MRS Final Project Proposal - Grid Map based multi robot exploration using Utility function.

Harinee Kannan	Karthick Subramanian	Praveen Paidi
1224733263	1223408524	1225713099
hkannan2@asu.edu	ksubra25@asu.edu	ppaidi@asu.edu

1 Abstract

Implementing surveillance in critical areas is a significant difficulty in circumstances of security and might be very dangerous to include human beings in some situations. In pursuit of information about an unfamiliar or a volatile environment will pose potential risks. To address this concern, this project aims to mitigate these risks by employing multi-agent circular robots for surveillance.

Our pipeline involves obtaining the target locations which need surveillance using a utility function by trading off cost and profit using density mapping[5],then customized the A star algorithm with orientation cost to plan the shortest path for each of the robots among the locations available. We employed Linear Quadratic Regulator (LQR) to control the movement of the robots. We tackled multiple cases for collision avoidance with known, dynamic(unknown) obstacles and among the other robot agents using strategies such as negotiation by involving waiting time, Artificial potential fields using estimated attractive and repulsive potentials, shape navigation function using vector fields and tangential velocity.

Negotiation Algorithm majorly used to navigate the robots while avoiding collision trading off reaching time to save cost.[5] We employed artificial potential field to detect the dynamic obstacles appearing on the grid trajectory and re-plan the path. But the local minima constrain constituted by the concave obstacle posed an challenge. Hence the algorithm was localized to reach the points path planned by A* star while going around the obstacles[8].In addition to this, the artificial potential field [7] also used the classical step shaped path which will increase the cost of the our robot due to orientation cost. Hence, we had implemented the shape navigation function[2] to obtain an effective path to move the robots around the dynamic obstacles.

This work is to facilitate monitoring indoor industrial areas, cluttered environment, planetary activities where the need of optimal surveillance with limited robots is required.

Pipeline of Project Being:

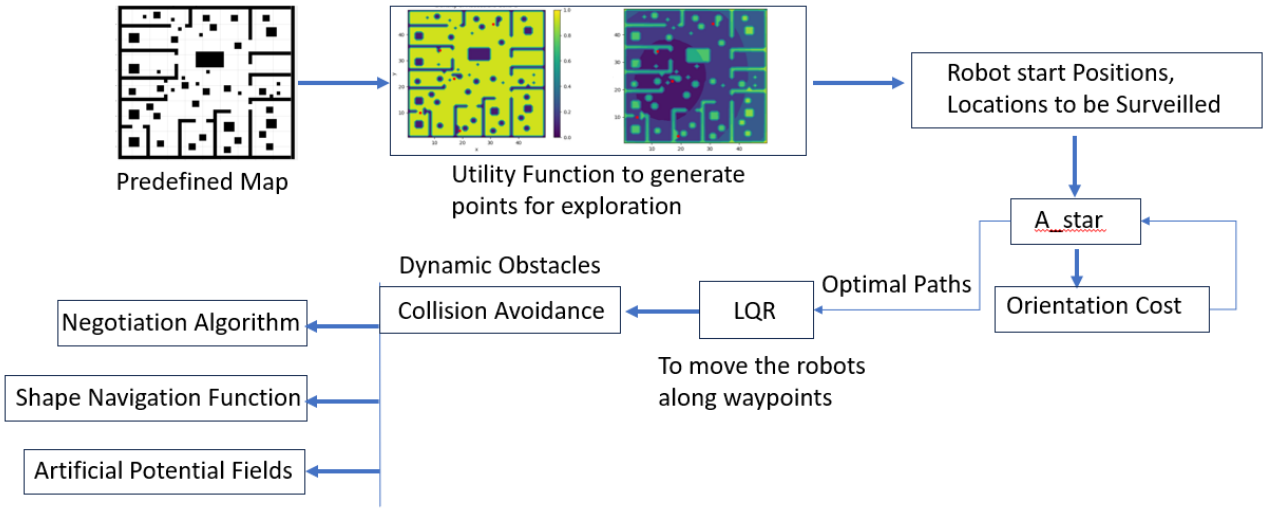

Figure 1: Flow Chart of Pipeline

2 Mathematical Model

2.1 Assumptions

We used a grid map which is of 50 x 50 size, Each grid cell is of (1 metre x 1 metre) making total map area of (50 metres x 50 metres).Robot specifications are as follows:
Robot size : Circular Robot of 0.3m radius.
Grid size: 1m x 1m.
Robot sensing area: 3 grids in front of robot.
Robot movement: Forward drive.
Cost to move in same orientation: 1
Cost to change to next orientation: 1
Environment : Bounded with static obstacles.
Sensing Capabilities: Semi-circular region, resolved to the three grid cells ahead of the robot.

This assumed robot has 2 wheels for the forward movement and one wheel at the front for orientation.The 90 degree orientation at the same position is achieved by front orientation wheel.This is supported by Rotary actuator and gear mechanisms for accurate 90 degree rotation.

Applications:

Our robot is mainly used in Indoor applications as security bots, warehouse monitoring as verifying inventory, ensuring safety compliance.As the robot size is almost of 75 percent of grid, only robot can one grid at a time and it shows as occupied or obstacle to other robot moving in that path.Even though obstacle size is smaller than grid size, we still consider that grid is occupied.We are assuming robot is given a predefined map.We assume the robots to

be equipped with the camera and LIDAR enabling the obstacle sensing and positional data sensing capabilities. Robot's location (x,y) is estimated using GPS localization and it has sensing capability of 3 full grids(similar to semi circle, but with grids, it is 3 full grids) front to it.Since our project does not extensively involve investigation utilizing localization, we have listed the assumptions considered while designing the problem.

We assume a scenario of warehouse inventory surveillance here for simulation with all the assumptions we made.In warehouse, paths are mostly straight as it has lanes and people working in robot collaboration and safety is primary concern maintaining buffer distance to robots which is taken as occupancy of grid.

2.2 Navigation function

For Collision avoidance, We used the navigation function ϕ to move around the obstacle and join the path again. As obstacles are considered as grids, they are square in shape. So equation of square is inequality and non differentiable. We implemented piece wise differentiation to get the objective done by the following algorithm.The equation of square is

$$\max(|x-a|, |y-b|) = \frac{s}{2}.$$

where s is the side length of the square.This equation indicates that the maximum absolute difference between x and a or between y and b should be half the side length of the square.The same equation with inequalities.

$$|\bar{x} - \bar{a}| \leq \frac{s}{2} \quad \text{and} \quad |y-b| \leq \frac{s}{2}$$

ϕ is the shape Navigation function equation. Shape equation of square obstacle is S and being treated each line equation as

$$S = s_1 + s_2 + s_3 + s_4$$

γ being $S(x, y)$ represented as

$$\gamma_1 = s_1(x, y), \gamma_2 = s_2(x, y), \gamma_3 = s_3(x, y)...$$
$$\beta_0 = R_0 - \|q\|^2.$$

Then We divided into $\phi = \phi_1 + \phi_2 + \phi_3 + ...$ each ϕ dealing with γ_1, γ_2 ,γ_3... with their respective shape equations(line equations as square is simply four line equations).

The shape navigation is given by

$$\phi(q) = \frac{\gamma^2}{\gamma^2 + \beta_0}$$

This is used to drive robot towards each line equation(s_x).To move robot in negative and positive directions of X and Y in grid map, we utilized ∂S.

$\psi = \begin{bmatrix} 0 \\ 0 \\ \gamma \end{bmatrix}$ and to make the tangential velocity we considered $\nabla \times \psi$ is a vector tangent to each curve's ϕ.This is with respect to individual robot and it is decentralized technique to control and move towards and around the obstacle.The final controller equation is as follows:

$$u_i = -\nabla_i \phi_i \cdot -\nabla_i \times \psi_i \cdot$$

2.3 Utility Function

Utility function utilizes grid map to find vacant cells and barriers using region-based technique [5] to explore the farthest cell with the least coverage. We map the robot's visibility zone based on its location and orientation on the grid and use the grid map to find the least frequented cells. The grid map is read as a binary image where 0 represents obstacles, robot position, and cells that are occupied and not available for navigation, and 1 represents cells that are unoccupied and available for navigation. This section uses 1 for unoccupied region to aid in correlation and distribution mathematical analysis. The robot viewable zones also use 0 to signify cells that are inaccessible for investigation or blocked by the environment. The obstacle density and robot coverage determine the navigation and surveillance cells. The occupancy matrix of the agents and obstacle is correlated with the Gaussian distribution to obtain the density function associated, which is then used to get the density matrix to analyze the region for surveillance information.

$$C(x, y) = \frac{S_a(x, y)}{S_b(x, y)} \tag{1}$$

Where S_a represents the density distribution of the agent and S_b represents the density distribution of the obstacles in correlation with the Gaussian distribution given as G with mean 0. The density distribution matrices are given by

$$S_a(x, y) = a \oplus G$$

and

$$S_b(x, y) = b \oplus G$$

a,b are the distributions of the agent and obstacle obtained by summing over individual robots. The cost of reaching each cell was calculated by estimating the robot's Euclidean distance from each cell based on its initial location and orientation. The current direction is additionally considered with the Euclidean distance to determine the minimal number of turns the robot needs to reach each grid or cell on the map, which improves robot cost estimates. The profit function is used to maximize the exploration .

$$P_{i,j} = I_{i,j} - C_{i,j} \tag{2}$$

for i = 1,2,. . . ,N and j = cell 1, 2, . . . , M . For each bot P_i corresponding cell with maximum profit is selected to select the target position of each robot from its current state.

2.4 Artificial Potential fields

Negative potential gradients are used in the artificial potential field approach to save energy. Our work adds the Artificial Potential Field[6] approach to the A* algorithm to improve its

navigation around dynamic barriers and collision prevention. The program uses an attractive potential to direct the bot to the objective and a repulsive potential to lower the cells' potential surrounding the barrier to prevent collisions. Our project uses a grid-based methodology, thus we assign potential to grid cells.

The attractive potential is given by the gain constant for estimation and by the distance estimated for our grid based approach,

$$U_{\text{attract}} = \frac{1}{2} k_a d_a$$

where d_a is the euclidean distance from the goal position and k_a attractive gain constant .The repulsive potential is given by

$$U_{\text{repulsive}}(q) = \frac{1}{2}(k_r)\left(\frac{1}{d_r} - \frac{1}{d_o}\right)^2$$

where d_r is the euclidean distance from current position to the obstacle, k_r is the repulsive gain constant and d_o is the distance required from the robot for manipulations.In addition d_r should always be greater that d_o to avoid any collision and to acquire better accuracy in the outcome else the repulsive potential is considered to be 0. These potentials are added to obtain the total potential at each point q is,

$$U(q) = U_{\text{attract}}(q) - U_{\text{repulsive}}(q)$$

Using the potential obtained we move along the negative gradient to reach the next step until the target position is reached,

$$\dot{x}(t) = -\nabla(U) \cdot x$$

While modelling the algorithm the robot motion is restricted along the x an y axis to accommodate the grid motion only and the bot movements take the Manhattan path.The grid based approach ensure maximum clearance for the robots to function since the grid size are considered to be larger than the base radius of the robot. The potential planning process is considered isotropic thus simplifying the path selection process such no restriction on the number of step taken,etc in the project. The algorithm itself is used to reinforce the trajectory planned by the A* algorithm to avoid the dynamic obstacle and connect back to the A* trajectory. Artificial potential field algorithm was used in the localized area to avoid the local minima issue arising do the concave obstacles. The algorithm works for obstacles limited to the size of one grid.

2.5 LQR based steering and speed control

In order to control the speed (v) and steering angle(δ) of the robot, we employ a Linear Quadratic Regulator(LQR) based controller[4].

The following parameters are taken as the robot states:

$$\mathbf{x} = \begin{bmatrix} x \\ y \\ \theta \\ v \end{bmatrix}$$

where:

x : Lateral position of the robot

y : Longitudinal position of the robot

v : Velocity of the robot

ϕ : yaw angle of the robot

The vehicle model is given by:

$\dot{x} = vcos(\theta)$

$\dot{y} = vsin(\theta)$

$\dot{v} = a$

$\dot{\phi} = vtan(\delta)/r$

where: δ is steering angle and r is the base radius of the robot

The Ordinary Differential Equation is:

$$\dot{z} = \frac{\partial}{\partial z} z = f(z,u) = A'z + B'u$$

where:

$$A' = \begin{bmatrix} \frac{\partial}{\partial x} vcos(\phi) & \frac{\partial}{\partial y} vcos(\phi) & \frac{\partial}{\partial v} vcos(\phi) & \frac{\partial}{\partial \phi} vcos(\phi) \\ \frac{\partial}{\partial x} vsin(\phi) & \frac{\partial}{\partial y} vsin(\phi) & \frac{\partial}{\partial v} vsin(\phi) & \frac{\partial}{\partial \phi} vsin(\phi) \\ \frac{\partial}{\partial x} a & \frac{\partial}{\partial y} a & \frac{\partial}{\partial v} a & \frac{\partial}{\partial \phi} a \\ \frac{\partial}{\partial x} \frac{vtan(\delta)}{L} & \frac{\partial}{\partial y} \frac{vtan(\delta)}{L} & \frac{\partial}{\partial v} \frac{vtan(\delta)}{L} & \frac{\partial}{\partial \phi} \frac{vtan(\delta)}{L} \end{bmatrix} = \begin{bmatrix} 0 & 0 & cos(\bar{\phi}) & -\bar{v}sin(\bar{\phi}) \\ 0 & 0 & sin(\bar{\phi}) & \bar{v}cos(\bar{\phi}) \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{tan(\bar{\delta})}{L} & 0 \end{bmatrix}$$

$$B' = \begin{bmatrix} \frac{\partial}{\partial a} vcos(\phi) & \frac{\partial}{\partial \delta} vcos(\phi) \\ \frac{\partial}{\partial a} vsin(\phi) & \frac{\partial}{\partial \delta} vsin(\phi) \\ \frac{\partial}{\partial a} a & \frac{\partial}{\partial \delta} a \\ \frac{\partial}{\partial a} \frac{vtan(\delta)}{L} & \frac{\partial}{\partial \delta} \frac{vtan(\delta)}{L} \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & \frac{\bar{v}}{Lcos^2(\bar{\delta})} \end{bmatrix}$$

These matrices represents the dynamics of the system.

In order to control the speed and the steering of the robot using LQR controller we model the system using error feedback and their corresponding derivatives for the states.

The state-space representation of the system dynamics using error is given by the following set of differential equations:

$$\dot{x} = Ax + Bu$$

where:

$$x = \begin{bmatrix} e \\ \dot{e} \\ \theta_e \\ \dot{\theta}_e \\ \delta_v \end{bmatrix} \qquad u = \begin{bmatrix} \delta \\ \text{accel} \end{bmatrix},$$

State and Input Matrices

The matrices A and B are given by:

$$A = \begin{bmatrix} 1 & dt & 0 & 0 & 0 \\ 0 & 0 & v & 0 & 0 \\ 0 & 0 & 1 & dt & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \qquad B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ \frac{v}{L} & 0 \\ 0 & dt \end{bmatrix}.$$

Control Law

The control law applied to the system is expressed as:

$$u^* = -Kx$$

where K is the gain matrix obtained through the Linear Quadratic Regulator (LQR) design process.

Control Calculation

The control inputs are calculated as follows:

$$\delta = \text{atan2}(Lk, 1) + \text{pi\_range}(u_1^*)$$

$$\text{accel} = u_2^*$$

Here, atan2 is the arctangent function, and pi_range ensures that the feedback steering angle is within the range of $-\pi$ to π.

2.6 Path Planning

Cost to reach next node in same orientation is defined as 1 and one orientation costs 1 for robot.
$G = (V, E)$ represents the graph, where V is the set of nodes (Traversal grids), E is the set of edges connecting the nodes.g(n) represents the cost function, which denotes the actual cost of reaching node n from the starting node.

$$g(n) = \text{cost from start node to node n}$$

h(n) represents the heuristic function, estimating the cost from node n to the goal node..

$$h(n) = \sqrt{(x_{\text{goal}} - x_n)^2 + (y_{\text{goal}} - y_n)^2}$$

f(n) represents the total cost function, which is used to determine the priority of nodes. It combines the actual cost g(n) and the heuristic estimate h(n) for each node:

$$f(n) = g(n) + h(n)$$

The A* algorithm selects nodes for evaluation based on this total cost function, choosing nodes with the lowest f(n) values.The algorithm terminates when the goal node is reached.Now, in our case orientation cost is also present, so I added r(n) which represents cost to turn at the same point.

$$r(n) = \text{Cost to change the orientation}$$

So we added it and customized the final cost equation by reducing the weight of h(n) to negligible as it drives to turn towards goal by increasing the orientation cost.So our final cost is modelled as follows.

$$f(n) = g(n) + r(n) + c.(h(n))$$

where c can be modelled according to our requirements, in this case, we reduced the value to decrease the weight of heuristic.

3 Theoretic analysis

3.1 Lyapunov Analysis for System Stability

The state-space representation of the system dynamics using the error is given by the following set of differential equations:

$$\dot{x} = Ax + Bu$$

where:

$$x = \begin{bmatrix} e \\ \dot{e} \\ \theta_e \\ \dot{\theta}_e \\ \delta_v \end{bmatrix}, \quad u = \begin{bmatrix} \delta \\ \text{accel} \end{bmatrix}$$

The matrices A and B are given by:

$$A = \begin{bmatrix} 1 & dt & 0 & 0 & 0 \\ 0 & 0 & v & 0 & 0 \\ 0 & 0 & 1 & dt & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ \frac{v}{L} & 0 \\ 0 & dt \end{bmatrix}$$

The control law applied to the system is expressed as:

$$u^* = -Kx$$

where K is the gain matrix obtained through the Linear Quadratic Regulator (LQR) design process.

The control inputs are calculated as follows:

$$\delta = \text{atan2}(Lk, 1) + \text{pi\_range}(u_1^*)$$

$$\text{accel} = u_2^*$$

The Lyapunov equation is given by:

$$A^T P + PA + Q - BK^T R^{-1} K B^T = 0$$

where:

P is a positive definite matrix,
Q is typically an identity matrix,
R is typically an identity matrix.

Given numerical values for dt, v, L, and K, the numerical solution for P is obtained by solving the Lyapunov equation.

$$P = \begin{bmatrix} 2.10177320 & 2.08085948 & 2.08100848 & 1.02502708 & -1.45476454 \\ 2.08085948 & 2.06565796 & 2.07080698 & 1.02002683 & -1.48869128 \\ 2.08100848 & 2.07080698 & 4.10115800 & 2.03012585 & -1.54079111 \\ 1.02502708 & 1.02002683 & 2.03012585 & 5.10049634 & -7.26891406 \\ -1.45476454 & -1.48869128 & -1.54079111 & -7.26891406 & 2.00005000 \end{bmatrix}$$

Since all the eigenvalues of P are positive, the system is positive definite, and therefore, the system is stable.

3.2 Artificial Potential Field

For the potential function to successfully achieve the target position while selecting a path, 1. The potential function should be 0 at the target/goal position.
2. There should be no cell in the grid that has infinite potential except for the cell with no readability, i.e., U(q) = ∞.
3. The local operator choosing the next state(given by q_0) where the potential function is less than the current state(given by q), $U(q_0) < U(q)$.
The local operator is designed to select the action that tends to minimize the potential at every step, $q^* = \arg\min_{q \in U(x)} U(x, y)$. The local operator is based on the attractive potential designed using the euclidean distance, we can visualize the attractive field as a parabola with its global minima at the desired position. Thus, the fundamental euclidean distance is used to generate the attraction field and enable the the global minima to be achieved by following the negative gradient of the potential,

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

while the repulsive function is treated as a charged line and the Coulomb's law given as

$$F = k \frac{|q_1| \cdot |q_2|}{r^2}$$

is used to account for repulsive potential to be estimated. q, charge in the equation is considered unity in this scenario and the k is equated to equated to the repulsion gain which is tuned as per requirements. In our project, we have set the k = $\frac{1}{2}k_r$. The distance r in the equation is compared to the distance of the obstacle from each bot which also accounts for the robot radius to avoid collision and to choose paths to fit the bot efficiently. This modelling analogy hold good for isolated/stand-alone obstacle's. But fails with concave obstacle as the create multiple local minima points and generation high repulsion field causing the obstacle to oscillate within the obstacle's perimeter with no escape. Hence we employed the shape navigation function to attempt to navigate around piece-wise continuous curve-square when it senses an obstacle in the nearby grid.

3.3 Shape Navigation Function

Stability of a proposed controller using $V(q)$ and its time derivative V'.

$$V(q) = \sum_i \phi(q_i)$$

The time derivative of $V(q)$ with respect to time (V') is calculated using the chain rule:

$$V' = \sum_i \nabla \phi(q_i)^T \dot{q}_i$$

Here, $V(q)$ is defined as the sum of $\phi(q_i)$ over i components. V' represents the rate of change of $V(q)$ concerning time. $\nabla\phi(q_i)^T$ denotes the transpose of the gradient vector of $\phi(q_i)$, and $\dot{q}_i$ represents the rate of change of q_i with time.Given the orthogonality between $\nabla\phi$ and $\nabla \times \psi$ as they are defined to drive towards and drive tangential,This orthogonal relationship implies that the gradient of ϕ is perpendicular to the curl of ψ,

$$\nabla\phi(q_i) \cdot (\nabla \times \psi(q_i)) = 0$$

As $\dot{q}_i$ is nothing but the dynamics equation of controller, which is

$$u_i = -\nabla_i\phi_i \cdot -\nabla_i \times \psi_i\cdot$$

replacing the $\dot{q}_i$ in V' time derivative,

$$V' = \sum_i \nabla_i\phi(q_i)^T \left(-\nabla_i\phi(q_i) - (\nabla_i \times \psi)\right)$$

The orthogonality relationship allows exclusion of the contribution of the cross product of $\nabla \times \psi$ from the derivative, resulting in the simplified expression for V'.

$$V' = \sum_i -k\|\nabla_i\phi(q_i)\|^2 \leq 0$$

The negative semi-definiteness of V' ($V' \leq 0$) is crucial for stability assessment.The negative sign in V' implies a fundamental property for systems exhibiting stability. This behavior suggests that the system tends to converge towards a stable equilibrium point or a minimum of $V(q)$, indicating stability.

General line equations such as $y + k_a$ and $x + k_b$ where k_a and k_b are constant values, these are only two line equations used in algorithm as our robot is moving in grids and and can only move in forward direction towards N, S, E, W directions only.

For the equation $y + k_a$

$$\frac{\partial S}{\partial x} = -\frac{2x(y+k_a)^2}{(-r + x^2 - 2k_a y - k_a^2)^2}, \qquad \frac{\partial S}{\partial y} = -\frac{2(y+k_a)(-r + x^2 - k_a.y)}{(-r + x^2 - 2k_a y - k_a^2)^2}.$$

For the equation $x + k_b$

$$\frac{\partial S}{\partial x} = -\frac{2(x+k_b)(r - y^2 + k_b.x)}{(r - y^2 + 2k_b x + k_b^2)^2}, \qquad \frac{\partial S}{\partial x} = -\frac{2y(x+k_b)^2}{(r - y^2 + 2k_b y + k_b^2)^2}.$$

For instance, robot is moving Y direction, obstacle information being perceived before 1 grid. s_1 equation for γ_1 being $Y + k_a$ and respective ϕ_1 shape navigation generated to drive towards and ψ_1 to move along the shape. When the obstacle is detected, it drives robot till the clearance point that is till that grid in which obstacle is present and then tangential velocity to move in line shape.The respective vector field is generated and displayed.

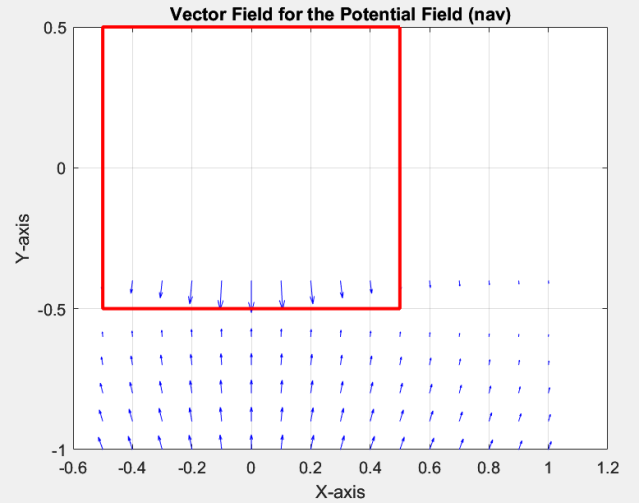

Figure 2: Vector field towards boundary

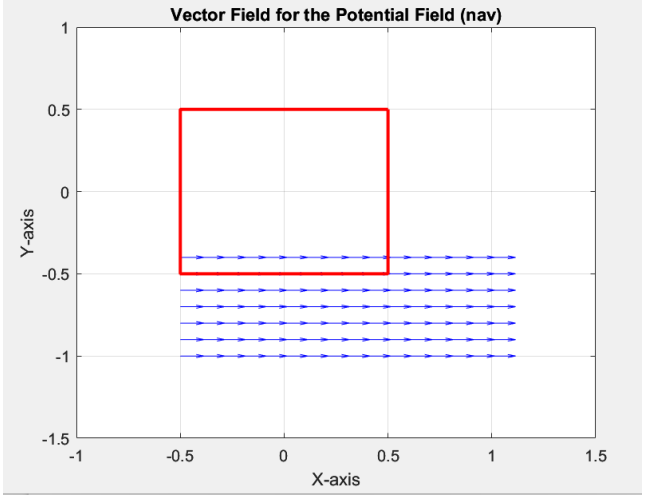

Figure 3: Vector field along the boundary

Then we provided a while loop to stop at next grid centre and again surrounding information being perceived,given no obstacles for this case,s_2 equation for γ_2 being $X + k_b$ and respective ϕ_2 shape navigation generated to drive towards and ψ_2 , it moves in y direction one more grid and again checks the same condition.This loop continues and till it reaches any of the nearest original path coordinates. The vector field figures are being displayed.

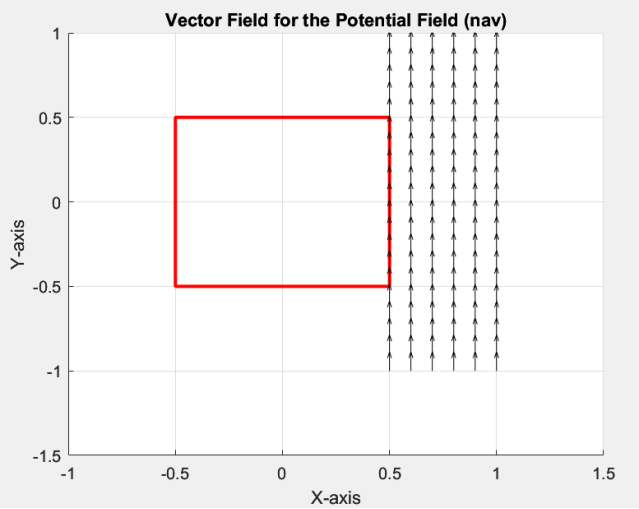

Figure 4: Vector field along the boundary

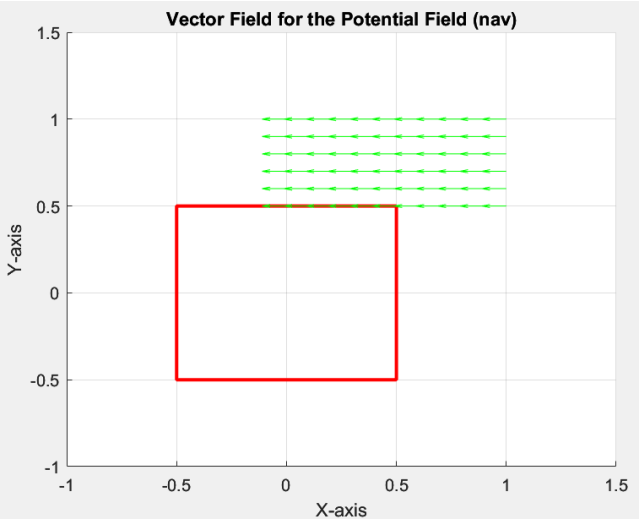

Figure 5: Vector field along the boundary

This is the shape followed by the robot when grid obstacle arrives, it shows it comes near the boundary buffer and then it moves along boundary maintaining distance.

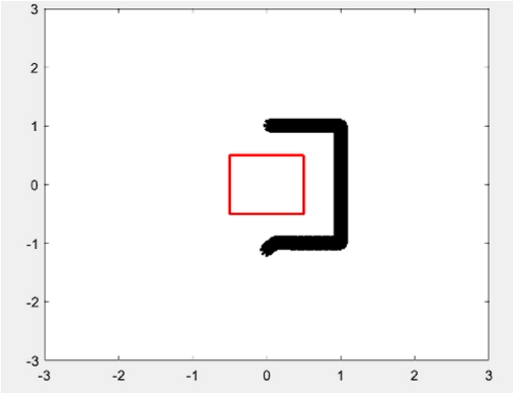

Figure 6: Shape function

3.4 Path Planning

Here is the top level view of Algorithm[1], to prove optimality of Algorithm.
Input: Graph G, start node *start*, goal node *goal*
Output: Shortest path from *start* to *goal*

1. Initialize an open set to store nodes to be evaluated, initially containing the start node.
2. Create dictionaries to store:
   - $g(n)$: Actual cost from start to each node (initialize all nodes with ∞, except start node with 0).
   - $f(n)$: Estimated total cost from start to goal through each node (initialize all nodes with ∞, except start node with heuristic estimate).
3. While the open set is not empty:
   - Extract node *current* with the lowest $f(n)$ from the open set.
   - If *current* is the goal node, reconstruct and return the path.
   - For each neighbor of *current*:
     - Calculate tentative $g(n)$ and $r(n)$ for the neighbor.
     - If tentative $g(n) + r(n)$ is lower than the current $g(n) + r(n)$, update f(n) using r(n), h(n) with calculated g(n).
     - Add neighbor to the open set if not already present.
4. If the open set becomes empty, no path exists.

Lets suppose, $c(N) = g(N) + r(N)$. Now, assume there is another goal node $n^{'}$ such that its cost $c(n^{'})$ is smaller than $c(n)$ which is given by A^*.

$$c(n^{'}) \leq c(n)$$

Case 1:
$n^{'}$ was on open list when n was picked up for expansion.As A^{*} picks the sorted final cost, it has to be

$$f(n) \leq f(n^{'})$$

As the n is the goal node, removing heuristic, cost to reach has to lesser as well which says,

$$c(n) \leq c(n^{'})$$

This is contradiction to our assumption.
Case 2:
$n^{'}$ was not on open list when n was picked up for expansion but neighbour(parent node) $n^{"}$ was present then

$$f(n) \leq f(n^{"})$$

As the n is the goal node, but $n^{"}$ is not a goal node, so heuristic is also added.

$$c(n) \leq c(n^{"}) + h(n^{"})$$

But $c(n^{"}) + h(n^{"})$ must be under estimating the cost of $c(n^{'})$ because of this is Manhattan distance and prior is euclidean distance.This can also be proved using admissibility of algorithm. which inturn says,

$$c(n) \leq c(n^{"}) + h(n^{"}) \leq c(n^{'})$$

It again contradicts our assumption.Hence A^{*} is optimal with involving orientation cost.

4 Simulation Results

We used python to generate a 50m x 50m grid map with each cell spanning 1m x 1m, with static obstacles to replicate a building/warehouse layout. The following is the final map obtained, In Figure, 8 the Purple spots indicates the obstacles and the the additional purple block in front of each bot is region being surveyed by the robot and hence represents the section of the grid that need no surveillance along side the obstacles.

4.0.1 Mapping and utility function

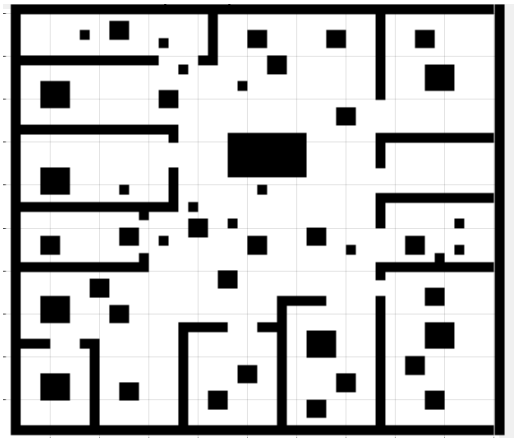

Figure 7: Grid Map

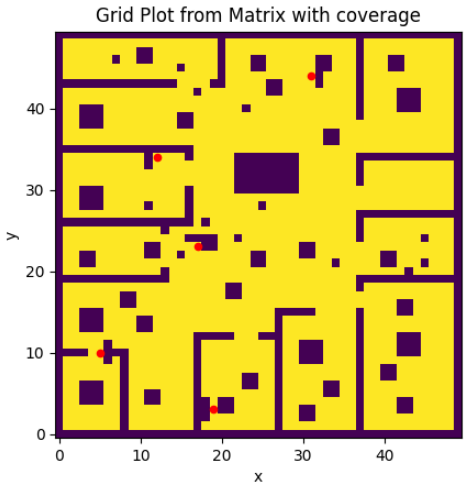

Figure 8: Robot Coverage Map

The next step was to random initialization of the robots and then estimating the hot spot to be survey the the unexplored parts using the robot position and the grid map with obstacles while the grids are maintained throughout the process. The heat-maps are generates on the grid for visualization of the hot spots. In Figure, 9, the yellow grid represents the passage available for movement without any obstructions which is used for estimation of new target position for each bot to achieve maximum exploration based. In Figure, 34, represent the section of the grid that are assigned to the bots as targets with "*" marker and the dark blue region represents the target coverage location. The target positions estimated using the utility function tries to navigate to the position which is the farthest from the current position ans the most unexplored.

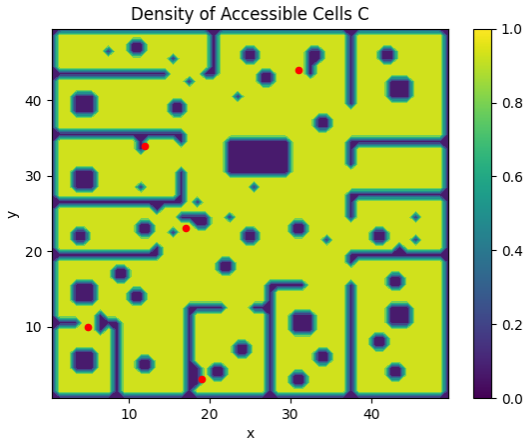

Figure 9: Grid Density

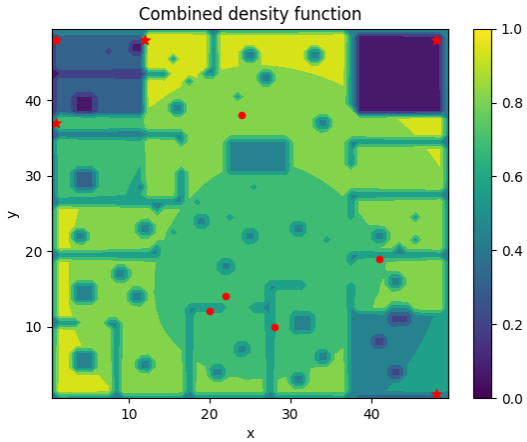

Figure 10: Coverage density

The target position select by the utility function is set as goal positions for the A* algorithm. During the initial iteration process, the bots were initialized at random position and then the chosen targets were used by A* algorithm for path finding.

4.0.2 A star and optimization of path

A star algorithm is employed to get the optimal path,As A star doesn't consider orientation cost[3], we tuned penalizing the heuristic cost by involving orientation costs.Below figures shows all the optimization stages.

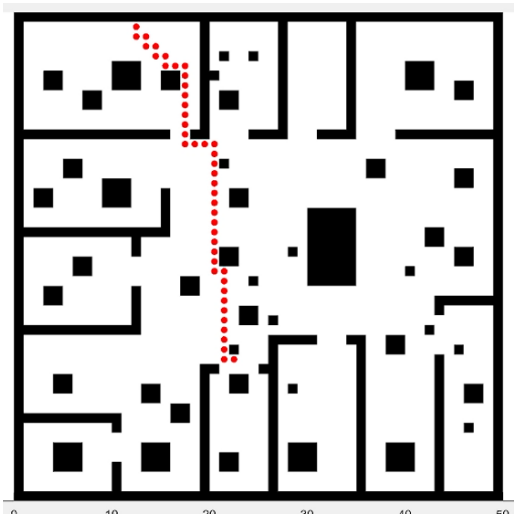

Figure 11: A star

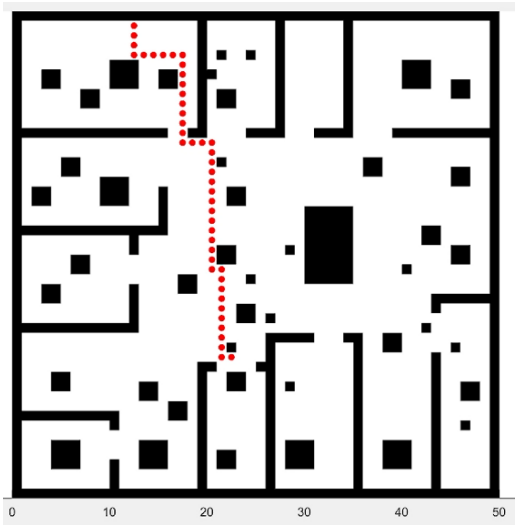

Figure 12: Optimizing path using all weighted costs

This is the optimized path utilizing robot cost for path, orientation and the penalized heuristic cost to reach the goal state.

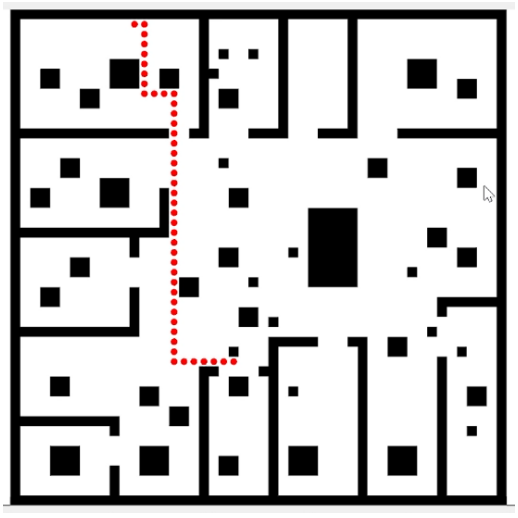

Figure 13: Optimized path by reducing the weight to heuristic cost

4.0.3 Artificial Potential Field

The robot can follow the trajectory generated by the A-star algorithm and traverse the path but based on our application the warehouse environment can have other autonomous or human counter-parts. These extreme disturbances might interfere with the trajectory determined by A-star algorithm, this is then sensed by the LIDAR setup and the artificial potential field is setup to read this interrupt already set trajectory and re-route around the dynamic obstacle that appeared and re-connected to the trajectory on the other side of the obstacle.

This is an obstacle avoidance maneuver the start position is set at (35,10) and the target is at(35,32) while the obstacle is set in its path. From the image it can be seen that the Artificial potential field algorithm decides on a path around the obstacle and reaches the potential.

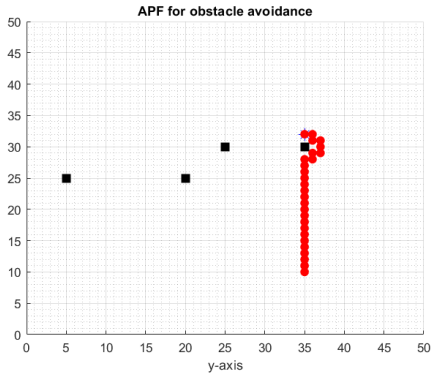

Figure 14: Artificial Potential field - Obstacle avoidance

During the course of the project, we attempted to use the artificial potential artificial for the navigation across the layout of the warehouse from the point of obstacle detection. During this process, we were encountered continuous oscillations in path and the bot trajectory did not reach the desired position under certain concave obstacle cases where the bot was enclosed on three side due to the building layout in addition to classical staircase navigation issue we encountered when the robot was able to reach the target position as exhibited in the figure 15.

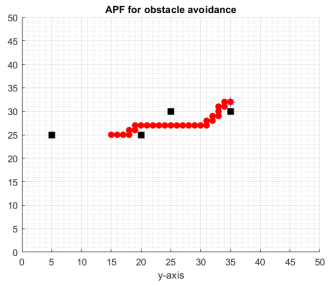

Figure 15: APF Path planning

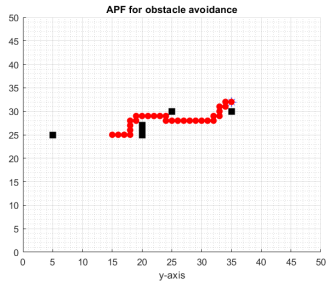

Figure 16: APF Path planning

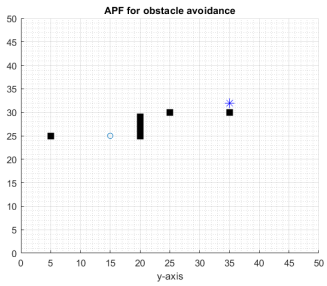

Figure 17: Oscillation Course

In the figure 15 and 16, the bot was able to navigate through the course with the obstacle but when the obstacle required the robot to a higher potential grid cell before it traverse to the minima, the artificial potential navigation was unable to achieve it. Based on these shortcomings, shape navigation has been adapted by out team to traverse the square grids considering each side as piece-wise linear.

4.0.4 Shape Navigation Cases

The following is case of dealing with bunch of dynamic obstacles using shape function. Robot is following the boundary of the obstacle and going to next point on predetermined path after the obstacle.

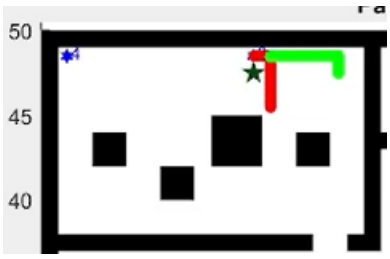

Figure 18: Robo and obstacle

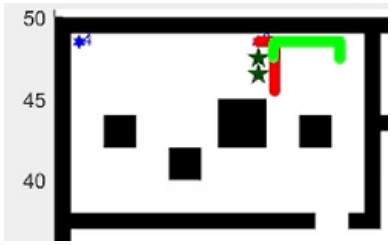

Figure 19: Moving on boundary of 1st obstacle

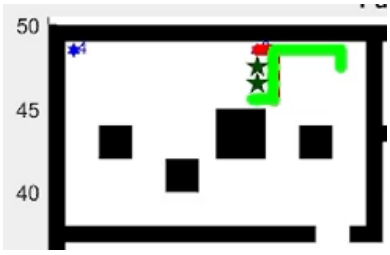

Figure 20: Moving along boundary of obstacles

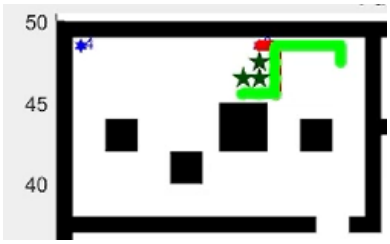

Figure 21: Moving along boundary of obstacles

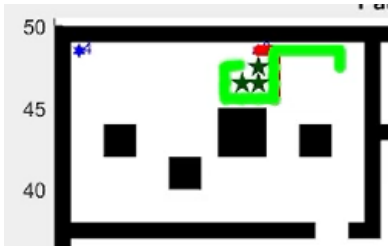

Figure 22: Moving along boundary of obstacles

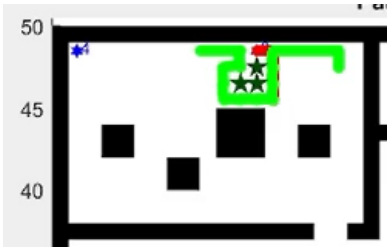

Figure 23: Reached the next point after the obstacle and following original path

Dealing with other agent on the path using shape navigation.

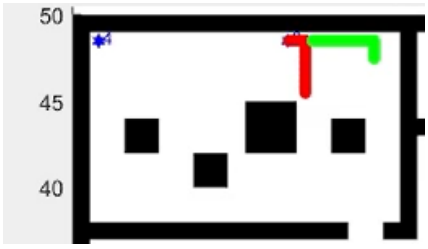

Figure 24: Detected other bot

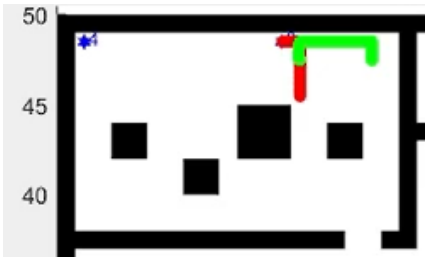

Figure 25: Follwing shape

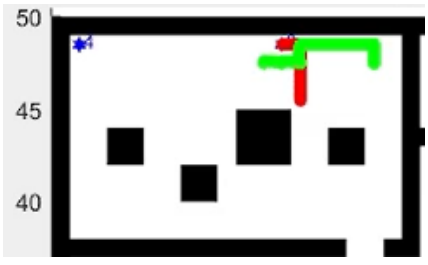

Figure 26: Follwing shape

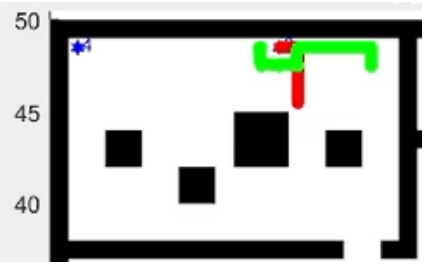

Figure 27: reaching next point on path

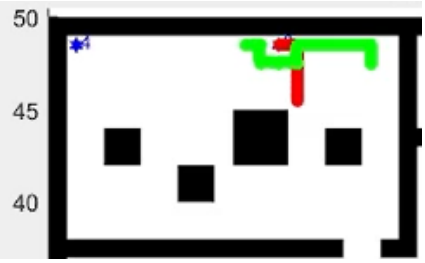

Figure 28: Continuing the path

4.0.5 Negotiation

The following is the case where robot is waiting for the other robot to pass.

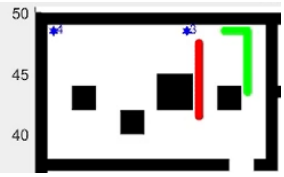

Figure 29: Robots approaching

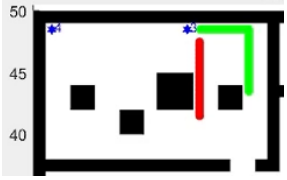

Figure 30: Robot waiting other to pass

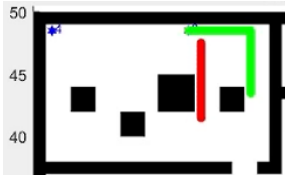

Figure 31: Robot waiting other to pass

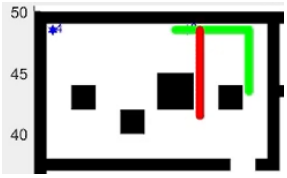

Figure 32: Robot moving

4.0.6 Final paths

The below are some of the final paths after applying different scenarios.

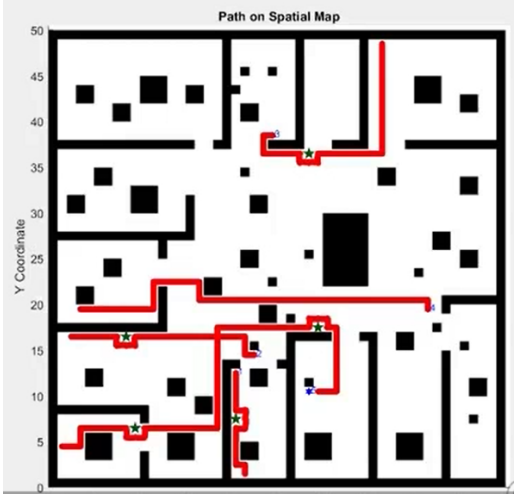

Figure 33: Different dynamic obstacles

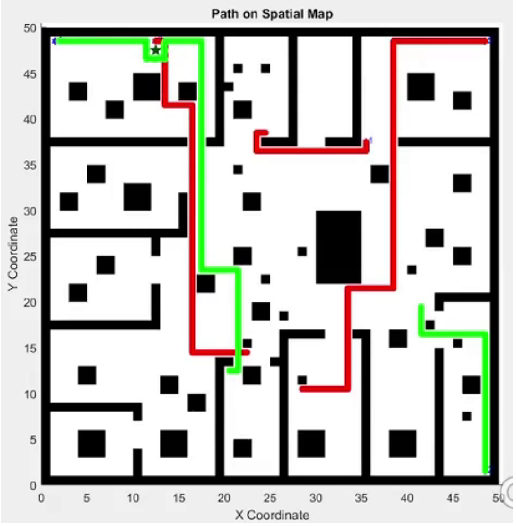

Figure 34: Dealing with other bots in path

5 Code and simulation videos

Please use link below to access simualtion files, or url mentioned in submission comment.

Link : Simualtion videos and code

6 Future Scope

Considering our project experience, we would like to investigate the possibility of incorporating a navigation function associated with the artificial potential field. This would involve integrating a random walk planner and analyzing various methods to reduce computation in order to overcome local minima. Additionally, we aim to expand the navigation functions to address the limitations of shape following navigation in both the sphere world and star world. We have implemented shape navigation for square grids by assuming piece-wise continuity. We are certain that we can further develop this capability and extend the analysis. In addition to working on the individual algorithms we would also like to work on combining the other path planning algorithm balance out the pros and cons each logic flow.

Division of Labour

Everyone Contributed equally for the each part of project and here is detailed view.

- Mathematical Modelling:
  - Shape navigation function model - Praveen Paidi
  - Utility function model - Harinee Kannan
  - Atificial potential fields model - Harinee Kannan
  - LQR controller model - Karthick Subramanian
  - A* shortest path search model - Praveen Paidi , Karthick Subramanian
- Analysis:
  - Lyapunov Stability analysis for LQR controller - Harinee Kannan, Karthick Subramanian
  - Artificial potential fields local implementation analysis: Praveen Paidi, Harinee Kannan
  - Shape navigation function analysis - Praveen Paidi, Karthick Subramanian
  - A* Path Planning Analysis - Praveen Paidi, Karthick Subramanian
- Simulation:
  - Grid mapping and utility function - Praveen Paidi
  - A* optimization of path - Karthick Subramanian
  - Artificial potential fields - Harinee Kannan
  - Shape navigation negotiation - Praveen Paidi, Harinee Kannan

References

[1] Bochun Cao et al. "Research on the star algorithm for safe path planning". In: *2023 IEEE International Conference on Control, Electronics and Computer Technology (ICCECT)*. 2023, pp. 105–109. DOI: 10.1109/ICCECT57938.2023.10141167.

[2] Mong-ying A. Hsieh, Savvas Loizou, and Vijay Kumar. "Stabilization of Multiple Robots on Stable Orbits via Local Sensing". In: *Proceedings 2007 IEEE International Conference on Robotics and Automation*. 2007, pp. 2312–2317. DOI: 10.1109/ROBOT.2007.363664.

[3] Boyoon Jung and Gaurav S. Sukhatme. "Cooperative Multi-robot Target Tracking". In: *Distributed Autonomous Robotic Systems 7*. Ed. by Maria Gini and Richard Voyles. Tokyo: Springer Japan, 2006, pp. 81–90. ISBN: 978-4-431-35881-7.

[4] Jesse Levinson et al. "Towards fully autonomous driving: Systems and algorithms". In: *2011 IEEE Intelligent Vehicles Symposium (IV)*. 2011, pp. 163–168. DOI: 10.1109/IVS.2011.5940562.

[5] Christian G. Quintero M., José Oñate López, and Francisco A. Bertel R. "Intelligent exploration and surveillance algorithms for multi-agents robotics systems". In: *2012 XXXVIII Conferencia Latinoamericana En Informatica (CLEI)*. 2012, pp. 1–10. DOI: 10.1109/CLEI.2012.6427178.

[6] E. Rimon and D.E. Koditschek. "Exact robot navigation using artificial potential functions". In: *IEEE Transactions on Robotics and Automation* 8.5 (1992), pp. 501–518. DOI: 10.1109/70.163777.

[7] Tuna Tandogan. "Collision-free indoor navigation using an artificial potential field controller". PhD thesis. Your University, 2015.

[8] Bo Wang. "Path Planning of Mobile Robot Based on A* Algorithm". In: *2021 IEEE International Conference on Electronic Technology, Communication and Information (ICETCI)*. 2021, pp. 524–528. DOI: 10.1109/ICETCI53161.2021.9563354.