Internship report - Assistant engineer

Rutgers, The State University of New Jersey

Control of a drone swarm

Autonomous Robotics

Jules Le Gouallec - FISE24

September 2023

Control of a drone swarm

Safety behavior applied to a decentralised multi-drones system with intermittent reception of the position of the other drones using interval estimation

Abstract

This internship took place in a university research team at Rutgers, New Jersey during summer 2023. The team assembled by my supervisor, Laurent Burlion, consisted of 4 PhD students and 4 students working on various robotics projects. Mr Burlion's research is about Interval analysis applied to drone flight. My task was to provide computer code that implemented a specific situation : the formation flight of UAVs (Unmanned Aerial Vehicle) in the case where each UAV has intermittent access to the position of the others. Since the UAVs can be controlled simply by sending velocity commands to the micro-controller, the calculations were limited to speed commands that satisfy the constraints set out above. Additionally, I explored an approach that involved using acceleration commands.

Since this subject combines several specific Guidance/Estimation techniques (formation flying, collision avoidance, intervals), it was naturally broken down into a series of progressive objectives. The final objective is to fly 3 UAVs in formation, maneuvering towards and away from each other in such a way that each UAV is always outside the range of the other UAVs. If the constraints are respected, any collision is theoretically impossible.

Following conclusive simulations in Python, mission tests were carried out using the Gazebo simulation software. Satisfactory results with Gazebo enabled us to start preparing a real flight mission for 3 UAVs, but due to lack of time, we were unable to complete the tests.

Control d'un essaim de drones

Comportement de sûreté dans le cas de reception inconstante de la position des autres drones en se basant sur une analyse par intervalles

Résumé

Ce stage s'est déroulé dans une équipe de recherche universitaire, à Rutgers, dans le New Jersey pendant l'été 2023. L'équipe réunie par mon maître de stage, Laurent Burlion, était composée de 4 doctorants et 4 étudiants travaillant sur différents projets de robotique. Les recherches de M.Burlion concernent l'analyse par intervalles appliquée aux vols de drones. Ma mission a été de fournir des codes informatiques qui mettent en application une situation particulière : le vol en formation de drones dans le cas où chaque drone a accès à la position des autres de manière intermittente. Les drones pouvant être simplement commandés en envoyant des consignes de vitesse au micro-contrôleur, les calculs se sont arrêtés aux commandes en vitesse qui satisfont les contraintes précédemment énoncées. Plus tard, une approche qui utilise des commandes en accélération a été étudiée.

Ce sujet combinant plusieurs techniques particulières de Guidage/Estimation (vol en formation, évitement de collision, intervalles), il s'est naturellement découpé en une suite d'objectifs progressifs. L'objectif final est de faire voler en formation 3 drones qui s'éloignent et se rapprochent de telle manière que chaque drone soit toujours en dehors de l'intervalle de présence des autres drones. Si les contraintes sont respectées, toute collision est impossible, en théorie.

Après des simulations concluantes en python, des tests de missions ont été réalisés avec le logiciel de simulation Gazebo. Des résultats satisfaisant sur Gazebo, nous ont permis de commencer à préparer une mission de vol réel pour 3 drones mais nous n'avons pas pu aller au bout des tests.

Contents

Context of the internship

Rutgers University is the public university of New Jersey. The New Brunswick campus has more than 30,000 students and has a lot of departments ranging from psychology and organic chemistry to economics, and mechanical engineering.

The drone laboratory of Professor Burlion is located in the Mechanical and Aerospace Engineering department. The lab includes a high-performance computer, a 3D printer 30x30x60 cm, and provides access to a drone cage of $35m^2$ and 7m high equipped with a safety net and a Vicon visualisation system (cameras that precisely locate the drones in a the space). M.Burlion's research is about Control theory and his fundings come from Rutgers and the United States Navy.

Objectives and challenges

The theme of my internship was initially defined as *Interval Analysis and control of a drone formation*. My task in the lab was to provide codes according to my teacher's instructions to simulate missions that gradually get more complex. The final goal is a flight of a triangle formation of three drones using interval analysis. The results that I would provide would be used by M. Burlion to make a presentation to the Navy about possible applications of Interval analysis applied to UAV control. Obtaining external funding is a major challenge for researchers in the United-States and due to the high demand for funding, competition for research grants is very strong.

I am going to work with softwares or versions of software different from the ones seen in class so I will need to learn how to use it by myself searching on the internet which is quite time-consuming. I will also rely on members of the lab when it comes to programming and flying drones.

One major challenge was to adapt the vision of professor Burlion to the physical constraints of the drones. As he is specialised in theory matters, he is less aware of certain constraints encountered in practice, for example related to the sensor limits, or software requirements. So I had to talk to him often in order to inform him of these difficulties and discuss the technical choices made.

1 Formation flying

My first task was to read a document given by Mr. Burlion about formation flying and provide a python code that uses its theory. The formation wanted is decentralized, which means that every drone is computing its own commands independently of others. On the contrary, a centralized formation includes a central entity that performs computations for all robots. In our case, all drones fly at the same altitude.

In our case, we assume that the drone only has access to the position of the other drones at every instant. We also suppose that we can send velocity commands to the micro-controller of the UAVs.

According to the document[1], the formation rely on a reference pattern represented by the variable **r**, which is the difference of the reference position between two drone of the formation. Let **n** represent the number of robots.

We have $\forall i \in [[1; n]]$, $\forall j \in [[1; n+1]] \setminus \{i\}$:

$$r_{0ij} = \bar{x_i} - \bar{x_j}$$

The drone **n+1** is a hypothetical robot which plays the role of **leader of the formation**. The tracking errors are therefore:

$$\epsilon_{ij} = x_i(t) - x_j(t) - r_{0ij}$$

and the desired velocity of the agent number i is defined as : $(k_p > 0)$

$$u_i = -k_p \cdot \sum_{\substack{j=1 \\ j \neq i}}^{n+1} \epsilon_{ij} = -k_p \cdot \sum_{\substack{j=1 \\ j \neq i}}^{n+1} x_i(t) - x_j(t) - r_{0ij}$$

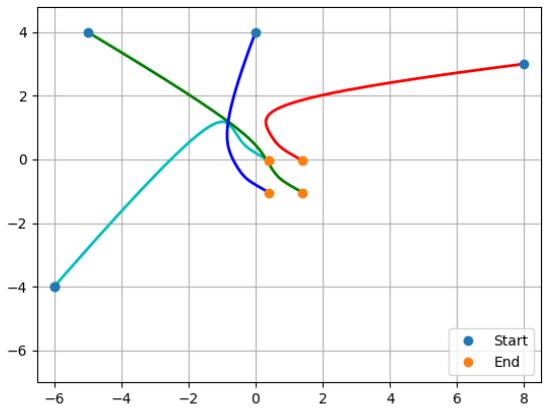


Figure 1: Example of a formation flight of 4 drones

2 Collision avoidance

To prevent crashes in a multi-drone system, it is crucial to have a collision avoidance program. In our case, we will use Control Barrier Functions(CBF) [2]. These prevent the commands sent to the drones from violating certain constraints by bounding them. As a result, the minimal distance between 2 drones always stays higher than a constant $\boldsymbol{d_{0inf}}$. In addition, these functions provide instability to the system which avoids gridlocks.

Theory

The paper [3] defines the variable $\phi_\perp$ that will traduce a orthoradial velocity constraint between 2 drones. This forces them to turn around each other :

$$\phi_\perp = -(y_1 - y_2) \cdot vx + (x_1 - x_2) \cdot vy$$

The squared distance between two drones is :

$$d_{02} = (x_1 - x_2)^2 + (y_1 - y_2)^2$$

The two previous equations give us :

$$\begin{cases} \dot{d_{02}} &= 2 \cdot (\dot{x_1} - \dot{x_2}) \cdot (x_1 - x_2) + 2 \cdot (\dot{y_1} - \dot{y_2}) \cdot (y_1 - y_2) \\ \phi_\perp &= -(y_1 - y_2) \cdot vx + (x_1 - x_2) \cdot vy \end{cases} \tag{1}$$

So we have the relation :

$$\begin{pmatrix} \dot{d_{02}} \\ \phi_\perp \end{pmatrix} = \begin{pmatrix} 2 \cdot (x_1 - x_2) & 2 \cdot (y_1 - y_2) \\ -(y_1 - y_2) & (x_1 - x_2) \end{pmatrix} \cdot \begin{pmatrix} u_x \\ u_y \end{pmatrix} + \begin{pmatrix} -2 \cdot \dot{x_2} \cdot (x_1 - x_2) \\ 0 \end{pmatrix} \tag{2}$$

$$\Leftrightarrow \begin{pmatrix} \dot{d_{02}} \\ \phi_\perp \end{pmatrix} = M(X_1, X_2) \cdot \begin{pmatrix} u_x \\ u_y \end{pmatrix} + \begin{pmatrix} D_1 \\ D_2 \end{pmatrix} \tag{3}$$

The two constraints that we want to be respected are : the minimal distance between two drones is always higher than $\boldsymbol{d_{0inf}}$ and their **orthoradial velocity** is never zero on their boundaries ($\boldsymbol{v_\perp^\#}$ is the tuning gain of the orthoradial component of the avoidance). Which can we traduced by :

$$\begin{cases} d_{02} &\geq d_{0inf}^2 \\ \phi_\perp &\geq d_{0inf} \cdot v_\perp^\# - k_2 \cdot (d_{02} - d_{0inf}^2) \end{cases}$$

According to the paper [3], this lead to :

$$\begin{cases} \dot{d_{02}} &\geq -k_1 \cdot (d_{02} - d_{0inf}^2) \\ \phi_\perp &\geq d_{0inf} \cdot v_\perp^\# - k_2 \cdot (d_{02} - d_{0inf}^2) \end{cases}$$

We define the saturation functions :

$$\begin{cases} h_1(X_1, X_2) &= -k_1 \cdot (d_{02} - d_{0inf}^2) - D_1 \\ h_2(X_1, X_2) &= d_{0inf} \cdot v_\perp^\# - k_2 \cdot (d_{02} - d_{0inf}^2) - D_2 \end{cases}$$

We obtain the saturated inputs as following :

$$\begin{pmatrix} u_{x,sat} \\ u_{y,sat} \end{pmatrix} = M^{-1}(X_1, X_2) \cdot \begin{pmatrix} Sat_{h1}^{+\infty}((1 \quad 0) \cdot M(X_1, X_2) \cdot \begin{pmatrix} u_x \\ u_y \end{pmatrix}) \\ Sat_{h2}^{+\infty}((0 \quad 1) \cdot M(X_1, X_2) \cdot \begin{pmatrix} u_x \\ u_y \end{pmatrix}) \end{pmatrix} \tag{4}$$

The function $Sat_a^{+\infty}(b) = Max(Min(b, +\infty), a)$ is called a *saturation function* (which is equivalent to a CBF). In this way, the velocity commands computed by the drone never violates the constraints defined previously.

Experiment

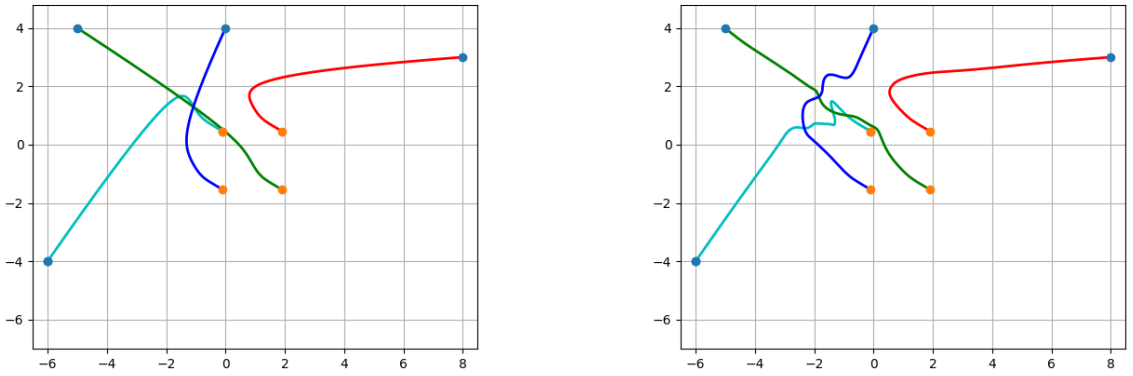
To apply this theory to my previous Python code, adjustments related to the sensor constraints are needed. Indeed, this method requires each drone to have access to the velocity of the others. This is not possible because even if they are equipped with IMUs, they are not supposed to communicate with each other. They only know other's positions thanks to the Vicon system. That is why I relied the **high-gain continuous-discrete time observer** of paper [4], which estimates position and velocity of an agent from discrete position data :

$$\dot{\hat{X}}_{ij}(t) = A \cdot \hat{X}_{ij}(t) - \theta \Delta_\theta^{-1} \cdot K_0 e^{-2\theta(t-\kappa_{ij}(t))} \cdot (\hat{p}_{ij}(\kappa_{ij}(t)) - p_j(\kappa_{ij}(t))$$

$$\hat{X}_{ij}(t) = \begin{pmatrix} \hat{p}_{ij}(t) \\ \hat{v}_{ij}(t) \end{pmatrix}, \Delta_\theta = \begin{pmatrix} I_2 & 0_2 \\ 0_2 & \frac{1}{\theta} I_2 \end{pmatrix}, K_0 = \begin{pmatrix} 2I_2 \\ I_2 \end{pmatrix}$$

$\hat{p}_{ij}(t)$ and $\hat{v}_{ij}(t)$ are the estimated position and velocity of agent j.
$p_j(t)$ is the position data of agent j.
θ represents the observer tuning parameter.
$\kappa_{ij}(t)$ is the last instant when agent i received the position data of agent j.

Fig. 2 shows the new trajectories taken by the drones :



(a) Without collision avoidance (b) With collision avoidance

Figure 2: Saturation technique applied to the formation flying

In this example, the minimal distance between 2 drones has been set at 1m. The shape of the formation is a square with 2m sides. The label OIST of Fig. 3 means Output-Input Saturation Technique. The drones deviate from their initial trajectories rather smoothly and the minimal distance (zone in gray) is globally respected.

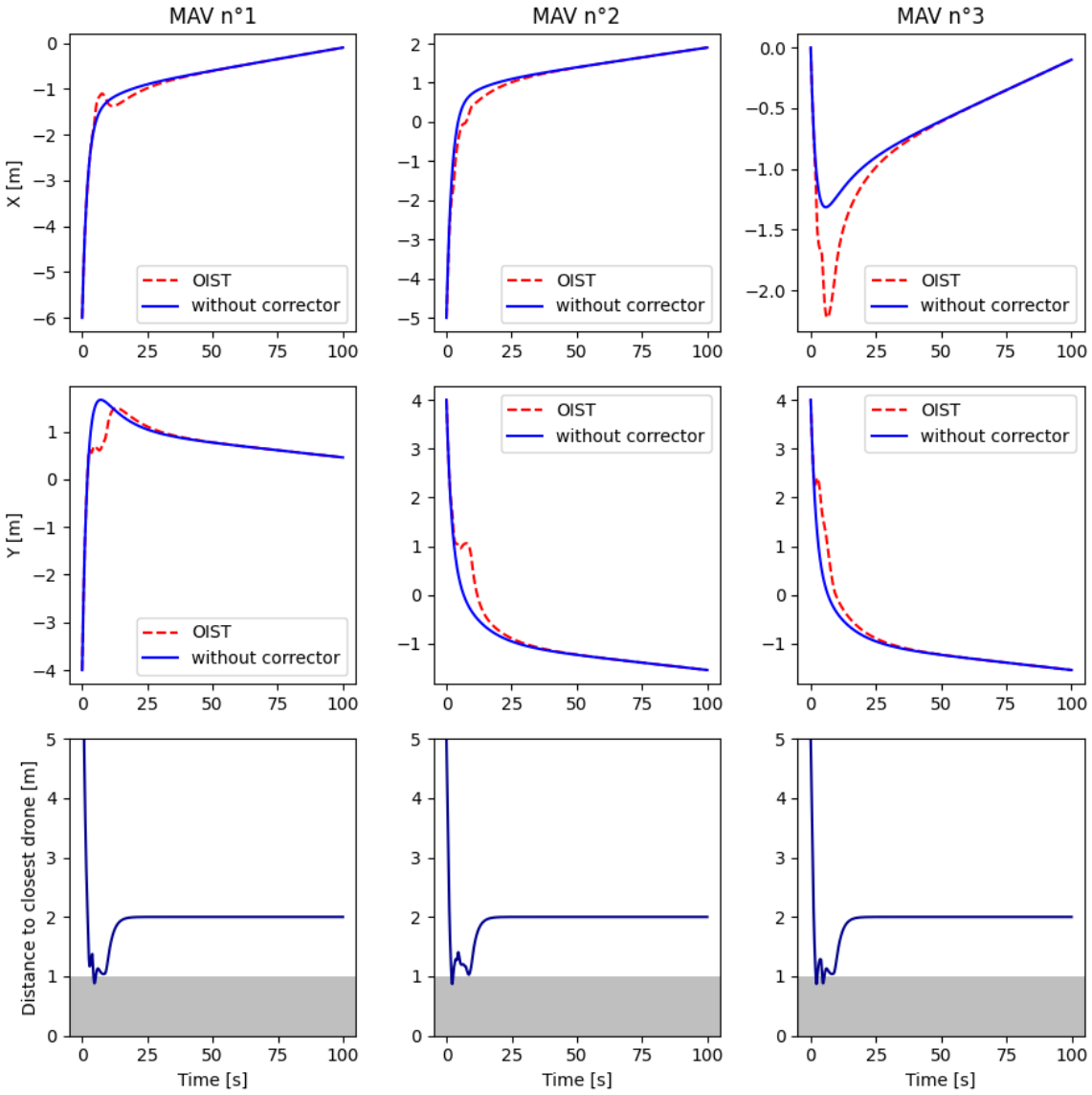


Figure 3: Deviation of trajectories and distances to obstacles during the flight (3 drones/4)

3 Intermittent data reception and Interval estimation

Now, in a safety purpose, the idea is to make the distance between drones vary over time in a special case : when the data reception of the position of the others is intermittent and fluctuating. We focus on **3 drones** flying in a **triangle**. When no data is received about the position of the other drones, the triangle is supposed to inflate, so each UAV is out of the **presence zone** of the others. Intervals are used to estimate the presence zone.

Remembering that our multi-robot system is decentralized, each drone knows exactly its position and receives intermittently the position of the other members.

Intervals with Codac

In class, we have used the codac library to do interval estimation. We used to manipulate "trajectory" objects and apply contractor networks to them. Here the approach is different because it turned that the trajectories are not convenient for real-time computation. Their calculations are too heavy.
Here the 2D position of an UAV is only represented by an interval vector $[x_j]$ that grows over time as following :

$$[x_j](t_{k+1}) \;=\; [x_j](t_k) + T_s \cdot [v_{max}]$$

For the mission, we have bounded the velocity of the drones to 2 m/s. The computations are slightly more pessimistic but it is faster. This evolution doesn't take into account the fact that the acceleration of the drone is bounded but it simplifies our problem.
When position data is received, the interval shrinks to the measurement taking into account uncertainties of measurement or delays :

$$[x_j](t_k) \;=\; [[y_{1,j}(t_k) - \epsilon \;;\; y_{1,j}(t_k) + \epsilon] \,,\, [y_{2,j}(t_k) - \epsilon \;;\; y_{2,j}(t_k) + \epsilon]]$$

Experiment

For the mission, we want our triangle to follow a circle. I artificially simulated the intermittent reception by creating a function that chooses randomly at every instant if position data is received by the drones or not.

Several approaches are possible to obtain the wanted behaviour. A first way could be to make the **minimal distance parameter** of collision avoidance increase and decrease, but it makes the drones turn around the presences zones. That is because of the orthoradial contraint seen previously.

Another solution is to make the **pattern scale** time-varying so that the drone stay out of the presence zones. Here it is a better solution because the velocity commands computed with the formation flying technique are rapid enough. The minimal distance parameter stay constant and so the collision avoidance is active only when the reception is frequent enough.

The Fig. 4 shows the point of view of UAV n°1 during the mission. He knows precisely its position and estimates others' with intervals The boxes in blue are the intervals of presence. This time the result is acceptable so it is ready to be tested on a real simulation software.

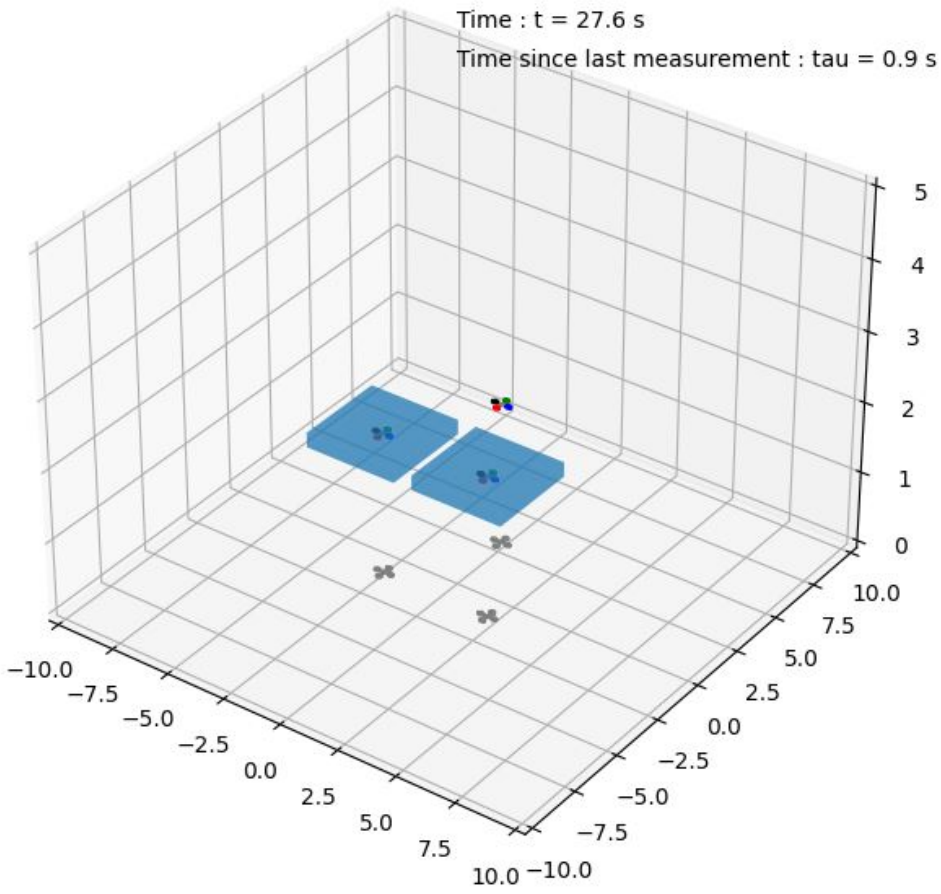


Figure 4: Triangle formation with intermittent position data reception - UAV1's point of view

Windows mission

M.Burlion wanted to show the utility of this approach in a particular case : when the drones have to go through a window. The drones have to wait in front of the window that the position data reception is frequent enough before going through.

Remembering that the orthoradial constraint of collision avoidance leads to a circular motion on the border, it results in a natural priority between 2 drones side by side : the one on the right side will pass first, as long as the presence zone of the other is small enough.

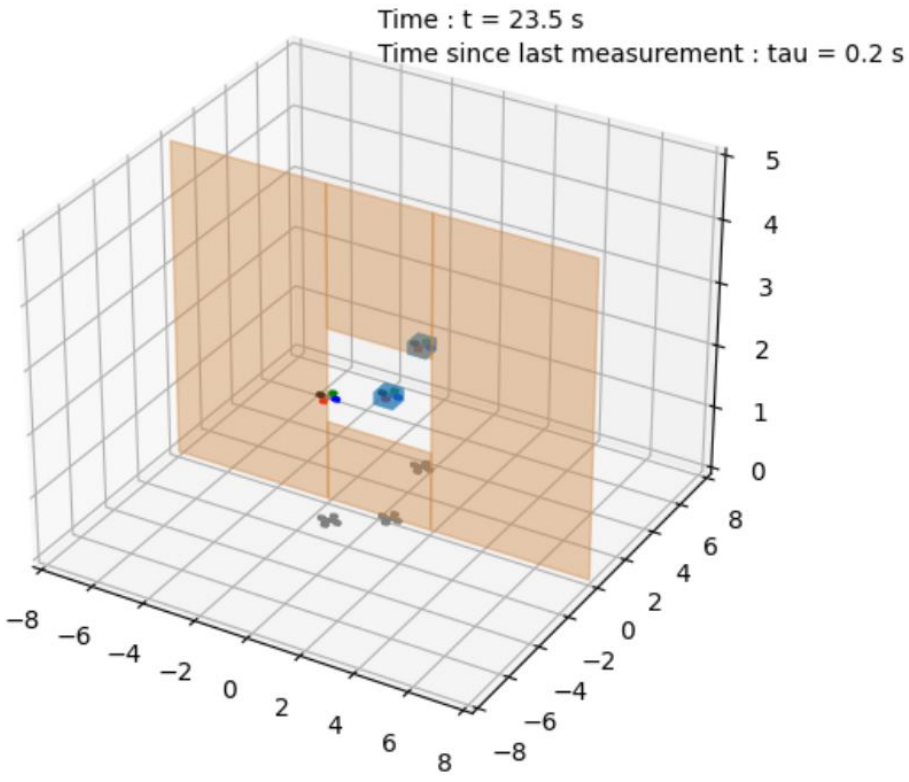


Figure 5: Triangle formation of drones passing through a window - UAV2's point of view

4 Gazebo simulation

To take your experiments further, it is necessary to switch to a real simulation software. Gazebo is the most common when it comes to drones. In professor Burlion's laboratory, the software is installed on a computer but it can simulate only one drone at a time. Consequently he gave me the link of an open source github project that is supposed to enable to fly multiple drones at the same time with Gazebo : **MRS UAV System** (https://github.com/ctu-mrs/mrs_uav_system). I have to read its documentation in order to find out how to install it on a computer and how to control it via a python code.

Installation

The recommanded installation of the toolbox has to be done with a Singularity image. Singularity is a containerization solution (like Docker) designed to facilitate the encapsulation and deployment of applications and their dependencies.

Although it is well documented, the installation was not easy because of the bugs that can occur concerning the installation of singularity and the creation of the image. I wrote the installation steps in a detailed document placed in the appendix.

Gazebo-ROS interface

Gazebo and the MRS packages autopilots communicate via ROS. It uses Mavros' functionalities to communicate with the sensors and the actuators. ROS is a bit different from ROS2 seen in class, so I have to adapt the writing of certain files as the CMakeLists.txt and the package.xml. The toolbox has its own controllers and multiple flight modes. It is possible to command a drone by giving it a target position, velocity commands or a trajectory to follow.

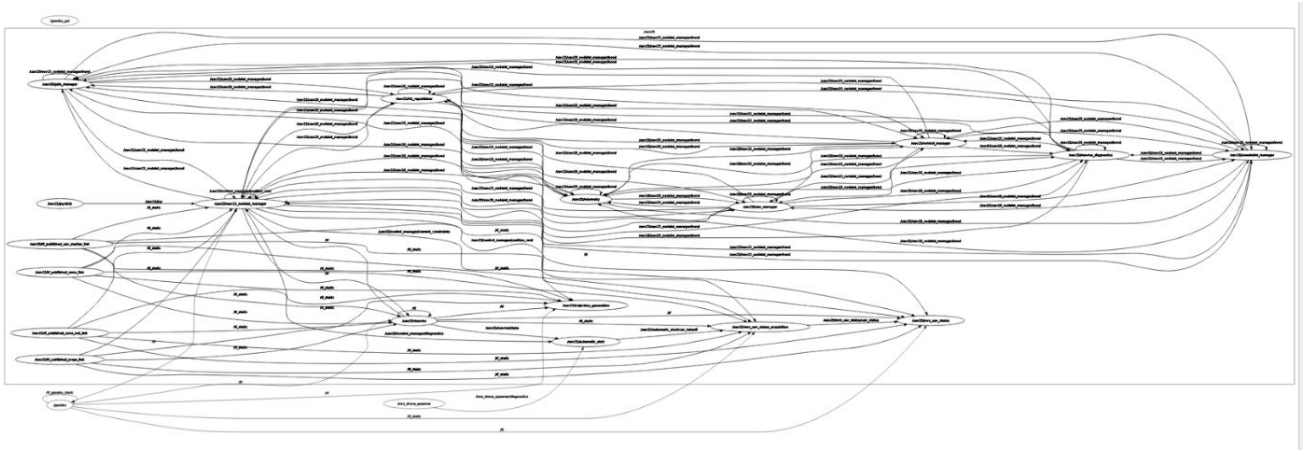


Figure 6: RQT ROS node graph of a single drone mission

The documentation could be more clear about the usage of several ROS messages and services. Moreover it doesn't indicate on which topic the positions of the drones can be read and precisely what are some services' type and their request type. Consequently there were many things to test : The Fig. 6 shows the node graph of a single drone mission. Knowing the number of active nodes and topics, I had to spend much time looking closely in the terminal using the commands **rostopic list** and **rostopic echo** to see what was published on the topics and try to make request to services.

How to control the simulation

Position of the UAV

Subscribe to this topic for x : **'/uav1/odometry/lkf_states_x'**
Subscribe to this topic for y : **'/uav1/odometry/lkf_states_y'**

Send the velocity command

Send a request to the service : **'/uav1/control_manager/velocity_reference'**
Service type : **VelocityReferenceStampedSrv**
Request type : **VelocityReferenceStamped**

Disable the package's collision avoidance systeme

Send a request to the service : **'/uav1/control_manager/mpc_tracker/collision_avoidance'**
Service type : **SetBool**
Request type : **bool** (send False)

Send a position command

Send a request to the service : **'/uav1/control_manager/goto'**
Service type : **Vec4**
Request type : **[x, y, z, yaw]** (for exemple [1., 0., 2., 0.5])

The simulation is started by running the file **start.sh** in a terminal. Once it is done, information about each drone is displayed. After a few seconds, the drones take off and wait in hover. Then, in another terminal, I open again a singularity container and I run my python script. First, I send the drones to their starting position, then I disable the MRS collision avoidance system which is very restricting. Once this is done, the mission begins : my python code sends velocity commands and subscribes to the positions of the drones.

Experiment

We are still using a decentralized approach, although I control the drone with only one single script to make things easier. This script takes into account the fact that the drones compute independently. We make them fly at the same height (2.5m). The maximum speed is still set at 2m/s.

Adaptation of the previous mission

I had to change several parameters of my previous code : the gains for the formation flying were to high, leading to oscillations. Moreover, the python script's time and gazebo simulation's did not go at the same speed. So I had to use the machine time to make sure the sampling period is respected.

As shown in Fig. 7, the drone on the top has a strange trajectory. The fact that all commands were computed and sent the same way made hard for me to find the bug.

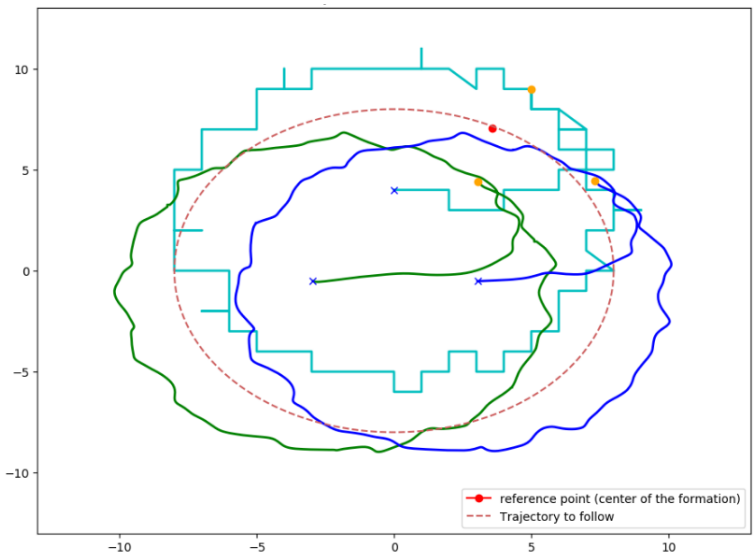
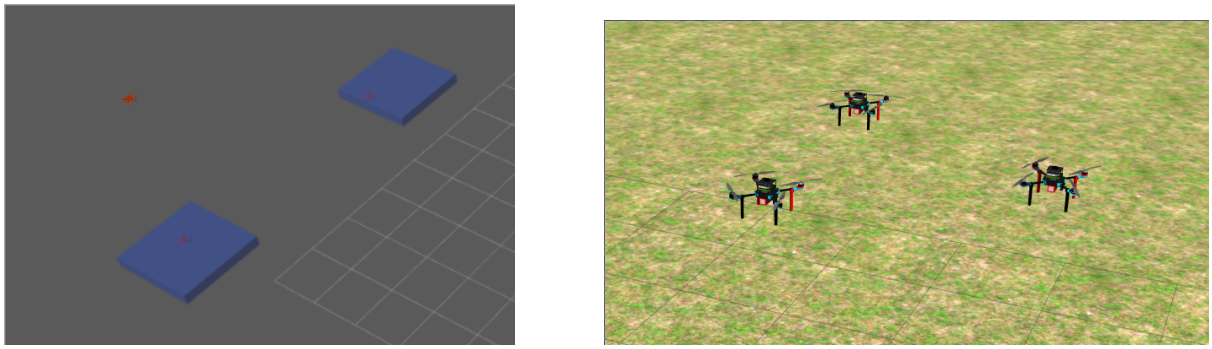


Figure 7: Example of UAV 2D trajectories during the first missions using Gazebo

I couldn't resolve the bug for a few days. I presented my problem during the weekly presentation with the members of the lab, and somebody evoked the idea that Gazebo was taking my command as an integer. Even if my code computed floats, I forced the command to be a float when it is sent in the code, and it worked. I modified the formation control law to remove the backlog comparing to the reference point (see Fig. 7). As professor Burlion is specialised in theory matters, the other lab members provided valuable assistance with technical issues. As shown in Fig. 8, I used RVIZ in addition to Gazebo to visualize the boxes of presence of the drones.



(a) RVIZ (b) Gazebo

Figure 8: Visualization of the drones

Intermittent position data reception

Now that a basic mission is up and running, I can add the intermittent data reception to the experiment. The issue here is to see if the UAVs are having the wanted behaviour, or if there are concerns that occur such as oscillations or lack of responsiveness. It is crucial to test it before a real flight test because the simulation results are close to what happens in reality.

After a few tests, I adapted the tuning gains. The robots are performing quite well : the triangle is inflating and shrinking as intended (see Fig 9).

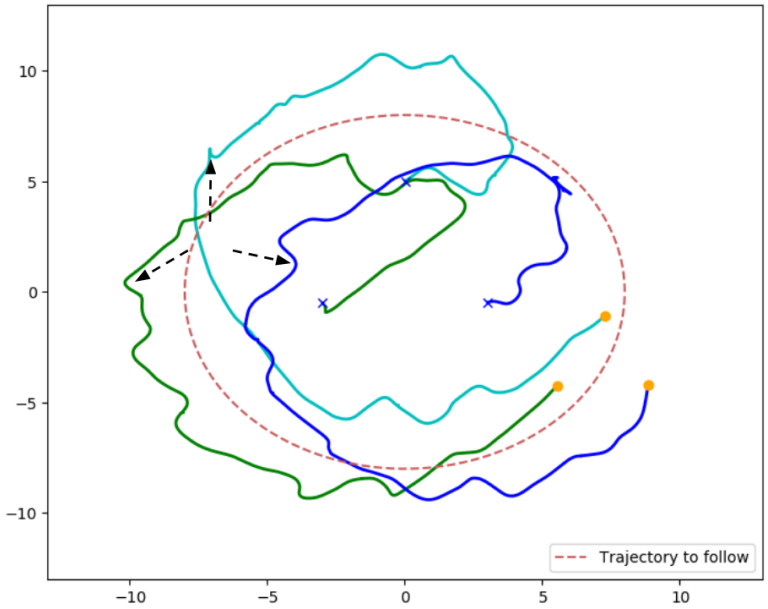


Figure 9: UAV 2D trajectories with intermittent position data reception using Gazebo

To verify that the goal is reached, I printed the distances to the center of the presence zones. The gray zone of the Fig. 10 represents the width of the intervals of each drone estimated by the others. The blue line is the distance to the center of the closest interval box. The graphs show that the drones always stay outside of the zones.

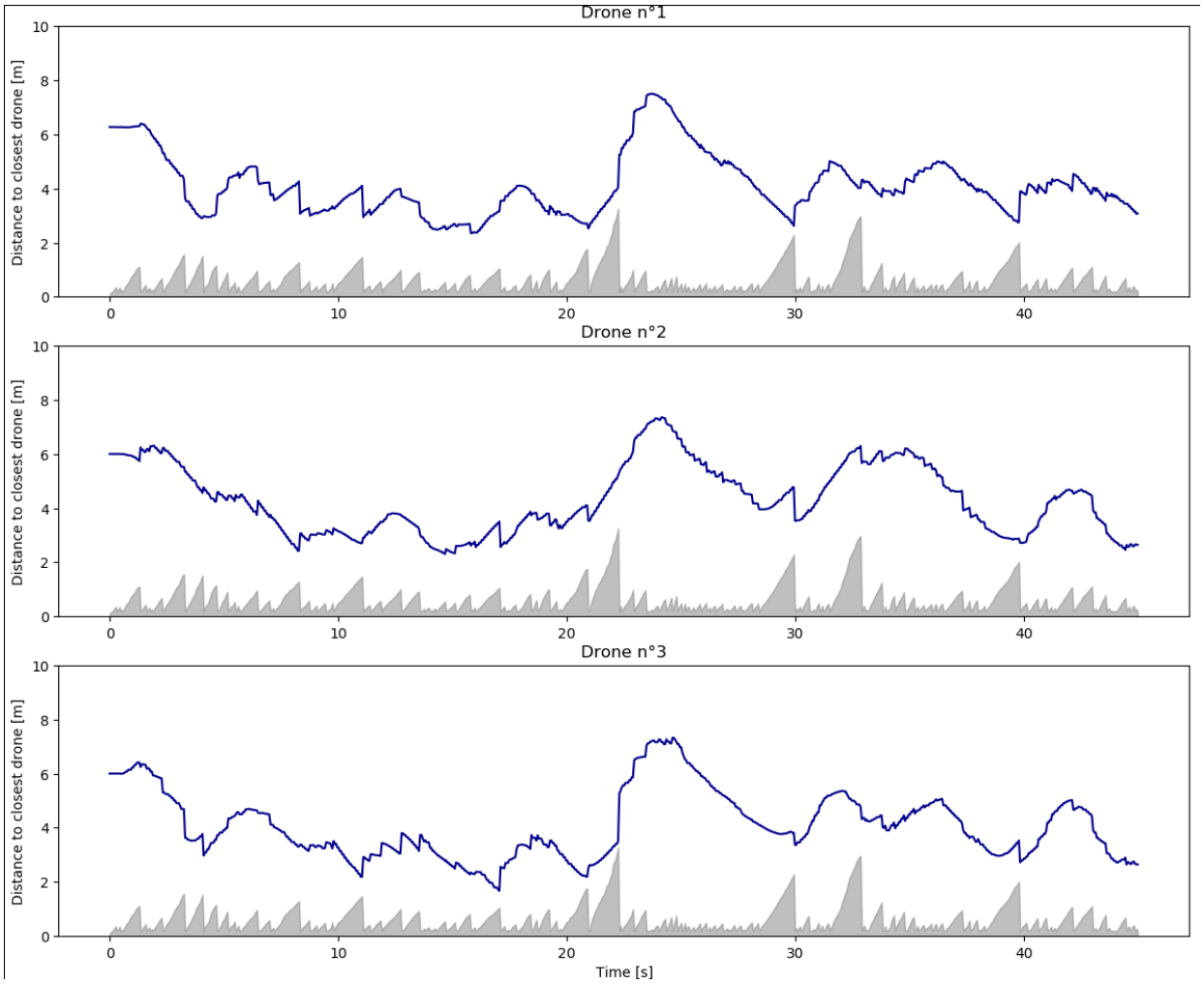
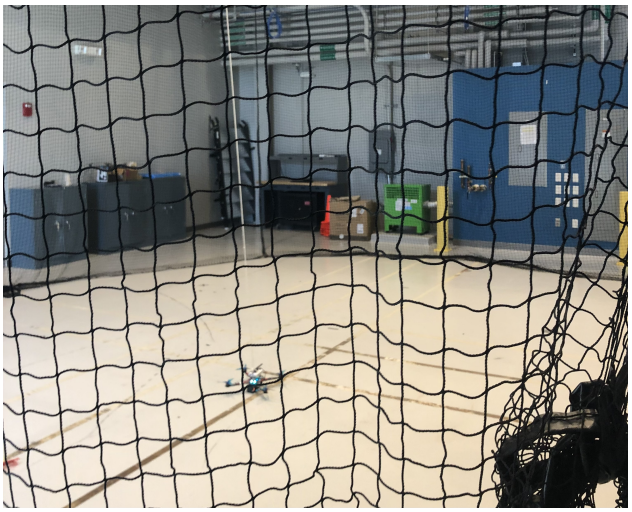


Figure 10: Distances between drones and the nearest presence zone

Even if the results are adequate, the UAVs overshoot and are not responsive enough. There is maybe a way so that the robots stay closer together, while respecting the constraint. However, it would be interesting to test this experiment in real conditions in the drone cage.

5 Real flight tests

To do the flight, I rely on a member of the lab : Agam. He is the one who knows how to configure the drones and how to write an onboard code. From an example of code he wrote for a drone to follow a circle, I wrote a code for each of the 3 UAVs. Basically, the onboard code is a python code that uses **rospy** and **Mavros** functions to get sensor values and send command to the motors. Normally, the code you write for a simulation is supposed to be put on the drone without modifications. But since Agam's installation of Gazebo and mine are differents, I had to adapt my previous code.



The picture on the left shows one of the quadcopters we have in the lab. The little gray spheres on the top of it are marks for the Vicon system cameras. It locates the robots to the nearest millimetre. The onboard computer is a Raspberry Pi 4 connected to the ESC, the IMU and the telemetry antenna. The motors used are Iflight Xing 2207 - 2450kV. On the UAVs, the code loop frequency is 50Hz, so is the VICON transmission rate.

The drone cage is located in the Richard Weeks Hall of Engineering building. It is $35m^2$ and 7m high surrounded by a safety net and 8 VICON cameras.

Setup

In the lab, we have 3 similar quadcopters, but we have encountered difficulties when we tested them separately. One of them cannot fly because there is a bug that occurs during the calibration phase with **Mission Planner**.

Consequently, we have chosen to adapt the mission to a flight of 2 drones. I adapted my code so the third drone is hypothetical and its position is known by the two other UAVs. The idea here is to begin by testing only the formation flying, and then go further in the experiment if possible.

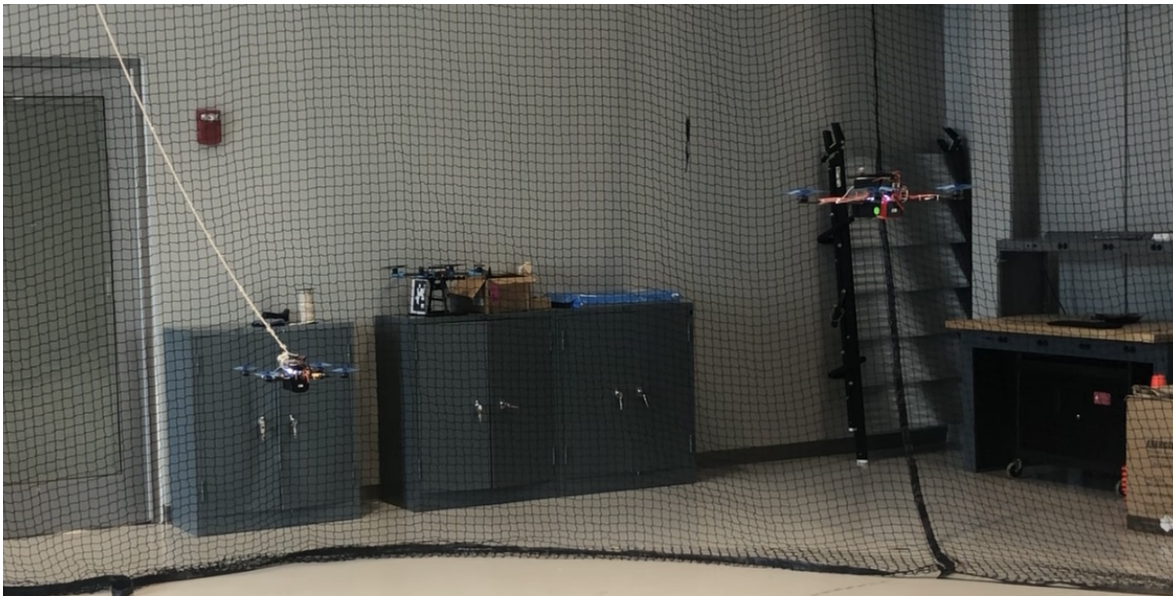


Figure 11: Simultaneous flight of 2 drones during the testing phase

The Fig. 11 shows the last test I did with Agam. To carry out the tests, the procedure is to start by arming the drones one by one with a transmitter and make them reach the minimal altitude of 2m so the onboard code takes over. Once a UAV is high enough, it is supposed to go to its starting position and wait each drone to be on its own. As soon as they are all in position, they must move back and forth in a triangular formation (taking into account the hypothetical UAV) for 60 seconds.

Results

Performing these tests has been time-consuming for many reasons. For each drone, we have to make a fly controlling it with a transmitter to verify if the autopilot is correctly configured. For exemple if the drone oscillates while flying, the roll (resp. pitch) tuning gain has to be adjusted. Sometimes the motors are listed in the wrong order. The rest of the procedure is also lengthy in itself because for every drone, it is necessary to open several terminals running simultaneously Mission Planner, ssh connection on drones etc. Furthermore, we spent a lot of time trying to calibrate the third drone, in vain.

We manage to make 2 drones fly at the same time and wait at their starting position several times, but we stopped at this step because we ran out of time. There were errors in my onboard codes since the only way to test it was in real conditions. I'd like to thank Agam for his expertise and patience during these sometimes tedious test phases.

6 2nd Order dynamics

Following to the results of the 1st order dynamics approach, M.Burlion would like to see the efficiency of a 2nd order dynamics method. That means we compute accelerations commands to send them to the microcontroller. It is also possible to convert these commands into desired thrust and angular velocity.

Formation flying

As before, the variable **r** is the reference pattern, which is the difference of the reference position between two drone of the formation. Let n still be the number of robots.

We have $\forall i \in [[1;n]]$, $\forall j \in [[1;n+1]] \setminus \{i\}$:

$$r_{0ij}{=}\bar{x_i} - \bar{x_j}$$

$$\dot{r_{0ij}}{=}0$$

Using the same reasoning as document[1], the tracking errors are now :

$$\begin{aligned}\epsilon_{ij}{=}&x_i(t) - x_j(t) - r_{0ij}\\ \dot{\epsilon_{ij}}{=}&\dot{x_i}(t) - \dot{x_j}(t)\end{aligned}$$

and the desired acceleration of agent i is defined as:

$$u_i{=}-k_p \cdot \sum_{\substack{j=1\\ j\neq i}}^{n+1}(\epsilon_{ij} + 3\dot{\epsilon_{ij}}) = -k_p \cdot \sum_{\substack{j=1\\ j\neq i}}^{n+1} x_i(t) - x_j(t) - r_{0ij} + 3(\dot{x_i}(t) - \dot{x_j}(t))$$

Collision avoidance

We rely on the paper [3] to re-calculate the CBFs. The relation (1) p.6 implies :

$$\begin{cases} \ddot{d_{02}} &= 2((x_1-x_2)\cdot(\ddot{x_1}-\ddot{x_2}) + (y_1-y_2)\cdot(\ddot{y_1}-\ddot{y_2}) + (\dot{x_1}-\dot{x_2})^2 + (\dot{y_1}-\dot{y_2})^2)\\ \dot{\phi_\perp} &= -(y_1-y_2)\cdot a_x + (x_1-x_2)\cdot a_y - (\dot{y_1}-\dot{y_2})\cdot v_x + (\dot{x_1}-\dot{x_2})\cdot v_y \end{cases}$$

So we have the relation :

$$\begin{pmatrix}\ddot{d_{02}}\\ \dot{\phi_\perp}\end{pmatrix} = \begin{pmatrix}2\cdot(x_1-x_2) & 2\cdot(y_1-y_2)\\ -(y_1-y_2) & (x_1-x_2)\end{pmatrix}\cdot\begin{pmatrix}u_x\\ u_y\end{pmatrix} + \begin{pmatrix}D_1\\ D_2\end{pmatrix} \tag{5}$$

$$\Leftrightarrow \begin{pmatrix}\ddot{d_{02}}\\ \dot{\phi_\perp}\end{pmatrix} = M(X_1,X_2)\cdot\begin{pmatrix}u_x\\ u_y\end{pmatrix} + \begin{pmatrix}D_1\\ D_2\end{pmatrix} \tag{6}$$

with :

$$\begin{aligned}D_1 &= 2(\dot{x_1}-\dot{x_2})^2 + 2(\dot{y_1}-\dot{y_2})^2 - 2\cdot\ddot{x_2}\cdot(x_1-x_2) - 2\cdot\ddot{y_2}\cdot(y_1-y_2)\\ D_2 &= -(\dot{y_1}-\dot{y_2})\cdot v_x + (\dot{x_1}-\dot{x_2})\cdot v_y\end{aligned}$$

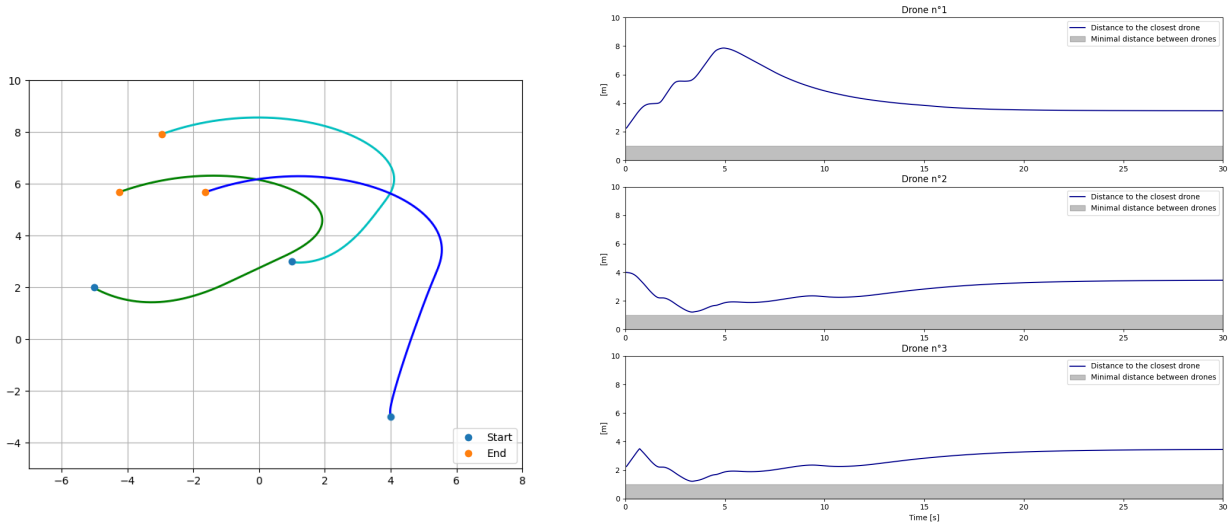
According to the paper [3], the saturation functions are now :

$$\begin{cases} h_1(X_1, X_2) &= -(k_1 + k_2) \cdot \dot{d_{02}} - k_1 k_2 \cdot (d_{02} - d_{0inf}^2) - D_1 \\ h_2(X_1, X_2) &= -k_3 \cdot \dot{d_{02}} - k_4 \cdot (\phi_\perp - d_{0inf} \cdot v_\perp^{\#}) - k_3 k_4 \cdot (d_{02} - d_{0inf}^2) - D_2 \end{cases}$$

Remembering that $\boldsymbol{v}_\perp^{\#}$ is the tuning gain of the orthoradial component of the avoidance. The saturated inputs are computed the same way as before (p.7).

Constant position data reception

Since we have one more level of integration in the dynamics, each drone needs to estimate the acceleration of the others, in addition to the velocity, for the collision avoidance computation. Fortunately, the **high-gain continuous-discrete time observer** of paper [4] already does it.



(a) Formation flying

(b) Collision avoidance

Figure 12: Second order dynamics - Constant position data reception

The Fig. 12 shows that the formation flying is working with the changed dynamics, so as the collision avoidance. Now we are able to test the efficiency of the approach when the data reception is intermittent. The maximum acceleration is set to 15 m.s^{-2}.

Intermittent position data reception

The goal is the same as before : keep each UAV out of others' presence zone. To do so, several strategies have been tested :

Strategy 1

Keep the minimal distance of the collision avoidance constant to the minimum, and play with the **formation pattern scale** to make the triangle inflate. That is the strategy used with the first order. This way, when the data reception is good, the drones avoid each other normally, and when it is bad, we count on the formation command to keep them far enough of the presence zones. Unfortunately, the system is not responsive enough, the gain is to small when the UAV is 1m from his desired position so it ends up in the zone.

Strategy 2

Make the **minimal distance of the collision avoidance** vary over time so the constraints force the UAV to stay out of the area. Since it doesn't know where are the others anymore, the drone keeps this minimal distance from the center of the presence zones. Once again, the UAV still ends up in the zone. The reason why the constraint is not satisfied is probably because the model of which we rely assumes that $\boldsymbol{d_{0inf}}$ is constant. Consequently, it is necessary to recalculate considering $\boldsymbol{d_{0inf}(t)}$.
This leads to adding the following terms respectively to h_1 and h_2 :

$$\begin{aligned} a_1 &= (\ddot{d_{0inf}}^2) + (k_1 + k_2) \cdot (\dot{d_{0inf}}^2) \\ a_2 &= (\dot{d_{0inf}}^2) \cdot v_{\perp}^{\#} + k_3 \cdot (\dot{d_{0inf}}^2) \end{aligned}$$

I didn't obtain good results while implementing these calculations in python. The UAVs, still infringed the constraints.

Strategy 3

I wanted to see the behaviour obtained when the drones react only when their distance to the nearest presence zone ($\boldsymbol{d_{sep}}$) falls below a certain limit. Since the zone is growing at constant speed, and we are working with 2 degres of integration, we should be able to express 2nd degree polynomial involving the acceleration. And with the sign of Δ, we should be able to find a minimum acceleration fleeing the area so there is no crossing. The instant t=0 is when $\boldsymbol{d_{sep}}$ falls below $\boldsymbol{d_{sep,min}}$.

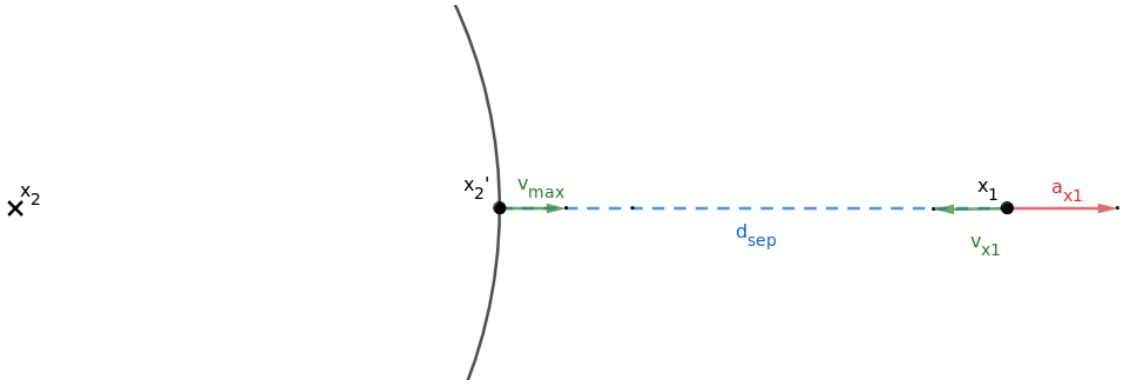


Figure 13: Diagram of the case only considering projections on the axis

We have :

$$\begin{aligned} &\ddot{x_1}(t) = a_{1,min} \\ \Rightarrow\ &\dot{x_1}(t) = a_{1,min} \cdot t + \dot{x_1}(0) \\ \Rightarrow\ &x_1(t) = a_{1,min} \cdot t^2 + dotx_1(0) \cdot t + x_1(0) \end{aligned}$$

We also have :

$$\begin{aligned} &\dot{x_2'}(t) = v_{max} \\ \Rightarrow\ &x_2'(t) = v_{max} \cdot t + x_2'(0) \end{aligned}$$

So :

$$\begin{aligned} d_{sep}(t) &= x_1(t) - x_2'(t) \\ d_{sep}(t) &= a_{1,min} \cdot t^2 + (\dot{x_1}(0) - v_{max}) \cdot t + x_1(0) - x_2'(0) \\ \Rightarrow \Delta &= (\dot{x_1}(0) - v_{max})^2 - 4 \cdot a_{1,min} \cdot (x_1(0) - x_2'(0)) \end{aligned}$$

We want $d_{sep} > 0 \Leftrightarrow \Delta < 0$, which gives us : $a_{1,min} = \frac{1}{4} \cdot \frac{(\dot{x_1}(0) - v_{max})^2}{x_1(0) - x_2'(0)}$.

To put this into practice, we need to project the velocity of the drone on the axis represented in Fig. 13. It has to flee the presence zones on the axis defined by the bisector of the segment connecting the center of the 2 zones. The Fig.14 show the result of this approach, the UAVs are much more responsive and they ensure distance from the zones without overshooting.

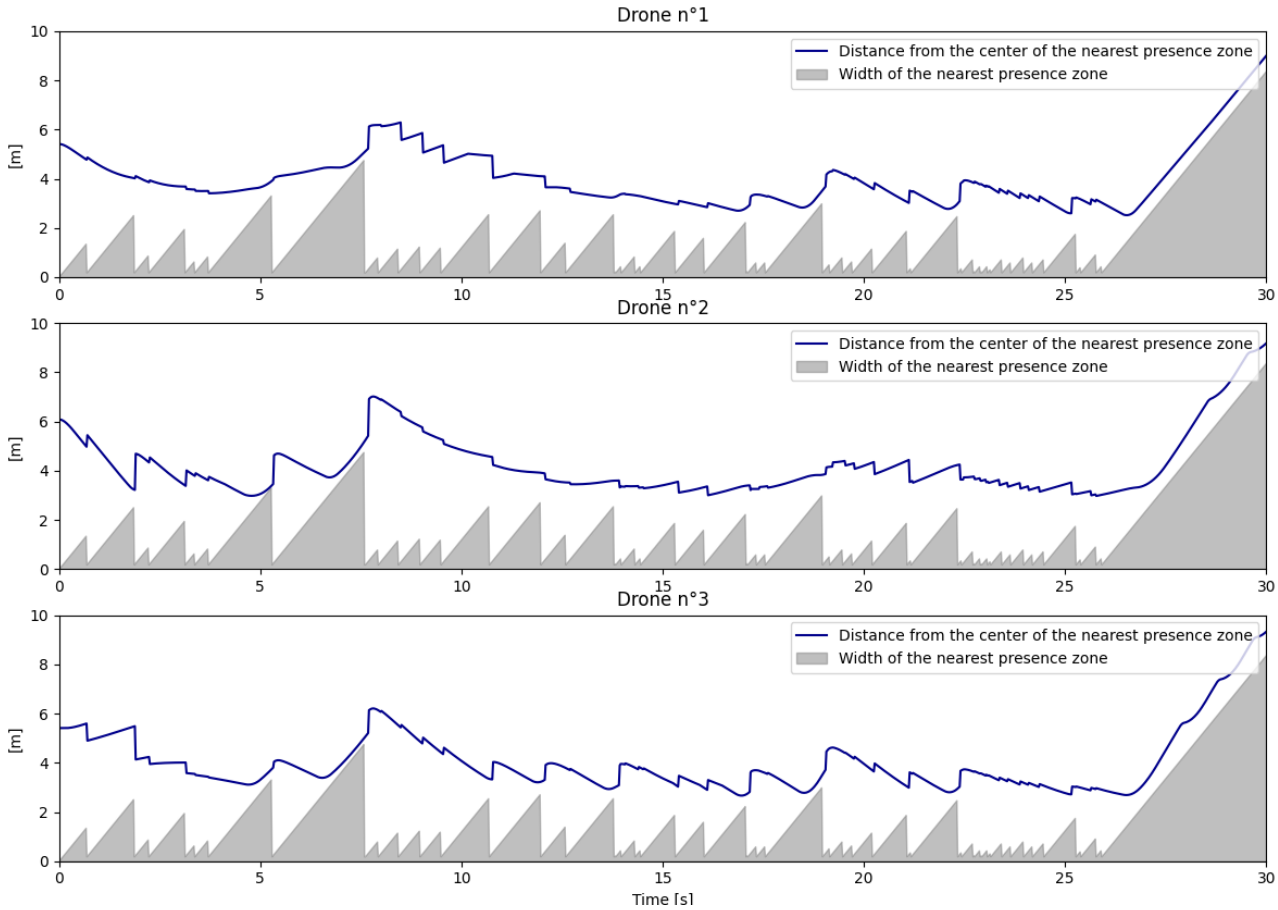


Figure 14: Distance between UAVs and the center of the nearest presence zone

Conclusion

Although a successful real flight has not been realised. Many issues have been tackled during this internship. The formation flying of the drone swarm and the collision avoidance work well with normal data reception. An appropriate method of estimation meets the sensor constraints. When it comes to intermittent position data reception, interval calculation has been used in a suitable way. The method is slightly more pessimistic than usual but fulfills the requirement for quick computation.

Several approaches have been tested to keep the UAVs outside the presence zones. Playing with the formation pattern scale works with velocity commands but becomes irrelevant with acceleration instructions. As the CBFs work well in normal conditions, it should be a solution to ensure the compliance with constraints when d_{0inf} is time-varying (as long as the bug I encountered can be found). The method computing the minimal acceleration required when the zone is too close is also a good lead (and the same can be done with velocity commands).

I have produced a manual describing how to install and use the MRS Gazebo Toolbox. I worked only with the example scripts of missions but they could be customized (add more drones, choose the starting point, automate the take-off and landing...) by looking the GitHub documentation and modifying the right files.

This internship provided me with an insight into academic research in the United States and the work of PhD student. I enjoyed implementing and testing the new theories that I read in papers thanks to the knowledge acquired in class. I had to work a lot autonomously with unfamiliar concepts so I've improved my ability to be self-taught.

References

[1] Laurent Burlion,S. Bertrand (?) *Drones Control and Coordination*, Rutgers in collaboration with Central Supelec and Université Paris-Saclay.

[2] Mrdjan Jankovic, Mario Santillo, Yan Wang (2022) *Multi-agent systems with CBF-based controllers – collision avoidance and liveness from instability*, Cornell University, EESS

[3] C. Chauffaut, Laurent Burlion (2015) *Collision Avoidance of multiple MAVs using a multiple Outputs to Input Saturation Technique*, ONERA - The French Aerospace Lab, Toulouse, France.

[4] Syed A. Ajwad , Emmanuel Moulay (2021) *Collision-Free Formation Tracking of Multi-Agent Systems Under Communication Constraints*, IEEE Control Systems Letters, Vol.5, No.4.

RAPPORT D'EVALUATION
ASSESSMENT REPORT

Merci de retourner ce rapport par courrier ou par voie électronique en fin du stage à :
At the end of the internship, please return this report via mail or email to:

ENSTA Bretagne – Bureau des stages - 2 rue François Verny - 29806 BREST cedex 9 – FRANCE
☎ 00.33 (0) 2.98.34.87.70 / stages@ensta-bretagne.fr

I - ORGANISME / *HOST ORGANISATION*

NOM / *Name* **Rutgers University, Mechanical and Aerospace Engineering Department**

Adresse / *Address* **98 Brett Road, Piscataway, NJ 08854, USA**

Tél / *Phone (including country and area code)* **+1-848-445-2046**

Nom du superviseur / *Name of internship supervisor*
Laurent Burlion
Fonction / *Function* **Assistant Professor**

Adresse e-mail / *E-mail address* **laurent.burlion@rutgers.edu**

Nom du stagiaire accueilli / *Name of intern* **Jules Le Gouallec**

II - EVALUATION / *ASSESSMENT*

Veuillez attribuer une note, en encerclant la lettre appropriée, pour chacune des caractéristiques suivantes. Cette note devra se situer entre **A (très bien)** et **F (très faible)**
*Please attribute a mark from **A (excellent)** to **F (very weak).***

MISSION / *TASK*

- La mission de départ a-t-elle été remplie ? (A) B C D E F
  Was the initial contract carried out to your satisfaction?

- Manquait-il au stagiaire des connaissances ? ☐ oui/*yes* ☒ non/*no*
  Was the intern lacking skills?

  Si oui, lesquelles ? / *If so, which skills?* _____

ESPRIT D'EQUIPE / *TEAM SPIRIT*

- Le stagiaire s'est-il bien intégré dans l'organisme d'accueil (disponible, sérieux, s'est adapté au travail en groupe) / *Did the intern easily integrate the host organisation? (flexible, conscientious, adapted to team work)*

  (A) B C D E F

Souhaitez-vous nous faire part d'observations ou suggestions ? / *If you wish to comment or make a suggestion, please do so here*_____

COMPORTEMENT AU TRAVAIL / *BEHAVIOUR TOWARDS WORK*

Le comportement du stagiaire était-il conforme à vos attentes (Ponctuel, ordonné, respectueux, soucieux de participer et d'acquérir de nouvelles connaissances) ?

Did the intern live up to expectations? (*Punctual, methodical, responsive to management instructions, attentive to quality, concerned with acquiring new skills)?*

(A) B C D E F

Souhaitez-vous nous faire part d'observations ou suggestions ? / *If you wish to comment or make a suggestion, please do so here* _____

_____

INITIATIVE – AUTONOMIE / *INITIATIVE – AUTONOMY*

Le stagiaire s'est –il rapidement adapté à de nouvelles situations ? (A) B C D E F
(Proposition de solutions aux problèmes rencontrés, autonomie dans le travail, etc.)

Did the intern adapt well to new situations? (A) B C D E F
(eg. suggested solutions to problems encountered, demonstrated autonomy in his/her job, etc.)

Souhaitez-vous nous faire part d'observations ou suggestions ? / *If you wish to comment or make a suggestion, please do so here* _____

_____

CULTUREL – COMMUNICATION / *CULTURAL – COMMUNICATION*

Le stagiaire était-il ouvert, d'une manière générale, à la communication ? (A) B C D E F
Was the intern open to listening and expressing himself /herself?

Souhaitez-vous nous faire part d'observations ou suggestions ? / *If you wish to comment or make a suggestion, please do so here* _____

_____

OPINION GLOBALE / *OVERALL ASSESSMENT*

❖ La valeur technique du stagiaire était : (A) B C D E F
Please evaluate the technical skills of the intern:

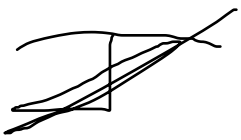
III - PARTENARIAT FUTUR / *FUTURE PARTNERSHIP*

❖ Etes-vous prêt à accueillir un autre stagiaire l'an prochain ?

Would you be willing to host another intern next year? [X] oui/*yes* [ ] non/*no*

Fait à **Piscataway, NJ, USA**, le **28/09/23**
In _____, on _____

Signature Entreprise _____ Signature stagiaire
Company stamp _____ *Intern's signature*



Merci pour votre coopération
We thank you very much for your cooperation