Predictive Drone Swarms with Limited Field of View

Robotics Project I (MICRO-580)

January 15th 2021

Student: KIMBLE Thomas **SCIPER**: 261204

Professor: FLOREANO Dario
Assistant 1: SORIA Enrica
Assistant 2: SCHILLING Fabian

EPFL

LIS

Contents

1 Introduction

1.1 Motivation

Some robotics tasks such as manufacturing, surveillance, or search and rescue require the use of multiple agents to be accomplished. Indeed some objectives would be too time consuming or even impossible to complete if left to a single robot. Cooperation is key in the success of multi-agent tasks, and we can look to nature for inspiration. These observations have lead to groundbreaking discoveries and algorithms such as the Reynolds flock algorithm [1].

Coordination is facilitated with a centralised approach with all agents sharing common information, but real life circumstances do not always allow for this. Therefore another way to approach the problem is by using distributed systems, where each agent is self-contained in its decision making as well as its sensing [2][3].

One standing issue with multi-agent robotics task is how to safely navigate an environment. Collision free trajectory generation is key for safety and success in cooperative missions. Obstacle avoidance can be successfully obtained with algorithms and concepts such as potential fields [4] and vector field histogram [5]. But many algorithms mitigate constraints, and assume that each agent is able to sense every other neighbouring robot [6]. Indeed most sensors are limited, thus adding concept of a limited field of view.

In this project, the effect of a limited field of view is to be introduced to a previously functional predictive model [2], combining state of the art collision avoidance algorithms with real world constraints such as the limited field of view found in off the shelf sensors.

1.2 Relevant Literature

The DMPC approach to trajectory generation and obstacle avoidance algorithm proposed by C. E. Luis, M. Vukosavljev, and A. P. Schoellig in *Online Trajectory Generation with Distributed Model Predictive Control for Multi-Robot Motion Planning* (2019) [2] is the main base for the combination of predictive collision avoidance and limited sensing in this project.

This paper proposes a model with a collision avoidance algorithm that can reduce on average around 50% of the travel time required to complete a multi-agent point-to-point navigation when compared to a Buffered Voronoi Cells (BVC) approach. Along with these results, the algorithm has a 90% success rate for point-to-point navigation with 30 agents in a $18m^3$ workspace.

Other important research on the effects of limited FOV sensing is *Collision Avoidance with Limited Field of View Sensing: A Velocity Obstacle Approach* by S. Roelofsen, D. Gillet and A. Martinoli (2017) [6], which shows that vision based collision avoidance can remain safe when the robot's sensors detect other robots in their limited FOV. Their approach is validated by experiments with real quadrotor agents.

2 Methodology

After stating the problem and the assumptions made, we will describe the online distributed model predictive control used to generate the trajectory and collision avoidance for the agents. We will compare different obstacle avoidance algorithms, and discuss the addition of a limited field of view for sensing.

2.1 Problem Statement

We have N agents with known linear dynamics in a 3-dimensional subset $\mathcal{W} \subset \mathbb{R}^3$, with N start points $\mathbf{p}_{start,i} \in \mathcal{W}$ and N end points $\mathbf{p}_{end,i} \in \mathcal{W}$. The scope of this project does not consider static obstacles, so for each agent i the goal is to compute inputs $\mathbf{u}_i[k] \in \mathbb{R}^3$ and $\mathbf{u}_{\theta,i}[k] \in \mathbb{R}$, for each time step k, such that:

- The agents reach their end position
- The agents do not collide with each other
- The agents remain within $\mathcal{W}$

At every time step k, each agent i has a position $\mathbf{p}_i[k]$, a velocity $\mathbf{v}_i[k]$, a heading angle $\theta_i[k]$, and a heading angle velocity $\omega_i[k]$.

2.2 Assumptions

We assume that all agents are equipped with a controller for position trajectory tracking where the inputs $\mathbf{u}_i$ are position references.

Due to implementation difficulties (part 2.5), inputs for position and heading angle were to be computed separately therefore giving us two different discrete linear systems for trajectory tracking dynamics (equation 2.1) and for heading angle dynamics (equation 2.2).

$$\mathbf{x}_i[k+1] = \mathbf{A}\mathbf{x}_i[k] + \mathbf{B}\mathbf{u}_i[k] \tag{2.1}$$

$$\mathbf{x}_{\theta,i}[k+1] = \mathbf{C}\mathbf{x}_{\theta,i}[k] + \mathbf{D}\mathbf{u}_{\theta,i}[k] \tag{2.2}$$

With inputs $\mathbf{u}_i[k] \in \mathbb{R}^3$ a position reference and $\mathbf{u}_{\theta,i}[k] \in \mathbb{R}$ a heading angle reference, and with states $\mathbf{x}_i[k] = (\mathbf{p}_i[k], \mathbf{v}_i[k])$ for position and velocity and $\mathbf{x}_{\theta,i}[k] = (\theta_i[k], \omega_i[k])$ for heading angle and angular velocity (around z-axis). For the position and velocity states we have a second order system with $\mathbf{A} \in \mathbb{R}^{6\times6}$ and $\mathbf{B} \in \mathbb{R}^{6\times3}$. For

the heading angle and angular velocity states we have a first order system with $\mathbf{C} \in \mathbb{R}^{2\times2}$ and $\mathbf{D} \in \mathbb{R}^{2\times1}$. We use a first order system for the heading angle because we had no knowledge of the rotational dynamics and for simplification purposes.

Each agent has the same limited field of view defined by a width angle α and a height angle β as shown in figure 2. We assume that agent j's states are known by agent i if and only if agent j is in agent i's field of view.

We assume that the agent's field of view range is higher than the size of the workspace $\mathcal{W}$ to ensure visibility of all agent's within the field of view, despite how far they are.

We assume that each agent knows the position of all other agents at the start of the each simulation, whether in an agent's FOV or not. We can assume that this information is given, or that each agent could perform a rotational manoeuvre to gather all of this information. From this we can test different heading angle initialisation methods as visited in part 2.4.2. However, once the trajectory generation starts, agents can only see others that are in their FOV.

2.3 Online Distributed Model Predictive Control

2.3.1 Trajectory

We use the same approach as in [2] which uses distributed model predictive control (DMPC). The input sequence is recomputed to be applied over a finite horizon on K time steps, where k_t is the time step at time t_0. A time step duration h is chosen and a continuous input signal $\mathbf{u}_i(t)$ for $t \in [t_0, t_0 + t_h]$, with $t_h = (K-1)/h$. The continuous input signal $\mathbf{u}_i(t)$ is a concatenation of Bézier curves.

Therefore the trajectory is defined by a concatenation Bézier curves, constructed thanks to Bernstein polynomials of degree p. The curve is characterised by a series of $p+1$ control points which serve as the optimisation variables to compute each agent's trajectory.

The Bézier curves are expressed in the power basis through a transformation B which converts control points into polynomial coefficients, before being sample thanks to another transformation Γ which then samples the polynomial at different time steps. The input is sampled and noted $\mathbf{U}_i \in \mathbb{R}^{3K}$ and it can be obtained from a linear combination of the control points of a continuous Bézier curve.

2.3.2 Model and constraints

The prediction model is the same as in [2]. Using the dynamics described in equation 2.1 with $\bar{\mathbf{x}}_i[k_t]$ the measured state at time step k_t and $\mathbf{U}_i$ the stacked input sequence, we can represent the stacked predicted state sequence over the horizon for agent i $\mathbf{X}_i$:

$$\mathbf{X}_i = \mathbf{A}_0 \bar{\mathbf{x}}_i[k_t] + \mathbf{\Lambda} \mathbf{U}_i \tag{2.3}$$

With $\mathbf{\Lambda}$ defined as:

$$\mathbf{\Lambda} = \begin{bmatrix} \mathbf{B} & \mathbf{0}_3 & \dots & \mathbf{0}_3 \\ \mathbf{AB} & \mathbf{B} & \dots & \mathbf{0}_3 \\ \vdots & \ddots & \ddots & \vdots \\ \mathbf{A}^{K-1}\mathbf{B} & \mathbf{A}^{K-2}\mathbf{B} & \dots & \mathbf{B} \end{bmatrix} \tag{2.4}$$

and $\mathbf{A}_0$ defined as:

$$\mathbf{A}_0 = \begin{bmatrix} (\mathbf{A})^T & (\mathbf{A}^2)^T & \dots & (\mathbf{A}^K)^T \end{bmatrix}^T \tag{2.5}$$

There are three constraints on the system that are to be taken into consideration during the optimisation step. Input continuity ensures trajectory smoothness and gives the equality constraint $\mathbf{A}_{eq}\mathbf{U}_i = \mathbf{b}_{eq}$. Dynamic feasibility limits the actuation and considers the environment dimensions giving us the inequality constraint $\mathbf{A}_{in}\mathbf{U}_i \leq \mathbf{b}_{in}$. Finally the collision avoidance described in part 2.3.3 gives the inequality constraint $\mathbf{A}_{coll}\mathbf{U}_i \leq \mathbf{b}_{coll}$.

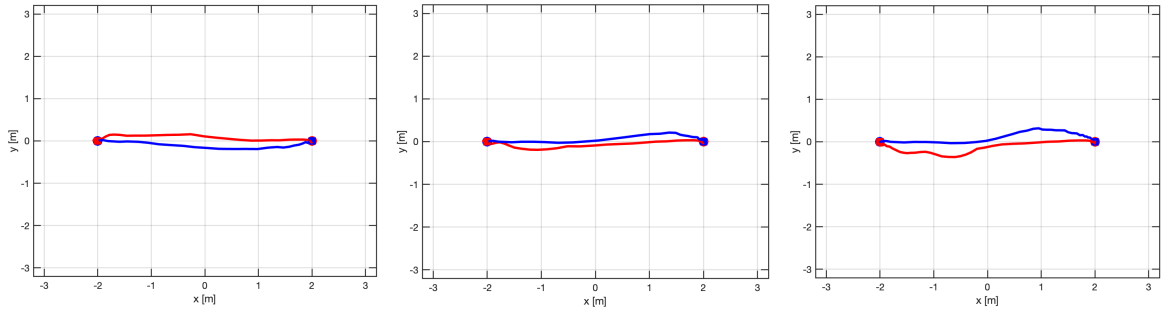
2.3.3 Collision Avoidance

For obstacle avoidance, the agents are required obey the inequality in equation 2.6 throughout the simulation. If the inequality is not obeyed between two agents, they are considered to have crashed and their trajectory execution is stopped until the simulation is terminated.

$$||\Theta^{-1}(\mathbf{p}_i[k_t] - \mathbf{p}_j[k_t])||_2 \;\geq\; r_{min}, \qquad \forall j \neq i \tag{2.6}$$

Where Θ is a scaling matrix for safety boundaries, and r_min is the distance where two agents are considered to have collided. The scaling matrix gives an ellipsoid with higher values for vertical positions allowing for the simulation capture the downwash effect of an agent's propellers.

Three obstacle avoidance algorithms were compared: On-demand Collision Avoidance in the input space, in the state space, and the Buffered Voronoi Cells method. Each method has an example trajectory shown in figure 1. For more information on each method refer to [2].



(a) On demand: Input space (b) On demand: State space (c) BVC

Figure 1: Comparison of trajectories of $N = 2$ agents switching positions while avoiding each other using three different obstacle avoidance methods

Collision avoidance between agents was tested with two agents switching positions being 4 meters apart. Ten experiments were run in each case, and the average execution times was recorded for each method. Figure 1 shows the smoothest trajectory to be the on-demand method with constraints imposed on the input space. As well as being the smoothest, the average cost was lower, and the execution time was the lowest with an average time of 6.2 seconds, as opposed to 6.8 seconds for the state space, and 8.0 seconds for BCV. Therefore, for the rest of the project, on-demand collision with constraints on the input space was used.

2.3.4 Cost Function

The cost function is made up of three quadratic terms , to be minimised in order to lead the agent to its goal position without collisions. The three terms are defined below.

Error to goal: equal to the sum of errors between the positions at the last $\kappa < K$ time steps of the horizon and the end location for each agent i

$$\mathcal{J}_{i,error} = \sum_{k=K-\kappa}^{K} q_k ||\hat{\mathbf{p}}_i[k|k_t] - \mathbf{p}_{end,i}||_2^2 \tag{2.7}$$

Energy: equal to a weighted combination of the sum of squared derivatives of the input

$$\mathcal{J}_{i,energy} = \sum_{c=0}^{r} \alpha_c \int_0^{th} ||\frac{d^c}{dt^c}\hat{\mathbf{u}}_i(t)||\, dt \tag{2.8}$$

Collision constraint violation: on-demand collision avoidance was implemented as soft constraints with a penalty term for violation

$$\mathcal{J}_{i,violation} = \zeta||\epsilon_{ij}||_2^2 + \xi\epsilon_{ij} \tag{2.9}$$

With q_k, α_c, ζ and ξ positive scalar weights. This leads the following standard quadratic programming problem to be solved:

$$\begin{aligned} &\underset{\mathbf{U}_i,\, \epsilon_{ij}}{\text{minimize}} \quad \mathcal{J}_{i,error} + \mathcal{J}_{i,energy} + \mathcal{J}_{i,violation} \\ &\text{subject to} \\ &\mathbf{A}_{eq}\mathbf{U}_i = \mathbf{b}_{eq}, \\ &\mathbf{A}_{in}\mathbf{U}_i \leq \mathbf{b}_{in}, \\ &\mathbf{A}_{coll}\mathbf{U}_i \leq \mathbf{b}_{coll}, \\ &\epsilon \leq 0 \end{aligned} \tag{2.10}$$

2.4 Heading Angle control

2.4.1 Limited Field of View

An agent i has a field of view F_i characterised by a width angle α, a height angle β and it's heading angle θ_i defined as:

$$F_i = \{\mathbf{p} \mid \ |\angle_{xy}(\mathbf{d}_i, \mathbf{p} - \mathbf{p}_i)| \leq \alpha/2,\ |\angle_z(\mathbf{d}_i, \mathbf{p} - \mathbf{p}_i)| \leq \beta/2, \} \tag{2.11}$$

With $\mathbf{d}_i = [\cos(\theta_i), \sin(\theta_i), p_z]^T$ the heading direction, $\angle_z$ the vertical angle, and $\angle_{xy}$ the horizontal angle between two points.

Angles α and β are common and identical for all agents and we have $0 < \alpha < \pi$ and $0 < \beta < \pi$. The field of view therefore resembles a pyramid with the summit at position $\mathbf{p}_i$ in the direction of the heading angle θ_i as shown in figure 2. In the scope of this project, the field of view range is assumed to be larger than the workspace, therefore we do not need to set a range limit. The field of view was chosen as such to resemble the field of view of common sensors such as cameras.

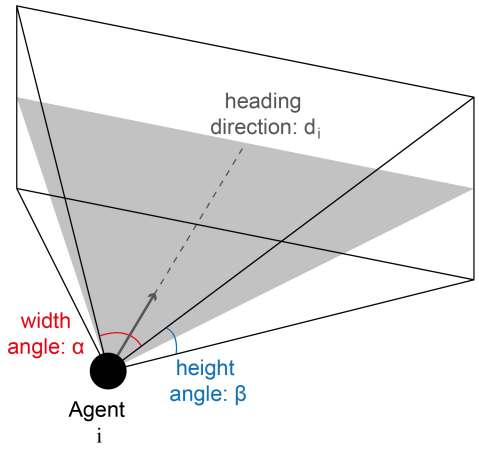


Figure 2: Field of view representation for an agent i

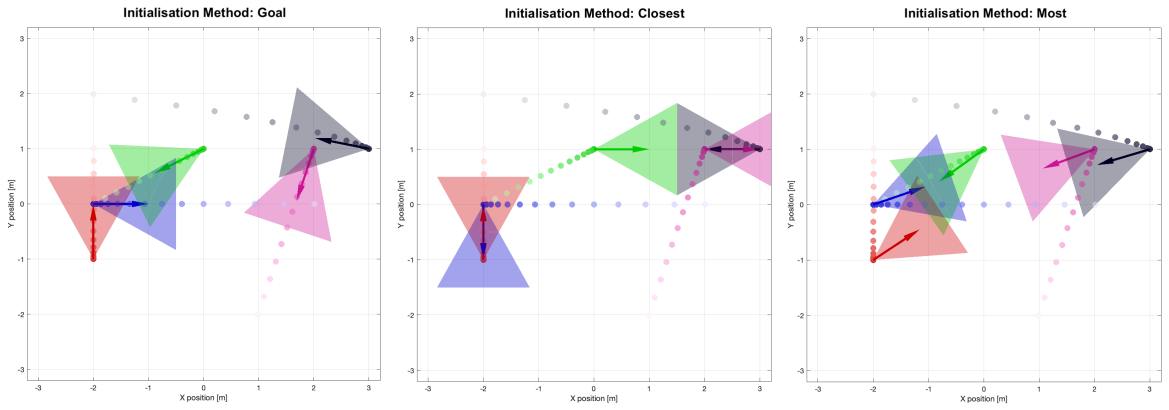
2.4.2 Initialisation Methods

We define three initialisation methods for each agent's heading angle at the beginning of each simulation: Goal, Closest and Most. Each method initialises heading angles according to a set of rules. Figure 3 illustrates the three initialisation methods with $N = 5$ agents, where each agent is at position $\mathbf{p}_{start,i}$, with its trajectory planned towards $\mathbf{p}_{end,i}$.

Goal:
Each agent i has an initial heading angle $\theta_i[k = 0]$ pointing towards its end position $\mathbf{p}_{end,i}$ as in figure 3a. This is simply done by calculating the angle between the $\mathbf{p}_{start,i}$ and $\mathbf{p}_{end,i}$ in the $[x, y]$ plane.

Closest:
Each agent i has an initial heading angle $\theta_i[k = 0]$ pointing towards its closest neighbour as in figure 3b.

Most:
Each agent i has an initial heading angle where the maximum amount of neighbouring agents are in its field of view as in figure 3c. For each agent i, we select an agent j and count the number agents within the field of view width α. We repeat for $j = 1, 2, ..., N$, $j \neq i$ and keep the agent j with the most neighbours in α. We choose the heading angle $\theta_i[k = 0]$ as the middle between the lowest and highest angle of the agents in the field of view α, ensuring that the most agents are in the field of view. Finally, we repeat for all agents $i = 1, 2, ..., N$.



(a) Goal (b) Closest (c) Most

Figure 3: Heading angle initialisation methods for $n = 5$ agents with FOV width $\alpha = 45°$

2.4.3 Control

After the initialisation step, the heading input $\mathbf{u}_{\theta,i}[k]$ is calculated thanks to a simple proportional controller. For each agent i we calculate find all agents within its field of view F_i. We then define a goal heading angle $\theta_{goal,i}$ which is equal to the mean angle between all agents in the FOV. We calculate the error from the goal at time step k $e_\theta[k]$, and apply a proportional controller with constraints on the actuation as in equation 2.12.

$$u_{\theta,i}[k] = K_p(\theta_{goal,i}[k] - \theta_i[k]) = K_p \cdot e_\theta[k] \tag{2.12}$$

Comparing a P, PD and PID controller for the heading control showed little difference in heading tracking performance during simulations. So we opted with the P controller as it is the simplest of solutions, adding fewer operations per agent per iteration, and saving memory by not saving previous error values for each agent.

2.5 Loose ends

The original plan was to have the heading angle be optimised using a predictive model as well. The goal was to implement the heading angle as a continuation of the state $\mathbf{x}_i[k] = (\mathbf{p}_i[k], \mathbf{v}_i[k])$, with $\mathbf{p}_i[k] = [p_{xi}, p_{yi}, p_{zi}, p_{\theta i}]$ and $\mathbf{v}_i[k] = [v_{xi}, v_{yi}, v_{zi}, v_{\theta i}]$. This would have given us a single discrete linear system for trajectory tracking dynamics including the heading angle with inputs $\mathbf{u}_i[k] \in \mathbb{R}^4$:

$$\mathbf{x}_i[k+1] = \mathbf{A}\mathbf{x}_i[k] + \mathbf{B}\mathbf{u}_i[k] \tag{2.13}$$

We would have then created a cost function for vision of other agents $\mathcal{J}_{vision}$ with new inequality constraints from the limited field of view $\mathbf{A}_{fov}\mathbf{U}_i \leq \mathbf{b}_{fov}$. The quadratic programming would have been formulated as such:

$$\begin{aligned}
&\underset{\mathbf{U}_i,\, \epsilon_{ij}}{\text{minimize}} \quad \mathcal{J}_{i,error} + \mathcal{J}_{i,energy} + \mathcal{J}_{i,violation} + \mathcal{J}_{vision} \\
&\text{subject to} \\
&\mathbf{A}_{eq}\mathbf{U}_i = \mathbf{b}_{eq}, \\
&\mathbf{A}_{in}\mathbf{U}_i \leq \mathbf{b}_{in}, \\
&\mathbf{A}_{coll}\mathbf{U}_i \leq \mathbf{b}_{coll}, \\
&\mathbf{A}_{fov}\mathbf{U}_i \leq \mathbf{b}_{fov}, \\
&\epsilon \leq 0
\end{aligned} \tag{2.14}$$

Unfortunately this solution was not explored fully because of implementation limitations. After implementing the heading angle into the agent's state, a trajectory was calculated but it was not optimal. An example trajectory is shown in figure 4. This solution should be reviewed and continued with future work.

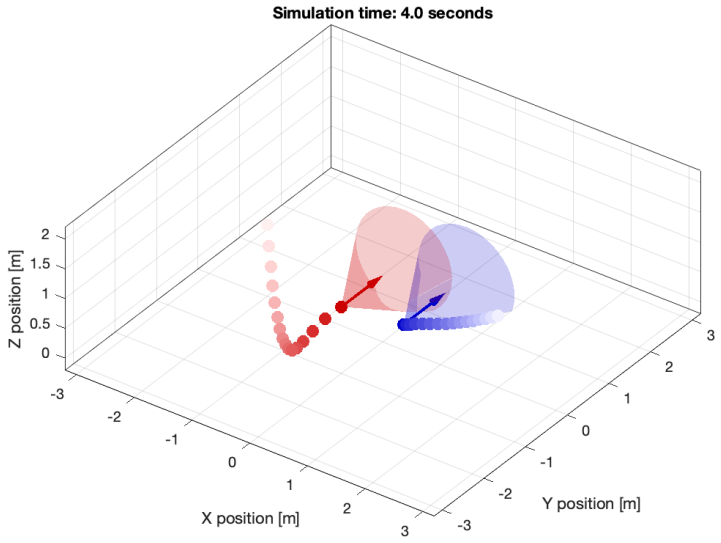


Figure 4: Non optimal trajectory generation of $N = 2$ agents while trying to implement heading angle into state. Note that the heading angle was represented as a cone at this stage of the project, and was not functional in this simulation.

3 Results and Discussion

In this section, we will describe the experiments ran to evaluate the performance of the model. We will report the results and compare different parameters for FOV and initialisation. Figure 5 shows a screenshot of a simulation being run to give a visual representation of the experiments run. This figure shows a swarm of $N = 4$ agents flying closely from one end of the workspace to the other with FOV angles $\alpha = 60°$ and $\beta = 30°$.

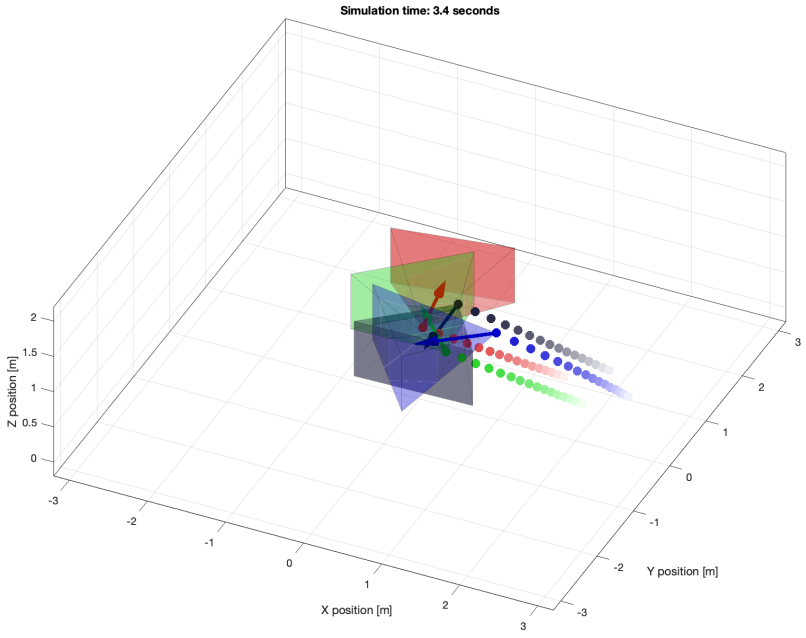


Figure 5: Screen capture mid simulation of the trajectory generation of $N = 4$ agents with visualisation of the heading angles (arrows), and the field of view (pyramids).

3.1 Experiment

The aim of the first set of experiments is to evaluate the performance of the model using different FOV height and width angles α and β, as well as how the heading initialisation impacts the performance.

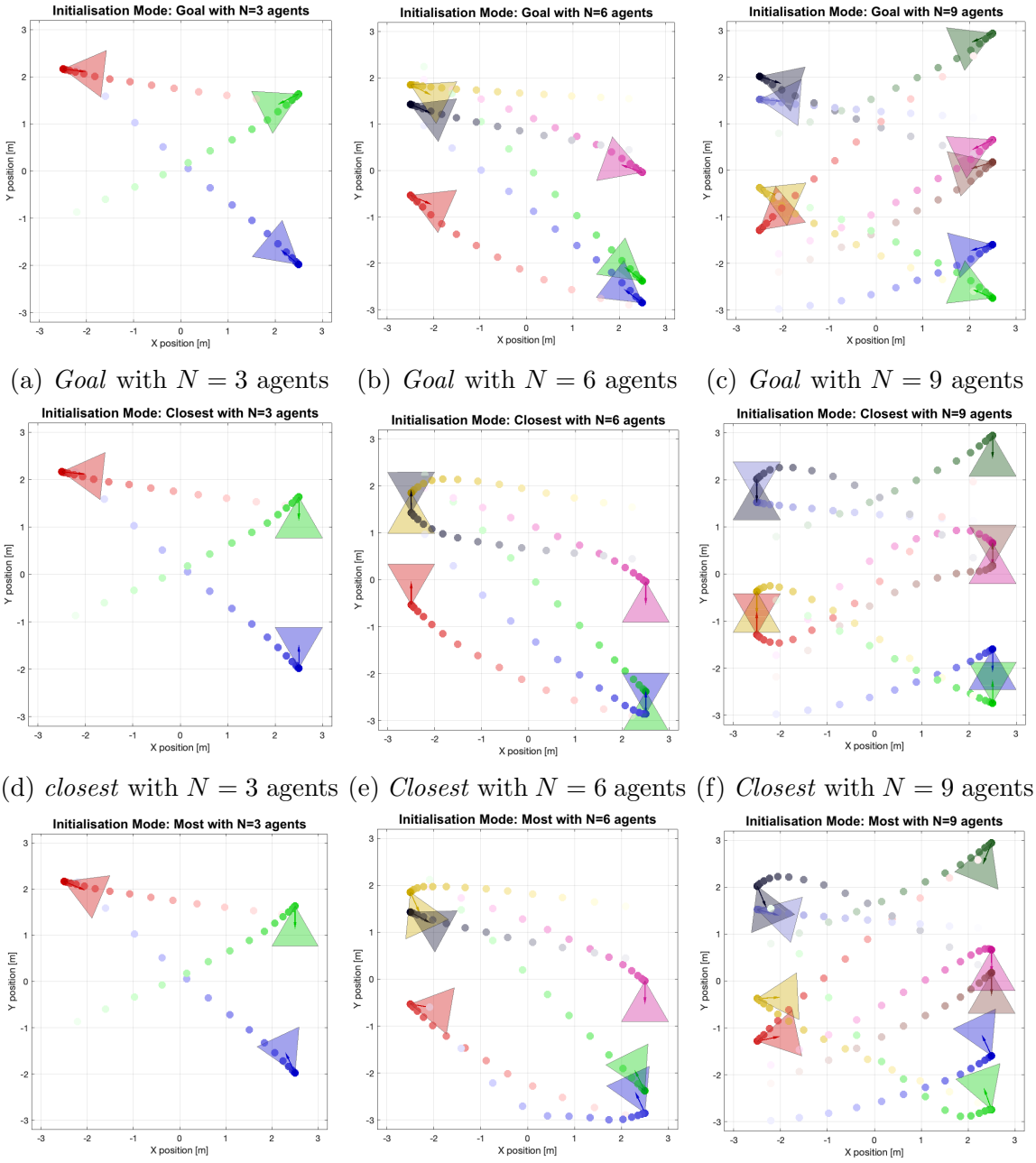
For each initialisation mode, we run simulations with $N = 3$, 6 and 9 agents. We vary the FOV angles from 15° to 180° with 15° increments to create a *heatmap* for each initialisation mode with each number of agents.

For each agent i, $i = 1, 2, ..., N$, if i is an odd number than it is placed on the $x = 2.5m$ axis, if i is an even number than it is placed on the $x = -2.5m$ axis. Agents are spread randomly along the y-axis within $y = [-3m \;\; 3m]$, and randomly along the z-axis within $z = [0.8m \;\; 1.2m]$ therefore giving us $\mathbf{p}_{start,i}$. End positions are determined the same way within the same boundaries for y and z except x values have their signs switched. This means that if an agent starts on the $x = 2.5m$ axis, it will finish on the $x = -2.5m$ axis, giving us $\mathbf{p}_{end,i}$. All agents are initialised at least $d = 0.3m$ a part.

This initialisation can be seen in figure 6 for each initialisation mode stated in part 2.4.2, with $N = 3$, 6 and 9 agents. This figure shows a view from above with a FOV width angle $\alpha = 60°$.

We use the average number of collisions over 5 experiments as a metric with each parameter combination to evaluate the models performance. A collision between two agents is reported when two agents are closer than $d = 0.3m$, and only one collision per agent pair is considered per simulation, avoiding counting multiple collisions for two agents stuck together.

We ensure that each experiment allows for all agents to reach their end position by setting the experiment length to $15 seconds$, a value chosen by previous tests to determine the maximal time for $N \leq 9$ agents. We choose to not evaluate agent speed, nor to evaluate the time to reach the goal in this experiment. Results are shown in figure 7.

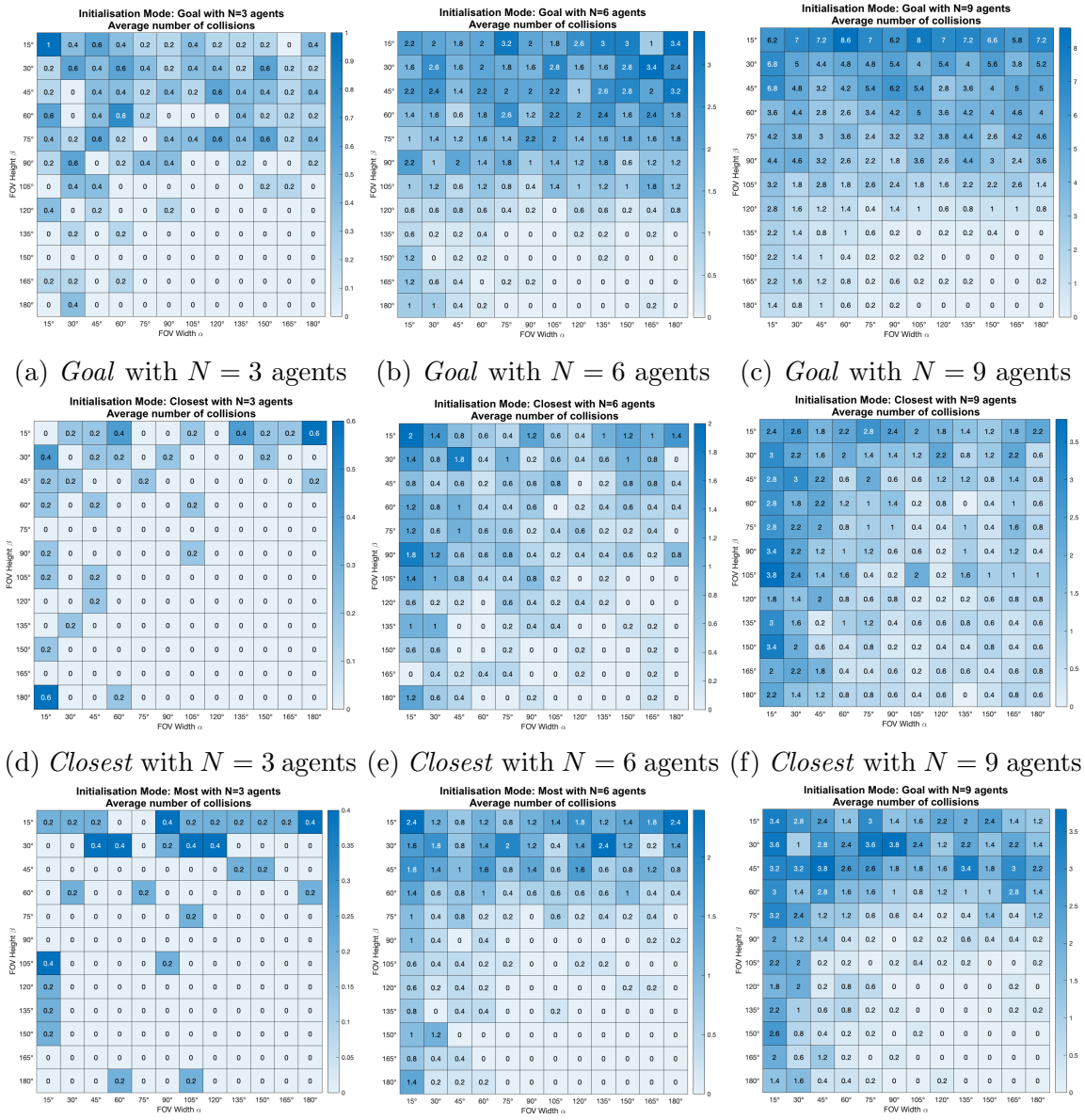


(a) *Goal* with $N = 3$ agents (b) *Goal* with $N = 6$ agents (c) *Goal* with $N = 9$ agents

(d) *closest* with $N = 3$ agents (e) *Closest* with $N = 6$ agents (f) *Closest* with $N = 9$ agents

(g) *Most* with $N = 3$ agents (h) *Most* with $N = 6$ agents (i) *Most* with $N = 9$ agents

Figure 6: In-simulation example of different initialisation modes with $\alpha = 60°$. Subfigures show results for $N = 3$, 6 and 9 agents with a), b), c) using *Goal* initialisation mode, d), e), f) using *Closest* initialisation mode and g), h), i) using *Most* initialisation mode



(a) *Goal* with $N = 3$ agents (b) *Goal* with $N = 6$ agents (c) *Goal* with $N = 9$ agents

(d) *Closest* with $N = 3$ agents (e) *Closest* with $N = 6$ agents (f) *Closest* with $N = 9$ agents

(g) *Most* with $N = 3$ agents (h) *Most* with $N = 6$ agents (i) *Most* with $N = 9$ agents

Figure 7: Average number of collisions using different FOV angle width and height angles α and β over 5 experiments for each parameter combination. Note that scales are not the same for each sub-figure. Sub-figures show results for $N = 3$, 6 and 9 agents with $a)$, $b)$, $c)$ using *Goal* initialisation mode, $d)$, $e)$, $f)$ using *Closest* initialisation mode and $g)$, $h)$, $i)$ using *Most* initialisation mode

3.2 Results

By comparing rows in figure 2.2 we can clearly see that the initialisation mode performances are the poorest with is *Goal*, followed by *Closest*, and the best performing is *Most*. Following is a more detailed overview of the performance with each mode.

Goal:
For each number of agents, we can see a trend where higher FOV angles induce fewer collisions. We specifically notice that the height angle β has more effect on the number of collisions compared to the width angle α.

From figure 7a with $N = 3$ we can see that the maximal average amount of collisions reported is 1 with the lowest width and height angles both at 15°. The best performance for three agents is with $\beta \geq 60°$ and $\alpha \geq 105°$.

Figure 7b with $N = 6$ shows the same trends but with performance with width angles lower than $\beta = 105°$ being very low, showing more collisions. The highest number of collisions happens with $\beta \leq 45°$ where the number reaches an average of 3.4 collisions.

In figure 7c we see very poor performances for $N = 9 agents$, with the same trends in angles as previously, but this time reaching up to an average of 8.6 collisions.

Closest:
We see a similar trend as with the *Goal* heading initialisation method, however with more variability in the number of collisions with different angles. Lower angles in both width and height lead to more collisions, however here we see that in general lower width angles α lead to more collisions than lower height angles β.

For $N = 3$ agents, figure 7d shows good performances for angles higher than 15°, with a maximum average of 0.2 collisions for all angles above this value.

In figure 7e we can see that the highest number of collisions occurs with $\alpha = \beta = 15°$ with on average 2 collisions for $N = 6$ agents. Best performances here are for high angles above 120° in both width and height.

Figure 7f shows that for $N = 9$ agents collisions have an average above 0.4 collisions across all angles. The maximal values are for $\alpha = 15°$ with up to 3.8 average collisions.

Most:
This heading initialisation method shows to be the most promising with slightly more collisions on average at lower width and height angles, but with a general better performance than the other methods.

We can see in figure 7g that the model performs well overall, with slightly more collisions with $\alpha = 15°$ and $\beta \leq 30°$. However the maximal average collision is 0.4 which is lower than the other two methods for $N = 3$.

For $N = 6$, figure 7h shows that medium to width angles $\alpha \geq 45°$ and height angles $\beta > 75°$ show very good results with a maximal average of 0.4 collisions only in two cases. For other angles, we can see that lower height angles β seem to lead to a higher number of collisions with a value of up to 2.4 average.

Finally in figure 7i for $N = 9$ agents good performances are seen for $\alpha \geq 60°$ and $\beta \geq 90°$. Below these angles we can get up to an average of 3.8 collisions for $\alpha = \beta = 45°$ for example.

3.3 Discussion

The previous results show how different heading initialisation methods and different FOV angles can affect the performance of agents agents avoiding each other during flights towards a goal.

Surprisingly, the different initialisation methods had a larger impact on the number of collisions than expected. With $N = 9$ agents, maximal average collisions varied from 3.4 to 8.6 when using different methods. The *Closest* method outperformed the other two methods in maximal average number of collisions, and by minimal FOV angle values with less than 1 collision on average. We find good performance with this method and FOV angles $\alpha \geq 60°$ and $\beta \geq 90°$.

When using the *Goal* method, agents generally start off the simulation with no others in their FOV, therefore leading to minimal heading corrections and imminent collisions that are only seen a moment before impact, and can not be avoided.

When using the *Closest* method, agents usually keep one other agent in their FOV, and as the closest agent is generally travelling in the same direction as itself, there is a low chance of trajectories coinciding and collisions happen with agents coming from further away. This method would perhaps work better if all agents within the swarm were heading in the same direction, but this hypothesis would need to be put to the test.

When using the *Most* method, by definition the agents were headed towards the maximal amount of others possible, therefore have more opportunities to correct their trajectory leading to less collisions. Indeed it is intuitive that an agent that can see more, can react accordingly.

If we focus more on the FOV results, independent of the initialisation mode we can see that in general a higher height angle β is required for fewer collisions when compared to the width angle α. This is due to the fact that the implementation of our model corrects the heading around the z-axis (yaw) and does not allow the heading and FOV to rotate up and down (pitch and roll). This leads to lower width angles α performing better than lower height angles β. Indeed in many cases, agents would

avoid others trajectories by flying under or over, therefore not seeing other agents from beneath or from above, leading to collisions.

3.4 Future Work

Although these first results are interesting, it would be necessary to add new metrics and experiments to evaluate and optimise trajectory speed. Here we would evaluate how long it would take different numbers of agents, in different environments to accomplish a certain task such as point-to-point navigation.

Adding another parameter to the limited FOV such as an azimuth as in figure 8 would allow to evaluate even more possible sensing configurations resembling what we can see in nature.

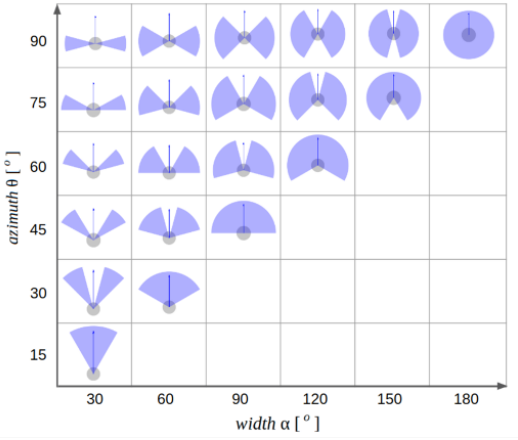


Figure 8: Graphical representation of the sensing configurations of an agent for different limited field of view including the azimuth [9]

It would also be enlightening to see how swarms would act together to accomplish a certain goal together, this could be by passing through obstacles, or reaching a destination as a swarm rather than flying against each other and avoiding collisions.

It would also be interesting to revisit the approach stated in part 2.5 to compute an optimal heading input using a predictive model like for the trajectory generation. This would allow to compute a solve quadratic programming that would include the heading and the limited FOV in the cost function and in the constraints. Heading initialisation could have less of an impact here as each agent's heading would be optimised at each iteration according to a predictive model, and would perhaps converge to a unique value independent of its initial state.

4 Conclusion

In conclusion, we introduced the effect of a limited field of view to a previously functional predictive model [2], combining a state of the art collision avoidance algorithm with real world constraints such as the limited field of view found in off the shelf sensors.

After comparing multiple collision avoidance algorithms, on-demand collision avoidance was selected as it showed the best performances for the model. An attempt to add heading and heading velocity to the agents states to solve the existing quadratic programming problem was made unsuccessfully, but to correct this issue the model was divided into two problems. An optimisation problem for the trajectory generation and obstacle avoidance, with a separate controller for the heading of the agents. After position and heading initialisation, we used a proportional controller to ensure that each agent was tracking other agents within their FOV to avoid collisions.

Field of view parameters such as width and height angles were compared with different initialisation methods to show that best performances occured with FOV width angles above 60° and height angles above 90°, when agents were initialised with the highest number of other agents possible with their FOV. We see trajectory generation and point-to-point navigation with up to 9 agents with on average less than a single collision with these parameters.

We notice that initialisation plays an important role in our model, and that it would be interesting to implement vertical heading control to lower the possible width angle lower than the reported 90° value for limited FOV.

Future work would include adding different FOV parameters such as an azimuth dividing the FOV into to separate regions. Trajectory completion speed would also have to be evaluated according to certain metrics to be defined. It would finally be interesting will to update the predictive model with the heading angle in the state space, and updating the quadratic cost function with limited field of view sensing rather than solving two separate problems.

References

[1] C. W. Reynolds, *Flocks, Herds, and Schools: A Distributed Behavioral Model* (1987)

[2] C. E. Luis, M. Vukosavljev, and A. P. Schoellig *Online Trajectory Generation with Distributed Model Predictive Control for Multi-Robot Motion Planning* (2019)

[3] R. Van Parys and G. Pipeleers, *Online distributed motion planning for multi-vehicle systems* (2016)

[4] D. E. Chang and J. E. Marsden, *Gyroscopic forces and collision avoidance with convex obstacles," in New trends in nonlinear dynamics and control and their applications* (2003)

[5] J. Borenstein and Y. Koren, *The vector field histogram-fast obstacle avoidance for mobile robots* (1991)

[6] S. Roelofsen, D. Gillet and A. Martinoli *Collision Avoidance with Limited Field of View Sensing: A Velocity Obstacle Approach* (2017)

[7] R. Bastien and P. Romanczuk *A model of collective behavior based purely on vision* (2020)

[8] G. Vásárhelyi, C. Virágh, G Somorjai, T Nepusz, A. E. Eiben, T Vicsek *Optimized flocking of autonomous drones in confined environments* (2018)

[9] E. Soria, F. Schiano, D. Floreano *The influence of limited visual sensing on the Reynolds flocking algorithm* (2019)