Improved Path-Finding Algorithm for Robot Soccers

Rahib H. Abiyev, Nurullah Akkaya, Ersin Aytac, Irfan Günsel, and Ahmet Çağman
Applied Artificial Intelligence Research Centre, Near East University, Lefkosa, North Cyprus, Turkey
Email: {rahib.abiyev, nurullah}@neu.edu.tr

***Abstract*—In robot soccer game mobile robot moves in complex, unpredictable, unknown dynamic environment. In such environment the design of efficient path finding algorithm that will find the path in a short time becomes important problem. The widely used path finding methods are potential field method, vector field histogram and A* algorithms. In real time operations finding the optimal path in a reasonable short time is not possible with the mentioned methods. The RRT (Rapidly Exploring Random Tree) algorithm finds a path in short time. But some times the length of path may be very long. This paper is devoted to the design of improved path finding algorithm that will find the near optimal path in a short time. The improved iterative RRT with path smoothing algorithm is developed and implemented in RoboCup Small Size robots. Through simulation it was shown that this algorithm can efficiently find desirable and near optimal solutions in short time.**

***Index Terms*—path finding, navigation, iterative RRT-smooth, robot soccer**

I. Introduction

In robot navigation the basic requirements for mobile robots is the ability to move to the goal, avoiding hazards and obstacles. Finding the collision free, near optimal path in a crowded environment in a short time is one of the greatest problems of path finding. The navigation algorithm must be able to determine whether there is a continuous motion from one configuration to the other, and find such a motion if one exists. The main goal of the navigation algorithm is to guide the robot to the goal point without colliding with the both static and dynamic obstacles. Many efforts have been paid in the past to develop various wheeled robot navigation algorithms. The wheeled robot navigation algorithms include a set of algorithms. These are path planning, path smoothing, and obstacle avoidance algorithms.

Given a map and a goal location, the path finding is used to determine a short route from robot's current coordinate location to another goal location along a set of waypoints. Path smoothing allows optimise the given path of robot using the local environment information. In fact, in mobile robots operating in unstructured environments, the a priori knowledge of the environment is usually absent or partial. Many times, the environment where robot moves is not static, i.e., during the robot motion it can be faced with other robots or obstacles, and execution is often associated with uncertainty. In most cases the environment is characterised with uncertainty, fast-changing dynamic areas with many moving obstacles. For a collision free motion to the goal, the path planning has to be associated with a local obstacle handling that involves obstacle detection and obstacle avoidance. There have been developed numbers of methodologies for robot navigation using path planning and obstacle avoidance. The most commonly used are the methods based on the use of Artificial Potential Fields (APF) [1], Vector Field Histogram (VFH) Technique [2], VFH+ Technique [3], Dynamic Window Approach [4], Agoraphilic Algorithm [5], Rule Based Methods [6], A star [7], fuzzy navigation [8], [9] etc

The use of the above methods for path finding requires a lot of time and the finding of this path will not complete in reasonable time for real-time operation. The search of the path in a dynamic environment is important. In this paper the quick and feasible path finding problem is taken in hand. Rapidly-exploring Random Trees (RRTs) algorithm developed by Kuffner and LaValle can be efficiently used for robot navigation in a dynamic environment [10]. RRT Connect works by creating trees starting at the start and the goal configurations. The trees each explore space around them and also advance towards each other through the use of a greedy heuristic. This efficiently solves the path planning problem, even in high dimensions it results in shorter time than the search methods given above. But in some cases the RRTs does not find a short feasible path, and some times the determined path becomes very long. Few algorithms have been developed in [11]-[16] to improve RRT. In [14], [15] authors by increasing sample size try to optimize the path length. The authors propose an extension to RRT algorithm called RRT* and the rapidly-exploring random graph (RRG) algorithm. Using results from random geometric graph theory, these methods retain the asymptotic computation complexity of RRTs under certain assumptions. The algorithm does large number of iteration in order to find optimal path, which needs more computation time. In [16] authors proposed algorithm, that refines the explored space by adding edges to the current roadmap to enable finding higher quality paths. The algorithm minimizes the number of collision checks while still retaining the optimality guarantees.

Manuscript received July 14, 2014; received November 30, 2014.

doi: 10.12720/joace.3.5.398-402

There are clearly a number of robust techniques for various key sub-problems in robot navigation. However, there is still no known technique or combination of techniques which will result in a robust, generalised performance. One of alternative and powerful ways of constructing efficient navigation system is the design of hybrid algorithm that has been proposed in the paper. In this paper the improved efficient algorithm that iteratively uses RRT algorithm with path smoothing procedure to find acceptable path of robot.

II. Design of the Robot

The mobile soccer robots and their navigation system are designed, manufactured and assembled in our research laboratory [17]. To design soccer robots we used holonomic robots that have holonomic wheels with 3 degrees of freedom. Fig. 1 depicts the designed robot soccer. The robot soccer has four omni-wheels with a diameter of 61mm. Wheel orientation is dramatically effects the robot speed and acceleration. The wheel orientation of the robot is 33 degrees with the horizontal axis in front wheels and 45 degrees with the horizontal axis in rear wheels. Designing the omni-directional wheels and determining the diameters of both the wheels and the roller is one of the critical points. The robot omni-wheels are connected to brushless DC motors and controlled by brushless DC motor drivers called Electronic Speed Controllers (ESC). The motors of the robot soccer are rotating the wheels through gear mechanisms. Motor drivers are driven by a microcontroller. Each robot has a micro-controller board that controls control circuits of all motors and a control circuit of kicking mechanism. Microcontroller is connected to the computer via wireless link. By changing the speed of the individual omni-wheels we can control direction, rotation and speed of the robot. Omni-wheels allow movement in any direction, at any angle, without rotating beforehand. For an omni-wheel robot to translate at a certain angle, each motor needs to go at a certain speed in relation to the other motors. They are able to vary each component of their position and orientation independently. They are able to turn on the spot, and move in any direction regardless of orientation.

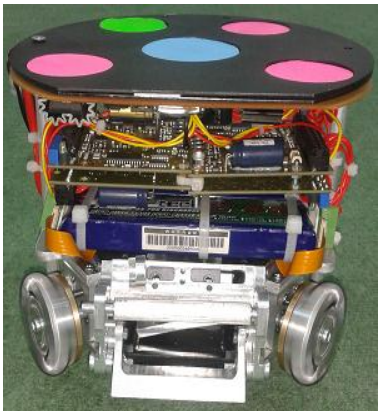


Figure 1. Omni-wheel robot

The environment where soccer robots move in is characterised with fast changing dynamic areas with moving dynamic obstacles. Collision free navigation of holonomic robot in such dynamic environment is difficult and very important. The structure of the robot soccer control system designed in this paper is given in Fig. 2. In the paper the omni-wheel robots are designed as soccer robots. Computer tracks the world using high speed overhead cameras. All objects on the field are tracked by a standardized vision system that processes the data provided by two cameras that are attached to a camera bar located 4 meters above the playing surface. SSL vision by using the cameras processes the map of the world and obtains the coordinates of soccer robots and balls then sends them to the computer/s. SSL-Vision uses a standard cartesian coordinate system in meters and radians. We accept that the centre of the soccer field is (0, 0). x is horizontal axis, y is vertical axis with headings given in radians. All communication from the SSL-Vision system to the clients is performed via network by UDP Multicast. In order to track the world in real time, the data is sent every 1/60 of a sec. Tracker module captures this data stream off the network and converts it into a data structure which will be used by decision making (DM) module. DM block using this data, makes strategic planning of soccer robots. Using the current coordinates of soccer robots and goal, the new path for each robot is determined. By controlling the speed of motors the control of movement of robots are performed.

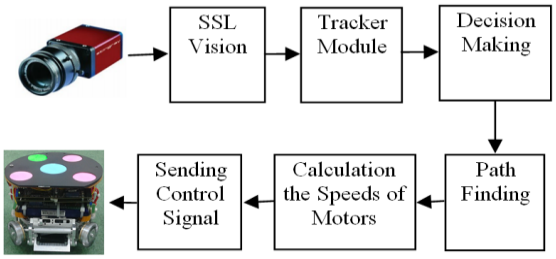


Figure 2. Structure of the designed control system of the Omni-wheel robots

III. Path Finding Procedure

In mobile robot navigation one of more used fast path finding algorithm is RRT algorithm [10]. The RRT algorithm is designed for efficiently searching nonconvex, high-dimensional search spaces. The RRT will search for a path from the start state to the goal state by expanding the search tree. The key idea of RRT algorithm proceeds by growing a single tree from the initial configuration until one of its branches encounters the goal state. Algorithm attempts to extend the RRT by adding a new vertex that is biased by a randomly-selected state. Fig. 3 demonstrates RRT planning algorithm. As shown in the algorithm, the inputs for RRT are map of the environment, start and goal position, RRT tree and the amount it is expanded at each step. The RRT algorithm is extremely simple and but it is not optimal. A path will be computed quickly but it is not guaranteed to be optimal and will results a different path for every search. In robot navigation the determination of shortest path in a short time is very important. The RRT algorithm is not optimal and contains many zig zags and unnecessary edges. In

order to deal with this problem, we extend the algorithm with proposing two new procedures. The first one is related to the extension of RRT. In this case we start to run RRT algorithm from two points, the first is the source point where robot is located, second one is the goal. In the results of two runs select more near optimal path. The second approach is the use of a simple path smoothing algorithm in RRT. The propose procedures is not too time consuming. In the paper extended RRT with quick path smoothing described is applied to optimize the selected path on the map. Fig. 4 depicts the fragment of the result of application of path smoothing algorithm to path obtained by the RRT. Given two nodes that are reachable A and B. Assume that A is start point of the path Fig. 4. This algorithm removes any nodes between A and B since we can go from A to B directly without going through all the nodes in between. A dashed line depicts the original path that robot try to get goal position, solid line demonstrates the optimal smoothed path. In the result of smoothing the path is optimized.

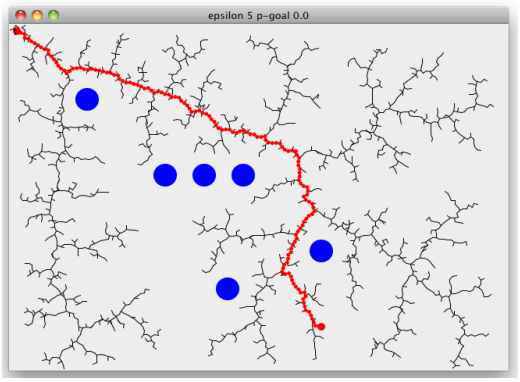


Figure 3. RRT-Plan algorithm when *pGoal*=0;

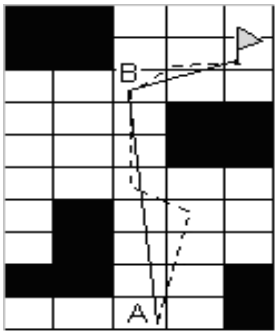


Figure 4. Smoothed optimal path. Dashed line- original path, curved line- smoothed optimal path

Using RRT algorithm and path smoothing procedure the iterative RRT-Smooth algorithm has been designed. At first stage using the coordinates of the robot and goal the rectangular area with certain width is defined. The value of width is determined according the size of search area. The RRT algorithm is run within this rectangular area (Fig. 5). If the path found then for this path the smooth algorithm is run. In other case the width of selected area is increased two times and for this area the process is repeated. The iteration is repeated until the acceptable path found. During the run of the RRT algorithm, at the first stage, "chooseTarget" determines where to explore the map by randomly selecting a point on the map giving bias towards the goal, with probability pGoal, that expands towards the goal minimizing the objective function of distance (Fig. 6). Here, using random number generator, uniformly points are generated to choose a target. Then algorithms determine nearest node using "nearest" procedure. If the distance between the node and target position is less than the distance between generated point and target position then the generated point is selected as new node where tree will be explored. In each iteration when the new node is selected, the distance between this nearest point and goal position is tested. If this distance is less than the required small value epsilon then the tree is returned as result of RRT algorithm. In other case tree is extended and explored. During extension of the tree the presence of obstacles are tested. In the case of presence of obstacles the tree is not extended toward to that direction. In other case the selected tree will be extended towards the chosen target and RRT algorithm will be iterated until goal position is reached (Fig. 6).

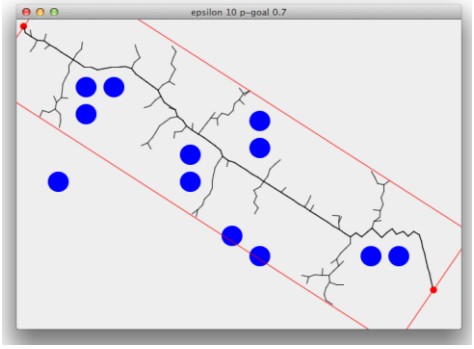


Figure 5. Iterative RRT-Smooth algorithm

```
function RRT-Plan (world, start, goal, epsilon, p-goal, tree)
  target ← chooseTarget(world, pGoal, goal)
  nearest ← nearest(tree, target)
  if( distance(nearest,goal) < epsilon )
    return tree
  else
    explored ← explore(world, nearest, target, epsilon)
    if (explored != nil )
      addNode(tree,explored)
   RRT-Plan(world, start, goal, epsilon, p-goal, tree)

function chooseTarget(world, pGoal, goal)
  p ← UniformRandom in [0.0 .. 1.0]
  if 0 < p < pGoal
    return goal;
  else
    return randomState(world)

function nearest(tree,target)
  point ← first(tree)
  for each node in rest(tree)
    if distance(node, target) < distance(point, target)
      point ← node

function explore(world, u, v, epsilon)
  explored ← extend(u, v, epsilon)
  if checkCollision(world, explored) = false then
    return explored
  else
    return nil
```

Figure 6. RRT planning algorithm

After reaching goal path smoothing algorithm is applied to the obtained RRT path. Smooth-path starts with first node in the path, it then calls dropwhile-walkable function which finds a node that is farthest from the first node that can be reachable without collision (Fig. 7). The function adds this node to the path and does the same operations for this node, this process is repeated until there are no more nodes on the path in which case the function returns. The use of path smoothing procedure with RRT-Plan allows to optimize the path of the robot.

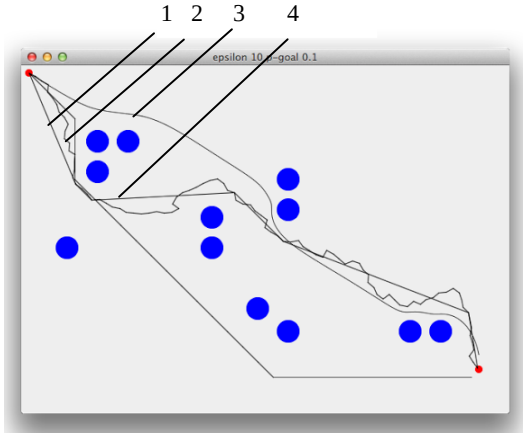
```
function smooth-path (isWalkable, path, curr-node, path-rest)
if(isEmpty(rest) == false)
path_rest = drop-while-walkable(fn(isWalkable,curr-
node,%),path_rest)
x =    first(path-rest)
xs = rest(path-rest)
smooth-path(isWalkable addNode(path-rest,curr-node),x,xs))
else
return addNode(path_rest,curr-node)

function drop-while-walkable (pred, path, curr-node)
if(isEmpty(path) == false && pred(first(path)) == true)
drop-while-walkable(pred, rest(path), first(path))
else
return addNode(curr-node, s)
```

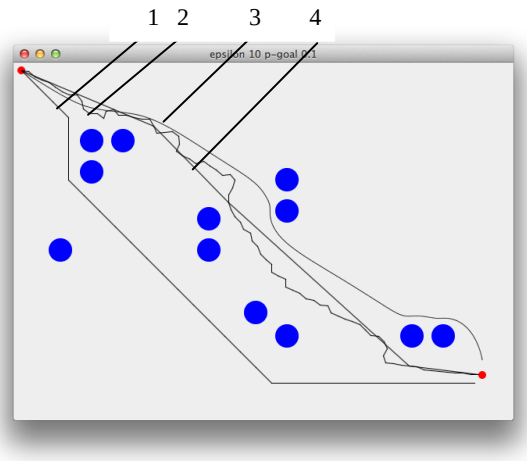
Figure 7. Path smoothing algorithm

IV. Simulation and Evaluation of Path Finding Procedure

Simulations of path finding algorithms have been done. When algorithm is run the input position of goal, start position of robot, positions of obstacles, threshold distance to obstacle are entered. After entering these values the robot start to path and move towards goal. We test different path finding algorithms in order to show the efficiency of the proposed improved iterative RRT-smooth algorithm. At first stage the map of the world with obstacles is selected. The implementations of APF, A-star, RRT-Plan, RRT-Smooth, IRRT and IRRT-Smooth algorithms for robot navigation have been done. The obstacles are shown with coloured circle (Fig. 8(a) and 8(b)). Robot analysing the map of environment plans its trajectory in real time in order to achieve the goal position. The source and goal points are entered for the given map. For comparative analysis of different methods the running time of algorithm and path length are selected. At first stage the simulations of APF and A star algorithms are done for the given map. Then the simulations of RRT-Plan and IRRT-Smooth algorithm are done for given map. The algorithms are run for the same values of Epsilon (the amount that the tree is expanded at each iteration) and pGoal (probability goal). After the simulation iterative RRT-Smooth algorithm is run for the same values of parameters of Epsilon and pGoal. In some cases the results of RRT-Smooth is the same as IRRT-Smooth. Fig. 8 depicts the fragment from the simulation of iterative RRT algorithm. During simulation the values of epsilon and pGoal are selected as 50 and 0.3 respectively. Fig. 9 depicts simulation results of different path finding algorithms. As shown from the figure IRRT-Smooth algorithm give better result in length. Table I demonstrates the average value for time and length of path finding results that it takes for achieving the goal for Fig. 9. The simulation results are averaged for 1000 (thousand) path finds. As shown from the figure and table 1 the IRRT-Smooth algorithm give better result in length. The results obtained for length of IRRT-Smooth algorithm better than other ones. The time results is better than others excluding RRT Plan.



(a)



(b)

Figure 8. Path finding, (a) 1- A star, 2- RRT-Plan, 3- APF, 4- RRT-Smooth, (b) 1- A star, 2- RRT-Plan, 4- APF, 3- Iterative RRT-Smooth.

TABLE I. Simulation Results Of Different Algorithms

Methods	time	length
A*	22.5347799	782.5483399
APF	102.477935	732.0000
RRT Plan	8.2619800	849.9
RRT Smooth	14.870870	748.336417
Iterative RRT	8.0726300	716.428945
Iterative RRT Smooth	12.651820	708.567281

V. Conclusion

Iterative RRT-Smooth path finding algorithm is designed for efficient navigation of soccer robots. The RRT and Smooth algorithms are iteratively applied to the environment in order to find near-optimal path. The comparison of iterative RRT-Smooth algorithm with the existing path finding algorithms has been performed using run time and path length parameters. The run time of A* and APF algorithms take more time than others and in some cases they are not efficient for real time operation where the environment changes quickly. The RRT algorithm quickly finds a feasible solution, but does not necessarily find the shortest path, and sometimes the length of calculated path may be very long. RRT-Smooth optimizes the path length of RRT plan algorithm. The proposed iterative RRT-Smooth efficiently finds feasible and near optimal solutions in short time and shortens the path length considerably. The described path finding algorithm is applied for control of the holonomic 4-wheel-driven soccer robots which are designed and manufactured in our research laboratory. The experimental results demonstrate the efficiency of the proposed algorithms in navigation of soccer robots using iterative RRT-Smooth in dynamic environments

References

[1] J. Borenstein and Y. Koren, "Real-time obstacle avoidance for fast mobile robots in cluttered environments," in *Proc. IEEE Int. Conference on Robotics and Automation*, vol. 1, May 1990, pp. 572–577.

[2] J. Borenstein, Y. Koren, and S. Member, "The vector field histogram-fast obstacle avoidance for mobile robots," *IEEE Journal of Robotics and Automation*, vol. 7, pp. 278–288, 1991.

[3] I. Ulrich and J. Borenstein, "Vfh+: Reliable obstacle avoidance for fast mobile robots," in *Proc. the IEEE Int. Conference on Robotics and Automation*, 1998.

[4] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidan-ce," *IEEE Robotics Automation Magazine*, vol. 4, 1997.

[5] L. Mc Fetridge, M. Y. Ibrahim, "A new methodology of mobile robot navigation: The agoraphilic algorithm," *Robotics and Computer-Integrated Manufacturing*, vol. 25, no. 3, pp. 545-551, 2009.

[6] K. Fujimura, *Motion Planning in Dynamic Environments. Secaucus*, NJ, USA: Springer-Verlag New York, Inc., 1992.

[7] W. Y. Loong, L. Z. Long, and L. C. Hun, "A star path following mobile robot," in *Proc. IEEE 4th Inter. Conf. on Mechatronics*, Kuala Lumpur, Malaysia, 2011.

[8] R. Abiyev, D. Ibrahim, and B. Erin, "EDUrobot: An educational computer simulation program for navigation of mobile robots in the presence of obstacles," *Int. Journal of Engineering Education*, vol. 26, no. 1, pp. 18–29.

[9] R. Abiyev, D. Ibrahim, and B. Erin, "Navigation of mobile robots in the presence of obstacles," *Advanced Engineering Software*, vol. 41, pp. 1179–1186, Oct. 2010.

[10] S. M. LaValle, *Planning Algorithms*, Cambridge, U. K.: Cambridge University Press, 2006.

[11] R. H. Abiyev, N. Akkaya, E. Aytac, and D. Ibrahim, "Behaviour tree based control for efficient navigation of Holonomic robots," *Int. Journal of Robotics and Automation, Actapress*, vol. 29, no. 4, 2014.

[12] D. Ferguson and A. A. Stentz, "Dynamic planning in high-dimensional search spaces," in *Proc. the 2006 IEEE International Conference on Robotics and Automation*, 2007, pp. 1310–1315.

[13] R. Geraerts and M. H. Overmars, "Creating high-quality paths for motion planning," *The International Journal of Robotics Research*, vol. 26, no. 8, pp. 845–863, 2007.

[14] S. Karaman and E. Frazzoli, "Incremental sampling-based algorithms for optimal motion planning," in *Proc. Robotics: Science and Systems*, (Zaragoza, Spain), June 2010.

[15] J. Nasir, F. Islam, U. Malik, Y. Ayaz, O. Hasan, M. Khan, and M. S. Muhammad, "RRT*-SMART: A rapid convergence implementation of RRT*," *International Journal of Advanced Robotic Systems*, 2013.

[16] R. Alterovitz, S. Patil, and A. Derbakova, "Rapidly-exploring roadmaps: Weighing exploration vs. refinement in optimal motion planning," in *Proc. IEEE Int. Conf on Robotics and Automation*, 2011, pp. 3706–3712.

[17] Neuislenders robotics team. [Online]. Available: robotics.neu.edu.tr

Rahib H. Abiyev graduated from the Azerbaijan State Oil Academy with High Honours in Electrical and Electronic Engineering. He continued studying and received Ph.D degree in Electrical and Electronic Engineering. He worked research assistant at the research laboratory "Industrial intelligent control systems" of Computer-aided Control System Department. From 1999-present he is working at the department of Computer Engineering of Near East University, Northern Cyprus. He is director of Applied Artificial Intelligence Research Centre. His research interests are Softcomputing, Control Systems, Robotics, Signal Processing, Pattern Recognition.

Nurullah Akkaya was born in Istanbul, on 17th September 1984. After finishing secondary and high school degree, he started his undergraduate degree in Computer Science at University of North Alabama. He graduated from Computer Engineering at Near East University. He has been working at Applied Artificial Intelligence Research Centre, Near East University. He is member of Robotics research group of Near East University.

Ersin Aytac was born in Antalya, on 16th July 1984. After finishing secondary and high school degree, he started his undergraduate degree in Mechanical Engineering at University of North Alabama and then transferred to University of Virginia. He graduated from Mechanical Engineering at Near East University. He has been working at Applied Artificial Intelligence Research Centre, Near East University. He is member of Robotics research group of Near East University.

Irfan Günsel graduated from Near East University. He has been working at Applied Artificial Intelligence Research Centre. He is member of Robotics research group of Near East University.

Ahmet Çağman graduated from Near East University. He has been working at Applied Artificial Intelligence Research Centre. He is member of Robotics research group of Near East University.