LITERATURE REVIEW AND SIMULATION OF ROBOT FOR PATH PLANNING OBSTACLE AVOIDANCE (RPPOA) MODEL IN HOMELAND ENVIRONMENT

NISHA P. ZODE¹ & DR. UJWAL A. LANJEWAR²

¹Student Department of ETC & Computer Science, RTMNU, Nagpur, India
²VMV Commerce JMT Arts &JJP Science College, Nagpur, India

ABSTRACT

This paper presents inclusive review of the researches published in obstacle finding and avoidance, and path planning for Self-directed Movable Robots. Methodologies, operational conditions, and boundaries of various approaches of obstacle avoidance and path planning have been argued. The comparison of the reviewed methodologies is also performed, and the rewards of a specific methodology over others are listed. Simulation of the obstacle detection and avoidance is for a self-directed movable robot using Arduino is illustrated.

KEYWORDS: Heading angle Finding Method, Calculation of the Gap Centre Angle & Simulation of Collision Avoidance Using matlab

Received: Sep 01, 2020; Accepted: Sep 20, 2020; Published: Oct 21, 2020; Paper Id.: IJCSEITRDEC20203

1. INTRODUCTION

There is a rapid increase in the use of self-directed movable robots and the usage is not only limited to partslike academics, industry, disaster recovery process, home cleaning instruments, entertainment for people, assistance for people which are disability, exploration, and transfer of goods. Movable robots are capable of effectively perform several useful tasks in a hazardous environment. For example, a self-directed robot that performs searching and transportation is used for many applications, which include military applications.

For a robot to be navigated in the environment safely, it should have the ability to understand its surrounding environment and should be able to move without hitting colliding with anything. Because of this reason, obstacle detection and avoidance, path planning is the top objective while designing a self-directed movable robot and this can be achieved by using sensory perception.

An excellent performance of for avoiding obstacle is difficult task while developing the assembly of robot. It assures the drive of the robot from its current position to the final destination without hitting any of the obstacles. The combination of successful collision avoidance with efficient path planning determines the safe transport of a movable robot from the initial position to the final destination. The technique which is used for obstacle avoiding is based on the sensor data obtained from the robot for finding the best effective path planning to navigate from one source destination to target destination. While designing a self-directed movable robot, there are two types of obstacle considered namely, negative obstacles that are beneath the ground on which the robot is moving and these obstacles have negative height. The second type of obstacle is positive obstacles that are above the ground on which the robot is moving and these obstacles have positive height.

For self-directed obstacle avoidance, a generally used approach is the wall-following method. In this
approach, the robot moves beside the walls until it senses an obstacle. If the sensors sense an obstacle, the robot thinks that obstacle as just another wall and continues moving along the obstacles contour until it traces back its original course. This navigation system is less demanding technologically as problems of determination of robots own position is made easier. Robot navigation by this approach is not much adaptable because of which it is suitable only for very specific applications. Another general method for obstacle avoidance is based on edge-detection. In this method, a developed algorithm is used to find an obstacle in the path and collects data about the position of the edge with respect to the robot and based on the information the robot changes its path around either edge. An obstacles boundary is represented by a line connecting the two edges. There is a disadvantage in the edge-detecting method for obstacle avoidance that the robot needs to stop in front of an obstacle if the robot wants to gather more accurate information.

Modelling a movable robot and executing algorithms before physical implementation supply a way to perform feasible and experienced applications. A simulation platform is a very beneficial and significant tool to envision the output of the developed algorithm. The testing algorithm using a simulation tool is a lot faster and economical as compared to testing the algorithm in real robots. The testing of a new algorithm on a new hardware platform without performing a simulation of the new algorithm before it is not an optimal way to design and test algorithms. Simulation helps in testing the algorithms in the absence of a physical robot and also proves vital in validating the performance of the developed algorithm in a physical environment. Apart from it, simulation also helps in identifying costly design error at the design phase.

In this paper, a literature review of the various methods proposed by researchers to accomplish the task of obstacle detection and avoidance is done. In the latter part of the paper, a CAD model of the Pioneer 3-DX robot is used to build a Simulink model and control algorithm of the robot to perform the simulation for collision avoidance of the robot in the virtual environment. In order to achieve the results Simulink, Simulink 3D editor, Simulink 3D animation and State flow is used. The CAD model used in this paper is an Adept movable robot (Pioneer 3-DX).

2. EXAMINATION OF OBSTACLE DETECTION AND AVOIDANCE FOR SELF DIRECTED MOVABLE ROBOT

Johann Borenstein (1989) has created an approach which is virtual field force. This approach includes two concepts.

- Certain grids shape for obstacle presentation
- Potential fields for finding the navigation

This algorithm uses fast movable robot and helps to sort out minimum trap problem. This robot uses the robot of up to 0.785 m/s in the VFF Method. This method of VFF does not require data storage and evaluation of corner problems. The obstacles are represented with the certainty grid technique. Certainty grid is created by dividing the robot’s work area into several square elements called as cells. Every grid cell has some value which indicates in terms of I and j form. The values of I and j shows higher level of confidence in the cell. Range reading is taken with respect to the motion of the robot using the ultrasonic sensors and projected into the certainty grid. The algorithm is running concurrently to examine 33*33 cells square window of the grid and during the examining of the square grid, the robot is always at its centre of the grid. Every cell inside the window exerts a repulsive force to the robot which pushes the robot away from the cell. The greatness of this force is inversely proportional to the square of the distance between the cell and the robot and proportional to the

Impact Factor (JCC): 9.7093  NAAS Rating: 3.76
cell contents $C(i,j)$.

The resultant force acting is summation of all forces. The resultant force vector is the sum of $F1$ and $F2$.

$$F(i, j) = F(i, j) \left( \frac{x_i - x_0}{d(i, j)} \hat{x} + \frac{y_i - y_0}{d(i, j)} \hat{y} \right)$$

(1)

In this experiment he chooses three motion modifications for the robot running at high speed. The results are as follows.

Obstacles in the certainty grid and the robot used is Cybernation K2A which has 24 Polaroid ultrasonic sensors.

- The first modification is that lower pass for steering control system. A positive change is occurred by implementing with time parameter constant 0.45 second. The filter smoothen the robot’s motion when moving alongside obstacles.

- The second modification is checked by increasing the strength of the repulsive forces when the robot moves toward an obstacle and reduces it when the robot moves alongside the obstacle. In order to achieve this enhancement, directional cosine is multiplying by the sum of the repulsive forces. The effect of this is that the disgusting force from the obstacle is of maximum magnitude if the robot is approaching then obstacle in the front as $\cos \Theta = 1$. As the robot turns the angle also changes which change the magnitude of the repulsive force and minimum magnitude of the repulsive force is when the robot runs parallel to the boundary.

- Another improvement is in the speed control. In VFF environment, the speed of a movable robot is directly proportional to the value of the repulsive force. The robot can move at the maximum speed if there is no obstacle examined, but when an obstacle is examined, repulsive forces from obstacles reduce the magnitude of the resultant force, thereby effectively reduces the robot’s speed. The speed of the robot does not change and the robot continues to move at its maximum speed if the robot moves alongside an obstacle boundary.

There is one problematic in the VFF method that the robot might get trapped if the robot runs into a dead end. To avoid trap situations, a disorder in the following equation (2) is monitored by the controller and if the condition is satisfied, then the system switches to a recovery algorithm which is a wall-following method.

$$|\theta_1 - \theta_2| > 90^\circ$$

(2)
Volkan Sezer (2012) developed a new method, that is, follow the gap method to avoid the obstacle. In this method obstacle and robot are circular shape and as the distance to obstacle and robot are changes because angle calculation is also changing. The Follow the Gap method is based on the building of a gap array around the vehicle and calculation of the best heading angle for heading the robot into the centre of the determined gap around, while simultaneously considering the goal point. Calculation of the gap distance with two parameters.

\[ \phi_{gap,c} = \arccos \left( \frac{d_1 + d_2 \cos(\phi_1 + \phi_2)}{\sqrt{d_1^2 + d_2^2 + 2d_1d_2 \cos(\phi_1 + \phi_2)}} \right) - \phi \]

**Heading angle Finding Method**

There is a boundary from left to right region which is shown in below figure. The left border angle is denoted by \((\phi_{lim_l})\) and the right border angle is denoted by \((\phi_{lim_r})\) comes from field of reference \((\phi_{fov})\). The final between angle is the gap between centre and goal angle. As the robot is comes towards obstacle first it considers the safety as compared to centre angle gap and vice versa.

Simulation the maximum value of gap is given by

\[ \phi_{final} = \frac{\alpha}{d_{mm}} \phi_{gap,c} + \beta \phi_{goal} \]

\[ \frac{\alpha}{d_{min}} + \beta \]
Calculation of the Gap Centre Angle

For calculating the centre angle, cosine method is used. The limitation is over cum by using MATLAB software. A Light Detection and Ranging sensor are used for performing the experiment. But there are some turning limitation in front of vehicle with a total 150-degree angle and 10-meter range. Simulation results of FGM for two different obstacle configurations are shown in Figure 4. Goal coordinates are [70–70] for Configuration A and [80–0] for Configuration B.

![Figure 4: Simulation Results for Dissimilar Obstacle.](image)

Shape

The linear $V(t)$ and angular $\omega(t)$ velocities of the robot can be computed by the equation below

$$
\begin{bmatrix}
V(t) \\
\omega(t)
\end{bmatrix} =
\begin{bmatrix}
1/2 & 1/2 \\
1/rc & -1/rc
\end{bmatrix}
\begin{bmatrix}
V_l(t) \\
V_r(t)
\end{bmatrix}
$$

(6)

Where, $rc$ is chassis radius of the robot. In a differential drive robot, kinematic equations depict the robot’s position [14]. A Simulink model of this made with four main blocks. The block Signal builder block is used to generate the signals (velocity/voltage) for the left and right wheels. The velocity of a robot depends on the radius of its wheel. The input of the DC-motor block is generated velocities. This block demonstrates the transfer function of the motor of left or right wheels with a scalar gain equal to the radius of the robot’s wheel. The computed transfer function of the motor is given by the equation below

$$
G_m(s) = \frac{\omega(s)}{U(s)} = \frac{K}{(LS + K) \cdot (JS + M_F) + K^2}
$$

(7)

The output of the motor block is provided to the velocity relationship block which converts these velocities of the wheels into linear ($V(t)$) and angular ($\omega(t)$) velocity of the robot. The resultant linear and angular velocities of the robot form inputs to the kinematic model, which computes the final whereabouts of the robot orientation in term of position $x$, $y$ and angle $\theta$. 
Figure 5: Flow Diagram of Simulink of the Robot.

Four-wheeled robot – The error of unequal diameter wheels and non-linearity’s is eliminated by using four-wheeled robot. Two front steering wheels change the angle which is the primary difference from the other configuration of the wheeled robot.

\[
\begin{bmatrix}
  x' \\
  y' \\
  \phi'
\end{bmatrix} = \begin{bmatrix}
  \cos \theta & \sin \theta & 0 \\
  -\sin \theta & \cos \theta & 0 \\
  0 & 0 & 1
\end{bmatrix} \begin{bmatrix}
  V(t) \\
  \delta
\end{bmatrix}
\]

(8)

Kinematic model of this configuration of the robot is given by the equation below where \( \delta \) is the steering angle.

Simulation: A series of simulations have been conducted to study the behaviour and to characterize the performance of the presented kinematic model. Four basic scenarios are taken with consideration on wheel velocities of the robot to demonstrate overall behaviour of the robot and the simulation is performed.

Figure 6: Graphical Picture of the Trajectories Followed by the Robot in (a), (b), (c), (d).

DistBug Algorithm – Similar to the Bug-2 algorithm but in this robot always computes and stores the information on the distance from the robot current position to the final destination and from the next position to the final destination. When the distance of the final destination from its next position is greater than the corresponding distance from its current position, then the robot leaves the edge following method. Proposed Intelligent Bug Algorithm (IBA) – When an obstacle is sensed, the behaviour of the robot is changed to obstacle avoidance and the robot starts following the edge of the obstacle until an obstacle-free path is sensed. The robot changes to move to goal in order to generate new reference path when an obstacle-free path is sensed.

Deepali Ghorpade (2017) works on the method of obstacle avoiding based on the principle of LIDAR, the hull
algorithm which is used to detect the actual geometrical body of the obstacle. In this method forward orientation angle uses to establish the best possible route finding. In this MATLAB is used for reliability

LiDAR Module: It is UTM-30LX 2D LIDAR, which measures and has a limit of 30 meters. It has horizontal view reference is about -45-degree o 225 degree with angular resolution is 0.254 degree.

Pre-processing: the filter which is implemented is applied on point cloud. The median value is taken as distance between robot and the point where effect of noise will be diminished. This preprocessing is made with median filter which reduces the complexity. Each point on the map has its relative position to can be computed from distance and scan angle value. We can calculate coordinates of each point using equation

![Figure 7: Laser-Point Cloud Creation.](image)

Obstacle detection: the segmentation is applied in the work of this project. It is laser point cloud which strikes the obstacle and discarding points which never toward the robot. This method is applying on each point into N blocks by varying start to end index of the block.

Obstacle avoidance: To determine optimal collision-free path, robot’s travelling direction has to be calculated in terms of orientation angle. The degree of freedom of robot decides configuration space given by two-dimensional space and limits of orientation angle. The proposed method assigns 0-180° angular degree of freedom to robot for obstacle avoidance.

![Figure 8: Determining Optimum Travelling Direction.](image)

Experiment Results: It is seen from results of implementation that proposed obstacle detection and avoidance method can effectively filter noise and split point cloud into segments and clusters quickly. The output of convex hull algorithm is set of points
Obstacle detection: Segmentation is proposed in this work to improve classification and performance of Obstacle Detection. Segmentation retains only those laser-point cloud data which encounter obstacle, discarding points which never return to robot. Then segmentation criteria is applied to each point in a group, splitting laser points into N blocks by satisfying criteria for start and end index of block. The segmentation divides 1082 laser points into N blocks, each containing spatial data of obstacles. To form perfect cluster criteria followed is that Euclidean distance between two consecutive laser points in block is less than width of robot. Thus after clustering we get N blocks such that each block represents obstacle data and distance between two blocks is such that robot can safely pass through the next step is to determine geometrical structure of obstacle. To map the object boundaries accurately, mathematical representation of obstacle is achieved using Convex Hull algorithm. Graham Scan algorithm based on polar angle is utilized to generate obstacle structure in terms of polygon.

Experiment Results: The results which are obtained to detection and avoiding the obstacle can be effectively filter the noise and vibration the output of this type of algorithm is the set points which are plotted and give the best result. To test validity of proposed method simulation tests were carried out on Matlab simulation platform. Simulation scenario is an area of 100mX100m (0, 0) is the starting point, (100, 100) is the termination point. Setting detection range of LIDAR to be 20m and 30m respectively, a series of obstacles were set up in those tests, and the results of those tests proved that obstacle in the path were accurately detected using convex hull algorithm and robot safely reached target location without any collisions as shown in figure 10.
Simulation of Collision Avoidance Using \textit{matlab}

In this part, simulation of collision avoidance will be presented for movable robots. An adept movable robot (Pioneer 3-DX) is used for this simulation. The source of the code is [9]. There is no change performed in the algorithm and the primary objective of this part is to illustrate how to create an algorithm for collision avoidance using Simulink. Initially, the CAD model of the robot used to create a Simulink/SimMechanics model to the robot and then designing the collision avoidance algorithm in a virtual reality environment using Simulink.

Creating a SimMechanics Model: The CAD model of the robot is converted into an .xml file using add-on software called SimMechanics Link. Before exporting the CAD file must be assembled into one file. The .xml file is imported in the Simulink using Matlab command (smimport(“filename.xml”)). The imported Simulink model is shown below.

Testing the Simulink model: The testing of the Simulink model is performed to verify the robot motion behaviour, and the results of each test are verified virtually by using the Buildin 3Dviewer which is Mechanics Explorer. Test 1 is performed to see the motion of the robot under the force of gravity. Test 2 is performed by adding actuation to the wheels of the robot. For this test, the motion to both the wheels is provided as input and torque are automatically computed. The Actuated_Pioneer3DXAssembly.slx file is used for test 2.

![Figure 11: Simulink Model Cast-Off for Test 2.](image)

Test 3 is performed to validate the motion of the robot for specific wheel velocity condition. The MotionConstraints_Pioneer3DXAssembly.slx file is used for test 3. Wheel angular velocity for both the wheels is computed. The initial model is updated by using a kinematic subsystem block which inputs the wheel angular velocity and calculates $P_x$, $P_y$, and $\theta$. Kinematic subsystem calculates the planar parameters of the robot. A planar joint block is installed between the kinematic subsystem and the robot body, and it defines the relationship between the robot body and the environment.

![Figure 12: Simulink Model Castoff for Test 3.](image)
Opening the robot in the custom environment: Saving the CAD model as a VRML file which has an extension of .wrl. Before saving the file in the building 3D viewer, open the file in the Simulink 3D Animation VRML. Detail information about the file structure can be seen after opening the file in the editor. The Custom environment is build using the file SL3DAnim_Pioneer3DXAssembly.slx. Simulation 3D animation is linked with the earlier designed Simulink model. In this file, VR sink block is used that gets actuation input for the robot animation. We can edit the block parameters by editing them in the properties of the block. VR sink requires input to drive the animation. In this simulation, the rotation matrix and the translation vector for the robot body is provided. Simulink/SimMechanics model is in the SmSL3 Dvinterface block. Interface block used the transform information from the model and performed basic coordinate geometry and transformation to input the data in the VR Sink block. Interface uses transform sensor which uses the coordinate transform with respect to the world frame as an input and computes parameters like rotation matrix and translation vector and feed that to the VR sink in order to drive the VR Sink block. After performing the simulation in the 3D animation using the SL3DAnim_Pioneer3DXAssembly.slx, we can see the motion of the robot in the custom environment.

Implementing Virtual Sensors: In this simulation, we are only considering eight front sonar sensors of the robot. We know the position of the robot given by Px, Py, and θ, based on these values we can determine the location of each sensor with respect to the world. We can make a predetermined binary lookup table from the virtual custom environment. Using the sensor position and the customer binary map of the environment, we can calculate the distance between a sensor and an obstacle.

- Whenever sensor 1 detects an object, the robot will perform turn right wide operation.
- Sensor 2 and 3 are coupled and if any of the sensors detect an object, the robot will perform a turn right tight operation.
- Sensor 4 and 5 are together and if any of the sensors detect an object, the robot will perform a turn in place wide operation. If the magnitude of sensor 4 is high, then robot rotates right, and if the magnitude of sensor 5 is high, then robot rotate left.
- Sensor 6 and 7 are coupled and if any one of the sensor detects an object, the robot will perform turn left tight operation
- Whenever sensor 8 detects an object, the robot will perform turn left wide operation.

Figure 13: Explanation of Simulated Sensors.
If in case any two sensor senses an object at the same time then the preference is given to the front sensor and move outward as preference decreases. This is since corrective action for the front sensor is most important. At any given opinion of time either the robot will be moving straight if any sensor does not detect an obstacle or the robot will be performing any of the turning operation. We can also think of the operations at states.

Developing a control algorithm: State flow is used to implement the supervisory logic algorithm. In this stepController_Pioneer3DXAssembly.slx file is used. If-else case or a Matlab code can also be used to implement the supervisory logic algorithm. Control logic algorithm takes sensor values and sensor ranges as input and based on those data sets the velocity of the right and left wheel. State flow is a high-level graphical language which makes it easier to understand. State flow also has a building debugger and animation that prove vital in debugging the logic. The rectangular block symbolizes the state of the robot and the arrows depict the transition from one state to another.

CONCLUSIONS

This paper obtainable an examination on the Obstacle detection and avoidance for a Self-directed movable robot. Virtual Force Field approach, Artificial Potential Field approach, Vector field Histogram approach and Follow a Gap style have been illustrated and the experiment results of the simulation achieved by these methodologies is also illustrated. In VFF approach, Potential field method is combined with a certainty grid and the obstacles are represented with the certainty grid technique. In APF approach, the goal is presented by an attractive potential field whereas the obstacles are represented by a repulsive potential field. VFH approach uses one dimensional Cartesian histogram grid which is used by the algorithm to select the lowest obstacle path. FGM computes a gap array around the robot and selects the best heading vector through the gap. APF have a drawback of a local minima and VFH does not take into account the nonholonomic constraints of the robot. Simulation of the Obstacle abscording of a movable robot is explained. CAD model of the robot is used to make a Simulink model and the motion constraints are applied on it. After simulating virtual sensors, a collision avoidance algorithm is designed and inspired using State flow.

REFERENCES

1. Abrar Alajlan, Khaled Elleithy, “Multi-Sensor Based Collision Avoidance Algorithm for Movable Robot”, IEEE LISAT,2015.
2. Deepai Ghorpade, Anuradha D. Thakare, Sunil Doiphode, “Obstacle Detection and Avoidance Algorithm for Self directed Movable Robot using 2D LiDAR”, 978-1-5386-4008-1, 2017IEEE.
3. Mariappan, Er Dr Murailndran, Vigneswaran Ramu, and Thayabaren Ganesan. 'Fuzzy Logic Based Navigation Safety System for a Remote Controlled Orthopaedic Robot (OTOROB).” Research and Development (IJRRD) 1.1 (2011): 21-41.
4. M. Takahashi, K. Kobayashi, K. Watanabe, and T. Kinoshita, “Development of prediction based emergency obstacle avoidance module by using LIDAR for movable robot”, Advanced Intelligent Systems (ISIS), 15th International Symposium,2014.
5. “Active and Passive Range Sensing for Robotics”, International conference on Automation and Robotics, IEEE2000
6. G. Giralt, “Movable robots,” NATO ASI Series, Vol. II, Robotics and Artificial Intelligence, New York, Springer-Verlag,1984
7. Manikanta, Yerrapothu, Seeram Srinivasa Rao, and R. Venkatesh. "The Design and Simulation of Rose Harvesting Robot." International Journal of Mechanical and Production Engineering Research and Development (IJMPERD) 9. 1, Feb 2019, 191-200
8. J. Iijima, S. Yuta, Y. Kanayama, “Elementary functions of a self-contained robot "YAMABICO3.1’," Proc. 11th Int. Symp. Ind. Robots, Tokyo, 1983, pp. 211-218.

9. Johann Borenstein, Yorem Koren, “Real-Time Obstacle Avoidance for Fast Movable Robots” IEEE transactions on systems, man and cybernetics, Vol. 19, no. 5, 1989.

10. J. Borenstein and Y. Koren, “Obstacle avoidance with ultrasonic sensors,” IEEE J. Robotics and Automat., Vol. RA-4, no. 2, pp. 213-218, 1988.

11. Mohapatra, Hitesh, et al. "A Comparative Analysis of Clustering Protocols of Wireless Sensor Network.” International Journal of Mechanical and Production Engineering Research and Development (IJMPERD) ISSN (P): 2249-6890.

12. Saat Karakaya, Gurkan Kucukyildiz, Hasan Oacak, “A New Movable Robot Toolbox for Matlab” J Intell Robot Systems, 2017.

13. Source: https://www.mathworks.com/videos/movable-robot-simulation-for-collision-avoidance-with-simulink-90193.html

14. J. L. Crowley, “Dynamic world modeling for an intelligent movable robot,” Proc. IEEE Seventh Int. Conf. Pattern Recognition, Montreal, Canada, 1984, pp. 207-210.

15. Patil, Vrishali, and Uttam Bombale. "Wireless Landmine Detection Robot.” International Journal of Robotics Research and Development (IJRRD) 9.1, Jun 2019, 35-42

16. B. H. Krogh, “A generalized potential field approach to obstacle avoidance control.” Int. Robotics Research Conf., Bethlehem, PA, Aug. 1984.

17. Volkan Sezer, Metin Gokasan, “A novel obstacle avoidance algorithm: Follow the Gap Method”, 2012

18. M.H. Ang, H. Krishnan, “Virtual obstacle concept for local minimum-recovery in potential-field based navigation”, IEEE International Conference on Robotics and Automation (ICRA), 2000.

19. Z. Xi-yong, Z. Jing, “Virtual local target method for avoiding local minimum in potential field based robot navigation”, Journal of Zhejiang University Science 43, 2003.

20. W. Nowak, A. Zakharov, S. Blumenthal, E. Prassler, “Benchmarks for movable manipulation and robust obstacle avoidance and navigation”, BRICS Deliverable D31, 2010.

21. O. Khatib, “Real-time obstacle avoidance for manipulators and Mquee robots”, IEEE, 1985.

22. Wenyao MA, “Simulation on vessel intelligent collision avoidance, based on artificial fish SWARN algorithm”, Polish Maritime Research, Special Issue, 2016 Vol. 23: pp.138-143

23. Rajibul Huq, Herve Lacheray, Cameron Fulford, "QBOT: an educational movable robot controlled in Matlab Simulink environment”, IEEE, 2009

24. L. Lapierre and R. Zapata. "A Guaranteed Obstacle Avoidance Guidance System", Self directed Robots, vol. 32, no. 3, pp. 177-187, 2012.

25. Angelo Nikko Catapang and Manuel Ramos, Jr. “Obstacle Detection using a 2D LIDAR System for an Self directed Vehicle”, International Conference on Control System, Computing and Engineering, IEEE 2006.

26. Cherif Foudil, Djedi Noureddine, Cedric Sanza, "Path finding and collision avoidance in crowd simulation” Journal of Computing and Information Technology, 2009.