Statistics and Applications {ISSN 2454-7395 (online)}
Volume 22, No. 1, 2024 (New Series), pp 57-71
http://www.ssca.org.in/journal

Eigenspace Based Online Path Planner for Autonomous Mobile Robots

Shyba Zaheer[1], Imthias Ahamed T.P.[1], Tauseef Gulrez[2] and Zoheb Zaheer[3]
[1]*Department of EEE, TKM College of Engineering, Kerala, India*
[2] *Department of Computing and Research, Syscon Private Limited, Melbourne, Australia*
[3]*Mitsogo Inc, Kerala, India*

Received: 03 August 2022; Revised: 10 December 2022; Accepted: 25 May 2023

Abstract

This paper proposes a target oriented online path planning algorithm which is capable of navigating a mobile robot autonomously in unknown environments. The proposed technique called Free Configuation Eigenspace (FCE) finds collision free path from laser sensor data by computing its eigenvectors. The paper describes an online 2D simulation method of FCE with static obstacles and start and goal positions. The proposed method is benchmarked aginst the well known online path planner Vector Field Histogram (VFH). In this 2D simulation, the robot model used is a differential drive robot and it is assumed that the robot is equipped with a laser scanner. Simulation experiments are done with start and goal positions on simulated 2D maps in MATLAB with different obstacle courses. The respective trajectories for different start and goal positions were generated on the map and path lengths analyzed

Key words: Online path planning; Online obstacle avoidance; Eigenspace; Eigenvector.

1. Introduction

An autonomous robot's ability to plan its motion in real-time has become a crucial part of modern intelligent robotics. Applications of path planning in online environments include the mining industry, planet exploration, reconnaissance, *etc.* This is also known as local path planning. Online path planning deals with the assessment of the dynamic conditions of the environment and identifying the positional relationships among various elements in the environment. In online navigation, the robot can autonomously decide its motion using equipped sensors such as laser sensors, ultrasonic range finders, sharp infrared range sensors, vision (camera) sensors, *etc.*

Pioneering work in online obstacle avoidance and path planning was initiated by Khatib (1986), konown as Artificialal Potential Fields method (APF) and is popular in mobile robotics. The idea of (APF) comes from the concept of the potential field in physics,

Corresponding Author: Shyba Zaheer
Email: s.shyba@gmail.com

which regards the movement of objects as the result of two kinds of forces. The robot is subjected to attractive forces from the target and repulsive forces from the obstacle. Under the action of the two forces, the robot moves toward the target point due to the resultant force and during the moving process, it can effectively avoid the obstacles and reach the target. Bug algorithms by Lumelsky and Stepanov (1987) are also used for online path planning which uses two-dimensional scenes filled with unknown obstacles. Bug algorithm assumes the robot as a point operating in the plane with a contact sensor or range sensor to detect obstacles. Bug algorithm uses a straightforward path planning approach to move towards the goal unless an obstacle is encountered, in which case it circumnavigates the obstacle until motion towards the goal is once again allowable. Another online planner in literature is the Dynamic Windows Approach (DWA), by Fox *et al.* (1997). This approach is derived directly from the motion dynamics of the robot and is therefore particularly well suited for robots operating at high speed. The dynamic window contains the feasible linear and angular velocities taking into consideration the acceleration capability of the robot.The collision cone concept based online path planning was proposed by Chakravarthy and Ghose (1998). The collision cone can be used to predict the possibility of collisions between two objects and to design collision avoidance strategies. In this method, a collision of a robot can be averted if the relative velocity of a robot with respect to a particular obstacle falls exterior to the collision cone.

One of the widely used sensor-based online path planning algorithms is Vector Field Histogram (VFH) by Borenstein *et al.* (1991). In VFH, a polar histogram is generated at every discrete point step to represent the polar density of the obstacles around the robot. The robot's steering direction is chosen based on the least polar density and closeness to the current steering direction. The VFH algorithm is fast, very robust, and insensitive to misreadings, allowing continuous and fast motion of the mobile robot without stopping for obstacles. But the VFH-controlled robot may get "trapped" in dead-end situations (as is the case with other local path planners). When trapped, mobile robots usually exhibit "cyclic behavior". Another limitation of this technique is that the polar histogram must be regularly generated for every time step. Hence in narrow hallways, the robot may move in an oscillatory fashion. Also, this method is suited for environments with sparse moving obstacles. Ulrich and Borenstein (1998) proposed a method known as **VFH+** that introduces some of the parameters tuning to accommodate the robot's width, also.

This paper is organized as follows: first it describes the materials and methods used for this study followed by the 2D simulation method of VFH with static obstacles and start and goal positions. Then it describes the proposed Free Configuration Technique(FCE) path planner proposed by us Zaheer *et al.* (2022). Detailed simulation results are included in section 4 and followed by result analysis and conclusion.

2. Materials and methodology

This section describes the materials and methods used for this study. The robot model used here is a differential drive robot and the sensor used is a laser sensor. In this 2D simulation, it is assumed that a vehicle is equipped with a scanning laser range sensor with a field of view of 240°. Also, vehicle location is known and only kinematic motion of the vehicle is considered. The simulation experiments are done with Start and Goal positions on simulated 2D maps with different obstacle courses. The performance analysis is done

for different scenarios: namely **Scenario-I** with 3 separate obstacles with and L-shaped wall obstacle and **Scenario-II and III** with differently shaped obstacles. The respective trajectories for different Start and Goal positions were plotted on the map and path lengths analyzed.

2.1. Differential drive kinematics

A differential drive robot consists of 2 drive wheels mounted on a common axis, and each wheel can independently be driven either forward or backward. For the robot to perform direction change in it's translational motion, the velocity of each wheel may be varied appropriately. The robot actually performs rotatory motion about a point along the common left and right wheel axis which is known as the ICC (Instantaneous Center of Curvature) as seen in Figure 1. Hence by varying the velocities of the two wheels, we can vary the trajectories that the robot take

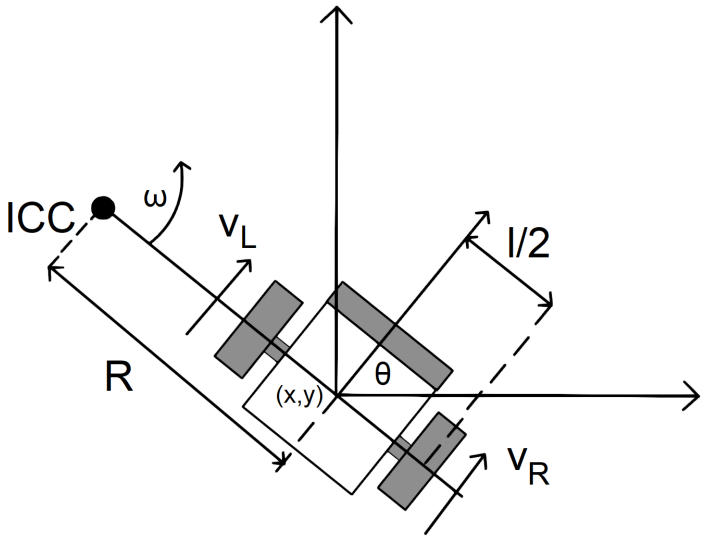


Figure 1: Differential drive kinematics

Since the rate of rotation ω about the ICC must be the same for both wheels, we can write the following equations:

$$\omega(R + \frac{l}{2}) = V_{\mathrm{R}} \tag{1}$$

$$\omega(R - \frac{l}{2}) = V_{\mathrm{L}} \tag{2}$$

where l is the distance between the wheels, V_R and V_L are the right and left wheel velocities along the ground respectively, and R is the signed distance from the ICC to the midpoint between the wheels. At any instance in time we can solve for R and ω:

$$R = \frac{l(V_{\mathrm{L}} + V_{\mathrm{R}})}{2(V_{\mathrm{R}} - V_{\mathrm{L}})} \tag{3}$$

$$\omega = \frac{V_{\mathrm{R}} - V_{\mathrm{L}}}{l} \tag{4}$$

In Figure 2, the velocity of the robot can be represented as a pair of vectors, $\vec{v}$ and $\vec{\omega}$, where $\vec{v}$ represents the linear velocity (forwards and backwards) of the robot and $\vec{\omega}$ represents the angular velocity of the robot. Given angular veocities of the right and left wheels ω_R and ω_L respectively, the linear and angular velocities of the differential drive robot are represented as shown in Equations 5 and 6.

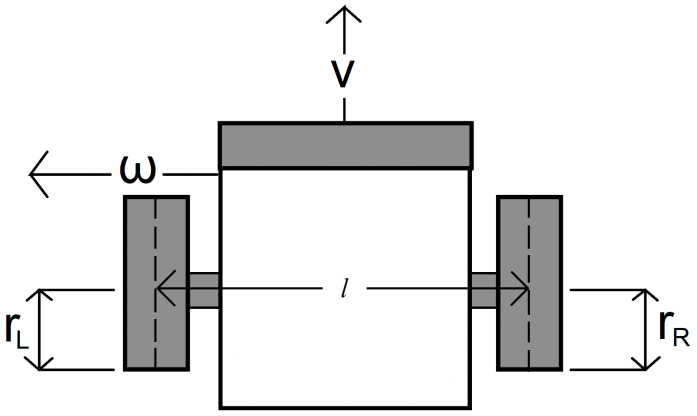


Figure 2: Physical configuration of the robot

$$v = \frac{r_R}{2}\omega_R + \frac{r_L}{2}\omega_L \tag{5}$$

$$\omega = \frac{r_R}{l}\omega_R - \frac{r_L}{l}\omega_L \tag{6}$$

where r_R and r_L are the wheel radii of the left and right wheels, respectively, and l is the width of the wheelbase as shown in Figure 2. The robot in the global coordinate frame is represented in Figure 3. The equations of motion are shown in Equation 7.

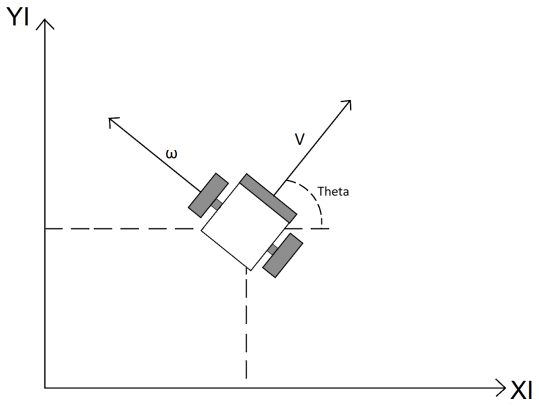


Figure 3: Robot's kinematics in global frame

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} v\ cos\theta \\ v\ sin\theta \\ \omega \end{bmatrix} \tag{7}$$

Suppose, at the initial time T_0 the pose is $[x_0, y_0, \theta_0]$; the pose at time t is $[x(t), y(t), \theta(t)]$; to find the pose at time T, we will have to integrate the variables at the pose $t + \Delta_t$ from within the limit T_0 to T which is added to the initial pose coordinates $[x_0, y_0, \theta_0]$, as follows:

$$x(T) = \int_{T_0}^{T} v(t)\cos(\theta(t))dt + x_0 \tag{8}$$

$$y(T) = \int_{T_0}^{T} v(t)\sin(\theta(t))dt + y_0 \tag{9}$$

$$\theta(T) = \int_{T_0}^{T} \omega(t)dt + \theta_0 \tag{10}$$

2.2. Principal component analysis (PCA)

PCA is a way of identifying patterns in data and expressing the data in such a way as to highlight their similarities and differences. Since patterns can be hard to find in data of high dimension, PCA helps us to identify patterns in data based on the correlation between features. It aims to find the directions of maximum variance in high-dimensional data and projects it onto a new subspace with equal or fewer dimensions than the original one.

In our case, with a laser range sensor, individual range measurements can be considered as an independent dimension. With such an approach, a range scanner with a 1^0 resolution and a 240^0 field view generates 240 observations which are called as point cloud

sensor data. Analysis of the point cloud data from such a scanner will show that adjacent range measurements are highly correlated. Thus, it is possible to use principal component analysis to determine a linear subspace with a minimum number of dimensions for representing an environment using a range sensor.

PCA finds Principal Components (PCs) that are linear combinations of the original variables ranked in terms of the variability in the data given by the variances. The corresponding orthogonal directions are given by the eigenvectors of the covariance matrix (C) of the data. The steps involved in PCA analysis are as follows :

- Standardize the dataset
- Compute the covariance matrix of the dataset.
- Perform eigen decomposition on the covariance matrix.
- Order the eigenvectors in decreasing order based on the magnitude of their corresponding eigenvalues.

Let S be the 2D point cloud data,where, $S_j = (x_j, y_j)$; $(S_j \in S \in R^2)$ and $\bar{S}$ is the mean of k no of sensor data points as given below.

To perform the PCA transformation, we have to compute the covariance matrix C of point cloud data set S using the below equation:

$$C_{2x2} = \frac{1}{K}\sum_{j=1}^{k}(S_j - \bar{S})(S_j - \bar{S})^T; \bar{S} = \frac{1}{K}\sum_{j=1}^{k} S_j \tag{11}$$

To perform transformation we have to solve the eigenvalue Equation 12

$$C\,V = \lambda\,V \tag{12}$$

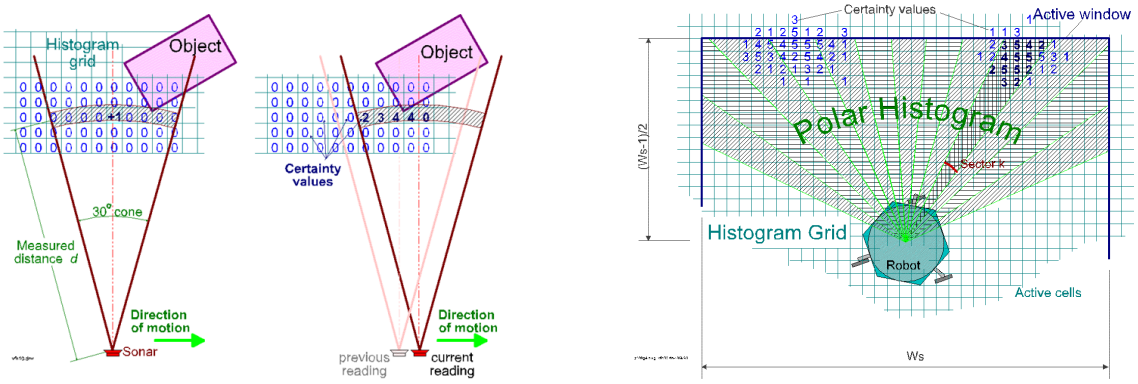
Solving the Equation 12 , we can get the eigenvalues λ, where $\lambda_1 \geq \lambda_2$ and eigenvectors V $[V_1, V_2]$.

These eigenvectors are called Principal Components (PCs). By applying a proper technique we can identify the pattern in 2D point cloud range data. In our case, the obstacle area or free area identification can be performed with this PCA technique .

3. Vector field histogram (VFH)

VFH method is executed in three main steps that are: Two dimensional cartesian histogram grid, Polar histogram sector and candidate valley selection. To begin with, on-board sensors such as ultrasonic sensor or laser rangefinder are used for mapping obstacles into histogram grid. In this step, the two-dimensional cartesian histogram grid is continuously updated in with range data sampled by the on-board range sensors as shown in Figure 4a.

At the second step, a one-dimensional polar histogram is constructed around the robot's momentary location by dividing the polar histogram into angular sectors of suitable width as shown in Figure 4b. At the third step the output of the VFH algorithm, which



(a) Cratesian histogram **(b) Polar histogram**

Figure 4: Algorithm steps - I and II

(a) Polar obstacle density of wide valley **(b) POD with narrow valley**

Figure 5: VFH Algorithm steps - III

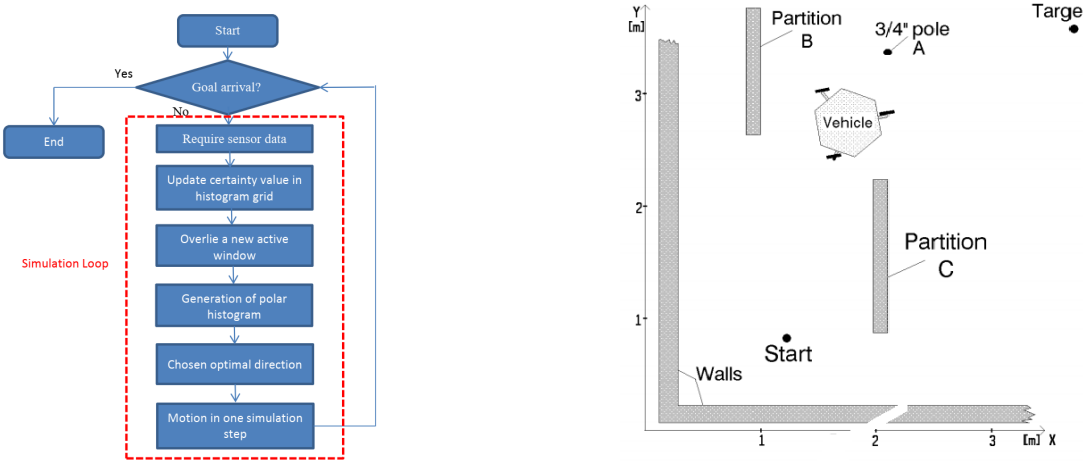
is the reference value for the new steering direction of the robot. The optimal direction is selected in each candidate valley such that every sector density is less than a suitable threshold value. The algorithm measures the size of the selected valley. Hence, two types of valleys are distinguished, namely, wide and narrow ones.A valley is considered wide if the no of consecutive polar sectors (S) are greater than 18 nos ($S_{max=18}$).

The sector nearest to target is denoted as **kn** and the far border sector is denoted as **kf** and is defined as $kf = kn + S_{max}$. The desired steering direction is then defined by $\theta = \frac{(kn+kf)}{2}$ and is closer to the goal or target directions as shown in Figure 5a.

In the second case, a narrow valley between closely spaced obstacles (shown in Pink), are shown in Figure 5b. Here the far border **kf** is less than S_{max} . However, the direction of travel is again chosen as $\theta = \frac{(kn+kf)}{2}$ and the robot maintains a course centered between obstacles as shown in Figure 5b.

3.1. VFH 2D simulation

In this section, we have done 2D Simulation of VFH algorithm in MATLAB. The **scenario-I** is an exact replica of the simulation done in the original paper by Borenstein *et al.* (1991). This scenario-I is shown in Figure 6b which contains three obstacles denoted as A,B,C with a L shaped wall.



(a) VFH simulation flow chart **(b) Original VFH simulation scenario**

Figure 6: VFH 2D Simulation

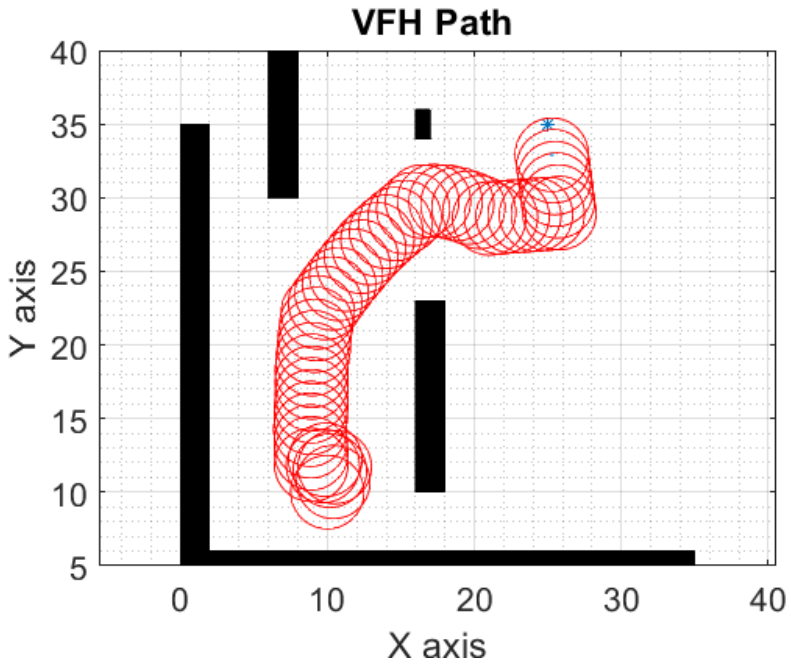


Figure 7: VFH Path: Start[10,10] , Goal[25,35]

The flow chart of VFH's 2D simulation is shown in below Figure 6a. 2D simulation is carried out for the VFH algorithm using MATLAB. The robot is assumed to be of circular shape of radius 0.025m and moving at a speed of 0.8 m/s. The Start position is at [10,10] and the Goal position is at [25,35] with a map scale 1 unit = 0.1m. The simulated robot trajectory is shown in Figure 7 and its path length is recorded in Table 1.

4. Free configuration eigenspace (FCE) formulation

The FCE method uses a two-stage sensor data reduction technique and three levels of sensor data representation :

- The first level represents the detailed description of the robot's environment. In this level, the two-dimensional cartesian map (world model) is created.
- At the second level, the high dimensional sensor space is reduced to a low-dimensional Eigenspace around the momentary location of the robot by computing Principle Components of sensor data. These PCs provides a spatial interpretation of the environment in terms of its variance of the sensor data.
- The third level of data representation is the output of the FCE algorithm, which selects the PC direction which is closest to the goal direction

4.1. Proposed FCE goal reaching algorithm

A 2D online pathplanning algorithm has been formulated with static obstacles using FCE philosophy by Zaheer *et al.*(2014). The below sections describe the algorithm formulation and implementation of FCE's 2D Trajectory generation with Start and Goal positions for a robot **A**. The flow chart of 2D simulation is shown in Figure 8.

The flowchart has the followings steps:

- The algorithm starts by computing the distance and the angle from the **Start** to **Goal** positions.
- If the goal is not reached, then the sensor cartesian data is acquired from the map's obstacle positions.
- The two PCs of sensor data are computed by applying PCA
- From the two PCs, the PC direction closer to the goal position is found.
- Then the velocity components of the new PC angle is calculated
- The new position is computed from the current position and the velocity components.
- The new position is added to robot path array .
- Finally, the robot traverses the generated path.

4.2. FCE's 2D simulation algorithm

Algorithm 1 FCE - Eigen vector trajectory

Input:
RobotA Start Position: **PosA**[ ]
RobotA Goal Position : **GoalA**[ ]
Scan Data: **S**[ ]
Initialize Simulation parameters :($v = 0.8m/s, r_A = 0.025m, t = 1s, P = 1, N = 100$)
Output: **Path**[ ] Path from **Start** to **Goal**

Compute distance (D) and angle between from Start to Goal(angA)
Path$[P, 1] = PosA(1)$
Path$[P, 2] = PosA(2)$
if (D$\geq$0) and(D $<$ 2.5) **then**
    GoalReached=1

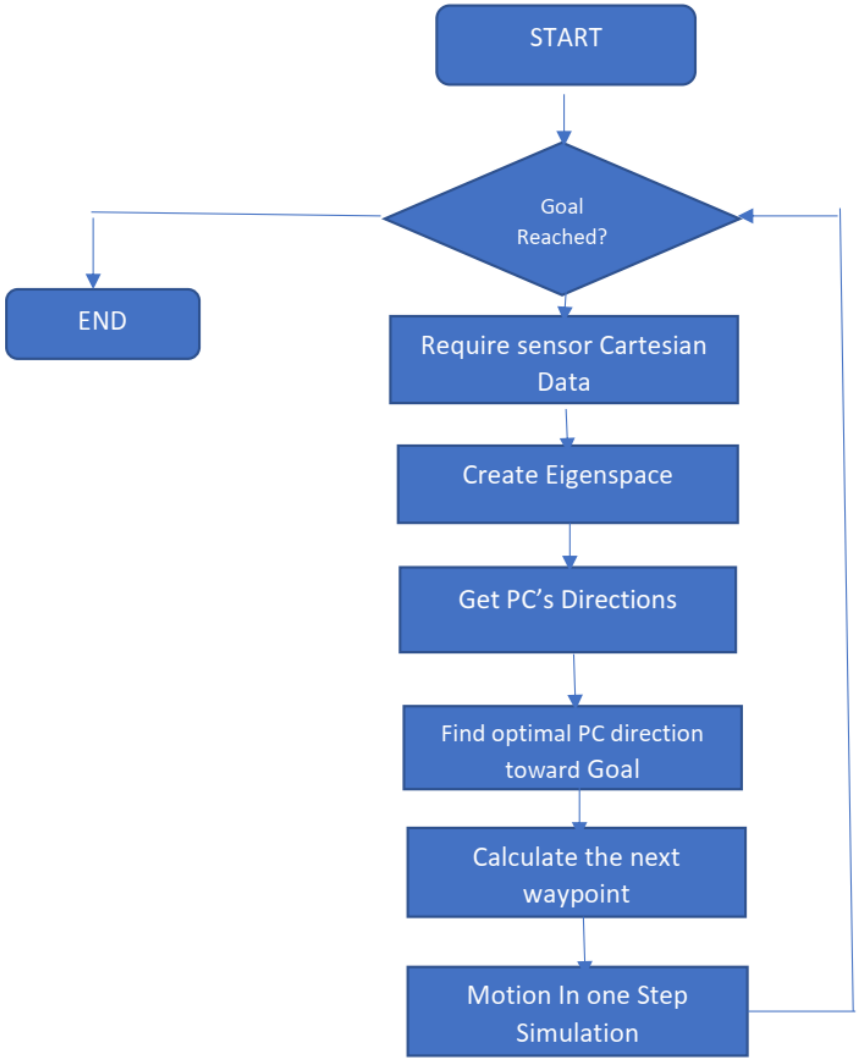


Figure 8: FCE 2D simulation flow chart

```
6: else
7:    GoalReached=0
8: end if
9: while (GoalReached==0) do
10:   repeat
11:      S=Get Cartetian Coordiante of Scan data
12:      [ PC ] =  eig (covariance (S));
13:      DirPC1 = atan2(PC(2, 1), PC(1, 1))
14:      DirPC2 = atan2(PC(2, 2), PC(1, 2))
```

newangleA=[DirPC1,DirPC2]
angleFuture=findFutureAngle(newangleA, angA);
angA=angleFuture
$V = [vcos(angA), vsin(angA)]$
$newX = PosA(1) + V(1) * t$
$newY = PosA(2) + V(2) * t$
$PosA = [newX, newY]$
$P = P + 1$
$Path[P, 1] = PosA(1)$
$Path[P, 2] = PosA(2)$
until (GoalReached==1) or (P==N)
end while
Plot the Path Points with robot **A** as circular shape

Algorithm 2 : Function: findFutureAngle()

Input: newangleA, angA
Output:angleFuture

$X1 = \cos(newangleA(1)) - \cos(angA)$
$Y1 = \sin(newangleA(1)) - \sin(angA)$
$Point1 = sqrt(X1^2 + Y1^2)$
$X2 = \cos(newangleA(2)) - \cos(angA)$
$Y2 = \sin(newangleA(2)) - \sin(angA)$
$Point2 = sqrt(X2^2 + Y2^2)$
if (Point1¡Point2) **then**
angleFuture=newangleA(1)
else
angleFuture=newangleA(2)
end if
Return (angleFuture)

In this simulation, the robot is assumed to be of circular shape with radius 0.025m and moving at a speed of 0.8 m/s. The simulation assumes that the robot is equipped with range sensor. The simulation starts with the input of initial Start position as (PosA) and the final Goal position as (GoalA). The initial step is to compute the distance (D) and the angle between the Start and the Goal Position (angA). If the distance value is greater than zero, then the obstacle free position which is more close to the Goal position has to be found from scan data. With FCE, this is achieved by computing the eigenvectors of the covariance matrix of the sensor cartesian data. This will give two PCs as shown in Figure 9. From these two PC's, the next suitable direction (angleFuture) to move towards the Goal with out hitting obstacles is identified by computing the distance between the new PCs direction point and the initial angle point (angA) and then selecting the minimum distance point among them as given in algorithm 2. Once the closest direction to the Goal is selected, the next step is to find the velocity components and the next position of the robot to move on. The next waypoint is computed by differential drive robot's kinematic equations and the values will be stored in the path array, which will give the obstacle free path from Start to

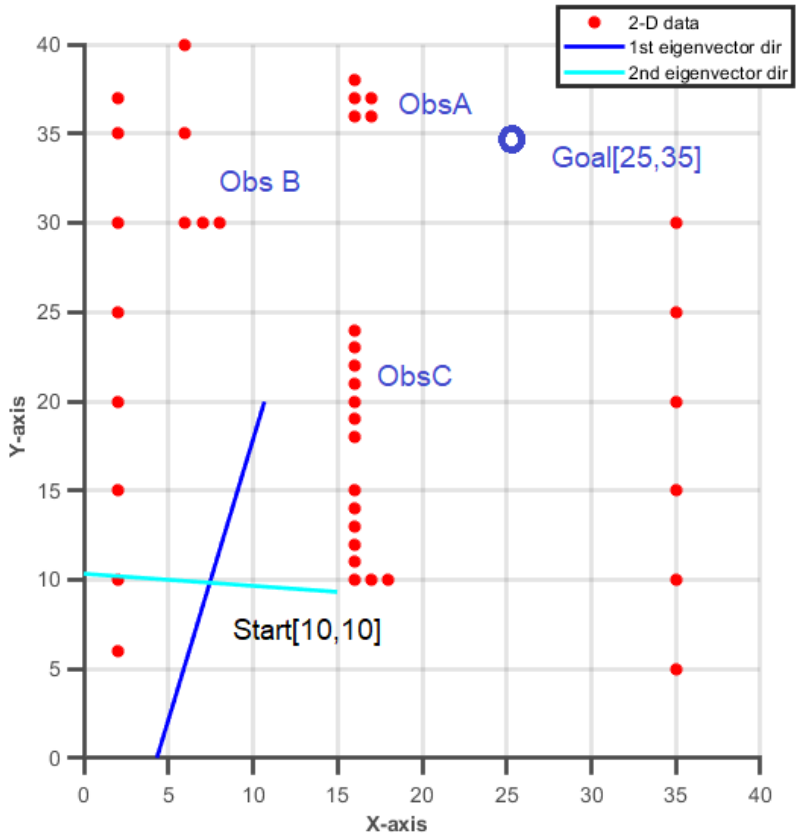


Figure 9: FCE cartesian map with eigenvectors

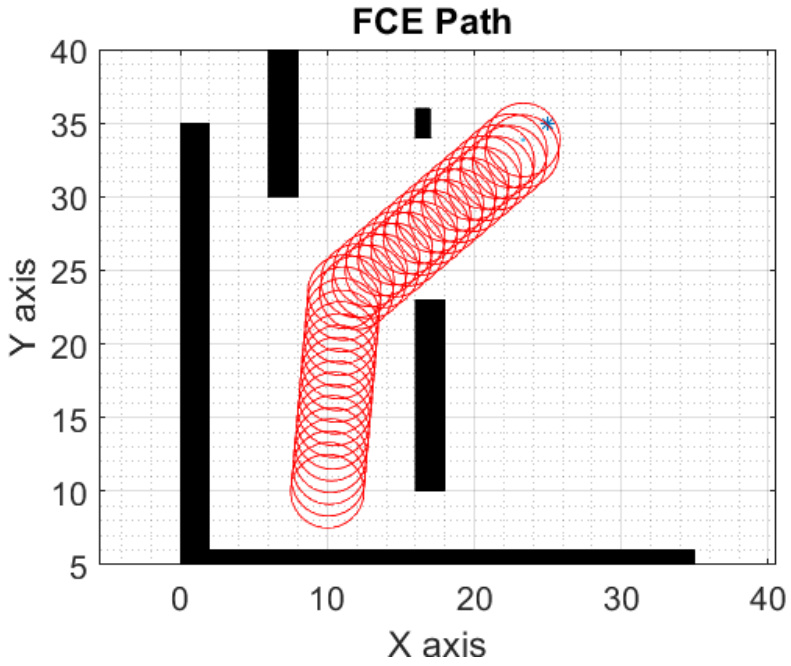


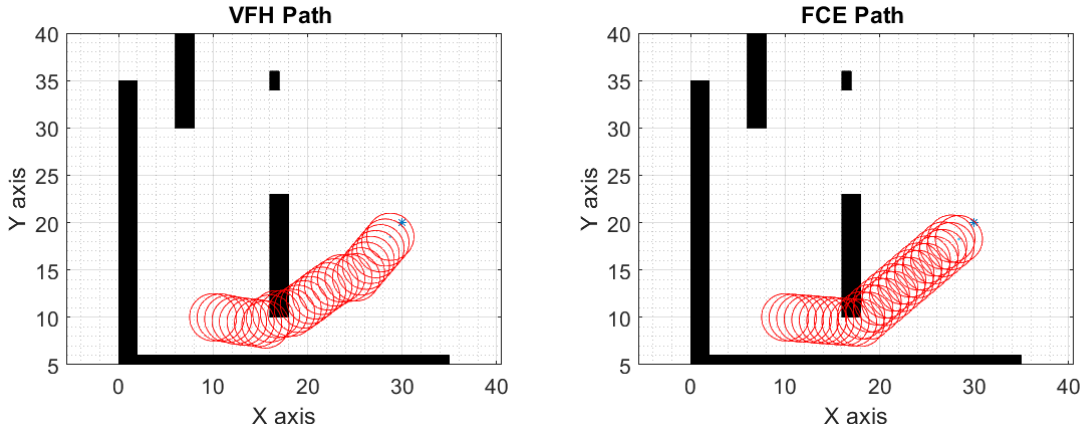
Figure 10: FCE result of scenario-I : Start[10,10], Goal[25,35]

Goal. Once the path is generated, the simulated robot trajectory can be plotted, assuming robot to be circular in shape.

The simulation scenario here is also the same as in the case of VFH described in the above section (Scenario-I) with **Start [10,10]** and **Goal [25,35]**. The FCE simulation result is shown in Figure 10 and the map scale is taken as 1 unit = 0.1m. Since the eigenspace generates only two PCs, the robot trajectory comprises of straight line segments as compared to the curved trajectory of VFH. This is shown in Figure 10 and the path length is recorded in Table 1.

5. Result analysis

Trajectory analysis of VFH and FCE technique was carried out by creating different obstacle configurations with different Start and Goal positions and the path lengths generated by each algorithm are computed. The analysis is done in two scenarios; Scenario-I, scenario-II and scenario-III having different Start and Goal locations. Then the VFH and FCE trajectory are plotted as shown in Figures 11, 12, 13 and 14 and the path lengths are recorded in Table 1. From the table it's can conclude that the VFH performance is better in terms of path length but have lots of abrupt change in directions. The FCE has straight line trajectory and shows some oscillations at some segments, as well.

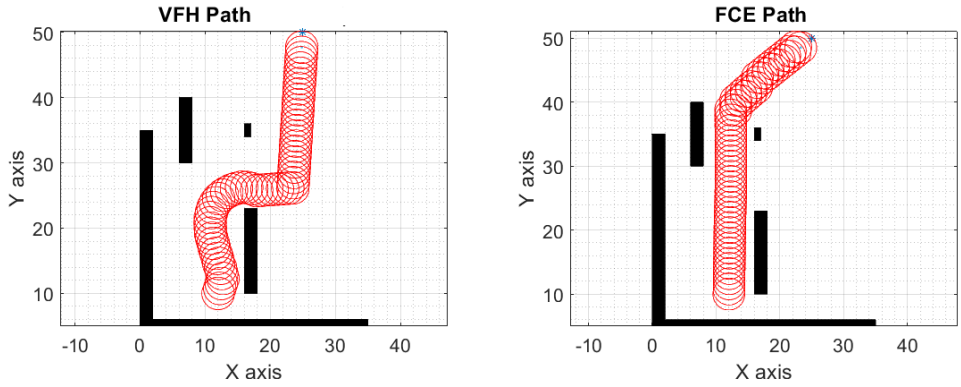


(a) Start[10,10],Goal[30,20] **(b) Start[10,10], Goal[30,20]**

Figure 11: Scenario -I

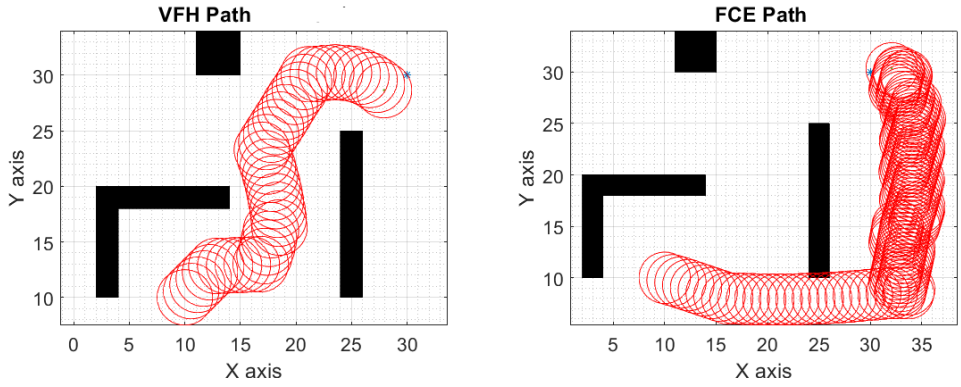
As shown in Figure 11 for Scenario-I with Start[10,10] and Goal[30,20], the path generated by both VFH and FCE are in the same direction but the VFH path length is shorter than FCE. Also, it is seen that the both trajectories are colliding with obstacle C.

As shown in Figure 12 for Scenario-I with Start[12,10] and Goal[25,50], the path generated by both VFH and FCE are in different coordinates but the VFH path length is shorter than FCE path length as seen in Table 1.



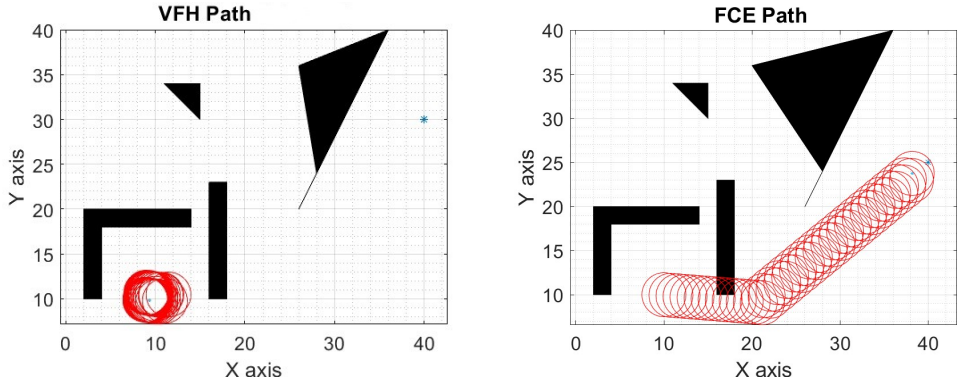
(a) Start[12,10],Goal[25,50] **(b) Start[12,10], Goal[25,50]**

Figure 12: Scenario -I



(a) Start[10,10],Goal[30,20] **(b) Start[10,10], Goal[30,20]**

Figure 13: Scenario -II



(a) Start[10,10],Goal[40,30] **(b) Start[10,10], Goal[40,30]**

Figure 14: Scenario -III

As shown in Figure 14 for Scenario-III which has more clutterd obastacles with Start[10,10] and Goal[40,30], VFH method robot is getting "trapped" in this Senario. When trapped, mobile robots exhibit "cyclic behavior", which is evident in Figure 14. But FCE method finds a path from start to goal as we can see in the Figure 14.

Table 1: 2D Simulation results

Technique	Scenario	Start	Goal	Path length(m)
VFH	Scenario -I	[10,10]	[25,35]	3.52000
FCE	Scenario -I	[10,10]	[25,35]	3.84000
VFH	Scenario -I	[10,10]	[30,20]	2.40000
FCE	Scenario -I	[10,10]	[30,20]	2.88000
VFH	Scenario -I	[12,10]	[25,50]	4.88000
FCE	Scenario -I	[12,10]	[25,50]	5.04000
VFH	Scenario -II	[10,10]	[30,30]	3.44000
FCE	Scenario -II	[10,10]	[30,30]	11.28000
VFH	Scenario -III	[10,10]	[30,30]	0.14000
FCE	Scenario -III	[10,10]	[30,30]	4.64000

6. Conclusion

Performance analysis with FCE and VFH for different scenrios shows that VFH gives the shortest path but have many changes in directions. But in the case of FCE, trajectory segments are almost straight lines but show some ocilllations in certain situations. The result analysis shows that the VFH perfomance is better in environments with unclutterd static obstacles. For exploring future scope, this research can be extended to path planning with dynamic obstacles.

References

Borenstein, J., Koren, Y., et al. (1991). The vector field histogram-fast obstacle avoidance for mobile robots. *IEEE Transactions on Robotics and Automation*, **7**, 278–288.

Chakravarthy, A. and Ghose, D. (1998). Obstacle avoidance in a dynamic environment: A collision cone approach. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, **28**, 562–574.

Fox, D., Burgard, W., and Thrun, S. (1997). The dynamic window approach to collision avoidance. *IEEE Robotics & Automation Magazine*, **4**, 23–33.

Khatib, O. (1986). Real-time obstacle avoidance for manipulators and mobile robots. *The International Journal of Robotics Research*, **5**, 90–98.

Lumelsky, V. J. and Stepanov, A. A. (1987). Path-planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape. *Algorithmica*, **2**, 403–430.

Ulrich, I. and Borenstein, J. (1998). Vfh+: Reliable obstacle avoidance for fast mobile robots. In *Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No. 98CH36146)*, volume 2, pages 1572–1577. IEEE.

Zaheer, S., Gulrez, T., and Thythodath Paramabath, I. A. (2022). From sensor-space to eigenspace–a novel real-time obstacle avoidance method for mobile robots. *IETE Journal of Research*, **68**, 1512–1524.