Multi-Agent Motion Planning and Coordination in Polygonal Environments using Vector Fields and Model Predictive Control

Rashmi Hegde and Dimitra Panagou

Abstract—In this paper, we extend earlier work on motion planning and coordination of multiple agents, to environments of arbitrary polygonal obstacles, using non-gradient vector fields to steer each agent towards their goal configurations while avoiding collisions. We formulate the vector fields so that the pattern of their integral curves depends on a parameter $\lambda$. By manipulating the value of $\lambda$, we obtain a set of vector fields whose integral curves define the flow lines for an a priori known obstacle environment. We use the vector field design in tandem with model predictive control to compute safe trajectories for multi-agent systems. The competence of the proposed methodology is demonstrated for both static and dynamic environment via simulation results. The efficacy of model predictive control in achieving control trajectories, free of chattering, for multi-agent coordination is validated through comparison to a state feedback coordination and control protocol.

I. INTRODUCTION

Motion planning and multi-robot coordination is a challenging problem in controls and robotics research. The major approaches encompass graph search algorithms, Lyapunov-based control methods, combinatorial planning methods, sampling-based planning and formal methods [1]–[3]. Dijkstra’s algorithm [4] is one of the graph search algorithm, which begins by evaluating each possible state with a path cost and iteratively moves on to the state with lowest cost. The computation time of this deterministic algorithm is not guaranteed, since it searches the entire state space. To overcome this drawback, $A^*$ algorithm [5] takes into consideration both the path cost and a heuristic cost, usually Euclidean distance, for the iterative search of a feasible path. However, there is still a possibility of a local minimum substantially increasing the computation time of the search algorithm. A concise discussion on the heuristic-based algorithms can be found in [6].

A widely used methodology to achieve feedback in motion planning with obstacles is to use Lyapunov-based control methods. Artificial Potential Field (APF) [7], [8] is a popularly used scalar function formulation to address such problems. However, the concern of being trapped in a local minimum reappears in the potential field method. There are various methodologies based on APF that attempt to tackle this, with different degrees of success: navigation functions [9]–[11], harmonic functions [12], [13] and stream functions [14] to name a few. A number of potential field based local navigation approaches, like the Virtual Force Field (VFF) [15] and the Vector Field Histogram (VHF) [16], [17], build a APF online for obstacle avoidance, but do not guarantee convergence.

Sampling-based algorithms such as the rapidly exploring random trees (RRT) [18]–[20] are widely favored for their quick exploration of state space and low probability to be caught in local minima. The disadvantage of RRTs is that its randomness may produce a path with unpredictable or even undesirable behavior. The RRT* [21] tackles the issue of optimality at the cost of increased computation, especially for nonholonomic constraints. Another commonly used algorithm is Probabilistic Road Map (PRM) [22], which is implemented in two stages. The first stage involves a learning phase, to sample the configuration space; followed by query phase, to specify the initial and goal configuration and connect to the roadmap. However, this method suffers from representational incompleteness since it is only probabilistically complete. More recent developments in sampling-based methodologies are reviewed extensively in [23]. A neoteric trend in motion planning is to use formal methods to model controllers for robots to comply with high-level specifications. These symbolic motion planning methods include model checking [24], supervisory control and reactive synthesis [25]. A descriptive survey of the formal methods can be found in [26].

In this paper, we extend the work done in [27] to use a set of non-gradient vector fields to guide a robot in an obstacle environment, to reach its goal location. An obstacle environment consisting of both convex and non-convex obstacles is assumed to be known a priori. In the presence of static obstacles, using vector fields for navigation results in a simple control law for the robot. In the multi-agent scenario, the obstacle avoidance problem is dynamic in nature. Hence, each agent has to avoid collision with both the static obstacles and other dynamic agents, and reach its goal configuration. The feedback solution from the vector field formulation is then used as a basis in model predictive control design for the coordination of multiple nonholonomic agents. Various methodologies of utilizing MPC in robot motion planning problems have been discussed in [28]–[31]. In the present work, we formulate a receding finite horizon problem using recentered barrier functions in the cost functional [32]–[34].

The organization of the paper is as follows: Section II gives the problem formulation by formalizing the robot and obstacle models, and Section III includes the construction of family of vector fields required for effective robot navigation.
Section IV presents the feedback motion plan for obstacle avoidance for a single agent, along with the control design and simulation results. Section V demonstrates the extension of the method for distributed coordination of multiple agents, the underlying control design and also the results from simulation. We summarize the proposed methodology and present our conclusions in Section VI.

II. PROBLEM FORMULATION

A. Robot Modeling: Unicycle Model

We consider a robot modeled as a closed circular disk of radius \( \rho \) whose motion is governed by unicycle kinematics:

\[
\begin{bmatrix}
\dot{x} \\
\dot{y} \\
\dot{\theta}
\end{bmatrix} = \begin{bmatrix}
\cos \theta & 0 & u \\
\sin \theta & 0 & 1 \\
0 & 0 & 0
\end{bmatrix}
\begin{bmatrix}
u \\
\omega
\end{bmatrix} \tag{1}
\]

where \( q = [r^T \theta]^T \) is the configuration vector of the robot with respect to (w.r.t.) a global coordinate frame \( G \), \( r = [x \ y]^T \) is the position vector, \( \theta \) is the orientation of the robot w.r.t. frame \( G \), \( u \) is its linear velocity and \( \omega \) is the angular velocity about \( z_G \) axis.

The robot moves in an environment with \( N \) polygonal obstacles, and needs to reach a goal location \( r_g = [x_g \ y_g]^T \) in the free space.

B. Obstacle Modeling

We consider an obstacle environment consisting of both convex and non-convex polygonal obstacles. We assume that the location of corners, or vertices, of obstacles are known. The boundary of the obstacle is inflated to account for both a safe distance from the obstacle edge \( \rho_s \), and the robot dimension \( \rho \) as: \( \rho_m = \rho + \rho_s \), where \( \rho_m \) is the minimum distance allowed between the robot and obstacle boundary. The corners of the obstacles are considered as circular obstacles of radius \( \rho_{oi} \). The vector field for the obstacle edge is defined between \( \rho_{ei} \) and \( \rho_{zi} \), with: \( \rho_{zi} = \rho + \rho_s + \rho_{oi} \). Non-convex obstacles are assumed to be composed of combinations of convex obstacles [35] and thus, the rules stated previously are applied in this case as well. The inflated model of an obstacle is illustrated in Figure 1(a). The dashed lines indicate the new obstacle boundary.

III. FAMILY OF VECTOR FIELDS FOR NAVIGATION

We consider a class of vector fields \( F : \mathbb{R}^2 \rightarrow \mathbb{R}^2 \) given by the analytical formula:

\[
F(r) = \lambda (p^T r)r - p(r^T r), \text{ with } p \neq 0 \tag{2}
\]

where \( \lambda, p \) are parameters that dictate the behavior of resulting flow lines around the origin. For more information the reader is referred to [27]. The vector field components read: \( F_x = (\lambda - 1)p_x x^2 + \lambda p_y xy - p_y y^2 \), and \( F_y = (\lambda - 1)p_y y^2 + \lambda p_x xy - p_x x^2 \). Our first task is to suitably define vector fields in free space, so that the robot moves along the vector field until it reaches its final destination. The final vector field formulation—taking into account all the cases mentioned below—is automated for simulation purposes, as shown in Figure 1(b).

A. Attractive Vector Field to the Goal

We define the attractive vector field to the goal to be a radial vector field: \( F_g = \frac{\begin{bmatrix} x_g - x \\ y_g - y \end{bmatrix}}{\sqrt{(x_g - x)^2 + (y_g - y)^2}} \), where \( r_g = [x_g \ y_g]^T \) is the goal position, and \( r = [x \ y]^T \) is the robot’s position. The normalized attractive vector field reads: \( F_n^{oi} = \frac{F_i}{\sqrt{F_{xi}^2 + F_{yi}^2}} \), and \( F_n^{g} = \frac{F_g}{\sqrt{F_{xg}^2 + F_{yg}^2}} \).

B. Repulsive Vector Field for the Obstacle Edges

The vector field (2) can be used to define the flows around edges for \( \lambda < 0 \), with an appropriate selection of vector \( p \).

Let the position vectors \( v_i = [v_{xi} \ v_{yi}]^T \), \( i \in \{1, \ldots, n\} \), of the \( n \) vertices of a polygon be known, and \( m_i = [m_{xi} \ m_{yi}]^T \) denote the position vector of the midpoint of the \( i \)-th edge defined by vertices \( i \) and \( i + 1 \). Then, the components of a vector field \( F_{ei} \) around the \( i \)-th edge, whose singular point is set at the midpoint \( m_i \), is given for \( \lambda = -1 \) by: \( F_{x{ei}} = -2 p_{sei}(x - m_{xi})^2 - p_{y{ei}}(y - m_{yi})^2 \), \( F_{y{ei}} = -2 p_{sei}(y - m_{yi})^2 - p_{x{ei}}(x - m_{xi})^2 \), and \( F_{sei} = [p_{sei} \ p_{y{ei}}]^T \), \( p_{sei} = \cos \varphi_ei \), \( p_{y{ei}} = \sin \varphi_ei \), and \( \varphi_ei = \arctan(-v_{yi}, -v_{xi}) \).

The vector field components for the \( i \)-th edge are normalized to get \( F_n^{sei} \) and \( F_n^{gie} \).

1) Orientation for Convex Obstacles: The direction of the vector fields along the obstacle edges \( \varphi_ei \) is such that the robot is driven towards its goal location. That is, the direction is either \( \varphi_ei \) or \( \varphi_ei + \pi \) depending on position of the goal configuration relative to the obstacle edge. This can be determined by using the geometrical properties of the obstacle environment. One of the methods is to use the difference between the angle of the obstacle edge w.r.t. the horizontal, and the angle of the line joining the midpoint of the obstacle edge and the goal, \( \psi_{di} \).

Let \( v_{ki} \) and \( v_{(k+1)i} \) be the vertices of the \( i \)-th edge. The angle of the obstacle edge w.r.t. the horizontal is: \( \psi_i = \arctan(v_y(k+1)i - v_{yk}i, v_x(k+1)i - v_{xk}i) \). Angle of the line joining the midpoint of the obstacle edge and the goal is given by: \( \psi_{gi} = \arctan(y_g - m_{yi}, x_g - m_{xi}) \). Hence, we have: \( \psi_{di} = \psi_{gi} - \psi_i \).

2) Orientation for Non-Convex Obstacles: For non-convex obstacle edges, there is a possibility of occurrence of a singular point. For instance, the vector fields of two adjacent obstacle edges might lead the robot to a common vertex, see Figure 1(b).

To avoid this situation, the orientation of the edge nearest to the goal, \( \phi_i \), is reversed. If \( d_{i1} \) and \( d_{i2} \) are the distances...
between the midpoint of the two edges, and the goal respectively, then: \( \phi_{ci} = \begin{cases} \phi_{c1i} - \pi, & d_{c1i} < d_{c2i} \\ \phi_{c2i} - \pi, & d_{c1i} > d_{c2i} \end{cases} \) with \( \phi_{c1i} \) and \( \phi_{c2i} \) calculated as done in the convex case.

### C. Vector Field for the Obstacle Corners/Vertices

1) Case-1: When there is no change in the direction of the edge field–

For \( \lambda = 1 \), the vector field (2) has circular flow lines around the origin. This field can be utilized to navigate the robot tangentially around obstacle corners. The vector field components are given by: \( F_{eci} = p_{eci}(x - v_{ei}) - p_{eci}(y - v_{ey}) \), and \( F_{eciy} = p_{eci}(x - v_{ey}) - p_{eci}(y - v_{ey}) \), with \( v_{ei} = [v_{xi} v_{yi}]^T \) are the coordinates of the vertices of the obstacle and \( p_{eci} = [p_{eci} p_{eci}]^T \). The vector field orientation is given by: \( \phi_{c1i} = \text{atan2}(v_{yi} - yg, v_{xi} - xg) + \pi \), and \( \phi_{c2i} = \cos \varphi_{ci}, \; \text{p}_{eci} = \sin \varphi_{ci} \). The vector field orientation is given by: \( \phi_{c1i} = \text{atan2}(v_{yi} - yg, v_{xi} - xg) + \pi \), and \( \phi_{c2i} = \cos \varphi_{ci}, \; \text{p}_{eci} = \sin \varphi_{ci} \).

2) Case-2: When two edge vector fields are directed towards the corner and goal is located within the purview of the sector of the obstacle corner–

In this case, we define a vector field out of (2) for \( \lambda = 0 \) and the vector field components read: \( F_{eci} = -p_{eci}(x - v_{xi})^2 - p_{eci}(y - v_{yi})^2 \), and \( F_{eciy} = -p_{eci}(x - v_{xi})^2 - p_{eci}(y - v_{yi})^2 \), with \( v_{xi} = \text{atan2}(v_{yi} - yg, v_{xi} - xg) + \pi \), and \( p_{eci} = [p_{eci} p_{eci}]^T \). The vector field orientation is given by: \( \phi_{c1i} = \text{atan2}(v_{yi} - yg, v_{xi} - xg) + \pi \), and \( \phi_{c2i} = \cos \varphi_{ci}, \; \text{p}_{eci} = \sin \varphi_{ci} \).

3) Case-3: When both edge vector fields are directed towards the corner and goal is located within the purview of sector of obstacle corner–

For \( \lambda > 1 \), the pattern of the integral curves around the singular point is dipolar. This field can be used to guide the robot towards the goal location. For \( \lambda = 2 \), the vector field components are given by: \( F_{eci} = p_{eci}(x - v_{xi})^2 + 2p_{eci}(x - v_{xi})(y - v_{yi}) - p_{eci}(y - v_{yi})^2 \), and \( F_{eciy} = p_{eci}(y - v_{yi})^2 + 2p_{eci}(x - v_{xi})(y - v_{yi}) - p_{eci}(x - v_{xi})^2 \), with \( p_{eci} = [p_{eci} p_{eci}]^T \), \( p_{eci} = \cos \varphi_{ci}, \; \text{p}_{eci} = \sin \varphi_{ci} \). If \( \varphi_{ci} = \text{atan2}(v_{yi} - yg, v_{xi} - xg) \), and \( \psi_{jc} = \text{atan2}(yg - c_{yi}, xg - c_{xj}) \), the angle of the line joining the goal and the geometric centre \( c_i = [c_{xi} c_{yi}]^T \) of the \( i \)-th obstacle, then the direction of the vector field is given by: \( \phi_{ci} = \begin{cases} \phi_{c1i} + \frac{\pi}{2}, & \phi_{c1i} \leq \psi_{jc} \\ \phi_{c2i} + \frac{\pi}{2}, & \phi_{c2i} > \psi_{jc} \end{cases} \). The vector fields for the obstacle corners are then normalized to get \( F^1_{eci} \) and \( F^1_{eciy} \). Figure 2 illustrates all four cases of vector field formulation for obstacle corners.

### D. Blending Function

1) Blending Attractive and Repulsive Fields: Let \( \gamma_i \) denote the perpendicular distance of the robot from the \( i \)-th edge of the obstacle and \( \rho_{zi} = \rho + \rho_s + \rho_{ci} \). To ensure a smooth transition between the two vector fields, we define a bump function \( \sigma_{ci}(\cdot) \):

\[
\sigma_{ci} = \begin{cases} 0, & \rho \leq \gamma_i \leq \rho_{zi} \\ 1, & \gamma_i \geq \rho_{zi} \\end{cases}
\]

where, \( a = \frac{-2}{(\rho_{zi} - \rho_{ci})^2} \), \( b = \frac{3(\rho_{zi} + \rho_{ci})}{(\rho_{zi} - \rho_{ci})^2} \), and \( c = \frac{-6\rho_{zi} \rho_{ci}}{(\rho_{zi} - \rho_{ci})^3} \).

2) Blending Attractive and Corner Fields: Let \( \beta_i \) denote the distance between the robot and the \( i \)-th obstacle corner, \( \beta_i = \sqrt{(x - xci)^2 + (y - yci)^2} \). To have a smooth transition between the vector fields, we define the blending function \( \sigma_{ci}(\cdot) \) as:

\[
\sigma_{ci} = \begin{cases} 0, & \rho_{ci} \leq \beta_i \leq \rho_{zi} \\ 1, & \beta_i \geq \rho_{zi} \end{cases}
\]

where, \( a, b, c, d \) are equal to that in (3).

### IV. Motion Plan for Static Obstacles

Assume a workspace \( W \) of \( N \) polygonal obstacles. Let \( Q_v = \{v_i| i \in \{1, \ldots, N\}\} \) be the set of vertices of all the \( N \) obstacles and \( U \) the total number of vertices or corners. The obstacle is positioned such that the distance between any two obstacle corners, \( d_{ij} = \|v_i - v_j\| \) satisfy: \( d_{ij} \geq \rho_{zi} + \rho_{s} + \rho_{ci}, \; i, j \in \{1, \ldots, U\}, \; j \neq i \). Then, the vector field is:

\[
F^* = \prod_{i=1}^{N} \sigma_{ci} \sigma_{ci} F_{ci} + \sum_{i=1}^{N} (1 - \sigma_{ci}) F_{ci} + (1 - \sigma_{ci}) F_{ci} \]

where, \( F_{ci}, F_{cix}, F_{cixy} \) are the normalized attractive vector field towards the goal location, and edge and corner vector fields around the polygonal obstacles, respectively.

The construction of the vector fields is such that it always directs the agent towards its goal configuration, except for a known set of initial conditions that lead to undesired singular points [27], due to vector field formulation with \( \lambda = 1 \).

#### A. Control Design

The control design for linear and angular velocity of the robot are given by the simple state feedback law:

\[
u = k_u \text{tanh}(\|r - r_g\|), \quad \omega = -k_\omega (\theta - \varphi) + \varphi
\]

where, \( \varphi = \arctan(F^*_{x}) \) is the orientation of the vector field \( F^* \) at a point \((x, y)\), with \( \varphi \) standing for the time derivative of \( \varphi \), and \( k_u > 0, k_\omega > 0 \).
dynamic agents in the region \( F \) the neighbor agent must be defined such that it does not
The vector field is then normalized to obtain \( \bar{F}_i \) given by:

\[
\bar{F}_i = \frac{\partial b_{ij}}{\partial x_i} \frac{x_i}{\|x_i\|} + \frac{\partial b_{ij}}{\partial y_i} \frac{y_i}{\|y_i\|} = \bar{x}_i \bar{y}_i
\]

where \( \bar{x}_i = \frac{x_i}{\|x_i\|} \) and \( \bar{y}_i = \frac{y_i}{\|y_i\|} \). Hence, the gradient vector field \( \bar{F}_i \) of function \( b_{ij} \). Then, we have the barrier function:

\[
B_{ij} = \left( \sum_{k=1}^{N} (1 - \sigma_{ck}) \right) F_{ck} + \sum_{j \in N_i} (1 - \sigma_{ij}) \bar{F}_{ij}
\]

V. MULTI-ROBOT SYSTEMS

A. Vector Field for Navigation

Consider a multi-robot system with \( M \) identical agents of unicycle kinematics. These agents are assigned the task of navigating through the polygonal environment to reach their respective goal locations \( q_{gi}, i \in \{1, \ldots, M\} \). Simultaneously, the agents have to avoid collisions with the static polygonal obstacles and the other dynamic agents. Each agent has a sensing region \( S_i \) of radius \( R_s \) centered at \( r_i = [x_i, y_i]^T \), denoted as: \( S_i : \{r \in \mathbb{R}^2 | \|r_i - r\| \leq R_s\} \). Hence, the vector field \( \bar{F}_i \) of function \( b_{ij} \). Then, we have the barrier function:

\[
B_{ij} = \left( \sum_{k=1}^{N} (1 - \sigma_{ck}) \right) F_{ck} + \sum_{j \in N_i} (1 - \sigma_{ij}) \bar{F}_{ij}
\]

B. Barrier Functions

1) Recentered Barrier Function for collision avoidance:

Agent \( i \) avoids collision with a neighbouring agent \( j \) if:

\[
c_{ij} = (x_i - x_j)^2 + (y_i - y_j)^2 - d^2_{ij} \geq 0
\]

We define a logarithmic barrier function as:

\[
b_{ij} = -\ln(c_{ij})
\]

Now, the gradient recentered barrier function for \( c_{ij} \) is given as:

\[
r_{ij} = b_{ij}(r_i - r_j) - b_{ij}(r_{id} - r_j) + \delta r_i, \ \text{where,} \ \delta r_i = \bar{r}_i - r_{id}, \ \bar{r}_i = \left[ x_{id}, y_{id} \right]^T \text{is the goal location of agent } i \text{, and } \nabla b_{ij} = \begin{bmatrix} \partial b_{ij} \partial x_i \\ \partial b_{ij} \partial y_i \end{bmatrix}
\]

2) Barrier function to limit velocity: To limit the linear velocity to \([-\omega_{max}, \omega_{max}] \), we define:

\[
B_v(\nu) = \frac{\|\nu\| - \omega_{max}}{\omega_{max}} + \frac{\|\nu\| + \omega_{max}}{\omega_{max}}
\]

These barrier functions ensure collision-free trajectories for the dynamic agents in a polygonal obstacle environment.

C. Control Design

When no agents are within the communication radius of agent \( i \), i.e., when \( d_{ij} > R_c, \forall j \in \{1, \ldots, M\}, j \neq i \), then the linear and angular velocity for the \( i \)-th agent are given as:

\[
u_i = u_{ai} \tanh(r_i - r_{id}), \quad u_{ai} > 0
\]

The bump function \( \sigma_{ij}(.) \) is defined as:

\[
\sigma_{ij} = \begin{cases} 0, & d_m \leq d_{ij} < d_c \\ 1, & d_c \leq d_{ij} < d_c \end{cases}
\]

where, \( a = \frac{2}{d_{ij} - d_c}, b = \frac{3(d_c - d_m)}{d_{ij} - d_c}, c = -\frac{6(d_m - d_c)}{d_{ij} - d_c}, d = \frac{d^2_{ij} - d^2_{ij}}{d_{ij} - d_c} \). Hence, the vector field for agent \( i \) in a polygonal obstacle environment with \( k \) vertices or corners is:

\[
\bar{F}^g_i = \prod_{j \in N_i} \sigma_{ck}\sigma_{ch}\sigma_{ck} F_{gij} + \sum_{k=1}^{N} (1 - \sigma_{ck}) F_{ck} + (1 - \sigma_{ck}) \bar{F}_{ij}
\]

\[
\bar{F}_i = \frac{\partial b_{ij}}{\partial x_i} \frac{x_i}{\|x_i\|} + \frac{\partial b_{ij}}{\partial y_i} \frac{y_i}{\|y_i\|}
\]

where \( \bar{F}_i = \frac{x_i}{\|x_i\|} \) and \( \bar{y}_i = \frac{y_i}{\|y_i\|} \). Hence, the gradient vector field \( \bar{F}_i \) of function \( b_{ij} \). Then, we have the barrier function:

\[
B_{ij} = \left( \sum_{k=1}^{N} (1 - \sigma_{ck}) \right) F_{ck} + \sum_{j \in N_i} (1 - \sigma_{ij}) \bar{F}_{ij}
\]
Here, \( L(q_i, \nu_i) \) is a positive definite function denoting running cost and \( M(q_i) \) is a positive definite function denoting the cost-to-go:

\[
L(q_i, \nu_i) = \frac{1}{2} z_i^T Q z_i + \nu_i^T R \nu_i + \mu_{agg} B_{agg}(r_i, r_{id}) + \mu_v B_v(\nu_i, \omega_i) + \mu_{poly} B_{poly}(r_i, r_{id}), \text{ and } M(q_i) = \frac{1}{2} z_i^T P z_i,
\]

where \( z_i = [x_i - x_{id}, y_i - y_{id}, \theta_i - \phi_i(x_i, y_i)]^T \), \( \phi_i(x_i, y_i) \) is the orientation of the vector field \( F_i^0 \) at a point \((x_i, y_i)\), \( Q, P \) and \( R \) are positive definite weighting matrices, and \( \mu_{agg}, \mu_v, \mu_{poly} > 0 \). The terminal cost \( M(q_i) \) is chosen to be a quadratic function of the position and orientation error of the robot w.r.t. the vector field \( F_i^0 \).

We solve the optimization problem (13) by using the infeasible Newton start method [36], described in Algorithm 1. In every run of MPC, while computing the trajectory, each agent takes into account the current position of other dynamic agents, at the beginning of the prediction horizon.

The terminal region \( \Omega_i \), containing the goal state \( q_{id,i} \), defined to be a neighborhood of the desired configuration, is:

\[
\Omega_i = \{ q_i | \| r_i - r_{id} \| \leq r_o, \| \theta_i - \phi_i(x_i, y_i) \| \leq \varepsilon_1 \}. \]

Once the agent is in the terminal region, the state feedback controller (12) becomes active, to establish the desired convergence properties [33].

### D. Simulation Results

We validated the vector field construction, and the control algorithm through multiple simulations of \( N = 10 \) dynamic agents in an obstacle environment – with different weighting matrices, prediction horizons, and initial and goal locations. One of the simulations is elucidated in this paper. To implement the MPC algorithm, we discretize the agent model with a sampling time, \( dT = 0.01s \). We use a prediction horizon of \( T_p = 25s \), and a control horizon of \( T_c = 5s \). The agents start from their respective initial positions, and navigate around the polygonal obstacles to reach their specified goal locations, denoted by square markers in Figure 4. The path followed by each of the agents is illustrated in Figure 5. Here, the ‘\( i \)’ marker indicates the initial positions of the agents, and the goal location is represented by the circular unicycle agents. The minimum inter-agent distance is always greater than the required value, see Figure 6. The linear velocities of the agents with the proposed MPC is compared with the state feedback control presented in [27] in Figure 7. We observe that the linear velocity is free of any chattering effect, even when the agents encounter dynamic and static obstacles within their communication radius. Also, the orientation error introduced due to this interaction is smaller with the barrier function based MPC, compared to the state feedback control law. This is illustrated in Figure 8 for one of the agents.

---

### Algorithm 1 Motion Planning for Dynamic Agents

**Input:** \( q_i \)

**Output:** \( u_i, \omega_i \)

**Initilaisation:** \( O, O_i, q_{id} \)

1. Find \( N_i \)

2. while \( q_i \neq q_{id} \) do

3. Calculate \( F_{gi} \)

4. for \( j \in N_i \) do

5. Calculate \( F_{ij}, \sigma_{ij} \)

6. end for

7. for \( i = 1 \) to \( v_N \) do

8. Calculate \( F_{ei}, \sigma_{ei} \)

9. Calculate \( F_{ci}, \sigma_{ci} \)

10. end for

11. Find \( F_i^0 \)

12. if \( d_{ij} > R_c \) then

13. Find \( u_i \) and \( \omega_i \) using (12)

14. else

15. Find the barrier functions (9), (10), (11)

16. Formulate the MPC problem

17. Solve for \( u_i \) and \( \omega_i \) using [36]

18. end if

19. end while

---

**VI. Conclusion**

In this paper, we have proposed a methodology to use non-gradient vector field for navigation in both static and dynamic obstacle environment. The current implementation considers the agents as identically modelled unicycle robots. In case of a single agent in an a priori known surrounding, a simple control law enables effective navigation of the robot. However, in case of a complex dynamic environment, when an agent has both static obstacles, and other dynamic agents within its communication radius, the control law needs to be more involved. We show that recentered barrier function based Model Predictive Control is competent to solve such
polygonal obstacle environment

by introducing curvature constraints on the reference fields.

of the method to double integrators and Dubins-like models

Fig. 7: Comparison of linear velocities $u_l$ of the dynamic agents in polygonal obstacle environment

Fig. 8: Comparison of Orientation error w.r.t reference vector field for Agent-10, while navigating in polygonal in obstacle environment

class of problems. Ongoing work focuses on the extension of the method to double integrators and Dubins-like models by introducing curvature constraints on the reference fields.

REFERENCES


