Indoor path planning and obstacle avoidance simulation

T Savu and B A Jugravu
Manufacturing Engineering (TCM) Department, University POLITEHNICA of Bucharest, 313 Spl. Independentei, 060042 sector 6, Bucharest, Romania
savu@ctanm.pub.ro

Abstract. The paper is describing the work performed for choosing and designing a path planning and obstacle avoidance solution for a car-like vehicle moving in an indoor environment. The proposed algorithm first consists in providing the vehicle with a set of passing points generated by a Dijkstra path planning algorithm. The vehicle is using the points for establishing its path and is avoiding the obstacles using scanned data from a distance sensor. In a simulation environment, initial tests were performed for estimating correct values of the vehicle’s forward and angular velocities and for estimating the effect of the scanning speed on the vehicle’s behaviour. Obstacle avoidance tests were performed for identifying specific situations possible to appear due to high velocities or to low scanning speed. Even not always choosing a smooth avoidance path, the algorithm proved to find a way for avoiding the obstacle in a clear and fast enough manner.

1. Introduction
When designing autonomous ground vehicles, two topics are between those of main importance: the path planning and the obstacle avoidance capabilities. Path planning algorithms are treated at various levels of complexity, starting from simple line following procedures [1] to detailed analytical models [2]. Obstacle avoidance is also the subject of numerous works, from graduation thesis [3] to detailed contributions in exhaustive books [4].

Nevertheless, when designing an autonomous ground vehicle for a specific task, existing literature must be reviewed and knowledge to be applied must be selected as close as possible to the particular case. The scope of the work presented in this paper was to choose or to design a path planning and obstacle avoidance solution for a car-like vehicle moving in an indoor environment, more specifically in a building. Vehicle’s hardware and software resources had to be considered not enough for performing high performance computing or for managing complex sensors sets. The paper is describing how a simulation environment was used for testing different options concerning the vehicle’s parameters and for also testing an obstacle avoidance algorithm specifically designed for the above-mentioned purpose.

2. The path planning and obstacle avoidance algorithm
The proposed algorithm consists in the following:

- a set of compulsory points, to be used for moving from one building’s space to a neighbouring one, are initially defined. When the vehicle must move from one point of the building to another, the set of the building’s compulsory passing points is used by a Dijkstra path planning algorithm [5] for establishing the vehicle’s path;
• the vehicle does not know the building’s configuration, but knows its current position, is receiving the array of path points from the Dijkstra algorithm and is selecting the next point it must reach. After reaching one point, the vehicle is setting its target to the next one in the array of records, till the last point is reached;
• for moving toward the current target point, defined by a target vector based in vehicle’s current position, the vehicle is first checking if there is any obstacle on the target vector’s direction. If no obstacle is detected, the vehicle is moving forward;
• if an obstacle exists on the target vector’s direction, then the vehicle is trying to avoid it by moving toward the obstacle’s edge which is closer to the current moving direction.

For example, in figure 1, there is an obstacle on the target direction. The green lines indicate the obstacle edges. The vehicle will try to avoid the obstacle by starting to move to the rightmost green line, which is closer to the vehicle’s speed vector.

**Figure 1.** Radar points indicating an obstacle on the target direction.

3. **Building the simulation environment**

The simulation environment, which is reproducing part of the authors’ faculty building, was initially built in a CAD environment (figure 2) and then described in the LabVIEW graphical programming environment [6], using the Robotics Starter Kit 1.0 functions library [7].

There are two areas in the environment which received special attention during the study:
• the departure room in the left side (figure 3), where the exit door is placed very close to a wall;
• an obstacle (figure 4), in the middle of one intermediate space, which is not known by the path planning software.

The building walls and the unknown obstacle were defined as box obstacles in the simulation environment (figure 5, figure 6). The data is stored in an array of clusters, where each cluster contains a subcluster with the three coordinates of the obstacle’s mass center and two other values representing the obstacle’s x-length and y-length. The obstacles’ height was left at the default value of 1 m. All the obstacles were set to the “kinematic” state, making them to behave as if they have infinite mass.
Figure 2. CAD model of the simulation environment.

Figure 3. Detail of the departure room.

Figure 4. The obstacle in the intermediate space.

Figure 5. LabVIEW diagram for defining the simulation environment.

Figure 6. Building in the simulation environment.
The simulated situation is one when the vehicle must move from the left-side room to another space in the right part of the building. The points of the vehicle’s path, generated by the Dijkstra algorithm, are stored in a global variable as an array of records, each record containing the x and y coordinates of one point (figure 7).

It is considered that a point on the path was reached by the vehicle when the distance to that point is less than a set value, initially 0.3 m.

![Figure 7. Points of the vehicle’s path.](image)

4. The simulated vehicle

The simulated vehicle (figure 8) is based on the one from the NI LabVIEW Robotics Starter Kit [8]. It has a steering frame composed of four simple fixed wheels with a radius of 50 mm. Only two wheels are active and the distance between them is 360 mm.

The active wheels are controlled, with a gear ratio of 2, by two 12V DC motors, with a maximum rotational speed of 152 rpm and a 2.1 N\(\cdot\)m torque. Each motor has its own optical quadrature encoder with 400 pulses per revolution.

![Figure 8. The simulated vehicle.](image)

The vehicle is equipped with a PING))) 40 kHz ultrasonic distance sensor for distance measurements between 2 cm and 3 m. The sensor is rotated around the vertical axis using a servomotor. The rotation angle can vary between -65° and 65°. Angle increment of the scanned points was initially set at 4°.

The overall vehicle dimensions are 405 mm x 368 mm x 150 mm and the total weight is 3.6 kg.

The vehicle’s initial position was set at coordinates (1,6,0) and its z-rotation was set to zero.

The vehicle’s distance sensor is storing the measured data in two arrays, one containing the angles and the other the distances of the scanned points belonging to the detected obstacles. Both angles and distances are expressed in a polar coordinate system having its origin in the vehicle’s origin.
According to the proposed algorithm, if the target vector’s direction is outside the ±65° distance sensor’s scanning interval, then the vehicle is moving toward that direction. If the target vector’s direction is inside the interval but the scanned distance is at the 3 m limit, then there is no obstacle on the target vector’s direction and the vehicle is also moving toward that direction.

If there is an obstacle on the target vector’s direction at less than 3 m, then the start and end scanning angles between which the obstacle was detected are computed (figure 9) and the one closer to the target direction is selected as the updated vehicle direction.

![LabVIEW diagram for checking the presence of an obstacle and for computing the avoidance direction.](image)

**Figure 9.** LabVIEW diagram for checking the presence of an obstacle and for computing the avoidance direction.

5. Initial tests

Initial tests were performed with the vehicle’s maximum forward velocity set at 2 m/s and the maximum angular velocity set at 2 rad/s.

The tests showed that the angular velocity is too low. Trying to rotate toward the target point, the turning radius is big, the vehicle is reaching close to a wall and moving in contact with this (figure 10). The second conclusion is that, due to the small rotational speed of the distance sensor, the vehicle fails to collect enough scanned points till it is reaching the exit door’s vicinity.

When the maximum angular velocity was increased to 5 rad/s, the turning radius decreased, and the vehicle was able to position itself faster toward the target point (figure 11).

![Moving along a wall due to big turning radius.](image)

**Figure 10.** Moving along a wall due to big turning radius.
Due to the increased rotational speed, there were tests during which the stability in moving toward the target’s direction was initially lost, but finally the vehicle succeeded to position itself on the right direction (figure 12).

Tests were performed with the maximum forward velocity set at 5 m/s and the maximum angular velocity set at 5 rad/s. It was only useful to demonstrate the simulator accuracy in reproducing the vehicle’s dynamic, because it was reached a situation when the vehicle climbed a wall with the backward wheels and got stuck with the distance sensor scanning the floor (figure 13).

The maximum forward velocity of 2 m/s still proved too big, since in some situations the vehicle didn’t succeed to break fast enough for almost hitting an wall (and was forced to move backward according to its “panic” algorithm) and also started turning right only after overcoming the second point (figure 14).
Reducing the maximum forward velocity to 1 m/s allows the vehicle to reach more precisely the specified path points. Some difficulties were still encountered when coming close to the wall on the right, because the distance sensor didn’t have enough time to scan the entire interval, so the vehicle does not have the complete data about its surroundings (figure 15).

Figure 14. Moving backward in “panic” state and point overcoming due to high speed.

Figure 15. Difficulties in finding the door due to slow scanning speed.

Changing the starting point coordinates from (1, 6, 0) to (1, 1, 0) gave enough time for the distance sensor to complete the scan, so the vehicle approached the room exit in a clean manner (figure 16).

Figure 16. Clear path following when scanning data completed.
The radar data contains the information about the forward-left wall and also, when coming close to the exit door, the data allow to clearly identify the open space (figure 17, figure 18).

6. Obstacle avoiding tests
When coming close to the unknown obstacle, due to the low scanning speed of the distance sensor, not all the obstacle’s width is completely scanned at a certain moment. In figure 19 it can be seen that the scanning started from the left side of the obstacle (from the vehicle’s point of view) and the obstacle’s right side the vehicle is seeing a clear path, while in figure 20 the scanning started from the right side and there is a still unknown portion on the obstacle’s left side.

In the first situation, the vehicle was choosing a detour point not right enough, so it reached close to the right side of the obstacle, close enough to hit it, entered the “panic” mode, moved back and then continued on a free path to the next target point (figure 21).
In a slightly different situation, the vehicle is sensing the right part of the obstacle from the beginning and it’s moving on an avoiding trajectory. Because the distance sensor is scanning only in the $\pm 65^\circ$ interval, the sensor is not detecting the obstacle when the vehicle is moving parallel to it. The vehicle started turning left too soon and the distance sensor detected now the right end of the obstacle. Even it was only an end of the obstacle, the vehicle entered the “panic” mode, moved backward and only afterwards continued to the next target point (figure 22).

![Figure 21. Incomplete obstacle avoidance when scanning from the left.](image1)

![Figure 22. Obstacle re-appearance due to limited scanning interval.](image2)

![Figure 23. Smooth obstacle avoidance.](image3)

![Figure 24. Difficulties in avoiding the obstacle.](image4)

An example of a situation when the obstacle avoidance run smoothly is presented in figure 23.
Final tests were performed with an increased forward velocity of 3 m/s. Due to the slow reaction, the vehicle had obvious difficulties in finding its way for avoiding the obstacle (figure 24) and in positioning on the exit door from the departure room (figure 25).

![Figure 25. Difficulties in positioning on the exit door.](image)

7. Conclusions and future developments

It can be concluded that, for the above described requirements, the chosen simulation environment allows an easy to use way of work for describing the simulated building, for bringing changes in its structure, for changing the vehicle’s speed parameters, for changing the behavior algorithm and for obtaining relevant data.

Choosing a Dijkstra algorithm, run on a separate server, for computing the passing points in the building, proved to offer a simple way for the vehicle to determine its next steps without consuming high processing resources. Future studies will focus on changing the points data stored by the vehicle by including intermediate points needed for obstacle avoidance.

The simple obstacle avoidance algorithm proved to be simple and robust enough, allowing to identify the causes of various anomalies either in non-optimal vehicle speed parameters or in the low scanning speed of the distance sensor. Simulations will be designed for higher performance scanning sensors. Further studies will be performed for other vehicle configurations and dynamic parameters.

8. References

[1] Jibrail S F and Maharana R 2013 *PID Control of Line Followers* (Rourkela: National Institute of Technology)
[2] Fazlollahtabar H and Saidi-Mehrab M 2015 *Autonomous Guided Vehicles – Methods and Models for Optimal Path Planning* (Cham: Springer International Publishing)
[3] Singh G 2015 *Sensor Fusion and Obstacle Avoidance for an Unmanned Ground Vehicle* (Daytona Beach: Embry-Riddle Aeronautical University)
[4] Fahimi F *Autonomous Robots – Modeling, Path Planning and Control* 2009 (New York: Springer Science+Business Media)
[5] Blokdyk G 2018 *Dijkstra's Algorithm: End-to-end Data Analysis* (Create Space Independent Publishing Platform)
[6] Bitter R, Mohiuddin T and Nawrocki M 2017 *LabVIEW: Advanced Programming Techniques, Second Edition* (Boca Raton: CRC Press – Taylor & Francis Group)
[7] Hossain A 2013 *Proc. Annual Conf. of American Society of Engineering Education* (Atlanta: ASEE Engineering Libraries Division) Paper ID #6116
[8] Siegwart R, Nourbakhsh I R and Scaramuzza D 2011 *Introduction to Autonomous Mobile Robots* (Cambridge: MIT Press)