Collision Avoidance and Formation Control

Mike Reynolds

***Abstract*—Formation control and collision avoidance are huge topics of research in multi-agent control. With many different techniques for both forms of control, this project was meant to find a solution to the problem of a formation moving along a known path through a static obstacle field while avoiding the obstacles and the other agents. This paper employed motion planning to drive each agent to a safe location while minimizing deviation from the target formation shape.**

I. Introduction

FORMATION control has been a topic of interest to the multi-agent control community for years. Although the control of these systems has been largely deduced, collision avoidance applications remain a popular question. Collision avoidance in single robot systems has been explored extensively, giving rise to motion planning methods that avoid predetermined obstacles or developing an additional safety controller to a system that keeps the system from operating unsafely. However, work in collision avoidance for multi-robot systems is still new and active. This project was meant to explore a couple of techniques that could be utilized for multiple agents.

II. Methods

A. Problem Formulation

For my formulation of this problem, I elected to simplify the dynamics to single-integrator dynamics. Having experience with formatting zeroing control barrier functions with these dynamics before, I thought this would be approachable for this problem.

Assumption 1.

$$\dot{x} = u \tag{1}$$

Where x is the state of any robot and u is the control input. This makes it quite simple to write the equations in terms of coordinates and implement those coordinates as control. Using these dynamics, I constructed the general form of the formation. Since I want the formation to maintain its shape as much as possible, I created the description for the position of one agent in the formation shown in **Fig 1**.

Given this formulation for the agent position, any formation shape is possible so far. However, this assumes that the agents do not have safe radii to avoid collision with each other. This brings us to my next assumption.

Assumption 2. *The formation is safe initially (i.e. in its nominal position the agents should be able to maintain formation without violating each other safe radii)*

Although this assumption may be relaxed in the future, for this project, I have assumed that the formation is possible without collision. For implementation purposes, I have also

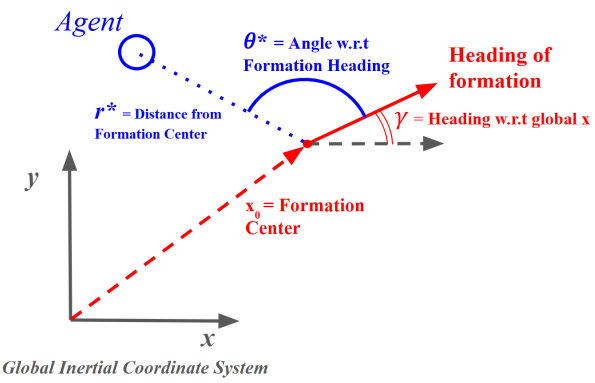

Fig. 1. Formation setup of Arbitrary agent position with respect to the formation center and the global coordinate system

elected to simplify the safety radii of the robots further with the following assumption.

Assumption 3. *$r_{s,i}$ is constant among all robots in the formation. Where $r_{s,i}$ denotes the safety radius of an agent i.*

Although the solution I have developed would certainly work without this assumption, the examples I will present all adhere to this assumption, so I will state it as an assumption here. My implementation also assumed a similar property of the obstacles in the playing field, so I will state that here as well.

Assumption 4. *All obstacles in the field have the same safety radius denoted as $r_{s,j}$ for any obstacle j*

Denoting the safety radii for the obstacles in an almost identical way may initially present some confusion about the differentiation between the agents and obstacles. However, it will become apparent why this will not matter for the solution formulation.

Some other assumptions that are critical for this problem are as follows.

Assumption 5. *The agents know both the positions of the other agents and the obstacles.*

This assumption is necessary for the problem solution that I have developed, but I will discuss the relaxation of this in the discussion section later on.

Assumption 6. *While one agent is moving or calculating, all other agents are stationary.*

This assumption allows for a decentralized computation scheme instead of relying on centralized computation. Further expansion of this assumption will be made in the discussion.

Assumption 7. *The formation center follows a predetermined flight path regardless of the movement of the agents.*

This assumption is crucial for agents to make their calculations for the discrete-time step at hand. However, this may be relaxed in the future, and recommendations on how will be made in the discussion section.

B. Flight Path

The first step in my implementation was determining a flight path through an arbitrary obstacle field. To accomplish this, I attempted my implementation of A*. Although my method worked, it ran incredibly slowly and it did not encapsulate feasible motion for a rigid body formation. I needed a motion planner that would work on the SE(2) manifold. Matlab 2023©has a motion planner based on A* that works with nonholonomic constraints and could be easily implemented into my code. The planner is Hybrid A* or the `plannerHybridAstar` method. Utilizing this planner, I could develop a feasible motion plan for the drone swarm as shown below.

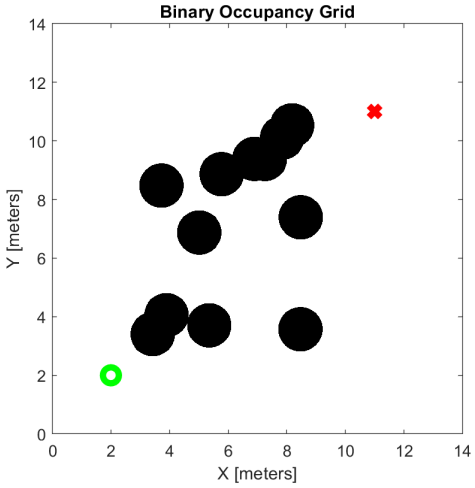

Fig. 2. Example field setup with the green marker as the start and the red as the finish. Plotted using `binaryOccupancyMap` function in MATLAB 2023©

With this motion planner working, I had a predetermined flight path through a randomly generated obstacle course. The Hybrid A* planner worked more efficiently than my A* star planner and accounted for heading which is representative of a rigid body formation that could rotate as well as translate.

C. Formation

Using the formation definition in **Fig 1**, it is possible to construct the ideal location of every agent in the formation with a given world coordinate for the formation center, the heading of the formation, and the polar coordinates of the agent with respect to the formation center.

$$x_i^*(n) = r_i^* \begin{pmatrix} cos(\theta_i^* + \gamma(n)) \\ sin(\theta_i^* + \gamma(n)) \end{pmatrix} + x_0(n) \tag{2}$$

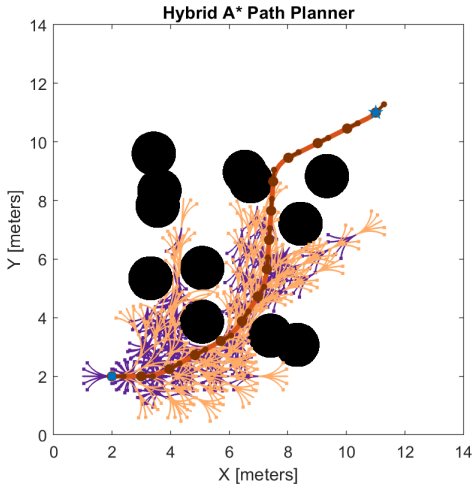

Fig. 3. Sample motion plan of center according to the `plannerHybridAStar` planner in Matlab 2023©. The path is illustrated in dark red with both position and heading.

Where $x_i^* \in \mathbb{R}^{2\times 1}$ denotes the designated "ideal" position for the agent at some time step n. However, while the agent is meant to be at this location, if this location falls inside of the safety radius of an obstacle or another agent, it is apparent that this location is not the best spot for that agent. With this in mind, I needed to develop a way of finding the location that would best match the geometry of the formation for a given agent without colliding with any obstacles.

D. Best Spot

Any formulation I developed for calculating the best spot had to obey some underlying postulates about safety.

Postulate 1. *To reach its best point, no agent nor its safety radius may violate any safety radius of another agent or obstacle.*

While potentially self-evident from a collision avoidance perspective, this postulate allows me to prove that the best point formulation I develop obeys the collision avoidance principles. From this postulate, the following lemmata are true.

***Lemma* 1.** *The best point must not fall inside the safety radius of an obstacle or agent.*

***Lemma* 2.** *When occupying its best point, an agent's safety radius must be entirely outside of the safety radius of any agent or obstacle.*

I will give no formal proof of these lemmata but rather state that if either of these lemmata is false, then the **Postulate 1** is violated at the terminal point of the agent's motion to the best point. This would be a contradiction proving that they are true.

There are some assumptions we will make about this best point for implementation reasons.

Assumption 8. *The best point lies inside the playing field.*

Assumption 9. *There exists a feasible path to the best point accounting for obstacles*

Using **Assumptions 8** and **9**, I only need to find the best point inside the playing field for a given agent to move ahead. Therefore, I propose the following formulation for the best point.

***Theorem* 1.**

$$x_{i,best} = \arg\min_{x_w}(f(x_w))$$

$$s.t \quad f(x_w) = \begin{cases} \infty, & \forall x_w \in X_\sigma \\ ||x_w - x_i^*||, & otherwise \end{cases} \tag{3}$$

$$X_\sigma = \{x_w| \quad ||\sigma_j - x_w|| \leq r_{s,j} + r_{s,i}\} \quad \forall i \neq j$$

Where $X_w \in \mathbb{R}^{2\times 1}$ is the set of world points in the playing field, $x_w \in X_w$, X_σ denotes the set of x_w that lies inside a radius equal to the sum of the safety radius of any obstacle or agent and the safety radius of the current agent, i, being calculated, and σ_j are the x_w for the obstacles and other agents.

The best point is simply the point closest to the 'ideal' location, x_i^*, that lies inside the playing field and outside the safety radii of the obstacles and the agents. Note this formulation extends the safety radii of the obstacles and agents, $r_{s,j}$, by the safety radius of the current agent, $r_{s,i}$. A sample of the calculation for the best point can be seen in **Fig 4**

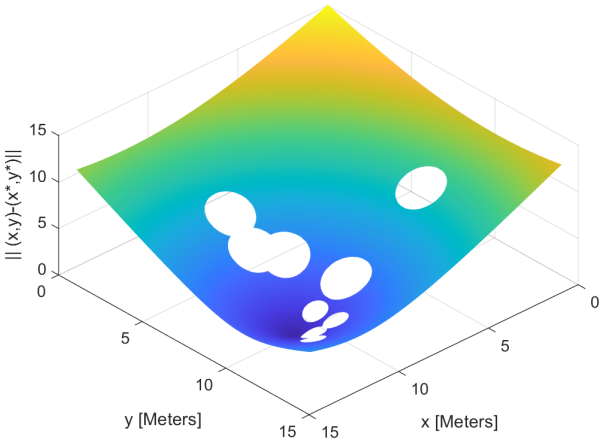

Fig. 4. Sample of $f(x_w)$ with $x_w = (x, y)$ and $x_i^* = (x^*, y^*)$. The best point can be seen at the darkest blue portion of this surface. It can also be seen that the $f(x_w \in X_\sigma)$ are shown as holes in this representation

By recreating the obstacles and other agents with artificially larger safety radii, we can now treat the agent as a point moving through the plane. This formulation also meets ***Lemmata 1*** and ***2*** since the best point from this calculation must fall outside the obstruction radius, and, by inflating the obstruction radius by $r_{s,i}$, the safety radius of the agent does not infringe upon that of the obstruction.

By keeping the obstructions safety radii large, I now let a motion planner generate a path from the current agent location to the best point. This creates a collision-free path that satisfies **Postulate 1**. Since all other agents are stationary by **Assumption 6**, the agent can move without needing to recalculate trajectory.

E. Solution Flow

Assuming the path has been predetermined by the Hybrid A* Algorithm for the whole formation. The Workflow for the algorithm can be seen in **Fig 5**.

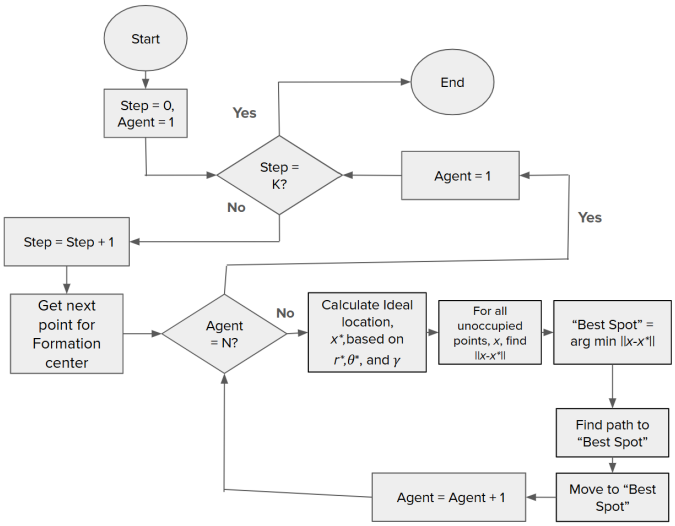

Fig. 5. This workflow is based on knowledge of the formation path, obstacle and agent locations, and the parameters for each agent's location in the formation as polar coordinates. Note: K is the number of points in the formation path and N is the number of agents.

III. Results

Although results for this formation control scheme are largely heuristic and qualitative, I will present an example of one an example formation moving through an obstacle course. For videos showing more examples see Appendix A.

A. Example Formation

1) Formation Path: In **Fig 6**, the output of the Hybrid A* motion planner for from start pose of $x_w = [2, 2]^T, \gamma = 0$.

2) Formation Setup: **Fig 7** shows the formation in its start position. The formation is defined by the following polar coordinates with respect to the center.

$$(r*,\ \theta^*) = \begin{pmatrix} 1, & 0 \\ 0, & 0 \\ 1.5, & \pi/4 \\ 1, & 2\pi/3 \\ 1.5, & \pi \\ 2, & -2\pi/3 \end{pmatrix} \tag{4}$$

3) Motion: To illustrate an iteration of the solution at a specific time instance, **Fig 8** shows an agent running path planning for along step along the formation path

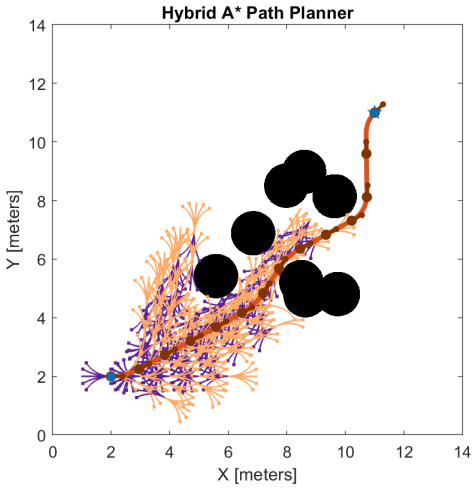

Fig. 6. Formation motion plan from Hybrid A*. Hybrid A* also incorporates nonholonomic constraints like turn radius.

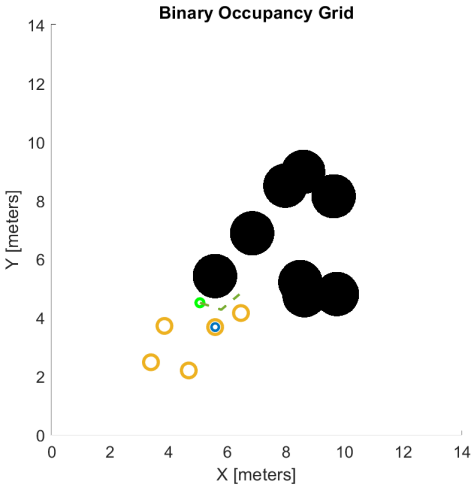

Fig. 8. Example of an agent moving to its best point after calculation. The bright green marker is the agent, the dull green is the path to the best point, and the marigold circles show the other agents' safe radii.

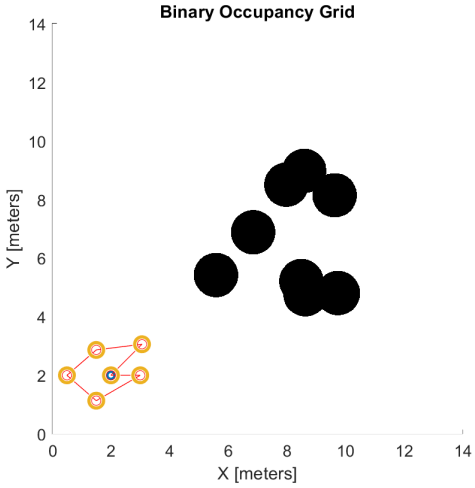

Fig. 7. Sample Formation for the given motion plan. The marigold circles denote the safety radius for each agent.

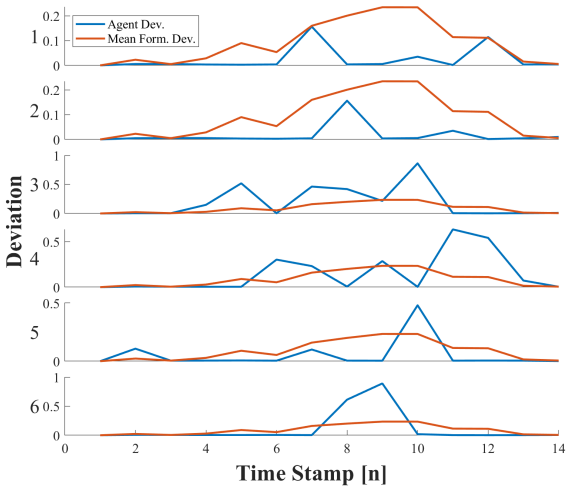

Fig. 9. Deviation of individual agents throughout the path. Red indicates the mean deviation of the whole formation, and blue shows the individual deviation.

4) Quantitative Deviation: Despite the semi-qualitative measurement of performance, I have also developed a way to visualize the performance through measurement. **Fig 9** shows the distance between the agents' ideal location and the best point, $||x_{i,best} - x_i^*||$, as the agent moves along the formation path.

The individual deviations can be seen to be reasonably small being on the order of 0.1-1 meters. The mean deviation of the whole formation, however, shows what an observer would expect from a formation shifting through an obstacle field. That is to say, the deviation gets larger as the formation moves through the obstacles, but then shrinks back to zero as the formation exits the obstacle field.

5) Video: For video results, please refer to the link in Appendix A, *"ObstacleFormationPath5.avi"*. This will give the reader a more intuitive understanding of how the formation moves as a result of the solution formulation.

IV. Discussion

A. Drawbacks

Although this method works and maintains as close a formation shape as possible while following a prescribed path, it has disadvantages. Firstly, it is computationally expensive to run a path planner for every step agent at every step along a path. However, this method could work faster if the agents were allowed to translate and disregard finishing orientation. If those assumptions can be made (e.g. for quad-rotor agents) then lighter motion planning algorithms such as Dijkstra or A* could be implemented. In addition, if a future implementation decided to find a local best point instead of a global best point, then this would drastically shorten the motion planning step. However, this implementation would need to have constraints on the minimum size of the local area to account for best points that may not have feasible paths.

B. Assumptions

While some assumptions are more easily relaxed such **Assumption 3** and **4** which deal with the safety radii of the obstacles and agents, some can not easily be relaxed. For example, **Assumption 2** that enforces the formation to feasible in the absence of obstacles would be hard to relax. If the formation has agents inside the safety radii of other agents, then the implementation of this method would struggle to plan a path to any best point given the start location is in a spot outside of the available motion planning space. Limited relaxations may occur when discussing the feasibility of the formation when discussing formation in which agents are outside of the safety radii of other agents, but their safety radii overlap.

Assumption 5, could be relaxed, however, a tertiary motion planner would need to be implemented on every step of an agent's motion to its next best point. This could result in a halted formation where the whole formation waits for an agent to reach a destination it may not be able to reach (i.e. obstacles become apparent as the agent moves to its best point that cuts the agent off from its best point). This also becomes computationally expensive and does not scale to small time steps well.

To relax **Assumption 6**, there would need to be an in-motion planner as with the relaxation of **Assumption 5** to allow an agent to receive positions with the agents it is connected to at any given time and recalculate the trajectory to the best point. This also would likely require a distance-based communication topology or undirected connected graph topology. Without these, there is no mechanism for an agent would receive the position of a disconnected agent that it may collide with.

Hand in hand with the previous assumption relaxations, relaxing **Assumption 7**, can be implemented with an online motion planner for the formation center. If this were relaxed then a sensor mechanism could easily be implemented for dynamic obstacle positions. This should be explored in the future.

C. Other Approaches

1) Zeroing Control Barrier Function: Although I have only discussed a decentralized motion planner strategy, I did attempt to solve this problem using a Zeroing Control Barrier Function (ZCBF) as described in [1] with quadratic programming. I developed a ZCBF that would create a safe set restricted by the minimum distance an agent could move before running into any other agent or obstacle.

$$\begin{aligned} h_{s,i} \;&=\; -max(r_{s,i} + r_{s,j} - ||x_i - \sigma_j||), \forall x_i \neq \sigma_j \\ s.t \quad X \;&= \{x\} \\ \Phi \;&= \{\phi\} \\ \Sigma \;&= X \cup \Phi \\ x_i \;&\in X \\ \sigma_j \;&\in \Sigma \end{aligned} \tag{5}$$

Where x_i denotes any agent i, σ_j denotes any obstacle or agent j, and $r_{s,i}$ denotes the safety radius of any σ_i.

While this formulation does meet the criterion for a ZCBF, an issue arose with implementation. To meet the Lipshitz Continuity condition, I use the *LogSumExp* smooth approximation for the *max* function since the *max* is not Lipshitz Continuous. However, at the boundary of the safe set for any given agent the smooth approximation would prove inadequate. This is because the ZCBF (disregarding the negative) would approach the 0 boundary from the left numerically (i.e. stay negative). But, since the *LogSumExp*, approximation is an upper bound of the *max* function, the smooth approximation would overshoot 0 and cause the system to become unsafe.

Although this zeroing control Barrier Function did not work, I believe a different ZCBF could potentially work and should be explored in future work.

2) Quadratic Programming: Square Constraints: To repurpose the use of quadratic programming I tried to create a safe set that was the inscribed square of the maximum possible safe radius defined in the ZCBF section. While this would technically work, the safe set constricts irreversibly when two agents or an agent and an obstacle get close. This would cause the agents to deadlock almost instantly when they reached the obstacles.

While a different ZCBF may show promise, I am dubious about the success of repurposing linear constraints to this problem.

V. Conclusion

The goal of this project was to explore a method to drive a formation through an obstacle field while maintaining as close to the ideal formation as possible. I developed a solution that used a motion planner to drive each agent to a safe location closest to the intended formation location at that time instance. The method accomplished safe planning in the trials it was subjected to and showed predictable mean deviation and individual deviation that were reasonably low. Future work on this topic could include lighter computational motion planning for agents, exploration of zeroing control barrier functions, relaxation of the a priori formation path, and an active sensor grid to observe a dynamic obstacle field.

References

[1] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs for safety critical systems," *IEEE Transactions on Automatic Control*, vol. 62, no. 8, pp. 3861–3876, 2017.

Appendix A
Documentation

Here is the link to a Git Repository with more videos of examples. In addition, I have also included a video presentation of the material in abbreviated form.