Robot Pathfinding Using Vision Based Obstacle Detection

Rahib H.Abiyev, Murat Arslan
Near East University, Department of Computer Engineering, Applied Artificial Intelligence Research Centre, Lefkosa, Mersin-10, North Cyprus
rahib.abiyev@neu.edu.tr; murat.arslan@neu.edu.tr

Irfan Günsel, Ahmet Çağman
Near East University, Applied Artificial Intelligence Research Centre, Lefkosa, Mersin-10, North Cyprus
igunsel@robotics.neu.edu.tr; acagman@neu.edu.tr

***Abstract*— This paper presents vision-based obstacle detection and pathfinding algorithms for a mobile robot using support vector machine (SVM) and A* algorithms. The used methods are applied for navigation of NAO humanoid robot. The camera which is located on NAO robot is used to capture the images of the world map. The captured image is processed and classified into two classes; an area with obstacles and area without obstacles. For classification of images, Support Vector Machine (SVM) is used. After classification, the map of the world is obtained as an area with obstacles and area without obstacles. This map is input for path finding algorithm. A* path finding algorithm is used to find the path from the start point to the goal. The considered algorithms are implemented for the guidance of NAO robot. The used algorithms allow to detect obstacles and find the near-optimal path.**

***Keywords*— Obstacle detection; Path-finding; A* algorithm; SVM**

I. INTRODUCTION

One of an important problem in robotics is the designing intelligent robots performing the human actions in certain fields. The designing of such intelligent robots needs to design set of modules such as object detection, image processing, path planning, obstacle avoidance, motion control etc.[1,2].

Object detection is a computer technology linked to computer vision and image processing that deals with detecting objects such as humans, cars, building, obstacles etc. in digital images and videos. One way of detecting objects is the use of image recognition. Using image recognition the world map can be classified as area with obstacles and area without obstacles. This information in future can be used for path-finding by robot.

Humans recognize large range of objects in images with small effort, but this task is still a challenge for computer vision systems. Object detection algorithms typically use extracted features and learning algorithms to recognize instances of an object category.

Robots are moving in unpredictable, cluttered, unknown complex and dynamic environments. During moving the robots need to detect obstacles and then avoid them to go to the destination point. Many obstacle avoidance algorithms are proposed. "Bug" algorithm follow the edges of obstacles without considering the goal [3]. They are time-consuming. The artificial potential field is most commonly used method that uses attractive and repulsive fields for the goals and obstacles, respectively [4]. But the APF has several disadvantages: when there are many obstacles in the environment the field may contain a local minima; the robot unable to pass through small openings such as through doors; the robot may exhibit oscillations in its motions. Vector Field Histogram (VFH) uses a two-dimensional Cartesian histogram grid as a world model and the concept of potential fields [5,6]. The VFH algorithm selects a shorter path than bug algorithms but it takes more time to manipulate. Other goal oriented algorithms are dynamic window [7], "agoraphilic"[8], fuzzy based approach [9-11]. Rapidly-exploring Random Trees algorithm is a faster algorithm and can be applied for pathfinding in dynamic environments [12,13]. But frequently the path determined by RRTs may be very long.

RRT-smooth algorithm [14,15] was proposed to short the RRT length. During finding of the path, the run time and length of the path are important parameters [16,17]. In this paper in order to choose an acceptable value for these parameters A* search algorithm was used [18,19]. A* is an informed search algorithm, called as a best-first search. The algorithm solves the problem by searching among all possible paths to the solution for the one that incurs the smallest cost (least distance traveled, shortest time, etc.), and among these paths, it first considers the ones that appear to lead most quickly to the solution. The considered algorithms are implemented for navigation of NAO robot for detection of obstacles and also pathfinding.

II. STRUCTURE OF THE SYSTEM

The block-scheme of the designed naviga- tion system of NAO robot is given in Fig. 1. The algorithm includes the following steps.

1. Image Capturing. In this step the image is captured by the camera which is located on NAO robot's forehead.
2. Obtaining Map of World that includes the following operations.

   -Reading the world images and converting the colour (rgb) images to hsv value to find the fixed area that robot moves.

   -Defining lower and upper white hsv values.

   -Thresholding the HSV image to get only fixed area.

Authorized licensed use limited to: TUFTS UNIV. Downloaded on December 17,2024 at 21:11:14 UTC from IEEE Xplore. Restrictions apply.
978-1-5386-2201-8/17/$31.00 ©2017 IEEE

-Removing small holes

-Fixing erosion

-Finding contours

-Getting coordinates of a rectangle around the corner

-Drawing the lines and dots to show the area

-the top-left, top-right, bottom-right, and bottom-left points are included to the array and is used for perspective correction.

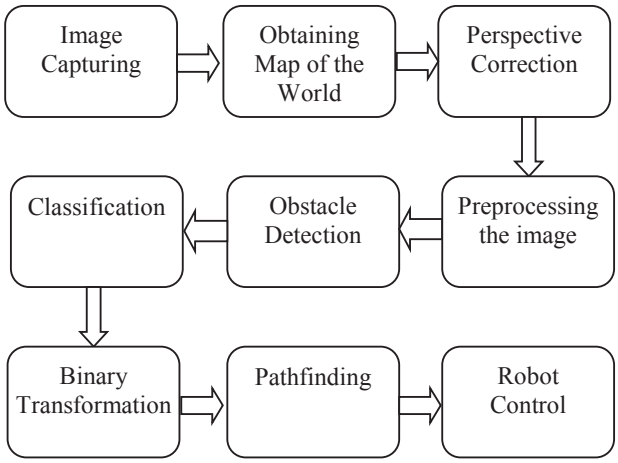

Fig. 1. Block-scheme of navigation system

3. Perspective Correction

- Computing the width of the new image, which is the maximum distance between bottom-right and bottom-left x-coordinates or the top-right and top-left x-coordinates
- Computing the height of the new image, which is the maximum distance between the top-right and bottom-right y-coordinates or the top-left and bottom-left y-coordinates
- Constructing the set of destination points to obtain a "birds eye view" of the image
- Specify points in the top-left, top-right, bottom-right, and bottom-left order. Finally save the image.

4. Preprocessing the Image
- Applying Canny Edge detection
- Applying Corner Harris Detection
- Applying Contour Detection

5. Obstacle Detection

-Splitting the preprocessed image into 20x20 pixels 768 pieces.

-Separating all the images into two classes: “negative” and “positive” .

-Using selected images the creating of dataset for training SVM.

6. Classification

-Selecting the images and obtaining their histograms values.

-Representing the world model (image) by 1 or 0 using histogram values of the all images.

-Training the SVM.

7. Binary Transformation

- Using 0 and 1 values create binary matrix of the map.

8. Pathfinding

-Using binary matrix apply A* algorithm to find shortest path.

-Finding the path.

9. Robot Control

-Using NAOqi framework’s movement commands to move NAO robot. Robot ip address and port are used for communication of the robot and computer.

III. CLASSIFICATION

Support vector machine tries to find out a hyperplane that has best separation which can be achieved by largest distance to the nearest training data point of any class. Let assume a binary classification have a data points (x_i,y_i), where $x_i \in R^p$ data points, $y_i \in \{-1,1\}$ classes. Each (x_i) is a vector. It needs to find the maximum-margin hyperplane that divides the points into two classes. It can be described as:

$$w.x + b = 1 \quad \text{and} \quad w.x + b = -1.$$

where w is the normal vector to the hyperplane. It needs to minimize $\|w\|$ to prevent data points from falling into the margin, it needs to add the following constraint: for each i either $w\,x_i - b \geq 1$ for x_i of the first class or $w\,x_i - b \leq -1$ for x_i of the second class. As a result, it can be written as as $y_i(wx_i - b) \geq 1,\ 1 \leq i \leq n$. The samples along the hyperplanes are called Support Vectors (SVs) and separating hyperplane with largest margin can be defined by $M=\frac{2}{\|w\|}$ that specifies support vectors means training data points closets to it. Taking into account the mentioned we can obtain the quadratic optimization problem:

$$\begin{aligned} &\text{Minimize} \quad \|w\| \\ &\text{Subject to:} \quad y_i\,(w\,x_i - b) \geq 1, \\ &\qquad\qquad \text{for any} \quad i=1,\ldots,n. \end{aligned} \tag{1}$$

The main goal in SVM is the maximization of the margin of separation and minimization of training error. The above problem can be transformed into Lngarange formulation.

Authorized licensed use limited to: TUFTS UNIV. Downloaded on December 17,2024 at 21:11:14 UTC from IEEE Xplore. Restrictions apply.

$$\max imize \quad L(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} y_i y_j \alpha_i \alpha_j K(x_i, x_j)$$

$$subject\ to \quad \sum_{i=1}^{n} y_i \alpha_i = 0 \qquad (2)$$

$$\alpha_i \geq 0, \quad i = 1, ..., n.$$

where $K(x_i, x_j) = \langle \varphi(x_i), \varphi(x_j) \rangle$ is a kernel function that satisfies Mercer theorem. Based on Karush-KuhnTucker(KKT) complemen-tarity conditions the optimal solutions α∗, w∗, b∗ must satisfy the following condition.

$$a_i^* [y(w^* \varphi(x_i) + b^* - 1] = 0, \quad i = 1, ..., m$$

where the α_i^* are the solutions of the dual problem. The resulting SVM for function estimation becomes

$$f(x) = \text{sgn}(\sum_{i=1}^{m} \alpha_i^* y_i K(x_i, x_i^*) + b) \qquad (3)$$

where m is the number of support vectors. SVM technique is a powerful widely used technique for solving supervised classification problems due to its generalization ability. In essence, SVM classifiers maximize the margin between training data and the decision boundary (optimal separating hyperplane), which can be formulated as a quadratic optimization problem in a feature space.

IV. PATHFINDING

A* is a computer algorithm which is commonly used for finding path between two nodes. Because of its performance and accuracy it is commonly used in finding of shortest path between the points.

A* solves problems by searching all possible paths to the goal for the one that incurs the smallest cost depending on the cost function. Among these paths the algorithm will first consider the ones that leads to least costly solution. It is formulated with weighted graphs, starting from a specific point. A* builds a tree of paths starting from source point, expanding the paths one step at a time, until one of the paths reaches at the goal point.

A* is iterative algorithm that determines the partial path at each iteration in order to expand it future and reach the goal node. It does this operation using an estimate of the cost (total weight) still to go to the goal node. Specifically, A* selects the path that minimizes $f(n)=g(n)+h(n)$ where n is the last node on the path, g(n) is the cost of the path from the start node to the n-th node, and h(n) is a heuristic function that estimates the cost of the cheapest path from n-th node to the goal. Depending on the problem there are multiple heuristic functions. For the algorithm to find the actual shortest path, the heuristic function must be admissible, which means that function can never overestimate the actual cost to get to the nearest goal node. As an example, when searching for the shortest route on a map, h(x) might represent the straight-line distance to the goal, since that is physically the smallest possible distance between any two points.

A typical implementation of A* uses a priority queue to perform the repeated selection of minimum cost nodes to expand. This priority queue is called an open set. At each iteration of the algorithm, the node with the lowest f(x) value is removed from the queue, the f and g values of its neighbors are updated accordingly, and these neighbors are added to the queue. The algorithm continues until a goal node has a lower f value than any node in the queue (or until the queue is empty). The f value of the goal is then the length of the shortest path, since h at the goal is zero in an admissible heuristic. If the heuristic h satisfies the additional condition $h(x)<=d(x,y)+h(y)$ for every edge (x,y) of the graph (where d denotes the length of that edge), then h is called monotone, or consistent. In such a case, A* can be implemented more efficiently—roughly speaking, no node needs to be processed more than once (see closed set below)—and A* is equivalent to running Dijkstra's algorithm with the reduced cost $d(x,y)=d(x,y)+h(y)-h(x)$.

Additionally, if the heuristic is monotonic (or consistent), a closed set of nodes already traversed may be used to make the search more efficient.

Figure 2 depicts the graphical simulation result of three different algorithms A*, APF and RRT used for path finding of mobile robot. Table 1 demonstrate the results of simulations using A*, APF and RRT path finding algorithms for figure 1. Here the results are obtained for 1000 runs. As shown the time result of RRT is better and the distance result of APF is better. A* algorithm is the more appropriate algorithm for selecting the method using both time and distance results.

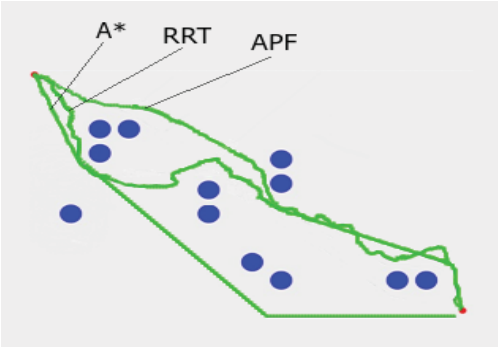

Fig. 2. Demonstration of simulation results of path finding algorithms.

TABLE I. SIMULATION RESULTS

Methods	Time	Length
A*	22.53477	792.5
APF	102.4779	732.0
RRT	8.261980	849.9

Authorized licensed use limited to: TUFTS UNIV. Downloaded on December 17,2024 at 21:11:14 UTC from IEEE Xplore. Restrictions apply.

V. Experimental Results

The above mentioned algorithms are used for navigation of humanoid NAO robot. Fig.3 shows NAO robot. NAO has 2 cameras, which are located on the head and chest. The camera located on the head is selected for capturing the map of world.

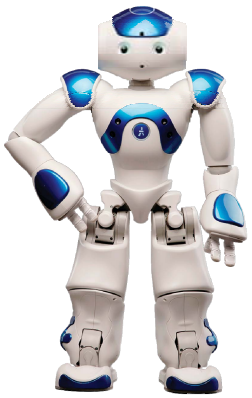

Fig. 3. NAO robot

Fig.4 shows the map where the robot should be moved. The obstacles which are different colors and shapes located on the fixed area. An image is captured using the on board camera on the NAO robot's head. At first the image of the world map is converted from RGB (Red, Green, Blue) to HSV(Hue Saturation Value) values. Next the borders of the area where robot should move are determined (in Fig. 4 the white colors) using the colours of the lower and upper boundaries of the fixed area. After finding the fixed area with erosion, filtering applied to the image such as closing technique. In image processing, closing is, together with opening, the basic workhorse of morphological noise removal. Opening removes small objects, while closing removes small holes. Closing fills small holes inside the foreground objects, or small black points on the object. The contour of fixed white area is determined and the coordinates of a rectangle's corners are determined. Next the width of the new image, which will be the maximum distance between bottom-right and bottom-left x-coordinates or the top-right and top-left x-coordinates is determined. The height of the new image, which will be the maximum distance between the top-right and bottom-right y-coordinates or the top-left and bottom-left y-coordinates is computed. Next the dimensions of the new image are determined and the set of destination points to obtain a "bird eye view" of the image are found. These data are used to specify points in the top-left, top-right, bottom-right, and bottom-left order. Finally save the image of fixed white area.

The capturing of the image of the world can be explained in detail as follows. Image capturing is handled by the on board camera. Captured image is transferred to the host computer for future processing. Image is read, by default OpenCV uses the RGB color model which is an additive color model in which red, green and blue light are added together in various ways to reproduce a broad array of colors. The name of the model comes from the initials of the three additive primary colors, red, green and blue, in order to make the image processing easier RGB model is then converted to HSV model which is the two most common cylindrical-coordinate representations of points in an RGB color model. This representations rearrange the geometry of RGB in an attempt to be more intuitive and perceptually relevant than the Cartesian representation. Then the image is thresholded using the colors of the obstacles and a closing rectangle of the surface is calculated. The image is blurred to get rid of imperfections and the contours of the surface is calculated. Perspective correction is applied to account for the location and height of the robot. The resulting image is saved for further processing.

Fig. 4. Map of the world

Surface area is extracted from the above process and then further processed to find the obstacle on the surface. It is done by applying canny edge detection (Fig. 5). Canny edge detector is an edge detection operator that uses a multi-stage algorithm to detect a wide range of edges in images. Then corner harries detection is applied which is an approach used within computer vision systems to extract certain kinds of features and infer the contents of an image in this case corners of the obstacles are detected. After, the corner detection the contours are applied to rectangles around and inside obstacles to describe safe area (Fig. 6).

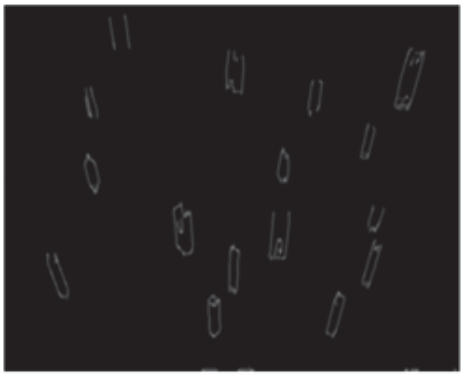

Fig. 5. Edge detection of objects

In next step, we crop our fixed area which is 640x480 pixels, into 20x20 pixels 768 pieces. After that we allocate the images into positive and negative folders. Positive area represents the area without obstacles. Negative area represents the area with obstacles. Mostly black images are area without

Authorized licensed use limited to: TUFTS UNIV. Downloaded on December 17,2024 at 21:11:14 UTC from IEEE Xplore. Restrictions apply.

obstacles and white images are obstacles. Using positive and negative images a dataset created to train the SVM.

Fig. 6. Result of edge detection

In the paper SVM is trained to detect obstacles and classify the image into areas with obstacles and without obstacles. Image histograms are used as input features for training the SVM. SVM training algorithm builds a model that assigns new examples into one category or the other, making it a non-probabilistic binary linear classifier. The system recognizes the image that has obstacle or does not have obstacles. After that we created a binary matrix which represents these fixed areas. In this matrix, 1 is area with obstacle and 0 is area without obstacle.

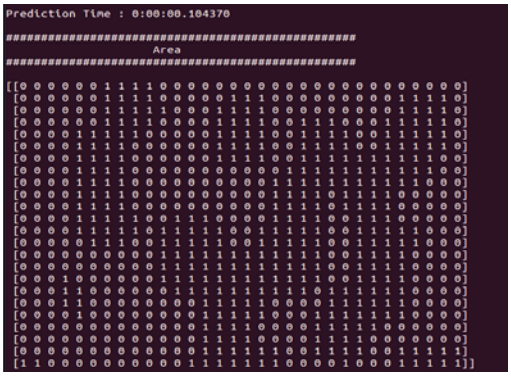

Fig. 7. Binary matrix obtained from the image

A* search algorithm is applied to the binary matrix to determine shortest possible path. Because A* search algorithm gives us shortest path and this path sometimes may be too close to the obstacles. During avoidance the obstacles by NAO robot the safe region is assigned to by-pass the obstacles carefully.

The path finding is applied to find a path to guide the robot to its destination, as the fixed area is classified into the areas with- and without obstacles.

Once the above steps are done. Mobile robot has the ability to detect new obstacles on the surface. At this point pathfinding can be applied to the surface using the previously trained SVM. Designed system works by taking picture of the surface detecting obstacles using the trained SVM which returns a binary matrix of the surface then applying A* pathfinding algorithm on the surface. Then the robot moves to the beginning position which is denoted by an orange marker. Finally robot follows the path calculated.

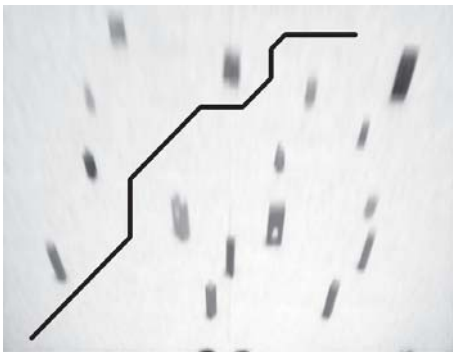

Fig. 8. Result of path finding

All the operations are combined and depicted in Fig. 9.

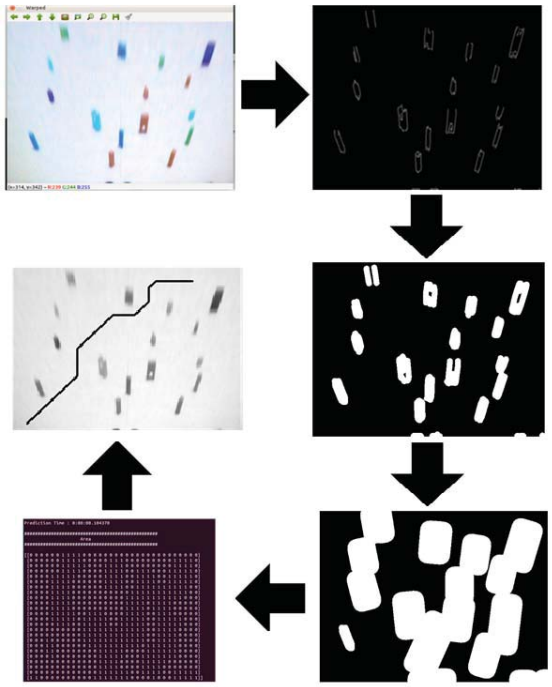

Fig. 9. The stages of image processing and path finding

VI. Conclusions

Robot navigation system based on Support Vector Machine and A* pathfinding algorithms are designed and used in real time application using NAO robot. The structure of designed robot navigation systems is presented. The functions of their main steps are described. SVM is chosen for classification of the images. A support vector machine which is a supervised learning algorithm is used to analyze data and create model of the real world. Results of SVM classification is used in path

Authorized licensed use limited to: TUFTS UNIV. Downloaded on December 17,2024 at 21:11:14 UTC from IEEE Xplore. Restrictions apply.

finding. After classification of world map, A* algorithm is applied for path finding problem. The image processing techniques, SVM algorithm and A* path-finding algorithm are used in navigation of NAO robot. Designed system relies on only vision data for navigation. This allows the mobile robot to navigate in environments where other traditional sensors (sonars, magnetometers etc.) won't work or does not work reliably. Designed systems applications include industrial automation of mobile robots in hazardous environments. For future work, the designed system will be modified to work with dynamic moving obstacles.

REFERENCES

[1] Borenstein, J., Everett, H. R., Feng, L., Navigating Mobile Robots: Systems and Techniques. A K Peters, Wellesley, MA.. 1996

[2] M. Yousef Ibrahim, Allwyn Fernmdes. Study on Mobile Robot Navigation Techniques. IEEE International Conference on Industrial Technology. Vol. 1, 2004,December 8-10p.230–236.

[3] Sezer, V., & Gokasan, M. (2012). A novel obstacle avoidance algorithm:"Follow the Gap Method". Robotics and Autonomous Systems

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

[5] 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.

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

[7] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," in IEEE Robotics Automation Magazine, vol. 4, 1997.

[8] M. Y. Ibrahim, "Mobile robot navigation in a cluttered environment using free space attraction "agoraphilic" algorithm", Proc. of the 9th Int. Conf. on Computers and Industrial Engineering, vol.1,pp.377–382, 2002.

[9] 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.

[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, 2010.

[11] Rahib H.Abiyev, Irfan Günsel, Nurullah Akkaya, Ersin Aytac, Ahmet Çağman, Sanan Abizada. Robot Soccer Control Using Behaviour Trees and Fuzzy Logic. 12th International Conference on Application of Fuzzy Systems and Soft Computing, ICAFS 2016 Book Series: Procedia Computer Science Volume: 102, 2016, Pages: 477-484

[12] S. M. LaValle, J. J. Kuffner, "Randomized kinodynamic planning," in Int. Journal of Robotics Research, vol. 20(5), pp. 378–400, 2001.

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

[14] Rahib H.Abiyev, Nurullah Akkaya, Ersin Aytac, Dogan Ibrahim. Behaviour Tree Based Control For Efficient Navigation of Holonomic Robots. International Journal of Robotics and utomation, Volume: 29 Issue: 1, 2014, pp: 44-57

[15] Rahib H. Abiyev, Nurullah Akkaya, Ersin Aytac, Irfan Günsel, and Ahmet Çağman. Improved Path-Finding Algorithm for Robot Soccers. Journal of Automation and Control Engineering, Vol.3, No.5, October 2015.

[16] Rahib H.Abiyev, Nurullah Akkaya, Ersin Aytac. Control of Soccer Robots using Behaviour Trees. ASCC, June, 2013, Istanbul.

[17] B. Erin, R. Abiyev, and D. Ibrahim. Teaching robot navigation in the presence of obstacles using a computer simulation program. Procedia – Social and Behavioral Sciences. Elsevier, Volume 2, Issue 2, 2010, pp. 565-571.

[18] Hart, P. E.; Nilsson, N. J.; Raphael, B. (1968). "A Formal Basis for the Heuristic Determination of Minimum Cost Paths". IEEE Transactions on Systems Science and Cybernetics SSC4. 4 (2): 100–107.

[19] Wong Yuen Loong, Liew Zhen Long and Lim Chot Hun. A star path following mobile robot. IEEE 4th Inter. Conf. on Mechatronics (ICOM), 17-19 May 2011, Kuala Lumpur, Malaysia

Authorized licensed use limited to: TUFTS UNIV. Downloaded on December 17,2024 at 21:11:14 UTC from IEEE Xplore. Restrictions apply.