Path Following and Obstacle Avoidance using Pure Pursuit and VFH

Experiment No. 2

EE615: Control and Computational Laboratory

April 10, 2025

Group 10

Full Name	Roll No.
Sahil Patil	24M1087
Shivangi Sharma	24M0027

Contents

1 | Introduction

In mobile robotics, smooth path following together with collision avoidance is a key challenge. The current experiment aims at the implementation and verification of two core navigation algorithms: Pure Pursuit for optimal path following and Vector Field Histogram (VFH) for optimal obstacle avoidance, based on MATLAB and Simulink as the development and simulation tools.
Pure Pursuit is a nonlinear path tracker controller, which computes the curvature required to arrive at a target look-ahead point along the reference path. The robot can follow curved and straight paths very accurately by continuously adjusting the steering angle towards the target point. This is especially useful in scenarios where the robot must stick to a pre-defined path, such as in warehouse navigation or self-driving car driving on pre-defined roads.
The Vector Field Histogram (VFH) algorithm is an extension of Pure Pursuit with the addition of dynamic obstacle avoidance. It takes real-time sensor measurements—usually laser or ultrasonic range data—into a polar histogram representing the density of obstacles in the neighborhood of the robot. The algorithm then chooses a steering direction which avoids interference caused by obstacles and continues to progress towards the destination.

2 | Aim

The aim of the experiment is to make the robot move from the point (x,y) = (2,4) to the point (x,y) = (10,4). The robot has to pass through the point (x,y) = (2,10) on its way to the destination. Further, the robot has to avoid obstacles on its way as it moves through the target goals

3 | Objective

1. To implement the Pure Pursuit algorithm in Simulink for accurate path tracking.
2. To develop the Vector Field Histogram (VFH) algorithm in MATLAB/Simulink for real-time obstacle avoidance, without using built-in functions such as `controllerVFH`.
3. To test the integrated navigation system under different scenarios with both static and dynamic obstacles.
4. To analyze and compare the robot's performance in terms of:
   - Path-following accuracy,
   - Smoothness of motion, and
   - Success in obstacle avoidance.

4 | Experimental Setup

- **Platform:** MATLAB & Simulink
- **Robot Model:** Differential drive robot with range sensor inputs
- **Inputs:**
  - Predefined path for Pure Pursuit
  - Range data (angles and distances) for VFH
- **Outputs:**
  - Steering direction
  - Velocity commands
- **Scenarios:**
  - Open space path following
  - Path with static obstacles
  - Path with dynamic obstacles

5 | Block Diagram

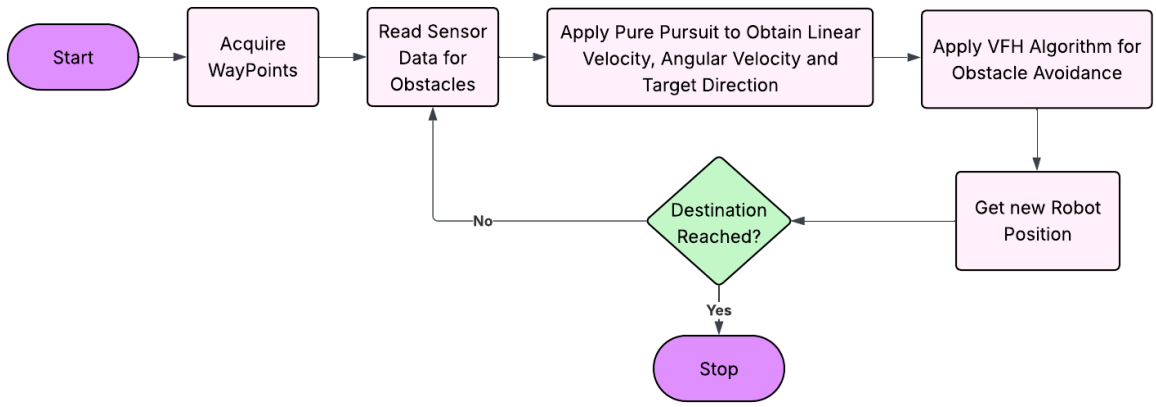
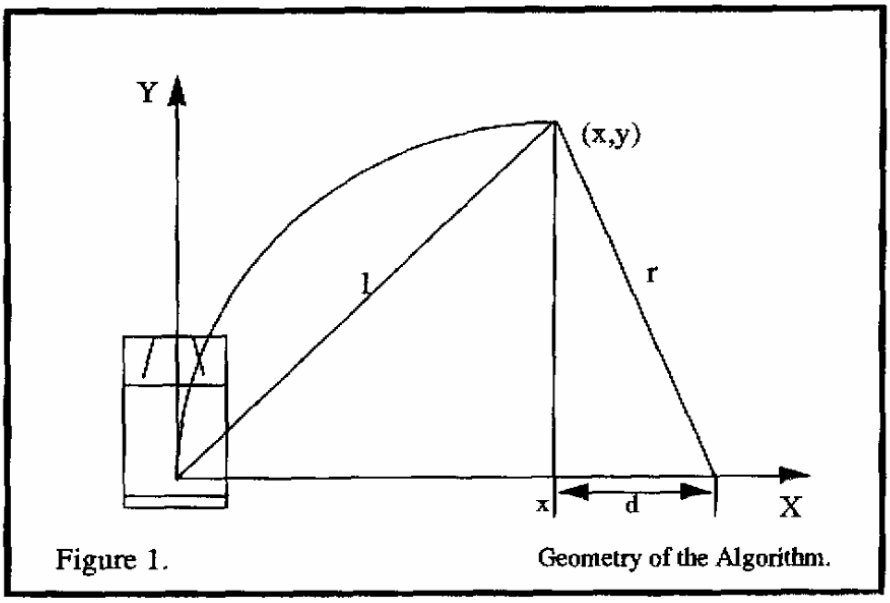


Figure 5.1: Block Diagram

6 | Procedure

6.1 | 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.



1. **Assumptions and Setup:**

   Assume the vehicle is located at position (x, y) with orientation θ in a 2D plane. A goal point is selected on the reference path at a fixed arc length ahead of the vehicle, denoted as the *look-ahead*

distance L_d. The coordinates of this look-ahead point, expressed in the vehicle's local coordinate frame, are (x_L, y_L).

The vehicle is modeled using a kinematic bicycle model, and the aim is to compute the required curvature κ so that the vehicle follows a circular arc towards this goal point.

2. **Relationship Between Geometry and Curvature:**

   From the geometry of a circle, the arc connecting the vehicle to the look-ahead point subtends an angle at the circle's center. Let α be the angle between the vehicle's heading and the line connecting the vehicle to the look-ahead point.

   Using basic trigonometry:

$$\sin(\alpha) = \frac{y_L}{L_d} \tag{6.1}$$

   The curvature $\kappa = \frac{1}{R}$ of the circular arc can be related to α as:

$$\kappa = \frac{2\sin(\alpha)}{L_d} \tag{6.2}$$

3. **Substituting to Eliminate α:**

   Substituting Equation (1) into Equation (2), we get:

$$\kappa = \frac{2y_L}{L_d^2} \tag{6.3}$$

   This is the fundamental control law used in the Pure Pursuit algorithm, providing the curvature needed to reach the goal point based on its position relative to the vehicle.

4. **Computing Steering Angle:**

   Once the curvature is known, the corresponding steering angle δ can be computed using the kinematic bicycle model:

$$\delta = \tan^{-1}(L \cdot \kappa) \tag{6.4}$$

   where L is the wheelbase of the vehicle.

   This steering angle δ is then applied to the vehicle to follow the computed circular arc, enabling smooth and accurate path tracking.

6.2 | VFH Algorithm

The **Vector Field Histogram (VFH)** is a real-time obstacle avoidance algorithm for mobile robots that enables fast, local navigation without relying on a global map. The method consists of the following steps:

1. **Histogram Grid Creation:**

   The robot constructs a two-dimensional Cartesian histogram grid from range sensor data. Each cell (i, j) stores an obstacle density value $m_{i,j}$ based on the proximity of detected obstacles:

$$m_{i,j} = \sum c^2 \left(a - b \cdot d_k\right)$$

   where:

   - d_k is the distance to the k^{th} obstacle point,
   - a, b are tuning constants,
   - c is certainity index

2. **Polar Histogram Generation:**

   The 2D histogram is converted to a 1D polar histogram divided into angular sectors. The obstacle density for each sector h_k is computed by summing the values of the cells that fall within the sector:

$$h_k = \sum_{(i,j) \in S_k} m_{i,j}$$

   where S_k is the set of grid cells mapped to sector k.

3. **Histogram Smoothing:**

   To reduce noise and produce smoother behavior, the polar histogram is filtered using a moving average:

$$\tilde{h}_k = \frac{1}{2w+1} \sum_{i=-w}^{w} h_{k+i}$$

   where w is the smoothing window size.

4. **Steering Direction Selection:**

   The steering angle is computed based on the sector index of the target direction and the candidate valley range from the polar histogram. Let:

   - $k_n = k_{\text{target}} + \Delta_{\text{sign}} \cdot \Delta_k$
   - $\text{sector\_count} = |\text{candidate\_valley}(1) - \text{candidate\_valley}(2)|$
   - $s_{\max}$ be the maximum allowed sector width
   - α be the angular resolution per sector

   [a] **If** $\text{sector\_count} > s_{\max}$, then:

$$k_f = \begin{cases} k_n + s_{\max}, & \text{if } \min(\text{candidate\_valley}) < k_f < \max(\text{candidate\_valley}) \\ k_n - s_{\max}, & \text{otherwise} \end{cases}$$

$$k_c = \frac{k_n + k_f}{2}$$

   [b] **Else**, when $\text{sector\_count} \leq s_{\max}$:

$$k_f = k_n + \text{sector\_count}$$

$$k_f = \begin{cases} k_f, & \text{if } \min(\text{candidate\_valley}) < k_f < \max(\text{candidate\_valley}) \\ k_n - \text{sector\_count}, & \text{otherwise} \end{cases}$$

$$k_c = \frac{k_n + k_f}{2}$$

   [c] **Steering Angle:**

$$\theta = k_c \cdot \alpha - \frac{\pi}{2}$$

   If no valid valley is found, then:

$$\theta = \text{NaN}$$

   This logic ensures the selected direction lies within the free space of the candidate valley and keeps the robot heading smooth and safe.

This method enables the robot to steer smoothly toward the goal while dynamically avoiding obstacles in real-time.

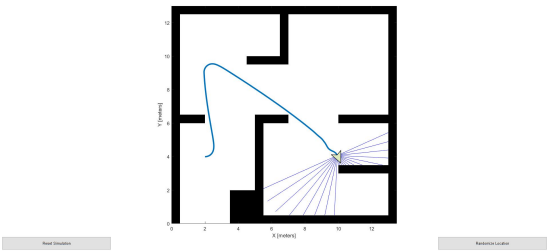
7 | Results

7.1 | Parameters Pure Pursuit

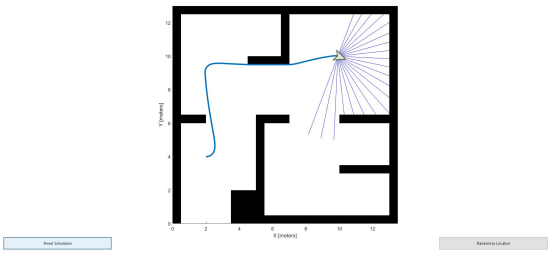
1. Look-ahead Distance : 1 m
2. Linear Velocity : 0.5 m/s

7.2 | Parameters VFH

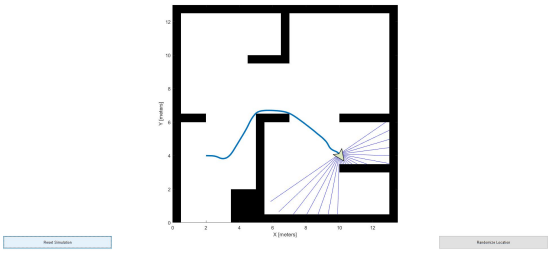
1. Histogram Constant : a = 6
2. Number of Sectors : 180
3. Scan Distance : $d_{max} = 6$
4. Threshold for Determining Valley : 2
5. Threshold to determine Candidate Valley $s_{\max}$: 10
6. Smoothing Window L: 10



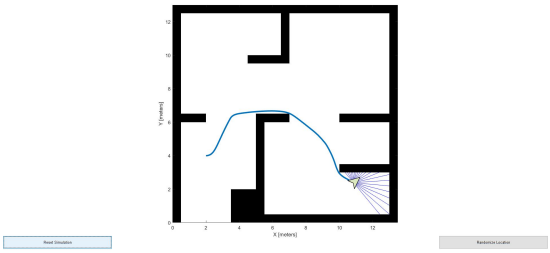
(a) WayPoints (2,4) - (2,10) - (10,4)



(b) WayPoints (2,4) - (2,10) - (10,10)



(c) WayPoints (2,4) - (10,4)



(d) WayPoints (2,4) - (4,7) - (12,2)

Figure 7.1: Overall caption for the 4 figures

8 | Challenges Faced

1. **Pure Pursuit Algorithm:**
   - *Overshooting at Sharp Turns:* The robot often overshoots tight corners, especially at higher speeds or when the lookahead distance is not properly tuned.
   - *Path Oscillation:* Inaccurate heading or frequent path corrections can lead to oscillatory motion, particularly in narrow passages.
   - *Lookahead Tuning:* Selecting an appropriate lookahead distance is crucial; too short causes instability, too long reduces tracking precision.
2. **Vector Field Histogram (VFH) Algorithm:**
   - *Local Minima Traps:* In complex environments, the robot may get stuck in local minima where no valley appears directly toward the goal.
   - *Parameter Sensitivity:* Performance heavily depends on tuning parameters like sector size, smoothing window, and cost function weights.
   - *Dynamic Obstacles:* Reacting to fast-moving obstacles is challenging as VFH relies on a static histogram at each time step.
   - *Computational Load:* Real-time performance requires efficient computation, especially for high-resolution histograms.

9 | Limitations

1. **Pure Pursuit Algorithm:**
   - Assumes accurate localization and odometry; performance degrades with noisy sensor data.
   - Struggles with dynamic environments since it does not account for obstacles.
   - Not optimal for narrow or cluttered paths as it may deviate significantly from the desired trajectory.
   - Turning radius constraints can limit its ability to follow sharp curves precisely.
2. **Vector Field Histogram (VFH) Algorithm:**
   - May fail in narrow passages if no suitable valley is detected in the histogram.
   - Relies on real-time sensor data; lag or low-resolution sensing reduces effectiveness.
   - Cannot handle long-term path planning; only makes local decisions.
   - Performance highly sensitive to parameter tuning and histogram resolution.