Autonomous Navigation and Obstacle Avoidance of VipaLab

Institut Pascal

Final Report as part of Internship work

Shubham M Wagh
B.Tech (Electrical and Electronics Engineering)
Visvesvaraya National Institute of Technology,Nagpur(INDIA)
July 10,2015

Abstract

At present , the topic of automated vehicles is one of the most promising research areas in the field of Intelligent Transportation systems (ITS). The use of automated vehicles for public transportation also contributes to reductions in congestion levels and to improvements in traffic low.

In this document autonomous navigation and obstacle avoidance of VipaLab is presented. It presents an algorithm for path following applications represented by way points in order to provide smooth tracking. The vehicle is provided with a multibeam Laser Imaging Detection and Ranging (LIDAR) sensor for obstacle detection. When an obstacle is detected in the planned path, the planned route is modified in order to avoid the obstacle and continue its way to the end of the mission. All this is done in a Cobaye Simulator i.e in a simulated environment as well as implemented successfully in real time environment.

Acknowledgements

I would like to thank my advisor Prof.Roland Chapuis without whose guidance and support this project would not have been possible. His suggestions and feedback greatly helped me in my project. He helped and encouraged me from all perspectives to complete this work.

I would like to thank Dr. Claude Aynaud and Mr.Gerald Lelong for their suggestions and positive feedbacks.

I would like to thank the staff members of Institut Pascal for all the help and support they had given me during my internship period at the University of Blaise Pascal. I would like to thank all my friends and well-wishers who had helped me from time to time.

Last but not the least; I would like to thank my family for their encouragement and support in all my endeavors. I owe all my success to them.

Contents

Abstract **i**

Acknowledgements **ii**

1 Introduction **1**

2 Waypoint Navigation **1**
2.1 Pure Pursuit 1
2.1.1 Theoretical Derivation 1
2.2 Implementation 3
2.2.1 Path Representation 3
2.2.2 Tracking Strategy 3

3 Laser based Obstacle detection **4**
3.1 Proposed Method 5

4 Path Planner **5**
4.1 Algorithm for Obstacle Avoidance 9
4.1.1 Case 1: Obstacle to the right($a < 90°$ and $b < 90°$) 9
4.1.2 Case 2: Obstacle to the left($a > 90°$ and $b > 90°$) 10
4.1.3 Case 3: Obstacle at the centre($a < 90°$ and $b > 90°$) 11
4.2 Calculation of new avoidance point 13

5 Results **13**

6 References **14**

List of Figures

1 Geometry of Algorithm 2
2 Geometric Transformations 6
3 Implementation of Proposed method 7
4 Obstacle within the corridor 7
5 Obstacle outside the corridor 8
6 Obstacle on the right of vehicle 9
7 Obstacle on the right with no clearance 10
8 Obstacle on the left of vehicle 11
9 Obstacle at the centre of the path 12
10 Calculation of new avoidance point 13

1 Introduction

Autonomous vehicles have been present in industry for decades as a means for automated transportation of goods and objects . They are usually guided in the environment using some kind of artificial markers on dedicated environments. The control system of an autonomous robot involves several functions such as perception, path planning and path tracking, that are strongly interconnected.Research on path planning for mobile robots is one of the most important aspects in mobile robots research. Path planning for a mobile robot is to find a collision-free route, through the robot's environment with obstacles, from a specified start location to a desired goal destination while satisfying certain optimization criteria.

The ability of a real vehicle to track a path depends on the characteristics of the path, so continuity in position, orientation and curvature are basic requirements to be satisfied by the path generator [1]. However, smooth variation in curvature is still needed because of the relation with the characteristics of the required steering motion.

Obstacle avoidance is a compulsory technology for autonomous vehicles. As a matter of fact, it is a critical aspect given the safety issues involved both for the occupants and for other road users.

The main purpose of this technical report is to describe in detail the implementation of the pure pursuit algorithm given the list of way points in the map. Given the general success of the algorithm over the past few years, it seems likely that it will be used again in land based navigation problems. This report also includes a geometric derivation of the method, and presents some insights into the performance of the algorithm as a function of its parameters. It also presents an algorithm to avoid obstacles in the planned path if an obstacle is detected.

2 Waypoint Navigation

VipaLab travels autonomously from a starting point to a desired target destination by following list of way-points, avoiding obstacles in the course, knowing the coordinates of those targets. The way-point navigation is achieved by using **Pure Pursuit** algorithm as explained below.

2.1 Pure Pursuit

Pure pursuit is a tracking algorithm that works by calculating the curvature that will move a vehicle from its current position to some goal position. The whole point of the algorithm is to choose a goal position that is some distance ahead of the vehicle on the path. The name pure pursuit comes from the analogy that we use to describe the method. We tend to think of the vehicle as chasing a point on the path some distance ahead of it - it is pursuing that moving point. That analogy is often used to compare this method to the way humans drive. We tend to look some distance in front of the car and head toward that spot. This lookahead distance changes as we drive to reflect the twist of the road and vision occlusions.

2.1.1 Theoretical Derivation

The pure pursuit approach is a method of geometrically determining the curvature that will drive the vehicle to a chosen path point, termed the goal point. This goal point is a point on the path that is at a certain lookahead distance from the current vehicle position. An arc that joins the current point and the goal point is constructed. The chord length of this arc is the lookahead distance, and acts as the third constraint in determining a unique arc that joins the two points. Consider the lookahead distance to be analogous to the distance to a spot in front of a car that a human driver might look toward to track the roadway

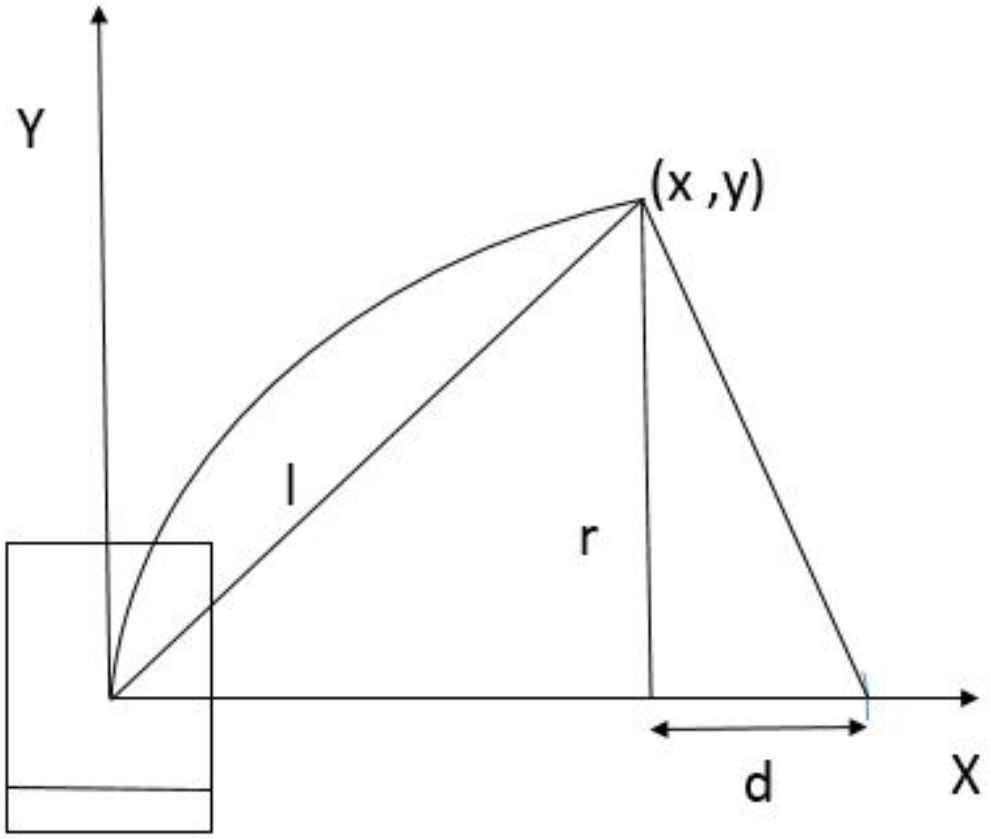


Figure 1: Geometry of the Algorithm

Consider Figure 1. The vehicle is pictured, with the axes of the vehicles coordinate system drawn. The X axis passes through the rear axle of the vehicle. The point (x,y), which is one lookahead distance 1 from the origin, is also shown. The point (x,y) is constrained to be on the path. The objective is to calculate the curvature of the arc that joins the origin to (x,y) and whose chord length is l.
The following two equations hold. The first is from the geometry of the smaller right triangle in Figure 1. The second from the summing of line segments on the x axis.

$$x^2 + y^2 = l^2 \tag{1}$$

$$x + d = r \tag{2}$$

- **Equation 1** describes the circle of radius l about the origin. This is the locus of possible goal points for the vehicle.
- **Equation 2** describes he relationship between the radius of the arc that joins the origin and the goal point, and the x offset of the goal point from the vehicle. This equation simply states that the radius of the arc and the x offset are independent and differ by d.

The next series of equations relate the curvature of the arc to the lookahead distance. The algebra is straightforward and requires no further explanation.

$$
\begin{aligned}
d &= r - x \\
(r-x)^2 + y^2 &= r^2 \\
r^2 - 2rx + x^2 + y^2 &= r^2 \\
2rx &= l^2 \\
r &= \frac{l^2}{2x} \\
\gamma &= \frac{2x}{l^2}
\end{aligned}
$$

The curvature has been related to the x offset of the goal point from the origin by the inverse square of the lookahead distance l. There is some similarity in form to a proportional controller where the gain is 2 times the inverse square of l. However the error in this form is the x offset of a point ahead of the vehicle.

2.2 Implementation

The method itself is fairly straightforward, as in the implementation. The only real implementation problems lie in updating the path with new information from the planner i.e. the **path information**.

2.2.1 Path Representation

A path is represented as a set of discrete points. (It is stored in the text file.) Typically a path point is of some PATH-TYPE that is a struct containing the following information:

- x location in global coordinates.
- y location in global coordinates.

2.2.2 Tracking Strategy

The vehicle changes its curvature by repeatedly fitting circular arcs to the goal point in a pure pursuit strategy [2]. This method is based on geometric considerations and it is very easy to compute. The required curvature of the vehicle (γ) is computed by :

$$\gamma = \frac{2x}{l^2} \tag{3}$$

where x is the x displacement of the centre of the goal point in vehicle coordinates and l is the lookahead distance between the current position of the centre of the robot and the goal point (Figure 1).

When the robot is tracking a previously defined path, the objective is to generate the control commands for the vehicle in order to follow a path in spite of external perturbations. So, it must take into account the present position and the constraints imposed by the vehicle and its lower motion controllers.

The controller approaches the knowledge of the points that form the trajectory, so it is able to make a better tracking of the desired path [3].

Since the vehicle cannot correct the errors with respect to the nearest point in the desired path, the goal point is chosen some distance ahead from the nearest point. The lookahead distance is measured over the desired path in order to make the search easier.

A bigger lookahead implies smoother control but it also means worse tracking. A smaller lookahead can reduce the tracking errors, however, command control increases and can even become

unstable [4]. Explicit path tracking is based on position estimation produced by the odometric system of the mobile robot.

Considering that the points that define the path and the successive positions of the robot (x_0, y_0) are referred with respect to the same coordinates system, the recursive algorithm for path tracking in each period of control consists of the following steps:

1. Obtain the initial position of the robot. This position is provided by the odometric system of the vehicle.
2. Generate the commands that obtain the tracking of the desired path in a smooth manner:
   (a) Choose a goal point (x_g, y_g) at a certain distance ahead from the vehicle:
      i. Calculate the point in the path closer to the vehicle. This nearest point is called (x_c, y_c). The algorithm searches for the nearest point in the proximity of the last nearest point. In this way, the robot is able to follow paths with intersection points.
      ii. Compute a certain lookahead distance.
      iii. Obtain the goal point (x_g, y_g) as the point over the path to follow that is separated from the nearest point (x_c, y_c) the lookahead distance.
   (b) Compute the desired curvature of the vehicle γ by using the pure pursuit control law defined in Eq. (3).
   In this equation, l and x are calculated as:

$$l = \sqrt{(x_g - x_0)^2 + (y_g - y_0)} \tag{4}$$

$$x = (x_g - x_0)\cos\phi_0 + (y_g - y_0)\sin\phi_0 \tag{5}$$

   where ϕ_0 is the current orientation of the robot, defined as the yaw angle with respect to global world.
3. Move the robot towards the goal point with the desired curvature.
4. Wait a period of control, obtaining the new position of the robot provided by the odometric system and go to point 2 of the algorithm.

The path tracking is aborted when the goal point is placed out of the desired path.

- For derivation purpose $x - y$ plane is assumed.
- In Cobaye simulator it is $z - x$ plane and y is the height.

3 Laser based Obstacle detection

Programming third-generation robot systems is very difficult because of the need to program sensor feedback data. A visualization of the sensor view of a scene, especially the view of the laser scanner, helps programmers to develop the software and verify action plan of a robot. A laser range scanner operates on a similar principle to conventional radar. Electromagnetic energy is beamed into the space to be observed and reflections are detected as return signals from the scene. The scene is scanned with a tightly focused beam of amplitude-modulated, infrared laser light. As the laser beam passes over the surface of objects in the scene, some light is scattered back to a detector that measures both the brightness and the phase of the return signal. The brightness measurements are assembled into a conventional 2-D intensity image.

The laser scanner has the advantage that it gives us a detailed description of the field of view. The laser scanner works by measuring the time of flight of laser light pulses. It is a non-contact

measurement device that scans its surroundings two dimensionally. The pulsed laser beam is deflected by an internal rotating mirror so that a fan shaped scan is made of the surrounding area and the shape of the object is determined by the sequence of impulses received. We can get a maximum scan angle of 180°with a resolution of 0.25°, 0.5°, or 1°. Now with this resolution and scan angle we get a clear profile of the path in front of our robot. We get data such as at every angle scanned the distance of the point of reflection of laser beam from any object in the field of view with its coordinates can be determined. Giving these values in the algorithm used for tracking the robot can avoid obstacles easily. The main strength of the laser scanner is the data accuracy and resolution that is returned from the scanned field.

3.1 Proposed Method

The proposed obstacle detection stage does not allow distinguishing between different types of objects (pedestrians, poles, vehicles, etc.) since it has been conceived for generic obstacle detection. In this case, it is not particularly relevant to know the type of object that appears in the vehicle trajectory, since it represents a real 3D obstacle that has to be avoided.

The most important variable is the relative location of the obstacle with respect to the pre-defined trajectory and current position of the micro-bus. The system can handle obstacles that are not in the vehicle path, such as lampposts or pedestrians on the pavements, by merging the global position of the host vehicle and the relative position of the detected obstacles. Note that this procedure is limited by the accuracy of both measurements, and some false positives can arise if they are very close to the road. In such cases, if the path planner is not able to provide a safety avoidance route, the vehicle will stop.

4 Path Planner

In the path-planning stage, trajectory and obstacles are managed in GPS Cartesian coordinates (X,Z), therefore the global position of the obstacle has to be calculated every time an obstacle is detected. A scheme of the geometric transformations applied for global position estimation is depicted in Figure 2. The waypoints that compose the trajectory, as well as the limits of the road, are shown in Figure 3. The trajectory is composed by a list of waypoints (pairs of X and Z).

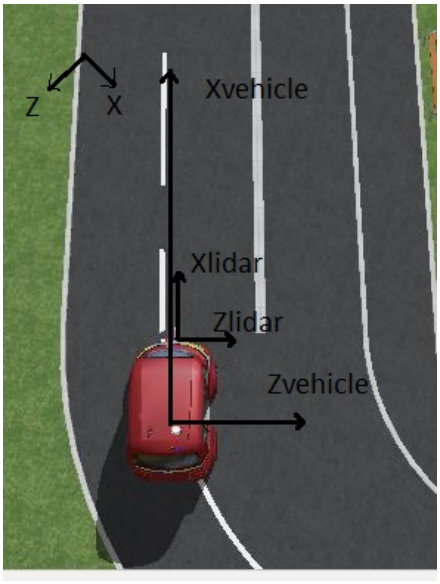


Figure 2: Two geometric transformations (rotation, translation) are calculated: the first one to convert from vehicle GPS position(A) to vehicle LIDAR position in cartesian coordinates(B). When cartesian coordinates of LIDAR position are calculated, a second transformation converts from LIDAR local coordinates to global cartesian coordinates(C).

The safety area is defined as the lateral space the vehicle needs to avoid a collision with an object. That distance is the addition of vehicle width, 0.5 metres on the left and 0.5 metres on the right. If there are one or more obstacles in the planned path , the trajectory needs to be re-calculated for obstacle avoidance. Free space on the left and right sides from the obstacle to road limits are calculated and the location of the bigger one is the side the vehicle is going use to avoid the object. If free space is not enough for driving safely, the vehicle will stop.

When an obstacle is detected in the safety area, a new trajectory has to be calculated to avoid the same. As depicted in Figure 3, the new trajectory is composed of one new point located near the obstacle but in a parallel trajectory to the original route.

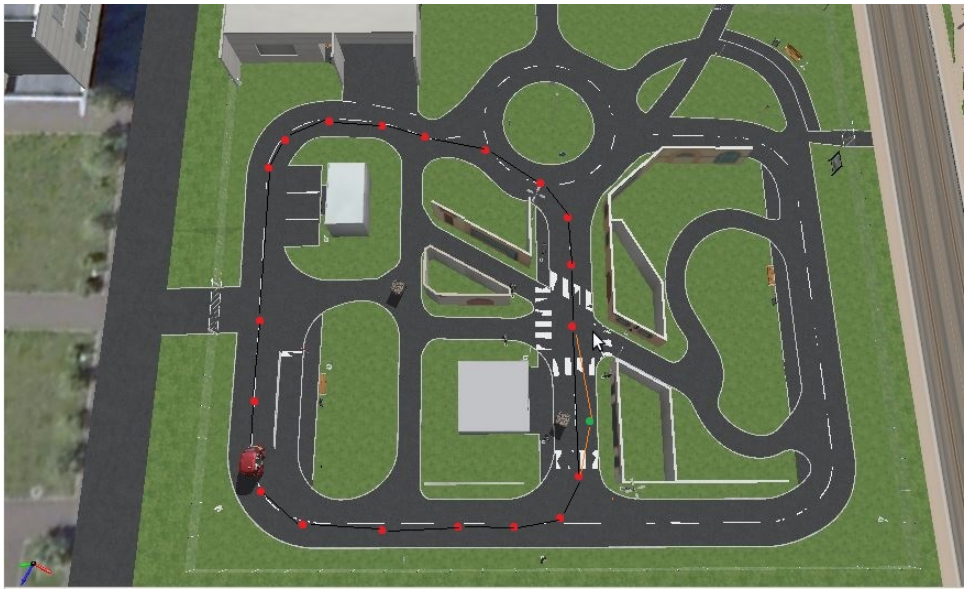


Figure 3: New points are added to the previous route close to the obstacle but shifted from the original path.

In order to guarantee the safety of the avoidance manoeuvres during the experiments, the system was strictly designed to avoid static obstacles. The avoidance of dynamic obstacles, which needs to introduce future position estimation techniques and adaptive avoidance trajectories generation, is an interesting research topic. However, it is out of the scope of this document and is considered as future research work.

In order to reduce unnecessary computation and consideration, we ignore two types of the obstacles that are given by the obstacle detection laser.

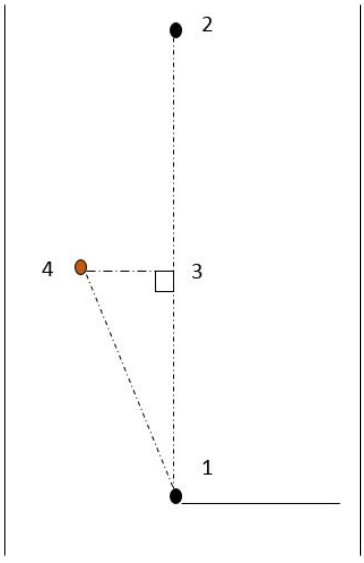


Figure 4: Calculating if an obstacle is within the corridor

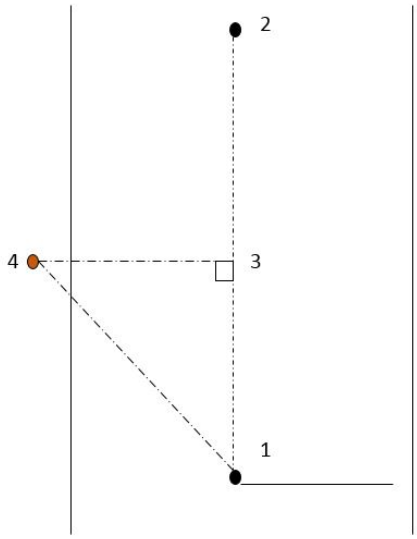


Figure 5: An obstacle is outside the corridor

One type is the obstacle that is outside the corridor segment, we use the sine rules to calculate that if an obstacle is outside the corridor(Figure 4). The corridor segment is determined by the current heading waypoint and passed waypoint. Assume P4 is the obstacle, we calculate the length of (P4 and P3), if it is larger(Figure 5) than the width of corridor, then we say the obstacle is outside the corridor.

Another type is the obstacle that is outside the heading range of the vehicle. This is done by limiting range of scanning angle as well as maximum distance range of LIDAR.

4.1 Algorithm for Obstacle Avoidance

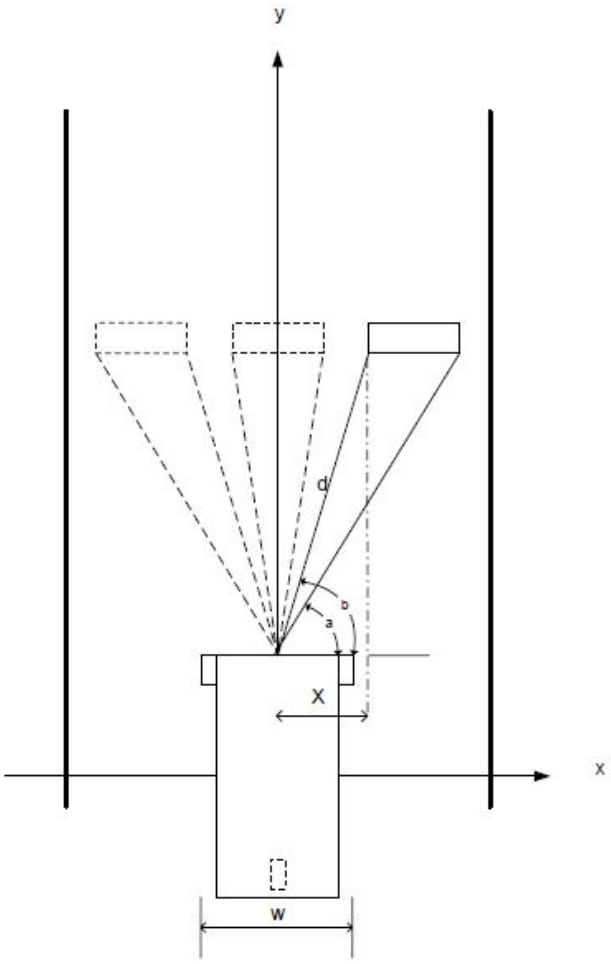


Figure 6: Obstacle on the right of the Vehicle with clearance

4.1.1 Case 1: Obstacle to the right($a < 90°$ and $b < 90°$)

1. Check if vehicle can maintain same path and avoid obstacle without offsetting the centroid of the vehicle from the centre of the track.
2. If $X = d\cos b$ and $X > \frac{w}{2}$ it can continue on straight path without offsetting vehicle centroid.
3. When $X = d\cos b$ and $X < \frac{w}{2}$, a new avoidance point needs to be calculated and to be placed on left side parallel to the planned route. (Taking into consideration the vehicle width and safety factor.)

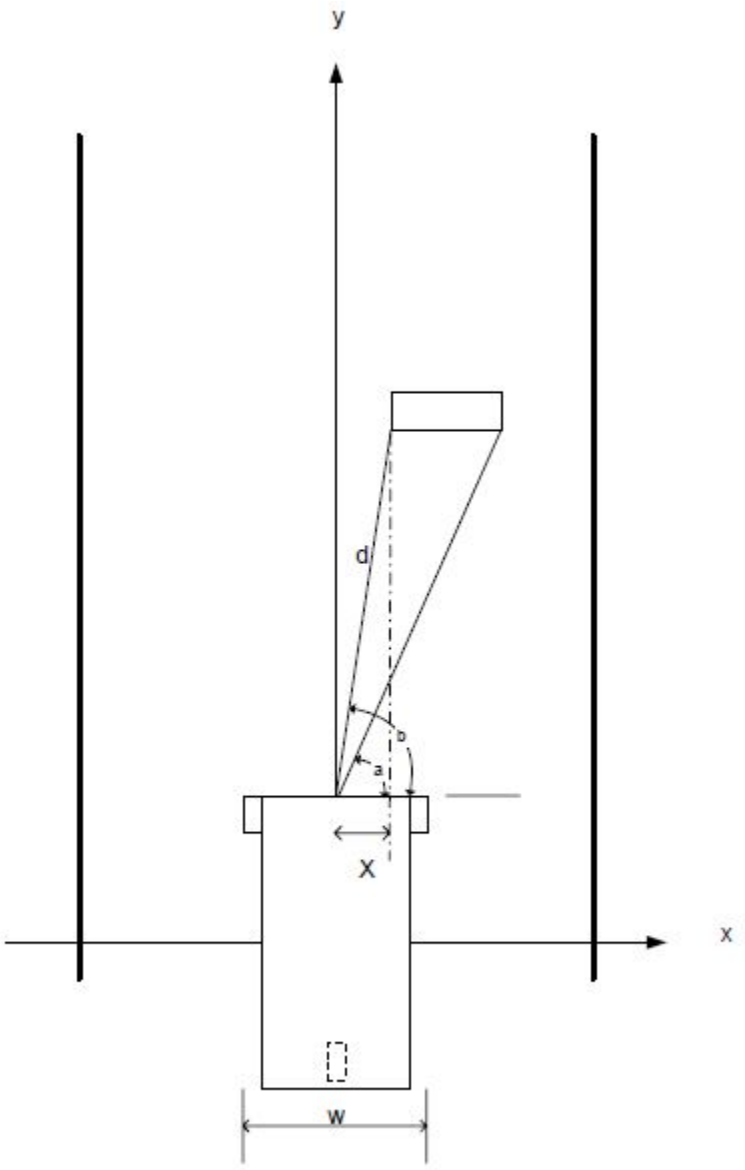


Figure 7: Obstacle on the right of the Vehicle with no clearance

4.1.2 Case 2: Obstacle to the left($a > 90°$ and $b > 90°$)

1. Check if vehicle can maintain same path and avoid obstacle without offsetting the centroid of the vehicle from the centre of the track.
2. If $X = d\cos b$ and $X > \frac{w}{2}$ it can continue on straight path without offsetting vehicle centroid.
3. When $X = d\cos b$ and $X <= \frac{w}{2}$, a new avoidance point needs to be calculated and to be placed on right side parallel to the planned route. (Taking into consideration the vehicle width and safety factor.)

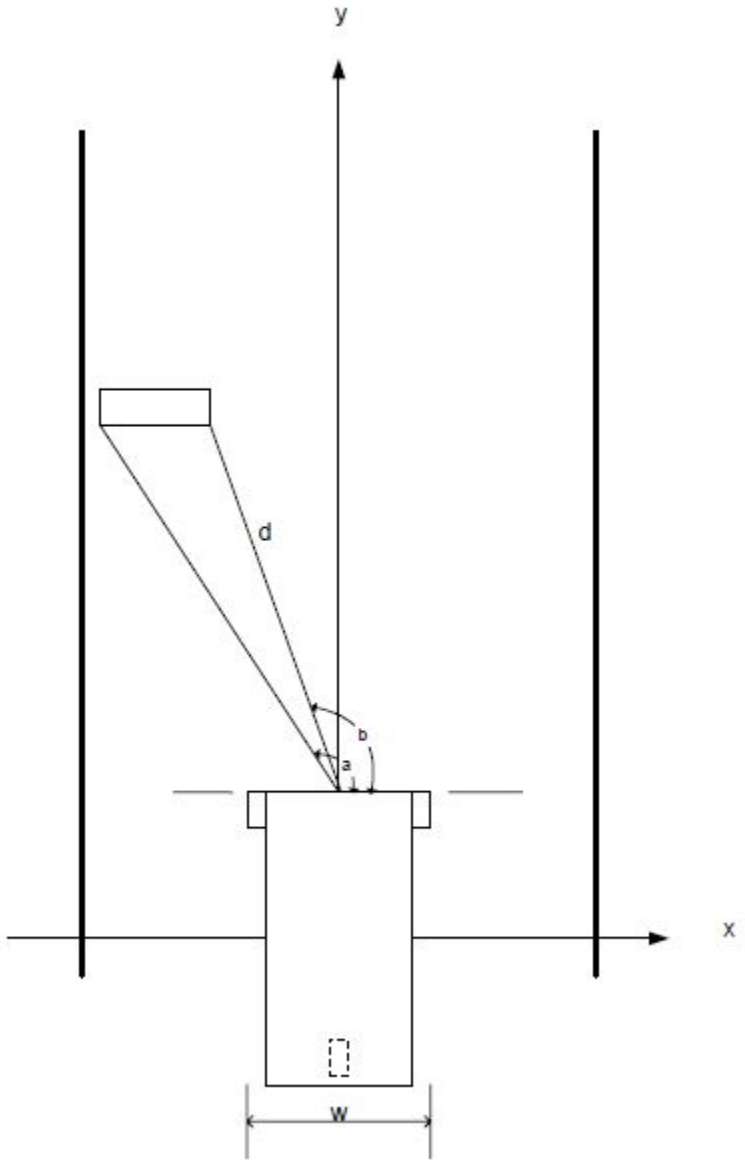


Figure 8: Obstacle on the left of the Vehicle with clearance

4.1.3 Case 3: Obstacle at the centre($a < 90°$ and $b > 90°$)

1. If $(180° - b) > a$, a new avoidance point should be placed on left side parallel to the planned route.
2. If $(180° - b) < a$, a new avoidance point should be placed on right side parallel to the planned route.

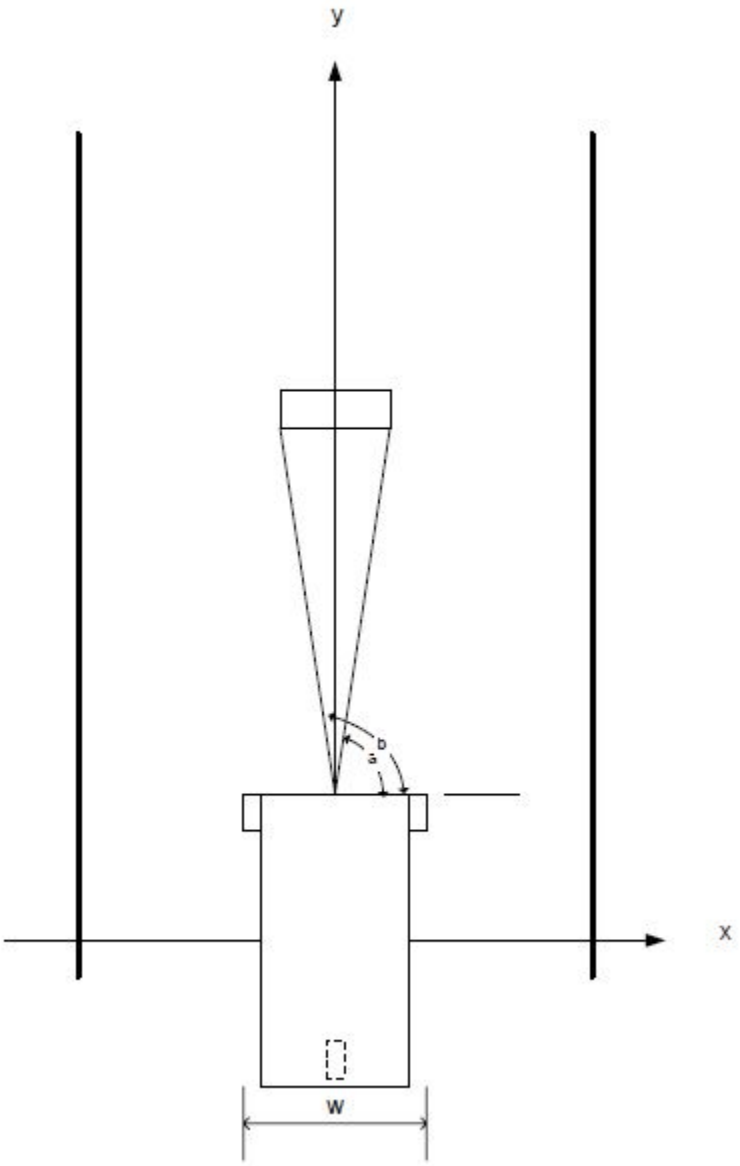


Figure 9: Obstacle at the centre of the path

4.2 Calculation of new avoidance point

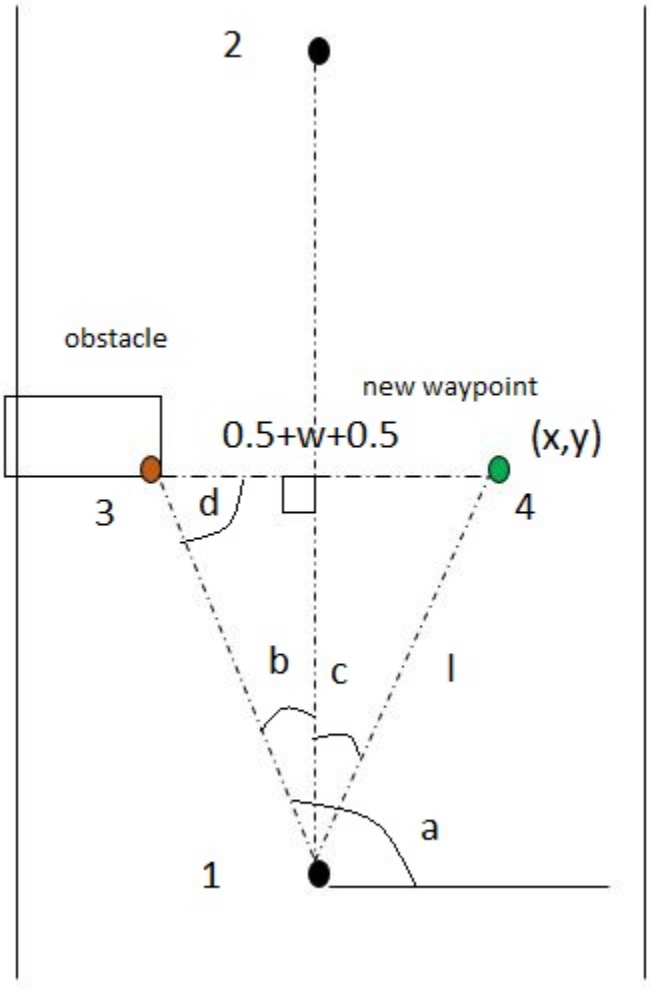


Figure 10: Obstacle on the left of the Vehicle with no clearance

1. The first contact angle a of LIDAR and distance from obstacle is calculated.
2. Then at fixed distance of $0.5 + w + 0.5$ a new avoidance point is to be calculated.
3. Using cosine law and trigonometric calculations, we can find distance l as well as the local co-ordinates (x, y) of new avoidance point in LIDAR reference frame as shown in Figure 10.

5 Results

- The vehicle in the simulator successfully avoids obstacle as per the proposed method given the desired goal destination.
- The code is successfully tested in Cobaye simulator, presenting excellent results in real time too.

6 References

[1] Rombaut, M., Segovia, A., Meizel, D. and Preciado, A. (1991),"*Displacement of a Mobile Robot in a Known Environment*", IMACS-MTCS Symposium, Villeneuve dAscq (France).

[2] Amidi, O. (1990), "*Integrated Mobile Robot Control*", Carnegie Mellon Univ. Robotics Institute, Technical Report CMU-RI-TR-90-17, Pittsburgh, Pennsylvania, (USA).

[3] Shin, D. H., Singh, S. and Whittaker, W. (1993), "*Path Generation for a Robot Vehicle Using Composite Clothoid Segments*", SICICA92 IFAC Symposium, Mlaga (Spain), pp. 533-538.

[4] Ollero, A., Garca-Cerezo, A. and Martnez, J. L. (1994), "*Fuzzy Supervisory Path Tracking of Mobile Robots*". Control Engineering Practice, Vol. 2, No. 2, pp. 313-319.

[5] R. Labayrade, M. Perrollaz, D. Gruyer, and D. Aubert. *Sensor Data Fusion for Road Obstacle Detection, Sensor Fusion and its Applications.* Ciza Thomas (Ed.), InTechopen, Chpater 16, 2010.

[6] Urdiales, C., Bandera, A., Arrebola, F. and Sandoval, F. (1998), "*Multi-Level Path Planning Algorithm for Autonomous Robots*", Electronics Letters, Vol. 34, No. 2, pp. 223-224.

[7] Sordalen, O. J. and Canudas de Wit, C. (1992), "*Exponential Control for Mobile Robot: Extension to Path Following*", IEEE Int. Conf. on Robotics and Automation, Nice (France), pp. 2158-2163.

[8] C. Deelertpaiboon and M. Parnichkun. Fusion of gps, compass, and camera for localization of an intelligent vehicle. *International Journal of Advanced Robotic Systems*, 5:315326, 2008.

[9] L. C. Fernandes, M. A. Dias, F. S. Osrio, and D. F. Wolf. A driving assistance system for navigation in urban environments. *IBERAMIA 2010, LNAI*, 6433:542551, 2010.

[10] D. F. Llorca, V. Milans, I. P. Alonso, M. Gaviln, I. G. Daza, J. Prez, and M. A. Sotelo. Autonomous pedestrian collision avoidance using a fuzzy steering controller. *IEEE Transactions on Intelligent Transportation Systems*, 12:390401, 2011.