Int. J. Engng Ed. Vol. 26, No. 1, pp. 18–29, 2010
Printed in Great Britain.

0949-149X/91 $3.00+0.00

EDURobot: an Educational Computer Simulation Program for Navigation of Mobile Robots in the Presence of Obstacles*

R. ABIYEV[1], D. IBRAHIM[2], B. ERIN[3]
Near East University, Department of Computer Engineering, Mersin 10, Turkey.
E-mail: [1]rahib@neu.edu.tr [2]dogan@neu.edu.tr [3]besime@neu.edu.tr

In engineering education different experimental exercises are provided to students in order to improve the teaching level and provide better understanding of theoretical materials. These exercises might be implemented using educational software tools. In this paper, an educational software tool called EDURobot has been developed to enhance the understanding of robotics for undergraduate and graduate students in computer and electrical and electronic engineering departments. The software tool mainly teaches students the navigation problems of a mobile robot avoiding obstacles in a static environment using different algorithms. The simulation environment is of a menu-driven variety where students can draw obstacles of standard shapes and sizes and assign the starting point of the mobile robot. The robot will then navigate among these obstacles without hitting them and reach the goal point given by the user. Parameters associated with the different algorithms may also be changed to observe their effects which will further enable comprehension of characteristics of different path planning algorithms. The program is developed in Visual C# for Windows platforms. Different algorithms employed in the software are potential field, vector field histogram plus and local navigation.

Keywords: robotics education; robot navigation; obstacle avoidance; path planning

1. INTRODUCTION

IN ENGINEERING EDUCATION one of the basic problems is understanding theoretical knowledge gained in engineering courses and its use in practice. For this purpose, various experimental exercises or real physical training systems are being developed for education of students. Because of the use of special hardware resources and their expensive nature the design of physical systems has become difficult. One efficient way is the use of different educational tools for improving the theoretical and practical background and also the educational level of undergraduate and graduate students. In this paper the development of an educational tool for robotics courses is considered.

Engineering education has gained impetus by explosion of the pace of development of information technologies. Because of this, students have become exposed to various educational platforms starting from the first year. The computer program which we are introducing in this paper serves the purpose of increasing the understanding of students as to how robots can move in a predetermined environment avoiding obstacles of standard shapes and sizes, utilizing different navigation and path-planning algorithms. The objective of the software is to let the students understand the main idea behind some of the well-known robot navigation algorithms and the differences in performance between them.

Different types of commercially available computer simulators are available in all branches of science and engineering education. Some examples are: electronic simulation package B^2 Spice [1], microcontroller electronic simulator Proteus [2], a chemical engineering simulator MDDS [3], a graphical simulation software for complex dynamic systems VisSim [4], a powerful mathematical tool for simulating all types of complex engineering systems MATLAB [5, 6, 7], and many more. The use of mobile robots in robotics education has gained popularity over the last two decades. Different applications can be discussed for mobile robot navigation which include trajectory tracking, target tracking, navigation, path planning and obstacle avoidance. Examples of such applications include behaviour-based navigation of a vehicle which has a combination of behaviours, including trajectory tracking [8, 9], target tracking [10], obstacle avoidance [11–13], landmark recognition systems and soccer robot navigation for mobile robot navigation [14]. Nowadays fuzzy logic-based mobile robot navigation is used both for indoor and outdoor applications of structured laboratory environments.

* Accepted 2 November 2009.

Researchers have proposed various control strategies for unknown and dynamic real world conditions using internet-based mobile robot navigation, an example of which is the enabling of users to remotely control a robot manipulator or a mobile robot.

Many educational institutions are now offering courses in robotics. For example, Carnegie Mellon University (CMU) offers undergraduate, as well as MS and Ph.D. level courses in robotics [15]. A summer camp program, RoboCamp [16], is offered by the National Robotics Engineering Consortium with the aim of teaching the practical aspects of building and programming mobile robots during the summer vacation. Another summer vacation educational robotics program is the Andrew's Leap [17], run by the CMU with the aim of teaching the programming of mobile robots to students. In addition to teaching, many colleges and universities organize robot competitions, such as robot navigation and speed competitions [18], maze navigation [19] competitions and so on.

Mobile robots systems are usually driven by microcontroller-based control systems. The architecture of these control systems provides different functionalities to accomplish specific robotic tasks. Traditionally, research in robotics is focused on the algorithms used to accomplish fundamental activities, such as path planning and robot navigation.

Robotics simulation is not new and several educational establishments with varying capabilities have been developed by various researchers in the past in order to inspire students' interest in mobile robotics and motivate them to participate actively in the learning process.

The educational robotic system presented in [20] has a multi-layered architecture. It has three types of tutoring tours: fully-guided, unguided and guided. The fully guided tour is a demonstration tour that demonstrates basic concepts without any intervention by the student. The unguided tour, or free tour, does not determine any order or any tasks at all. This tour provides generic tools to the user and lets him/her customize the experiment according to needs. These generic tools include a 2D model for the robot and the lab, an odometry data panel, a sonar data panel, a laser data panel, a motion controller and a network camera.

The guided tour aims to present different specified courses with direct interaction between the student and the tutor. The authors show that the EDURobot allows students to run any type of simulation of path planning algorithms with any set of desired parameters. In this system, commands are given to send the robot to a certain room and it is generated using the sequencer by combining various simple skills such as GoTo-Point, FindDoorLaser, Approach-Door, Cross-Door and WallFollowing. In addition, a low-level commands editor is now under development to facilitate interaction with the remote robot using low-level commands or a group of commands in the form of a programming script. The user will be able to teleprogram the robot or to send low-level commands via the Internet to the robot's base server to perform many tasks, such as adjusting the robot control parameters, increasing/decreasing robot velocity, reporting battery state, setting the Watch Dog timer, that can be used for safety or diagnostic purposes, etc.

The educational mobile robot platform MBR-01 presented by Gunes & Baba [21] was designed as an educational tool for the course 'Control and Robotics'. Its aim is to teach the fundamental skills of control systems, autonomous robots, FLC, image processing, edge detection, RF communication techniques and internet-based remote control. The speed control of MBR-01 has a trajectory-tracked curvature using fuzzy algorithms. This simulator gives students the ability to control the speed and position of an autonomous mobile robot, thus providing experience of using a real environment situation. This software develops and simulates students' cognitive skills since, after each attempt to find the correct parameters, students are required to develop strategies. It can be said that the accelerated learning gives students an invaluable understanding of the course in a short space of time, providing a safe environment for teaching and learning.

The MBR is controlled either manually or automatically on the variable trajectory. In manual operation, the speed reference value is entered into the PC screen and position control can be achieved with constant speed on the selected trajectory. The fuzzy edge detection user interface allows students to have an overall understanding of fuzzy edge algorithms, to simulate and modify every phase of the operation at anytime, to observe the effects of a change in the membership functions on the control action, to observe each active rule and its output value and to investigate its effects on the controller.

System RoboLab [22] allows students to work with a simulation of an industrial robot and carry out operations with a real robot through tele-operation. Students perform exercises on the simulated virtual environment and then, after checking that the results are correct, they can execute them in the real system by means of the tele-operation option. Students are able to practice and carry out correct movement sequences. Once a correct simulation has been effected, students can request the 'main server' to remotely execute the movement sequences with the real robot. Although the user interface is always available for executing simulations in the virtual environment, students must identify themselves as authorized to use the tele-operation capabilities. The 'main server' also manages the access of several users to the robots, guaranteeing their orderly access.

A computer program named RoboKol is introduced in [23], which performs path planning for redundant manipulators and mobile robots using potential field-based algorithms. Through a

friendly user interface, the user can interact with the program and can perform a variety of operations such as drawing obstacles and robots on the screen, obtaining two- and three-dimensional images of the potential field and performing robot simulations. Although RoboKol was originally developed for research purposes, it is a convenient tool to teach potential fields, path planning for redundant manipulators and mobile robots in engineering education.

Much previous work in robotics education concentrates on actual robot movements rather than the path planning or obstacle avoidance methods. The development of an educational software tool that includes several methods allows students to compare and analyse their results and improve their understanding of the problems of robotics. In this paper three different algorithms—potential field, vector field histogram and local navigation—have been implemented for robot navigation problems. The development of such a system will allow students to a better understanding of the problems of obstacle avoidance and their solution mechanisms. We have designed a simulator for robot path planning which basically draws a path for a mobile robot in face of obstacles. This path may be modified with respect to changes to various parameters for each of three different algorithms. The start and goal positions may also be modified and the simulation run again as many times as desired.

The paper is organised as follows: in section 2, potential field, vector field histogram plus, and local navigation algorithms are described. Section 3 describes features of the simulation software. Section 4 includes simulation results and analysis of obtained results. Section 5 discusses pedagogical assessment of the tool and its impact on teaching and learning. Finally, Section 6 discusses conclusions and suggestions for future work.

2. MOBILE ROBOT NAVIGATION ALGORITHMS

Navigation is one of the basic problems in robotics. The research paper [24] classifies robot navigation algorithms as global and local, depending on surrounding environment. In global navigation, the environment surrounding the robot is known and the path which avoids the obstacles is selected. Here graphical maps which contain information about the obstacles are used to determine a desirable path [25–28]. The global navigation problem is solved using different path planning algorithms. In local navigation, the environment surrounding the robot is unknown, or partially known, and sensors are used to detect the obstacles; a collision avoidance system is incorporated into the robot to avoid these obstacles.

In this paper global navigation using path planning algorithms and local navigation algorithms are modelled for a mobile robot to avoid obstacles. Path planning methods for mobile robots is based on the idea of finding an optimal and smooth path consisting of many points close enough to each other avoiding obstacles. It is possible to have a path consisting of spline curve segments. The software which is the subject of this paper utilizes three methods.

2.1 Potential field method

Robot dynamic characteristics and navigation law are important in path planning, where information about location of obstacles is used to determine the desirable path. One of the path determination methods used in the simulator software is the potential field method (PFM) [25]. The philosophy of the potential field approach is that the mobile robot moves in a field of forces. The goal position to be reached is an attractive potential while each obstacle generates a repulsive potential. A potential field can be viewed as an energy field and so its gradient at each position is a force. Potential fields can be applied locally while path determination, trajectory planning and control steps are achieved in one step in real time.

The idea of obstacles exerting virtual repelling forces towards a robot, while the target generates a virtual attractive force uses a similar concept that takes into consideration the robot's velocity in the vicinity of obstacles. In one example called the Brooks implementation [27], if the magnitude of the sum of repulsive forces exceeds a certain threshold, the robot stops, turns into the direction of the resultant force vector and moves on. This method also requires the robot to stop and thus is not suited for tele-autonomous operation. The theory of the potential field method is given below briefly.

A potential, $\phi(r)$ is defined by the Laplace equation $\nabla^2\phi$ in a closed region, Ω, of continuous, equal connectivity. The boundary of Ω, $\partial\Omega$ does not have to be connected. It includes the surfaces of all obstacles and the goal point. $\phi(r) = \phi_2$ at the surfaces of obstacles and $\phi(r) = \phi_0$ at the goal point. There are no local minima on $\phi(r)$. Nevertheless, the exponential decay of the field from any point leads to areas where the magnitude of the gradient on ϕ, $|\nabla\phi|$, is very small while the range of $|\nabla\phi|$ may be very large. The field decays rapidly near the goal, and far from the goal there is only a slight change in the field.

In this project, the potential value is calculated for each point on the grid by using the Laplace equation where the value of the field at the goal point is set to the value of -2^{124} and boundary points are zero.

The Laplace equation in two dimensions is represented on an equally spaced and connected grid by the following partial differential equation:

$$\phi_{(i,j)} = \frac{\phi_{(i+1,j)} + \phi_{(i-1,j)} + \phi_{(i,j+1)} + \phi_{(i,j+1)}}{4} \quad (1)$$

where i is position on the grid in the x direction, j is position on the grid in the y direction. The

potential value given above for each grid point is referred to as the gridValue, a two-dimensional array. gridSize is the parameter that determines the length and width of the grid cell.

Field values are calculated for any point in the workspace by using linear interpolation.

Passing two parameters *a* and *b* into the function field, we use the following sets of equations to determine the field value at any given point that is anywhere on the canvas (i.e. can be anywhere within the grid). In this algorithm, *gridSize* is the length and width of each grid cell. *numberofGridLinesX* and *numberofGridLinesY* are the number of grid cells along the X and the Y axes. *dx* and *dy* are the differential values for x and y values. *sx* and *sy* are simply approximations. *bot1* and *bot2* are calculated values which eventually are used to calculate the *potential field* value at any given point, i.e. this is linear interpolation. Computing of potential field is given in the following fragment:

```
x = a / gridSize
y = b / gridSize;
gridx = numberOfGridLinesX + 1
gridy = numberOfGridLinesY + 1

dx = x—Floor(x)
dy = y—Floor(y)
sx = Floor(x)
sy = Floor(y)
bot1 = (1.0—dy) * gridValue[sx, sy] + dy *
   gridValue[sx, sy + 1]
bot2 = (1.0—dy) * gridValue[sx + 1, sy] + dy *
   gridValue[sx + 1, sy + 1]

field(a,b) = dx * bot2 + (1.0—dx) * bot1
```

The mathematical function *Floor(x)* returns the largest integer less than or equal to the specified double-precision floating-point number.

One problem with potential fields is that they may have local minima where the robot gets trapped before reaching the goal. Another situation which may cause local minima is closely spaced obstacles. Because of this, the potential field approach can also be applied globally by means of numerical potential fields or navigation functions that can be defined on a grid without local minima. Another problem with potential fields is unstable oscillation where the dynamics of the robot and/or the environment becomes unstable. This may be caused by high speeds, narrow corridors or sudden changes in movement.

Since a point robot can be defined as a mobile robot with no dimension, it can simply reach the goal by following the gradient descent. The mobile robot is moved towards the goal in such a way that the centre of the robot is moved from the current point to the next point on the path while the orientation of the robot is taken as the tangent angle of the current point.

Finding the tangent angle at each point of a given path and then calculating the updated robot position is a simple process as specified below:

$$top = field(robx, roby - 1)$$
$$bottom = field(robx, roby + 1)$$
$$left = field(robx - 1, roby)$$
$$right = field(robx + 1, roby)$$

robx and *roby* are the current x and y positions of the mobile robot respectively.

$$\alpha = \tan^{-1}\frac{top - bottom}{left - right} \tag{2}$$

$$x_{\min} = lineLength * \cos(\alpha)$$
$$y_{\min} = lineLength * \sin(\alpha)$$
$$robx = robx + x_{\min}$$
$$roby = roby + y_{\min}$$

where, *lineLength* is a parameter which sets the distance the mobile robot travels at each simulation tick and the routine *field*(x,y) calculates the potential field value given by the gridpoint x and y.

2.2 Vector field histogram plus method

The Vector Field Histogram Plus (VFH+) method [26] includes four stages for computing direction of robot motion. In the first three stages the two-dimensional map grid is transformed into one-dimensional polar histograms. These are implemented using a primary polar histogram, a binary polar histogram and a masked polar histogram. In the last stage, using a masked histogram and cost function the algorithm selects the suitable direction for the robot. A brief description of these stages is given below.

In the VFH+ method, there exists a circular window with diameter w_s where the robot scans its environment. This forms our histogram grid which has dimension w_s x w_s. In this histogram grid, each cell has a certainty value $c_{i,j}$ which has the value 1 where we are confident that there exists part of an obstacle and has value 0 where there is no obstacle. In the developed algorithm where obstacles are represented as rectangles and ellipses in a static environment, the circular window mentioned above is obtained from a two-dimensional array called savepoint which holds the information whether each cell is part of an obstacle or not. Next step is to build a primary polar histogram. In order to do that, we need to calculate the vector magnitude $m_{i,j}$ of each cell.

$$m_{ij} = c_{ij}^2\left(a - bd_{ij}^2\right) \tag{3}$$

Where, $c_{i,j}$ is the certainty value of active cell, $d_{i,j}$ is the distance from active cell to the RCP (Robot Centre Point) and the parameters a, b are chosen according to the following equation:

$$a - b\left(\frac{w_s - 1}{2}\right)^2 = 1 \tag{4}$$

Based on the obstacle vectors, the primary polar histogram H^p is built. H^p has an angular resolution α so that $s = 360/\alpha$ is an integer, resulting in a fixed number of angular sectors over which the histograms are built. The method uses an analytically determined low-pass filter to compensate for the width of the robot. Obstacle cells are enlarged by the $r_{r+s} = r_r + d_s$ where r_r is the robot radius and d_s is the minimum distance between the robot and the obstacle. With the obstacle enlarged by r_{r+s}, the robot can be treated as a point-like vehicle. The width compensation method is implemented efficiently by enlarging the obstacles while building the primary polar histogram. The equation to determine primary polar histogram can be calculated using the enlargement γ_{ij} angle, vector direction β_{ij} and polar obstacle density H_k^p calculated for each k-th sector.

$$\gamma_{i,j} = \arcsin\left(\frac{r_{r+s}}{d_{i,j}}\right); \quad \beta_{i,j} = \arctan\left(\frac{y_j - y_0}{x_i - x_0}\right) \tag{5}$$

$$H_k^p = \sum_{i,j \in C_a} m_{i,j}.h'_{i,j} \tag{6}$$

where

$$h'_{i,j} = \begin{cases} 1, & \text{if } k \cdot \alpha \in \left[\beta_{i,j} - \gamma_{i,j}, \beta_{i,j} + \gamma_{i,j}\right] \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

Here, x_o , y_o are present coordinates of the Robot Centre Point, and x_i, y_i are coordinates of the active cell $c_{i,j}$.

The result of this process as mentioned above is a primary polar histogram that takes into account the width of the robot. The h' function also serves as a low-pass filter and smoothes the primary polar histogram. Since the histogram is built around the current robot position, independent of its orientation, this first stage of the algorithm can be implemented very efficiently by the use of tables of the size w_s x w_s. Based on the primary polar histogram H^p and the two thresholds, τ_{low} and τ_{high}, a binary polar histogram (H^b) is built. The sectors of H^b are either blocked (1) or free (0). The binary polar histogram indicates which directions are free for a robot that can instantaneously change its direction of motion. At time n, the binary polar histogram are updated by the following rules.

$$H_{k,n}^b = \begin{cases} 1, \text{ if} H_{k,n}^p > \tau_{\text{high}}, \\ 0, \text{ if} H_{k,n}^p < \tau_{\text{low}}, \\ H_{k,n-1}^b, \text{ otherwise} \end{cases} \tag{8}$$

The VFH+ method uses a simple, but closer approximation of the trajectory of most mobile robots. It assumes that the robot's trajectory is based on circular arcs (constant curvature curves) and straight lines. The maximum trajectory curvature of a mobile robot is often a function of the robot velocity. The faster the robot travels, the smaller the maximum curvature. The minimum steering radius can be zero for a differential drive robot if it has zero traversal speed otherwise it has a certain value which is fixed in our software. There is another parameter on the other hand which is called minDistance(the minimum distance between the robot and the obstacle) that can be varied between 1 and 10. It is used in the building of primary polar histogram. The next stage after binary polar histogram is the masked polar histogram that shows which directions of motions are possible. It is determined according to the following rules:

Determine φ_b. Set φ_r and φ_l equal to φ_b.

Here φ_l is left angle, φ_r is right angle, and $\varphi_b = \theta + \pi$ is defined as direction opposite to the current direction. The limit angles φ_l and φ_r are obtained by checking every active cell using the following two conditions:

An obstacle cell blocks the directions to its right if:

- Condition 1: $d_r^2 < (r_r + r_{r+s})^2$

An obstacle cell blocks the directions to its left if:

- Condition 2: $d_l^2 < (r_l + r_{r+s})^2$

For every cell $C_{i,j}$ in the active window C_a with $c_{i,j} > \tau$

If $\beta_{i,j}$ is to the right of θ and to the left of φ_r, check condition 1. If condition is satisfied, set φ_r equal to $\beta_{i,j}$

If $\beta_{i,j}$ is to the left of θ and to the right of φ_l, check condition 2. If condition is satisfied, set φ_l equal to $\beta_{i,j}$

If the robot's sensors are not very reliable, φ_r and φ_l could also be determined in a more stochastic way. Instead of comparing the cell certainity values to a threshold, we could build a polar histogram whose sector values indicate the certainity that a sector is blocked because of robot dynamics. The values for φ_r and φ_l could then be determined by applying a threshold to this histogram. As the first method is more efficient, the second method should only be applied if really necessary.

With φ_r, φ_l and the binary polar histogram, the *masked polar histogram* can be built.

$$H_K^m = 0 \text{ if } H_K^b = 0 \text{ and } (k, \infty) \, \epsilon\{[\varphi_r, \theta], [\theta, \varphi_l]\} \tag{9}$$
$$H_K^m = 1 \text{ otherwise}$$

Fourth stage is the selection of the new steering direction and is done according to the following rules:

The masked polar histogram shows which directions are free of obstacles and which ones are blocked. However, some free directions are better candidates than others for new direction of motion. The VFH+ method first finds all openings in the masked polar histogram and then determines a set of candidate directions. A cost function that takes into account more than just the differ-

ence between the candidate and the target direction is then applied to these candidate directions. The candidate direction k_d with the lowest cost is then chosen to be the new direction of motion. In the case of a goal-oriented robot, which is our case, we obtain between one and three candidate directions for each opening in the masked polar histogram.

Next we need to define an appropriate cost function that selects the new direction of motion φ_d. The cost function has three terms. The first term is responsible for goal-oriented behaviour while the second and the third terms make the mobile robot *commit* to a direction. The cost function is as follows:

$$g(c) = \mu_1 \, \Delta(c_i k_i) + \mu_2 \, \Delta\left(c_i \frac{\theta_a}{\alpha}\right) + \mu_3 \, \Delta\left(c_i k_{d,n-}\right) \tag{10}$$

where,

$$\Delta(c_1, c_2) = \min\{|c_1 - c_2|, |c_1 - c_2 - s|, |c_1 - c_2 + s|\}$$

The first term of our cost function *g(c)* represents the cost associated with the difference of a candidate direction and the target direction. The larger this difference is, the more the candidate direction will guide the robot away from its target direction, and hence the larger the cost.

The second term represents the cost associated with the difference of a candidate direction and the robot's current wheel orientation. The larger the difference is, the larger the required change of the direction of motion.

The third term represents the cost associated with the difference of a candidate direction and the previously selected direction of motion. The larger the difference is, the greater the change of the new steering command.

Only the relationship between the three parameters is important. To guarantee a goal- oriented behaviour, the following condition must be satisfied:

$$\mu_1 > \mu_2 + \mu_3.$$

If a smooth path is more important than variations in the steering commands, then μ_2 should be set higher than μ_3. If smoothness of the steering commands is more important, then μ_3 should be set higher than μ_2. Experiments have shown that a good set of parameters for a goal-oriented mobile robot is:

$$\mu_1 = 5, \mu_2 = 2, \mu_3 = 2.$$

2.3 Local Navigation Method

A Local Navigation Method (LNM) with obstacle avoidance is considered for mobile robots in which the dynamics of the robot are taken into consideration. The goal is known but the geometry and the location of the obstacles are unknown. The mobile robot position is represented by the Cartesian coordinates and can move in three directions, forward, left or right. The starting point and goal points of robot are given. Using these points, the directional angle of robot $\theta(t)$ $(0 \leq \theta(t) \leq 2\pi)$ is determined. There may be obstacles in the plane of motion and the objective is to navigate the robot to the goal avoiding the obstacles.

To determine optimal path the following navigation law is used [24]:

$$\dot{\theta}(t) = -\eta[\theta(t) - \theta^*(t)] \tag{11}$$

Where $\theta(t)$is current directional angle of robot, $\theta^*(t)$ is desirable path, η is positive constant.

The problem is to navigate the robot to its goal avoiding obstacles, by switching $\theta^*(t)$ based on the information available from three—left, centre and right—distance sensors.

The only information needed about the local environment is the distance of obstacles determined in three directions: distance to the obstacle a degrees to the left of the center, d_l , distance to the obstacle a degrees to the right of the center, d_r , distance to the obstacle along the path to the goal, d_c. If the sensors measure a distance further than d_{max}, the sensor returns a –1 value.

The angle $\theta^*(t)$ at which the robot will travel is determined according to the values of the d_l (distleft), d_r (distright) and d_c (distcentre). The angle at which the robot will turn is given by $\theta_\alpha(t)$ (thetaalpha) while the angle to the goal is given by $\theta(t)$ (theta). $\theta_\alpha(t)$ is determined as:

```
if (distcenter > 0)
  if ((distleft < 0) and (distright < 0))
    thetaalpha = theta + pi/2;
  else
    if ((distleft < 0) or (distright < 0))
  thetaalpha = theta + Sign(distleft—distright)
*(pi/2—epsilon) + Sign(distleft—distright) * pi;
else thetaalpha = theta +Sign(distleft—distright) *
(pi/2—epsilon);
  else
    thetaalpha = theta;
```

here, *pi =3.14,*
epsilon = $\tan^{-1}((\text{dcos} \propto - \text{distcentre})/\text{dsin} \propto)$
$\theta(t)$ is determined as

thetat = $\tan^{-1}\left(\frac{goalY - roby}{goalX - robX}\right)$

```
if (thetat < = theta)
thetat = thetat + pi
else thetat = thetat—pi
```

The new angle at which the robot will travel is determined as:

```
  if ((distcenter < 0) and ((distleft > 0) and(dist-
right > 0)))
    thetanew = thetaalpha;
  else if (distcenter > 0)
  if (((thetat > thetaalpha) and (thetaalpha >
theta)) or ((thetat < = thetaalpha) and (thetaalpha
< = theta)))
    thetanew = thetat;
```

```
else thetanew = thetaalpha;
else if (((distleft < 0) and(thetat > = theta)) ||
((distright < 0) or (thetat < = theta)))
  thetanew = thetat;
else thetanew = thetaalpha;
theta = thetanew
```

The resulting path is modified with the parameters d_{max} and α. Where d_{max} = max(distleft,distright). As you adjust the parameter d_{max}, the user of the program can observe that the path will get away from the obstacle. As you drop alpha below 10^{o}, the user will see that there will be distortions in the path.

3. THE SIMULATOR SOFTWARE

The software package has been developed for implementation of path planning and navigation algorithms on the PC, using visual C# in Windows environment. The selection of navigation algorithm and algorithm parameters can be observed and modified via the user interface. The program menu includes File, Edit, Draw, Simulate, Settings, Parameters and Help pull-down menus. Under the File menu, there are the options New, Open, Save, Save As and Quit that will allow students to save the configuration to a file and later retrieve it. Under the Edit Menu, there is Cut, Copy, Paste, Cancel or Clear.

The procedure which was followed to develop the software was to first design a user interface where we can draw obstacles of standard shapes (rectangles and ellipses) and sizes using the Draw menu.

The next step was to develop a user interface under the Settings menu where the user can enter the starting and ending coordinates of the path of the mobile robot. Under this menu, there is also a dialogue box where students can enter the unit size of the grid and the number of grid lines along the x axis and the number of grid lines along the y axis as well as an interface which allows them to adjust the linelength between 0.1 and 1.

Linelength is the amount of length on the path undertaken before the algorithm checks the surroundings again.

Under the Simulate menu, students are given the option to run a potential field algorithm, adaptive navigation algorithm, or vector field histogram plus an algorithm on the same configuration. One of the nice points about the simulator is that the three algorithms can be run at the same time and this enables students to compare the results and learn more about the differences between the algorithms. The software also lets the user know about the running times of each algorithm and the number of points taken to reach the goal.

Since this software is created for educational purposes, a pull-down Parameters menu is created which allows students to change the appropriate parameters for each algorithm. For potential field method, the parameters are the number of iterations and gridSize (the length of each grid cell). For adaptive navigation, these parameters are alpha and dmax. For vector field histogram plus, the parameters are workspace size(w_s), b, alpha(the meaning of which is different from the alpha in adaptive navigation). Note the fact that these are the parameters which are allowed to change, not necessarily the only parameters within the algorithm.

4. SIMULATION RESULTS

During a typical run of the program using user interface, different obstacles can be created, and starting and goal positions of the robot can be selected. Then, an algorithm is selected and path planning or navigation of the mobile robot is performed. In all of the software runs, the gridSize has been taken as 1.0 and linelength has been taken as 0.1. gridSize is the length and width of each cell on the grid. The number of points generated indicates the total number of points required to reach the goal with the selected linelength. That is, the array that contains the path points has this number of members.

Figure 1 is a typical run of the EduRobot software with only three obstacles. As can be observed from the figure, all three obstacles were avoided successfully by the robot using all the algorithms. Local navigation and Vector Field Histogram (plus) algorithms gave similar results while potential field seemed to stay away from the obstacles. The fastest algorithm was the local navigation, followed by VFH+ and PFM, as can be observed from Table 1. PFM algorithm reached the goal using the minimum number of points, followed by the VFH+, and then the Local Navigation algorithm.

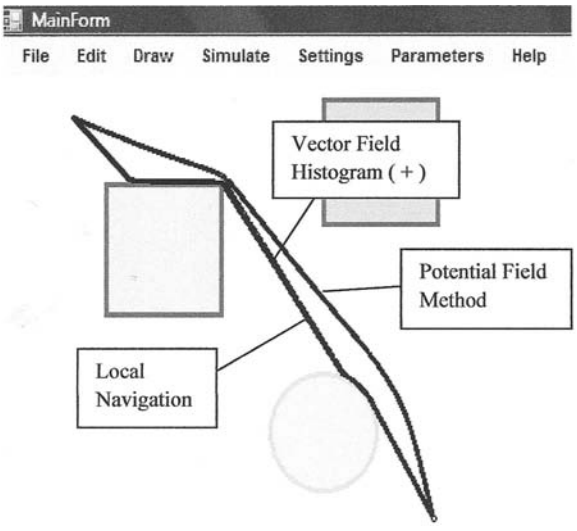

Fig. 1. Example with three obstacles showing the paths of three different algorithms. (Start position: 50,50; Goal position: 300,300).

Table 1. Statistics for Fig. 1.

Algorithm	Time taken to reach goal (ms)	Number of points generated
PFM	24024	5380
VFH+	10764	5652
LNM	608	5804

In Fig. 2, four obstacles are used and the three algorithms follow a different path having differed due to configuration and particularly sizes of obstacles. Potential Field and VFH+ follow one side and local navigation follows the other side of obstacles. The arrangement in terms of time taken to reach goal is same as the configuration in Fig. 1. The number of points generated in this case is least with local navigation, higher with PFM and the highest with VFH+. Table 2 shows the statistics for Fig. 2 with the fastest algorithm being the local navigation, followed by the VFH+.

In Fig. 3, five obstacles are considered in order to see differences between the simulation results of the algorithms. In this run, parameter alpha of local navigation is varied to obtain the correct obstacle-avoiding behaviour. Possibly due to this reason local navigation gives the highest number of points generated although it is the fastest, while VFH+ and PFM come next in number of points generated, with VFH+ being much faster than PFM.

The last simulation is shown in Fig. 4 and Fig. 5 where the parameters are changed from their default values. Figure 4 shows the results of the

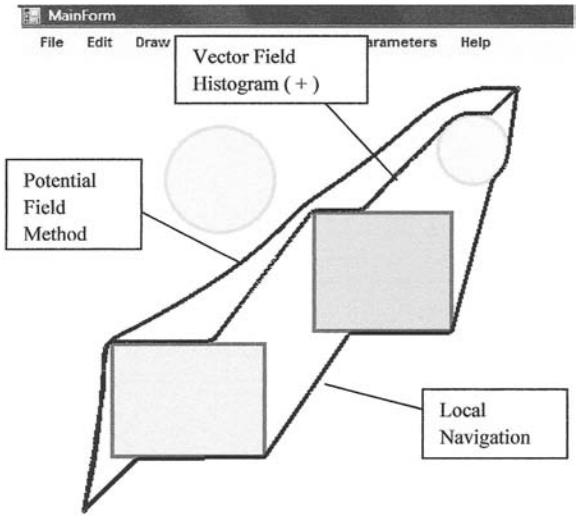

Fig. 2. Another configuration with four obstacles with the start and end points modified.

Table 2. Statistics for Fig. 2.

Algorithm	Time taken to reach goal (ms)	Number of points generated
PFM	26020	3718
VFH+	6474	3743
LNM	405	3759

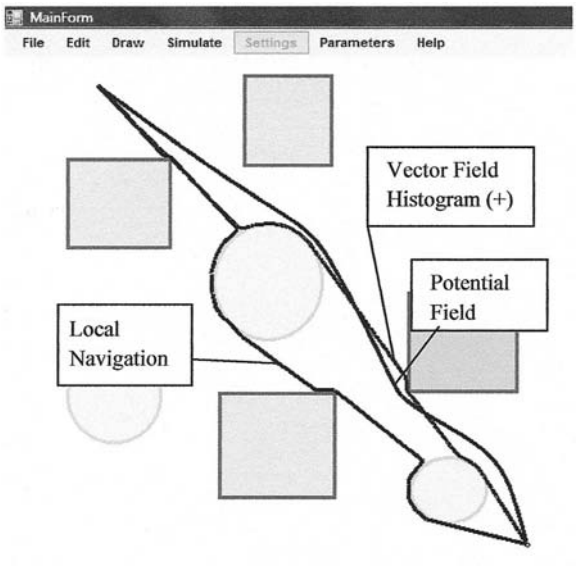

Fig. 3. A complex configuration where the path avoids up to five obstacles. (Start position: 80,50; Goal Position: 450,450).

Table 3. Statistics for Fig. 3.

Algorithm	Time taken to reach goal (ms)	Number of points generated
PFM	30763	5610
VFH+	9640	5650
LNM	702	6367

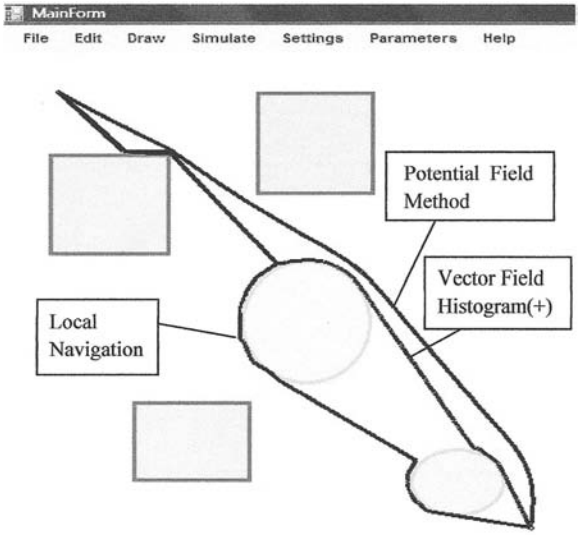

Fig. 4. Graph before parameters were changed. The parameters are: PFM: Number of Iterations: 1000, gridSize = 1.00. VFM(+): w_s = 12, minDistance = 1, b = 1.5, alpha = 2.0. LNM: d_{max} = 4 units, alpha = 85 degrees (It had to be fixed from 30 degrees to get the correct path), linelength = 0.1.

Table 4. Statistics for Fig. 4.

Algorithm	Time taken to reach goal (ms)	Number of points generated
PFM	2682	5431
VFH+	1325	5538
LNM	764	6304

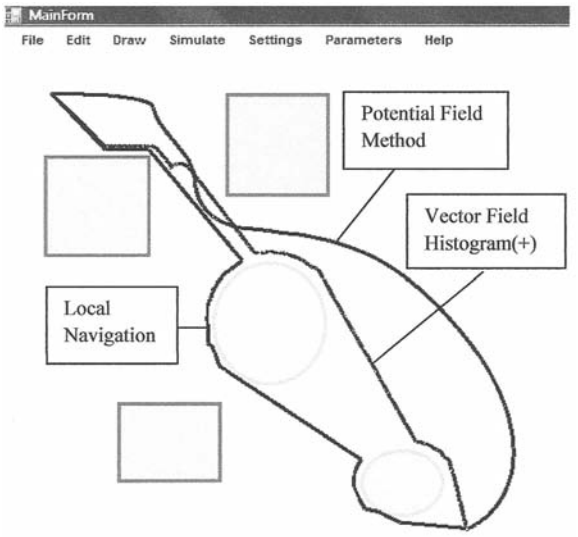

Fig. 5. Graph after parameters are changed. The parameters are: PFM: Number of Iterations: 1500, gridSize = 5.00. VFH(+): w_s = 22, minDistance = 10, b = 1.5, alpha = 2.0. LNM: d_{max} = 8.5 units, alpha = 85 degrees (It had to be fixed from 30 degrees to get the correct path), linelength = 0.1.

Table 5. Statistics for Fig. 5.

Algorithm	Time taken to reach goal (ms)	Number of points generated
PFM	45396	6436
VFH+	24070	5779
LNM	1544	6465

three algorithms before the parameters were changed and Fig. 5 shows the situation after the parameters were changed. The values of the parameters are shown underneath the figure.

What has happened is that in potential field method the values got out of line as the gridSize was increased to 5.00 from 1.00 and the time increased twofold as the number of iterations was increased from 1000 to 1500. For Vector Field Histogram plus the workspace size(w_s) was increased from 12 to 22 which dramatically increased the time taken to reach goal from 11325 msecs to 24070 msecs. The fact that we increased minDistance from 1 to 10 has taken the path away from the obstacles. The other parameters for VFH+ remained constant (namely b and alpha). Finally there is the Local Navigation parameters where the parameter d_{max} has been increased from 4 units to 8.5 alpha, for local navigation had to be increased from 30 to 85 degrees to fix the path. The change was again a slight movement of the path away from the obstacles.

5. CASE STUDY—OPINION OF STUDENTS

A pilot study was carried out at the Near East University to find out the opinions of students to using the EDURobot simulation program. The study consisted of a carefully prepared questionnaire, completed by undergraduate students who used the EDURobot simulation software as part of their normal lecture sessions during one month. The aim of the questionnaire was to learn the opinions of students about the usefulness of the system.

Forty-four questions from eight subjects were prepared using the Likert-5 scale. participants were given time to complete the questionnaires at the end of their training. Training included the presentation of theoretical and practical sections. In the theoretical section the navigation problem of mobile robots, the methodologies used for navigation were explained. After the theoretical section students attended their laboratory sessions. In these laboratories the EDURobot simulation software was used with different parameters.

After completion of the practical section, evaluation of the simulator software was done by students by completing the questionnaire. The results of the questionnaire were analysed using the SPSS statistical package and below is a summary of the important outcomes:

- A majority of students (85%) strongly agreed that the EDURobot simulation software is easy to use and is user friendly.
- Over 90% of students agreed that the effects of parameter changes can easily be observed.
- A majority of students (89%) agreed that the simulator had helped them to understand the functional differences between the three different algorithms.
- Over 94% of students agreed that they were able to thoroughly understand the PFM, VFH+ and LNM algorithms.
- Finally, all of the students agreed that a computer simulation is an effective way for them to learn.

6. CONCLUSIONS AND SUGGESTIONS FOR FUTURE WORK

The development of a computer simulation program called EDURobot for navigation of mobile robots in the presence of obstacles has been described. The different local and global navigation methods are considered for software application. The objective of the software was to improve student understanding of robotics navigation.

EDURobot is an interactive program used successfully in laboratory sessions at the Near East University. A software workspace is set by students using the GUI facilities of the software. The workspace consisted of drawing obstacles and robot starting and goal points on the screen. Various parameters of the algorithms can be adjusted by students and this helps them to

compare the advantages and disadvantages of the algorithms.

Results show that the software tool increased student knowledge and understanding of robotics and gave them a better insight into the various robotic path planning and navigation algorithms. Considering the interest and enthusiasm of students it is planned to include the developed tool as a permanent experiment of undergraduate robotic laboratory work.

REFERENCES

1. B2Spice electronic Simulation Software, Beige Bag Software, http://www.beigebag.com [Accessed June, 2009].
2. Proteus Simulation Software, Labcenter Electronics, http://www.labcnter.co.uk [Accessed June, 2009].
3. MDDS Simulator, http://dambiren.tripod.com/ mddssimSetup.zip [Accessed September, 2009].
4. J. A. Sokolowski and M. B. Catherine, *Principles of Modeling and Simulation: A Multidisciplinary Approach*, Wiley, 2009.
5. A. Gilat, *MATLAB: An Introduction with Applications*, Wiley, 2008.
6. B. Hahn and D. Valentine, *Essential MATLAB for Engineers and Scientists*, Newnes, 2007.
7. S. J. Chapman, *MATLAB Programming for Engineers*, CL-Engineering, 4th Ed., 2007.
8. T. Das and I. N. Kar, Design and implementation of an adaptive fuzzy logic-based controller for wheeled mobile robots, *IEEE Transactions on Control Systems Technology*, **14**, 2006, pp. 501–510.
9. G. Dongbing and H. Housheng, Receding horizon tracking control of wheeled mobile robots, *IEEE Transactions Control Systems Technology*, **14**, 2006, pp. 743–749.
10. R. C. Luo, T. M. Chen and K. L. Su, Target tracking using hierarchical grey-fuzzy motion decision-making method, *IEEE Transactions on Systems, Man. and Cybernetics-Part A: Cybernetic,* **13**, 2001, pp. 179–186.
11. C. Ye, N. H. C. Yung and D. Wang, A fuzzy controller with supervised learning assisted reinforcement learning algorithm for obstacle avoidance, *IEEE Transactions on Systems, Man. and Cybernetics-Part B: Cybernetics*, **33**, 2003, pp. 17–27.
12. T. Fairchard and P. Garnier, Fuzzy control to drive car-like vehicles, *Robotics and Autonomous Systems,* **34**, 2001, pp. 1–22.
13. M. D. Hurley, W. L. Xu and Glen Bright, Implementing fuzzy logic for machine intelligence: a case study, *Int. J. Eng. Educ*., **21**(5), 2005, pp. 178–186.
14. P. Vadakkepat, O. C. Miin, X. Peng and T. H. Lee, Fuzzy behaviour-based control of mobile robots, *IEEE Transactions on Fuzzy Systems,* **12**, 2004, pp. 559–564.
15. Robotics Institute of Carnegie Mellon University, http://www.ri.cmu.edu [Accessed July, 2009].
16. National Robotics Engineering Center, http://www.rec.ri.cmu.edu [Accessed July, 2009].
17. Andrew's Leap, http://www.cs.cmu.edu/~leap [Accessed August, 2009].
18. MIT Autonomous Robot Design Competition, http://web.mit.edu/6.270/www [Accessed August, 2009].
19. I. M. Verner and D. J. Ahlgren, Robot Contest as a Laboratory for Experiential Engineering Education, *Journal on Edu. Resources in Computing*, **4**(2), 2004.
20. A. Khamis, F.Rodriguez, R. Barber, M. A. Salichs, An Approach for Building Innovative Educational Environments for Mobile Robotics, *Int. J. Eng. Educ.* **22**(4), 2006, pp. 732–742.
21. Mahit Gunes, A.Fevzi Baba. An Approach for Building Innovative Educational Environments for Mobile Robotics, *Int. J. Eng. Educ.* **22**(4), 2006, pp. 732–742.
22. Fernando Torres, Francisco A. Candelas, Santiago T. Puente, Jorge Pomaresi Pablo Gil, Francisco G. Ortiz, Experiences with Virtual Environment and Remote Laboratory for learning Robotics at the University of Alicante,International *Journal of Engineering Education*, **22**(4), 2006, pp. 732–742.
23. E. S. Conkur. RoboKol: A computer program for Path Planning for Redundant and Mobile Robots. *Computer Applications in Engineering Education*, **14**(3), 2006, pp. 198–210.
24. Atsushi Fujimori, Peter N. Nikiforuk, and Madan M. Gupta. Adaptive Navigation of Mobile Robots with Obstacle Avoidance. *IEEE Transactions on Robotics and Automation,* **13**(4), 1997.
25. O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *Proceedings of the IEEE Conference on Robotics and Automation*, 1985, pp. 500–505.
26. Iwan Ulrich and Johann Borenstein. VFH+: Reliable obstacle avoidance for fast mobile robots, Proceedings of the 1998 IEEE International Conference on Robotics and Automation, pp. 1572-1577, May 1998.
27. R. A. Brooks. A Robust Layered Control System for a Mobile Robot, IEEE Journal of Robotics and Automation, Vol. RA-2, No. 1, 1986, pp. 14–23.
28. R. A. Brooks and J. H. Connell. Asynchronous Distributed Control System for a Mobile Robot. *Proceedings of the SPIE*, Vol. 727, 1987, pp. 77–84.
29. T. Arai, E. Pagello, and L. E. Parker, "Advances in Multi-Robot Systems", *IEEE Trans. Robot. Autom.,* Vol. 18, no. 5, pp. 655–661, October, 2002.
30. R. Alami, R. Chatila, S. Fleury, M. Ghallab and F. Ingrand, An architecture for autonomy, *The International Journal of Robotics Research*, **17**(4), 1998, pp. 315–337.
31. M. Boada, *Control system for autonomous mobile robots based on reactive skills*, Ph.D. thesis (in Spanish), Universidad Carlos III de Madrid, 2002.
32. R. Barber and M.A. Salichs, A New Human-Based Architecture For Intelligent Autonomous Robots, *The 4th IFAC Symposium on Intelligent Autonomous Vehicles*, Sapporo, Japan, 2001, pp. 85–90.
33. D. Ibrahim Teaching the Principles of Modern Electricity Metering, *Int. J. Eng. Educ*. **24**(6), 2008, pp. 1163–1169.

34. B. Erin, R. Abiyev, D. Ibrahim Teaching robot navigation in the presence of obstacles using a computer simulation program. *WCES-2010, World Conference on Educational Sciences, Istanbul, 2010* (accepted).

APPENDIX

Extract from Questionnaire

QUESTIONNAIRE

Criteria	Sub-Criteria	Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree
User Friendliness	EduRobot is easy to use and user-friendly	17	4			
	My interaction with the EDURobot tool is clear and understandable.	16	5			
	EDURobot is not difficult to use	12	9			
	The tool has visual capability to facilitate the understanding of the various path planning algorithms	14	7			
	Changing of parameters is easy and has significant impact	15	6			
Application-specific self-efficiency	I have the ability to set the start and and goal positions of the path plan	14	6	1		
	I have the ability to place obstacles anywhere in the grid	14	7			
	I have the ability to run any of the three different algorithms for any configuration	13	6	2		
	I can observe the effects of changes in the parameters easily	12	8	1		
	I can thoroughly understand the differences in functioning between the three different algorithms	10	8	3		
The features and functionality of the software	I can observe the numerical results easily		12	9		
	I can observe the graphical results of EDURobot easily	11	10			
	I can observe the location, sizes and shapes of the obstacle easily	13	8			
	I can easily record the planning parameters	10	8	3		
The suitability of	I liked working with EDURobot in teams		18	3		
	Working in groups with EDURobot developed my interpersonal skills	9	8	4		
	Working with EDURobot enabled me to develop a better understanding of path planning algorithms	14	7			
	It was beneficial to turn my theoretical experience to practice	9	10	2		
	The simulator can help me to improve the quality of my vocational education	10	8	3		

Dogan Ibrahim completed his secondary and higher education in Cyprus. In 1975 he graduated from Salford University (UK) with First Class Honours in Electronic Engineering. He then completed an MSc course in Automatic Control Engineering at the Manchester University (UK). His Doctoral study was at the City University (UK) where he obtained a Ph.D. in 1980 in the field of digital signal processing. Currently he is the Head of Department of Biomedical Engineering at the Near East University. His research interests include automatic control, signal processing, robotics, and distant engineering education.

Rahib Hidayat Abiyev received his Ph.D. degree in Electrical and Electronic Engineering from Azerbaijan State Oil Academy (old USSR) in 1997. He worked as research assistant at the research laboratory Industrial intellectual control systems of computer-aided control system department. From 1999 to the present he has worked at the department of Computer Engineering of Near East University, North Cyprus. He is chairman of Computer Engineering Department and director of Applied Artificial Intelligence Research Laboratory. His research interests are soft computing, robotics, control systems, pattern recognition, signal processing.

Besime Erin completed her primary and secondary education in North Cyprus. She then received her B.S. degree in Computer and Electrical Engineering from Purdue University in West Lafayette, Indiana, U.S.A. in 1992, and MS degree in Management, Management Information Systems(M.I.S.) and Finance in 1994 from the same University. She is about to receive her Ph.D. in Computer Engineering from Near East University, North Cyprus where she is a Senior Lecturer at the Computer Engineering Department. Her areas of interest include robotics, digital logic design, object-oriented computing in general and distributed databases.