Behaviour Trees Based Decision Making for Soccer Robots

RAHIB H.ABIYEV[1], ŞENOL BEKTAŞ[2], NURULLAH AKKAYA[1], ERSIN AYTAC[3]
[1]Department of Computer Engineering, [2]Department of Electrical & Electronic Engineering,
[3]Department of Mechanical Engineering, Near East University, P.O. Box 670, Lefkosa, North Cyprus
Phone: +0392 2236464 (383), Fax: +0392 223 66 22,
E-mail: [1]rahib@neu.edu.tr; [1]nurullah@nakkaya.com; [3]ersin@aytac.me

Abstract:- This paper proposes a novel behaviour tree based decision making mechanism for control of soccer robots. Decision Making (DM) is a basic block that analyses the current state of world model and makes decision about new positions of robots. Using hierarchical tree structure the behaviours of robot soccer are designed. BT has high and low level behaviours and its nodes are operating using certain behaviour rules. High level behaviours are implemented using low level behaviours. The use of BT approach allows to model complicated situations easily that show advantages of this technique over finite state machines widely used in robot control. After defining behaviours and making decisions, path finding module determines the path of robot. In the paper a novel path finding algorithm that uses a Rapidly Exploring Random Tree (RRT) and path smoothing techniques has been developed to find the path of the robot in a short time. The obtained simulation and experimental results show that the constructed navigation system of soccer robots efficiently finds desirable and feasible solutions in short amount of time.

Key-Words: - Behaviour trees; robot navigation; path finding; soccer robot

1 Introduction

The main goal of mobile robot navigation is steering a robot in any direction, in particularly towards a goal, and to know its precise location in the environment at any time. Many efforts have been paid in the past to develop various algorithms for navigation and control of robots in dynamic environments. Control and navigation of robot includes a set of algorithms. These are vision module, decision making, path planning, and control algorithms. The soccer robots used in RoboCup uses SSL vision system. Decision Making (DM) is the basic intelligent module in robot navigation. This module analyses the current state of the world and makes decisions about the new positions of soccer robots and plans a motion for the robots. In the literature different approaches-finite state machines [1,2], Petri nets [3,4], behaviour based control [5-7], and topological graph map [8] have been considered to develop DM module for soccer robots.

Finite state machine (FSM) is widely used for robot control [1,2,9]. Modelling behaviour with hierarchical finite state machines has expressiveness; it is more intuitive by utilizing more compact behaviour descriptions. For actions layer a FSM has been designed to properly select among the actions. The algorithm has been tested in simulation and provided satisfactory results in many researches. But for complex situations the use of FSM for robot games increases the number of states required to encode the behaviour of the robots, along with the number of transitions between the states. Sometimes the modelling of these situations becomes very difficult with FCM.

Petri nets are preferred to FSM due to their larger modelling power and because one can model the same state space with a smaller graph [3,4]. Composition of Petri nets leads to an exponential growth in the state space. In some cases the building of Petri nets become too large to generate all states of the system and its analysis becomes difficult.

In the paper the hierarchical BT approach that simplifies the decision making process is purposed. In mobile robots, operating in unstructured environments, a priori knowledge of the environment is usually absent or partial. Frequently, the environment where robot moves is not static, i.e., during motion the robot can face with other moving robots or obstacles, and thus execution is often associated with uncertainty.

After making decision using BT approach, the path planning is used to determine a route from the robots current coordinate location to another goal location along a set of waypoints. There have been developed a number of methodologies for robot navigation using path planning and obstacle avoidance. These approaches require certain time for finding of feasible path [10,11]. Moreover soccer robots operate in the environment having many moving obstacles and goal, and the next positions of the goal and obstacles are unknown to the robot. The use of the traditional methods for path finding of soccer robots require certain time, and the finding of this path may not complete in

ISBN: 978-1-61804-168-5

reasonable time for real-time operation. For such cases, finding feasible, "near-optimal" robust path in short time becomes important. In this paper, for constructing efficient path finding system, the use of a path smoothing algorithm is considered that allows optimizing the given robot path using the local environment information.

2 Behaviour tree based decision making

BT is goal oriented. Each tree is assigned a goal, that will be achived. The robot behaviour is a control law that satisfies a set of constraints to achieve a particular goal. Each behaviour is defined by the set of actions.

A BT enables modularity, making states nested within each other and thus forming a tree-like structure, and restricting transitions to only these nested states [6,7]. The root node branches down to the tree until the leaf nodes are achieved. The leaf nodes are the base actions that define the behaviours. A BT is made up of three types of nodes, action, decorator, composite (Fig.1). Composite and decorator nodes are used to control the flow within the tree and action nodes. Actions are used to change states such as calculating a new path or kicking the ball. Composite nodes include set of nodes such as selector, sequence nodes, and their parallel and random versions.

Selector and sequence nodes are workhorse internal nodes (Fig. 1). A sequence represents a series of behaviours that we need to accomplish. A sequence will try to execute all its children from left to right. If all of its children succeed, sequence will also succeed. If one of its children fails, sequence will stop and return failure (Fig.1(a)). Fig. 1(a) describes a hypothetical pass sequence which will start executing its children from left to right, first checking if the receiver is at the correct assist spot waiting for a pass, then make sure a pass is safe. If both these conditions are met, only then it will execute the pass and return success up the tree.

Selector node will try to execute its first child. If its first child returns success, it will also return success. If the child fails, it will try executing its next child until one of its children returns success, or the node runs out of children at which point the node will return failure (Fig. 1.(b)). This property allows us to choose which behaviour to run next. The tree in Fig.1(b) chooses between shooting at the goal directly or executing a pass. Selector will start executing its children from left to right beginning with Shoot Goal sequence. If both Can Shoot to goal? and Shoot actions succeed, sequence will succeed, which will cause the selector to succeed. If Shoot Goal sequence fails, it will keep trying to execute its children until one succeeds, in which case selector will also succeed. If all fails selector will also fail.

Decorators take their names from the software design pattern Fig.1(c). In the context of BTs a decorator node is a node with a single child. Decorator modifies the behaviour of the branch in some way. Fig.1(c) shows the same Pass sequence with a decorator as its root. What limit decorator does is that, it will try to play pass. If a pass is made it will return success up the tree, if its children fails, it will try it n more times (1 in the case of this tree) to execute a pass. If a pass can not be made within n times then it will return fail up the tree.

The other behaviour nodes that can be used for composing tree are "non-deterministic-selector", "non-deterministic-sequence", "parallel", "forever", "until-fail", "until-success", "limit", "inverter", "interrupter" etc.

Using above described technique, the BT of complicated behaviours can be described. In the paper the BT approach is applied to control soccer robots. Hierarchical BT that uses high and low level trees are constructed. Each behaviour is reacting to the set of input data obtained from SSL vision. Decision making (DM) block of the robot uses a hierarchy from higher level behaviours to lower level behaviours. Robots are controlled by composing lower level behaviours into more complex higher level behaviours, and then these behaviours can be dispatched to the robots.

Each low-level behaviour handles a specific low-level action: *Move to* behaviour provides move function for other behaviours; *Pass* behaviour controls two robots and plays a pass; *Move to ball* behaviour is used to catch the ball; *Defend Goal*. In this case defender-robot will place itself to the intersection of ball heading and the goal line; *Shoot* behaviour is used to kick the ball; *Assistant* behaviour searches for a free space for receiving a pass; *Marker* behaviour selects a robot to intercept passes; *Move with ball* behaviour handles dribbling; *Penalty-taker* behaviour specializes in shoot task for penalties; *Penalty keeper* behaviour is optimized for keeping penalties.

As an example in Fig.2, *Pass* behaviour tree is described. *Pass sequence* begins by taking control of two robots using a *parallel sequence,* which acts like a *sequence* but executes its children in parallel aligns, passer and receiver for a pass, then the tree checks if a pass is safe. If a pass can be made passer shoots the ball. Next, the *sequence* waits for the ball to start moving, once that happens, we again wait

ISBN: 978-1-61804-168-5

for either the ball to stop or the ball gets within a robot diameter to the receiver at which point receiver moves and captures the ball.

The low level behaviours are composed to form high level behaviours. Some of our high level behaviours include: *Formations* behaviour places robots on various hard coded spots on the field; *Offensive Game* behaviour focuses on scoring goals against the opponent, as opposed to defence plays, which focus on preventing goals being scored into our own goal; *Defensive Game* focuses on preventing the opponent from scoring goals, as opposed to offence plays that focus on scoring against the opponent; *Game Selection behaviour* is used to select the main tactic.

3 Design of Efficient Path Finding Procedure

The next important problem is designing of the path finding algorithm in mobile robot navigation. One of the commonly used fast algorithms for path finding is a Rapidly-exploring random tree (RRT) algorithm [12,13]. The RRT algorithm is designed for efficiently searching nonconvex, high-dimensional search spaces. RRTs incrementally reduce the expected distance of a randomly-chosen point to the tree. The RRT algorithm is extremely simple and cheap to calculate but it is not optimal. A path will be computed quickly but it is not guaranteed to be the cheapest, and will results a

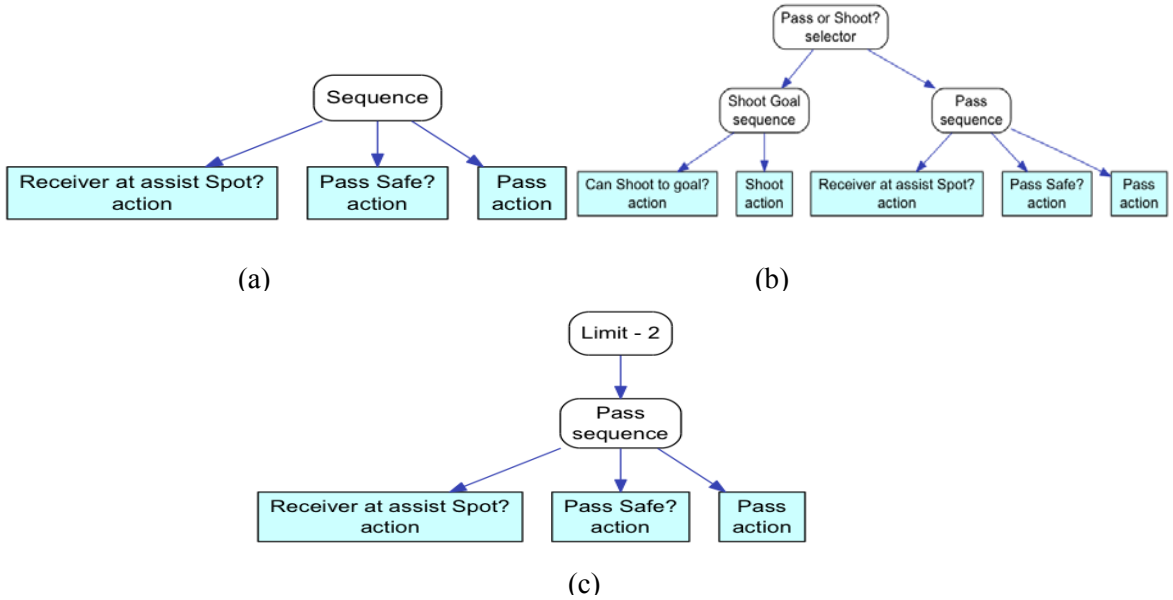


Fig.1. Behaviours: (a) Sequence nodes, (b) Selector nodes, (c) decorators

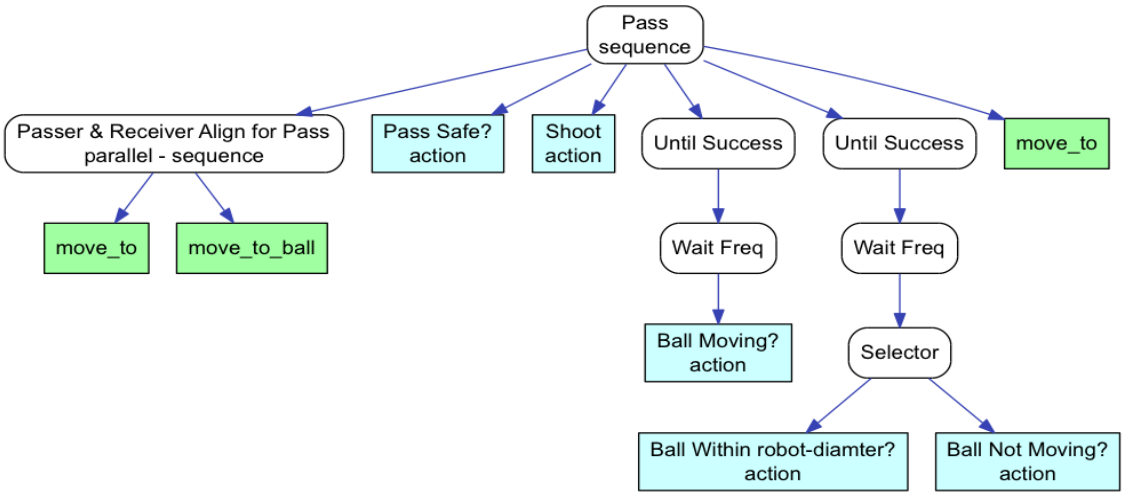


Fig.2. *Pass* behaviour tree

ISBN: 978-1-61804-168-5

different path for every search. Fig. 3 depicts the result of the RRT algorithm. As shown RRT algorithm finds many paths, then selects the path that can achieve the goal. RRT contains many zig zags and unnecessary edges. In order to deal with this problem, we use a path smoothing algorithm. Given two nodes that are reachable A and B. Assume that A is start point of the path. 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 (Fig.4). Here a dashed line depicts the original path, solid line demonstrates the optimal smoothed path. In the process of smoothing, the path is optimized. The use of path smoothing procedure with RRT-Plan allows to optimize the path of the robot.

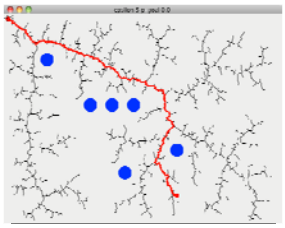


Fig.3.RRT-Plan algorithm. when *pGoal*=0.

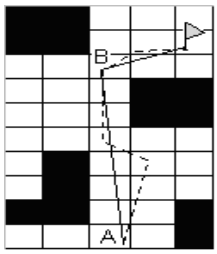


Fig.4.Smoothed optimal path. Here _ _ _ Original path, ____ Smoothed path

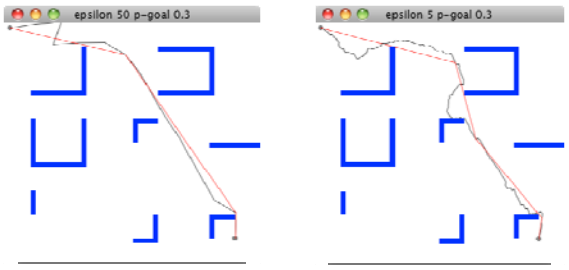
```
function smooth-path (isWalkable, path, curr-node,
                      path-rest)
 if(isEmpty(rest) == false)
    path_rest = drop-while-walkable(fn(isWalkable,
                curr-node,%),path_rest)
    x =   first(path-rest)
    xs = rest(path-rest)
    smooth-path(isWalkable, addNode(path-rest,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

Fig.5. Path smoothing algorithm

Smooth-path is a recursive algorithm (Fig.5) that will keep dropping nodes that are reachable from the given node. *Smooth-path* starts with first node in the path, it then calls *drop-while-walkable* function which finds a node that is farthest from the first node that can be reached without collision. 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 simulation studies were done using RRT-Plan and path smoothing algorithms. Simulation of the RRT-Plan algorithm has been done for different values of parameters- *Epsilon* (the amount that the tree is expanded at each iteration) and *pGoal* (probability goal). The resulting trajectory found by RRT-Plan is not guaranteed to be optimal, which causes the planed trajectory to be longer than the optimal path found by A* and APF [14]. To overcome this deficiency, the path found by RRT-Plan is optimised by a "quick path smoothing" algorithm. Fig.6 (a,b) depicts the results of simulation of RRT-Plan and quick path-smoothing algorithms, when *epsilon=50, pGoal=0.3* (a), *epsilon=5, pGoal=0.1* (b) and *epsilon=5, pGoal=0.3* (c). In figures dashed lines indicate results for the RRT-plan algorithm only, and solid lines indicate the RRT-plan and path-smoothing algorithm results. In Fig.6 (a) two different cases are given for the RRT-plan and path-smoothing algorithms, when *epsilon=50, pGoal=0.3.*



(a) (b)

Fig.6. Simulation results of RRT-Plan and quick path-smoothing algorithms, when *epsilon=50, pGoal=0.3* (a), and *epsilon=5, pGoal=0.3* (b). Here black lines - RRT-plan algorithm, red lines- RRT-plan with path-smoothing algorithm.

Table 1 demonstrates the simulation results of RRT-plan, using path smoothing algorithm for Fig.6 for different values of *pGoal* and *epsilon*. As shown from the tables, the use of path smoothing allows optimizing the tree and decreasing the path. The simulation results of A* and APF algorithms for Fig. 6 for different values of grid sizes are given in Table 2. Comparing these results with the ones given in Table 1 we can show that the run time of RRT and path smoothing algorithm are considerably lower than the run time of A* algorithm. Because of the optimization, the length of path used by RRT and path smoothing algorithms are close to those of the A* algorithm. From this it follows that the use of RRT and path smoothing algorithms allow efficient control of the robot in dynamic environments cluttered with obstacles.

ISBN: 978-1-61804-168-5

Table 1. RRT-Plan and quick path smoothing algorithm results

epsilon	pGoal	time-raw	average-raw-length	time-smooth-quick	average-smooth-length
15	0.1	562.6580	926.4784	549.6829	734.4640
15	0.2	545.6197	922.4163	568.7634	738.3867
15	0.3	623.1779	894.1692	595.6740	741.4081
15	0.4	730.5443	897.7102	709.2538	731.4458
15	0.5	714.7288	896.6968	891.5002	730.7849
30	0.1	226.7948	917.9369	229.9944	751.9652
30	0.2	241.7770	908.6983	260.9923	747.4403
30	0.3	265.9063	917.6009	290.2578	759.9219
30	0.4	298.1229	901.2486	312.5456	749.4017
30	0.5	354.7271	908.5616	373.4415	761.2640
50	0.1	131.4685	907.5731	141.6986	772.2194
50	0.2	138.2280	925.2939	151.8176	765.1810
50	0.3	163.2630	900.4459	173.0801	757.1586
50	0.4	167.2672	919.1979	204.5359	761.8144
50	0.5	193.8853	906.1210	216.6600	766.4267

Table 2. Simulation results of A* and APF algorithm

Methods	grid-size	time	length
A*	1	28833.62801	883.893577750
	2.5	4834.85994	870.286976033
	5	1332.355361	819.325035256
APF		676.493099	679.760236374

4. Real time implementation

The structure of the robot soccer navigation system designed in this paper is given in Fig. 7. The mobile soccer robots and their navigation system are designed, manufactured and assembled in our research laboratory [15]. To design soccer robots we used holonomıc robots that have holonomic wheels with 2 degrees of freedom. Before applying the robot navigation software on soccer robots, we've tested the algorithms using a simulation software package called grSim [16]. grSim acts as the vision system of the robot which sends coordinates of all robots and balls, and in addition simulates the movements of the robot by receiving control signals. In real life the same coordinates are sent by the SSL-Vision system.

SSL-Vision is a vision system of soccer robots of RoboCup Small Size League. Computer tracks the

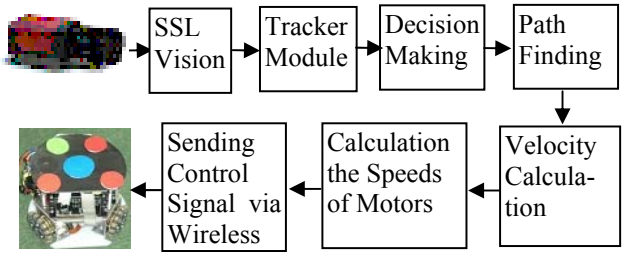


Fig.7. Structure of the designed control system of the holonomıc robots

world using high speed overhead cameras. SSL vision processes the map of the environment using the cameras and obtains the coordinates of soccer robots and balls then sends them to the computer/s. In order to track the world in real time, the data is sent every 1/60th of a sec. Tracker module captures this data stream off the network and converts it into a data structure which is used by DM module. DM block using this data, makes strategic planning of soccer robots. Behaviour tree based control with RRT-Plan and path smoothing algorithms are applied for decision making and path finding of the holonomıc robots. After selecting certain behaviours, path finding block find the new coordinates of each holonomıc robots. These coordinates describes a path, which is then sent to the *Velocity* block where a velocity vector is calculated and updated to guide the robot along the path. Between each velocity update a set of four motor speeds, one for each wheel, is calculated and sent to the soccer robots. Motor drivers are driven by a microcontroller. 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. The formula used to calculate the speeds of each wheel, for a robot with four wheels, is given below,

$$\begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{bmatrix} = \begin{bmatrix} -\sin\theta_1 & \cos\theta_1 & 1 \\ -\sin\theta_2 & \cos\theta_2 & 1 \\ -\sin\theta_3 & \cos\theta_3 & 1 \\ -\sin\theta_4 & \cos\theta_4 & 1 \end{bmatrix} \times \begin{bmatrix} v_x & v_y & R_\omega \end{bmatrix}^T$$

Here v_1, v_2, v_3, and v_4 are motor speeds. v_x *and* v_y are the speed of robot in horizontal and vertical direction, R_ω is the rotation speed. After calculation the wheel speeds, the computer sends these values to the robot soccer through wireless. A PID controller is used to control rotational speed of the robot soccer which keeps it pointing to an angle determined by the DM block.

Fig.8 demonstrates the fragment of robot navigation using considered behaviour tree based control and RRT-Plan with path smoothing algorithms. The path taken by the holonomıc robot is shown in black line. Here start point of robot is

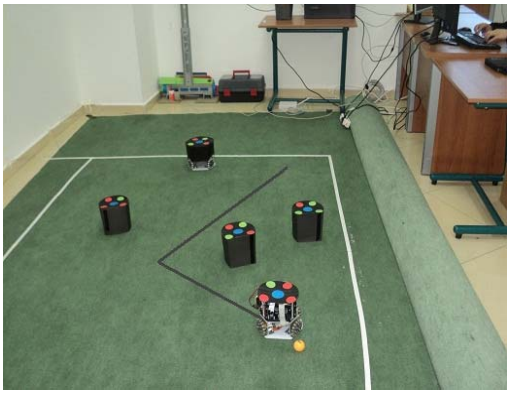


Fig. 8. Robot navigation

ISBN: 978-1-61804-168-5

start point of black line, goal position is ball position. Detailed description of the robot navigation is given in web-pages http://staff.neu.edu.tr/~rahib/aairc/research_groups.htm or http://robotics.neu.edu.tr/wiki/index.php/Media

5 Conclusion

The hierarchical BT based algorithm is developed for efficient navigation of soccer robots. BTs easily define complex states. The modularity of the behaviour tree based control design allowed for incremental development and ease of maintainability and extendibility of control system. The algorithm has been extensively tested in simulation and through experimental tests and provided satisfactory results. Also improved path finding algorithm is presented to provide navigation path in required shortest time. Through simulation it was shown that the proposed algorithm efficiently finds feasible and near optimal solutions in short time and shortens the path length considerably. The described BT based control and path finding algorithm are applied for control of the holonomic soccer robots which are designed and assembled in our research laboratory. The experimental results demonstrate the efficiency of the proposed algorithms in navigation of soccer robots .

References

[1] Damas, B., and Lima, P., (2004), "Stochastic Discrete Event Model of a Multi-Robot Team Playing an Adversarial Game", 5th IFAC/EURON Symposium on Intelligent Autonomous Vehicles - IAV2004, Portugal

[2] Peter van de Ven. Finite state automata applied to robot soccer. 2005. http://www.ie.wtb.tue.nl/Robocup/Robots/Intelligence/Report.pdf.

[3] Costelha, H., Lima, P. Modelling, analysis and execution of robotic tasks using Petri nets. IEEE International Conference on Intelligent Robots and Systems,IROS2007,pp.1449–1454

[4] D. Milutinovic, P. Lima, Petri Net Models of Robotic Tasks, Proc. Of the IEEE International Conference on Robotics and Automation, 2002.

[5] M. Mataric. Behaviour-based control: examples from navigation, learning, and group behaviour. Journal of Experimental & Theoretical Artificial Intelligence, Volume 9, Issue 2-3, 1997

[6] Alex J. Champandard's. Getting Started with Decision Making and Control Systems, AI Game Programming Wisdom 4, section 3.4, pp. 257–264, Course Technology, 2008

[7] Chong-U Lim, Robin Baumgarten, Simon Colton. Evolving Behaviour Trees For the Commercial Game DEFCON**.** Applications of Evolutionary Computation Lecture Notes in Computer Science, Springer, 2010

[8] Concalo Neto, Hugo Costelha, Pedro Lima. Topological Navigation in Configuration Space Applied to Soccer Robots. D.Polani et al.(Eds): RoboCup 2003, LNAI, Springer-Verlag Berlin Heidelberg, 2004, pp.551-558.

[9] Nobuyuki Kurihara, Ryotaku Hayashi, Hikari Fujii, Daiki Sakai, Kazuo Yoshida. Intelligent control of autonomous Mobile Soccer Robot adapting to dynamic environment. D.Polani et al.(Eds): RoboCup 2003, LNAI, Springer-Verlag Berlin Heidelberg, 2004, pp.568-575.

[10] R.Abiyev, D. Ibrahim, and B. Erin. EDURobot: An Educational Computer Simulation Programm for Navigation of Mobile Robots in the Presence of Obstacles. International Journal of Engineering Education. Vol.26. No.1, pp.18-29, 201.

[11] R.Abiyev, D. Ibrahim, and B. Erin. Navigation of mobile robots in the presence of obstacles. Advances in Software Engineering, Vol.41, Issues 10-11, 2010, pp.1179-1186.

[12] S.M. LaValle and J.J. Kuffner. Randomized kinodynamic planning. *Inter. J. of Robotics Research*, 20(5):378-400, May 2001.

[13] S. M. LaValle. Planning Algorithms. Cambridge University Press, U.K., 2006.

[14] Rahib H.Abiyev, Nurullah Akkaya, Ersin Aytac. Navigation of Mobile Robot in Dynamic Environments. IEEE CSAE 2012, Zhangjiajie, China, May 25-27, 2012.

[15] NEUIslenders robotics team. http://robotics.neu.edu.tr/wiki/index.php/NeuIslanders

[16] grSim RoboCup small size robot soccer simulator. http://www.parsianrobotic.ir/grsim/

[17] Rahib Abiyev, Nurullah Akkaya, Ersin Aytac, Mustafa Arici, Muhlis Bayezit. NEUIslanders 2012 Team Description Paper. http://small-size.informatik.uni-bremen.de/_media/neuislanders_2012_tdp.pdf

[18] Rahib Abiyev, Nurullah Akkaya, Ersin Aytac, Dogan Ibrahim. Behaviour tree based control for efficient navigation of holonomic robots. International Journal on Robotics and Automation. (Accepted).