theres is an obstacle and depth < max_recursive then
  {
    subgoal=SearchPoint(trajectory,obstacle,environment);

    trajectory1=GenerateSegment(trajectory.start,subgoal);
    trajectory1=FastPathPlanning(environment,trajectory1,depth+1);
    trajectory2=GenerateSegment(subgoal,trajectory.goal);
    trajectory2=FastPathPlanning(environment,trajectory2,depth+1);
    trajectory=JoinSegments(trajectory1,trajectory2);
  }

  return trajectory;
```

The function *Collides* determines if there is any obstacle in the trajectory and if this is the case, the function returns the position of the closest obstacle in the trajectory.

The function *SearchPoint* assigns a new point (subgoal) at the side of this obstacle. This point is located from the obstacle to a distance equal to the robot diameter and 90 or -90 degrees related to the path between the initial point to the obstacle (the sign is a function parameter). See Fig. 4a. In the case the new point collides with an obstacle, the function keeps moving the point one robot diameter in the same direction, until an obstacle free point is found.

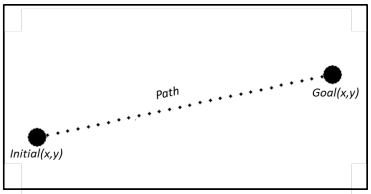


Fig. 3. Path without obstacles

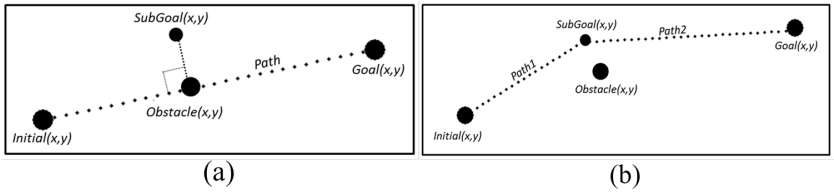


Fig. 4. (a) Subgoal Selection. (b) New paths created with subgoal

The function *GenerateSegment* generates a straight path between two points. It is used to create two new paths. The first one between the initial state and the subgoal and the second one between the subgoal and the goal state. These new paths will be analyzed recursively as shown in Fig. 4b. Fig. 5 shows all the steps of execution until an obstacle free path is found.

Finally, the returned paths should be joined; this is performed by the function *JoinSegments*. It returns the path to reach the target point avoiding obstacles. Fig. 6 shows an example of a game situation in which the algorithm performed recursion twice.

As we mentioned before, two possible subgoal points may be returned by the function *Collides*, one at 90 degrees and another at -90 degrees. This means that the obstacles will always be avoided in the same direction, either always in the 90 degree direction, or in the other one. In order to optimize the path length, both options are tested and the shortest path is chosen.

The Fig. 7a shows the two possible paths found by the algorithm in a scenario with multiple obstacles. Fig. 7b shows a continuous line for the trajectory to be followed.

Finally, at Fig. 8 we present the solution found by the algorithm in a sample random scenario with multiple obstacles.

4 Results and Evaluation

The evaluation and comparison of the algorithms involved three different metrics. Namely, processing time required for the path generation, path smoothness and total path length. Additionally, a weighted sum evaluation (Eq.1) of the trajectories was made according to Xiao's proposal [11] with some modifications, where W_t,W_s and W_d are the weights assigned to each criterion according

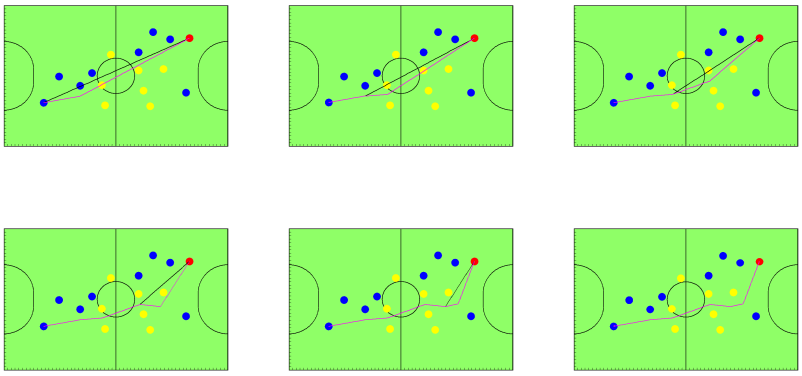


Fig. 5. The figures show the recursive steps of the algorithm. At each step, the black line shows the computed straight path and the magenta segments denote the alternative route to avoid the found obstacle. After computing an alternative route, the algorithm is applied over it, recursively.

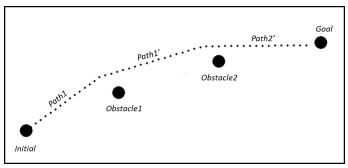


Fig. 6. Example of the proposed algorithm with recursion

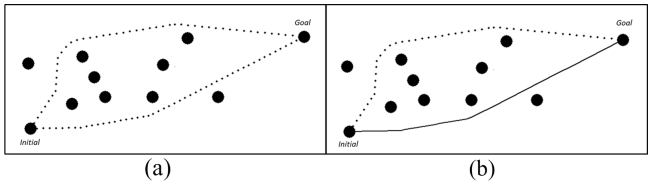


Fig. 7. (a) Possible paths (b) Path selection

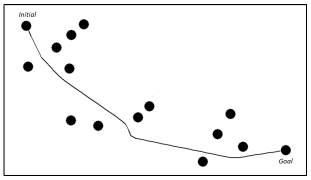


Fig. 8. Solution found by the algorithm in a random scenario with multiple obstacles

to the desired relevance of processing time, smoothness and distance traveled respectively.

$$Eval(p) = W_t time(p) + W_d dist(p) + W_s smooth(p) \quad (1)$$

We define *time*, *smooth* and *dist* as:

- $time(p)$, processing time
- $dist(p) = \sum_{i=1}^{n-1} d(m_i, m_{i+1})$, total path length, where $d(m_i, m_{i+1})$ is the distance between two adjacent nodes m_i and m_{i+1}.
- $smooth(p)$ measures the smoothness of the path, defined as

$$smooth(p) = \frac{\sum \theta_i}{dist(p)}$$

where θ_i is the angle between adjacent segments of the route.

According to Eq.1, smaller $Eval(p)$ values represent better performance measures. In addition, for the case of Small Size League, we assign more relevance to processing time $W_t = 0.5$, next in significance the smoothness $W_s = 0.3$ and finally the path length $W_d = 0.2$. As *time*, *smooth* and *dist* variables must be normalized, we normalized each variable using the higher value obtained for it from both RRT and the proposed algorithm.

The algorithm evaluation process was performed in two parts. The first one involves static random scenarios and second one includes real game dynamic environments taken from the RoboCup 2013 contest.

4.1 Part 1: Static scenario

We consider two different game cases and for each one we generate 100 paths using the RRT algorithm. From these paths, we take average values of each one of the three criteria and then we compare with the trajectory generated by the proposed algorithm in the same case. We did not run our algorithm several times, since it would generate the same result every time, as it is not randomized.

Case 1 Fig. 9a shows the trajectories generated by both algorithms in the shortest time. According to criterion 1 (processing time), the path generated the faster by RRT algorithm took less time than our proposal's. Fig. 9b shows processing time for the 100 trajectories generated by RRT (green), their average (blue) and the time spent by the proposed algorithm (red). The obtained results are shown in Table 1.

Regarding the second criterion (smoothness of the path), Fig. 10 and Table 2 exhibit the results obtained for the same case. Last, in Figure 11 and Table 3 we show the results according to the third criterion (path length).

The performance evaluation of this case is shown in Table 4. The path generated by the proposed algorithm is 3.4 times better than the average of those generated by RRT.

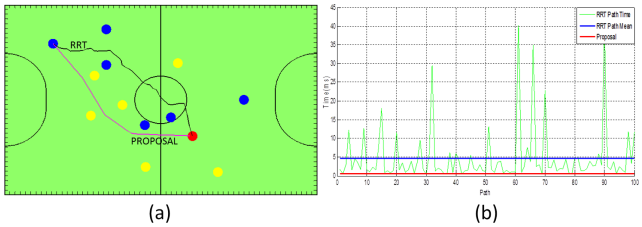


Fig. 9. (a) Proposal and RRT algorithm (Path that took less time to create). (b) Results Case 1, Criterion 1 (Process time)

Table 1. Case 1 results: Processing time analysis

CASE 1, CRITERION 1 (time)			
ALGORITHM	**MINIMUM(ms)**	**MAXIMUM(ms)**	**MEAN(ms)**
RRT	0.4223	40.3556	4.7193
PROPOSAL	0.5303	0.5303	0.5303

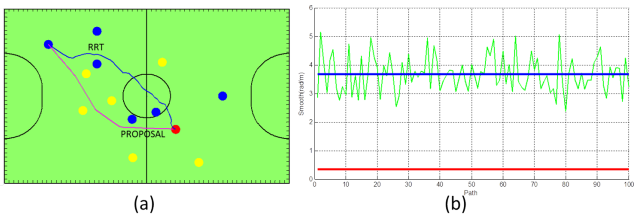


Fig. 10. (a) Proposal and RRT algorithm (smoother path). (b) Results Case 1, Criterion 2 (path smoothness)

Table 2. Case 1 results: smoothness analysis of path

CASE 1, CRITERION 2 (smoothness)			
ALGORITHM	**MINIMUM(rad/m)**	**MAXIMUM(rad/m)**	**MEAN(rad/m)**
RRT	2.378	5.148	3.668
PROPOSAL	0.3447	0.3447	0.3447

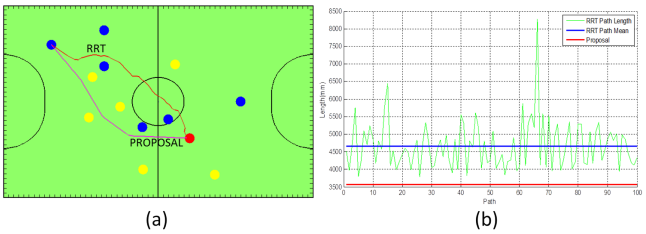


Fig. 11. (a) Proposal and RRT algorithm (shortest path). (b) Results Case 1, Criterion 3 (Path length)

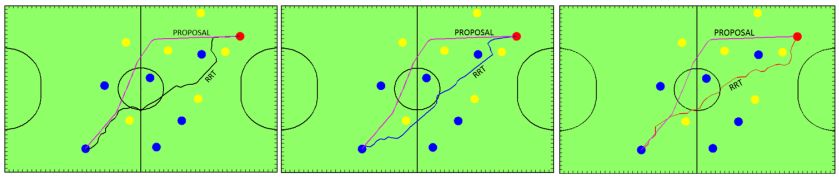
Table 3. Case 1 results: path length analysis

CASE 1, CRITERION 3 (length)			
ALGORITHM	**MINIMUM(mm)**	**MAXIMUM(mm)**	**MEAN(mm)**
RRT	3780	8267	4667
PROPOSAL	3569	3569	3569

Table 4. Case 1 results: Weighted performance evaluation

CASE 1, EVALUATION				
ALGORITHM	**TIME(p)**	**SMOOTH(p)**	**DIST(p)**	**EVAL(p)**
RRT	0.116	0.712	0.564	0.385
PROPOSAL	0.013	0.066	0.431	0.113

Case 2 Fig. 12 shows the second case, with a) the fastest generated path by RRT, b) the smoothest and c) the shortest in length. All the results for this case are shown in Table 5.



(a) (b) (c)

Fig. 12. Case 2. (a) Fastest generated path. (b) Smoother path. (c) Shortest path

The evaluation of this case is shown in Table 6. We observe that for this game situation, the proposed algorithm is 3 times better than RRT.

4.2 Part 2: Dynamic Scenario

For this case, we took data from a segment of one game of RoboCup 2013 (STOx's vs CMDragons). There are 281 continuous scenarios in total and for each one we generated trajectories using both RRT and the proposed algorithm. Full data of this match can be obtained from [12] and a video showing the segment can be found in [13].

The obtained results are shown in Fig. 13. This graphic presents a) processing times for the 281 trajectories for both algorithms, b) the smoothness and c) distance traveled. Table 7, consolidates results obtained from both algorithms.

Finally, Table 8 shows the evaluation of both algorithms for the dynamic scenario and results indicate that the average of the proposed algorithm performance is 2.4 times better than RRT's.

Table 5. Case 2 Results

CASE 2				
CRITERION	**ALGORITHM**	**MINIMUM**	**MAXIMUM**	**MEAN**
TIME(ms)	RRT	1.710	68.3259	17.7726
	PROPOSAL	0.6523	0.6523	0.6523
SMOOTH(rad/m)	RRT	2.143	4.131	31.708
	PROPOSAL	0.3234	0.3234	0.3234
DIST(mm)	RRT	4842	7647	5776
	PROPOSAL	4883	4883	4883

Table 6. Case 2 Results: Evaluation

CASE 2, EVALUATION				
ALGORITHM	**TIME(p)**	**SMOOTH(p)**	**DIST(p)**	**EVAL(p)**
RRT	0.260	0.768	0.755	0.511
PROPOSAL	0.010	0.078	0.639	0.156

Table 7. Dynamic Scenario Results

DYNAMIC SCENARIO				
CRITERION	**ALGORITHM**	**MINIMUM**	**MAXIMUM**	**MEAN**
TIME(ms)	RRT	1.16	35.595	7.891
	PROPOSAL	0.052	15.051	1.494
SMOOTH(rad/m)	RRT	1.398	6.178	2.546
	PROPOSAL	0.001	0.871	0.149
DIST(mm)	RRT	6062	9445	8310
	PROPOSAL	6344	7199	6728

Table 8. Dynamic Scenario Evaluation

CASE 1, EVALUATION				
ALGORITHM	**TIME(p)**	**SMOOTH(p)**	**DIST(p)**	**EVAL(p)**
RRT	0.222	0.412	0.880	0.410
PROPOSAL	0.042	0.024	0.712	0.171

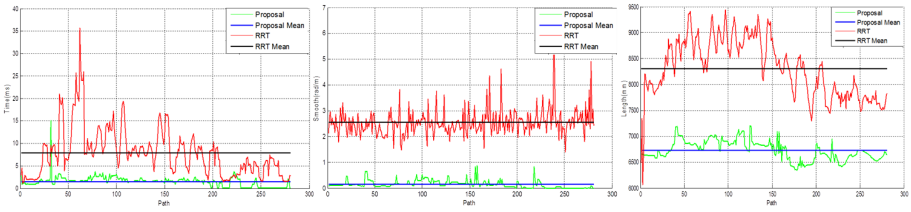


Fig. 13. Dynamic Scenario. (a) Time results. (b) Smooth results. (c) Dist results.

5 Conclusions

We consider RRT to be a robust and flexible algorithm when applied to path planning. However, we observe that its generality can turn into its own weakness, when aspects like processing time or path smoothness are critical. Thus, we have proposed an ad-hoc heuristic for path planning in non-cluttered dynamic environments. We have tested our approach over a set of artificial and real RoboCup Small Size League situations, and we have found it finds smoother and shorter paths faster in the average case.

It is clear that the good performance of the proposed algorithm is related with the constraints of the environment where we apply it. However, we claim that the characteristics of the SSL league with regards to path planning are similar to many other planning situations, e.g. autonomous car driving.

When testing the proposed heuristic, we have found that it outperforms the RRT implemented version by approximately 30% on average, using the proposed metrics. Additionally, we have also observed that the proposal outperforms RRT in all three individual metrics, i.e. path length, smoothness and processing time.

After competition in RoboCup 2013, we can assert that the proposed algorithm generated paths avoiding obstacles and also processed all necessary data in real time, which was crucial to be among the top eight teams in the SSL.

6 Future Work

We propose to test the new algorithm on more complex environments to assess the validity of our claims regarding the applicability of our heuristic.

According with the outcomes of these tests, we will modify algorithm to make it more robust.

We also acknowledge the need to compare this algorithms with newer and improved versions of RRT-based path planners. Future work will include these comparisons.

Integrating the robots dynamics information into the planning process is also our priority. We believe that this is crucial to close the gap between the planned paths and the actual navigated ones, as well as to reach optimal plans.

Acknowledgements

This work has been funded by "Octava Convocatoria Interna de Proyectos de Investigación FODEIN 2014 #049" at Universidad Santo Tomás Colombia, entitled "Construcción de un enjambre de robots omnidireccionales". Many thanks to Martin Llofriu for his comments and suggestions on the work presented here.

References

1. Kitano, H., Asada, M., Kuniyoshi, Y., Noda, I., Osawa, E.: RoboCup: The robot World Cup initiative. In: Proceedings of IJCAI-95 Workshop on Entertainment and AI/Alife, pp. 340–347. IEEE Press, Montreal (1995)
2. RoboCup Information, http://www.robocup2013.org/about-robocup/
3. Small Size Legue Information, http://robocupssl.cpe.ku.ac.th/rules:main
4. Ould-Khessal, N.: Botnia: A Team of Soccer Plating Robots. In: 2nd International Conference on Autonomous Robots and Agents, pp. 429–433. Palmerston North (2004)
5. LaValle, S.M. Rapidly-exploring random trees: A new tool for path planning. Technical Report No. 98-11 (1998)
6. Tsubouchi, T., Arimoto, S.: Behaviour of a mobile robot navigated by an iterated forecast and planning scheme in the presence of multiple moving obstacles. In: 1994 IEEE International Conference on Robotics and Automation, pp. 2470–2475. IEEE Press, San Diego (1994)
7. Han, K., Veloso, M.: Reactive Visual Control of Multiple Non-Holonomic Robotic Agents. In: 1998 IEEE International Conference on Robotics and Automation, pp. 3510–3515. IEEE Press, Leuven (1998)
8. Kuffner, J.J., LaValle, S.M.: Rapidly-Exploring Random Trees: Progress and Prospects. In: Donald, B.R., Lynch, K.M., Rus, D. (eds.) Algorithmic and Computational Robotics: New Directions, pp. 293–308. AK Peters, Massachusetts (2001)
9. Bruce, J., Veloso, M.: Real-Time Randomized Motion Planning for Multiple Domains. In: Lakemeyer, G., Sklar, E., Sorrenti, D.G., Takahashi, T. (eds.) RoboCup 2006 Robot Soccer World Cup X. LNCS, vol. 4434, pp. 532–539. Springer, Heidelberg (2006)
10. Desaraju, V.R., How, J.P.: Decentralized path planning for multi-agent teams in complex environments using rapidly-exploring random trees. In: 2011 IEEE International Conference on Robotics and Automation, pp. 4956–4961. IEEE Press, Shanghai (2011)
11. Xiao, J., Michalewicz, Z., Zhang, L., Trojanowski, K.: Adaptive Evolutionary Planner/Navigator for Mobile Robots. IEEE Transactions On Evolutionary Computation. 1, 18–28 (1997)
12. Log of Quaterfinal4 Small Size League RoboCup 2013 http://er-force.de/gamelogs/robocup2013/2013-06-29-165351_stoxs_cmdragons.log.gz
13. STOx's Team webpage. http://www.stoxs.org/index.php/en/projects/robocup-ssl-2/86-english-categories/stox-s-english/168-fast-path-planning-algorithm-en