Internship Report

Quadcopter Swarm - Indoor positioning, control and obstacle avoidance

---

Pierre Perraud

p4perrau@enib

16th March 2020 - 14th August 2020

Supervisor :
Luis Mejias Alvarez

Referring teacher :
Cédric Buche

Abstract

This report will present the work done during my final internship for my engineering course at the National Engineering School of Brest (ENIB) and my Intelligent, Interactive and Autonomous Systems (SIIA) master degree in computer science at the University of Western Brittany (UBO). This internship took place at the Queensland University of Technology (QUT) in Brisbane

The subject of my work was to research and develop algorithms allowing the control of a swarm composed of quadcopter called crazyflies using an indoor positioning system based on the UltraWide Band technology. The swarm would have to go from waypoints to waypoints avoiding obstacles on the way.

Introductions

With the progress in the robotic and computer programming fields, the human can create robots smaller, smarter and more powerful than ever before. They will most likely be used in indoor situations enabling access to areas where humans could maybe not go as fast as drones or not go at all such as a building was gas leak, where sending drones in recognition could prevent humans from taking risks.

During my work at QUT I tried to help advance the research in the positioning of indoor flights for Unmanned Aerial Vehicles first by exploring the use of the UltraWideBand technology for positioning drones and by developing tools to move the drones while avoiding obstacles.

Queensland University of Technology

The University :

The University is located in Brisbane, Queensland in Australia and was created in 1989 after the merger of Queensland Institute of Technology and the College of Advanced Education. The buildings are separate in two campuses in Brisbane : Gardens Point and Kelvin Grove. Gardens Point hosts the faculties of Business, Law and Science and Engineering while Kelvin Grove hosts the faculties of Creative Industries, Health and Education.

QUT is ranked in the top 10 Australian universities and in the top 1% universities worldwide.

Fig 1 : aerial view of QUT Gardens Point campus

The 11th floor :

My internship took place in the science building of the Gardens Point campus. The 11th floor of this building hosts the robotic research laboratory of this building. Lot of professors, doctors, Phd students, engineers and master students work on various projects covering artificial intelligence, computer vision, unmanned aerial vehicles etc…

Work Places

Open space :

I was working on one of the open spaces on the 11th floor. For the few days I worked their, I was mostly working on my laptop at my desk, reading articles and programming. I also had access to a small test room (3m by 3m of possible fly area) where I could fly the crazyflies in order to test them.

Working from home :

Due to the global pandemic, the Queensland state instituted measures to restrict the spread of the COVID-19 virus and therefore a lockdown was imposed. The university had no choice than to send all the non-essential workers to their home, including me. All these measures were taken only one week after I started working... I was at that time staying in an AirBnb while looking for a place to stay for a longer time. The room I had was not the best place to work since the house was quite noisy and had a very small desk. Moreover I could not test the positioning system of the drones in this area. So most of the work I achieved in this period was research and a little bit of programming.

This situation lasted for weeks as I struggled to find a place to stay for an extended period… The lockdown was frightening everyone who wanted to rent a place, and the fact that I was French was even worse because at this time France had a lot of people infected with the virus. Therefore most of the time when I was asking for a visit, I was refused because they were afraid I would contaminate them…

During the month of may, I finally found a place I could stay until the end of my internship. I had a room in a flat that was quiet and with enough place to work. Every week I had a meeting with my supervisor through Zoom to explain the work I have done and what would be the next steps.

Crazyflies

The crazyflies are nano-quadcopters developed by the Swedish company Bitcraze. Their size, weight and robustness are a great benefit for indoor flights and that is part of the reason that they are widely used for uav research purposes. They also have the advantage of having full control on the onboard and offboard software with the complete open source library available on Github. The onboard software is developed in C language whereas the offboard uses python but tools allow development in Java, Ruby, C/C++, C# and Javascript.

Figure 2 : Crazyflie 2.0

The Crazyflie is composed of a mainboard that holds the microcontrollers, the usb connector and power management unit. The motor, propellers, battery and landing legs are all removable to make it easier to replace if it's damaged. The 3.7 V battery with a capacity of 250mA allows the drone to fly for approximately seven minutes without payload.
In addition to those parts, it is possible to expand the capacity of the Crazyflies using the two GPIO connectors available on the main board. Bitcraze sells various extensions accessories such as : LED-ring deck, UWB deck, ultrasonic sensors etc… But since the software is open source, it is possible to create a new one that can be connected through the GPIO connectors.

To communicate, the Crazyflies use the nRF51822 microcontroller that allows the drones to use Bluetooth and 2.4 GHz radio frequencies. Since each drone is equipped with this chip, they can transmit information to any other drone if they are connected on the same channel.

A USB-radio dongle called CrazyRadio is also available to connect the drones to other devices, like a computer allowing commands to be sent to the Crazyflies.

The Crazyflies have lot of advantages for research purposes thanks to to their specificities :

- They are repairable : the spare parts allow easy reparations if something breaks
- They are modifiable : the GPIO connectors is great if features must be added to the drone
- They allow measures : thanks to the onboard radio, it is possible to transmit the data collected either from the integrated captors or from the expansion deck.

Thanks to these advantages, the Crazyflies are vastly used in universities around the world.

Literature review

In order to prepare for this internship, we were asked to do some bibliographic research. This allowed me to have a better idea of what was already achieved in this field. I also did some research in the beginning of my internship.

Positioning

Most of the modern drones include an IMU (Inertial Measurement Unit) on their board. This unit allows the drone to have information about its position and speed. Most of the IMUs are composed of gyrometers and accelerometers that respectively give angular speeds and accelerations of the drone. These values can then be derived to obtain estimations of positions and inclinations.

As seen in the article of Milton C. P. Santos[1], using only the IMU is for most of the time not enough. Indeed since the values are derived, errors can appear. A solution discussed in the article is to merge these values with others obtained by another captor using an Extended Kalman Filter (EKF). The same method is also discussed in the Master thesis of F. Poirier[2] in which he explains how to recover the positions of the Crazyflie using live video of the drone. In his method, each drone is equipped with a paper target which allows it to compute its localisation thanks to the open source method Whycon. Then the data from the IMU and the Whycon are merged using the EKF.

Fig 3 Screenshot of the Live video use by the Whycon system

Other positioning methods exist like the Ultra Wide Band (UWB) system. It's one of the more popular with the Crazyflies since the company Bitcraze is selling its own positioning system using UWB : the loco positioning system (LPS). The LPS has many advantages :

- its responsivity : Using electromagnetic wave allow the signal to travel very fast
- its wide frequency band : reduce the possibility of interferences
- its obstacle resistance : the transmission is resistant to obstacles where a camera based one would lose the signal if in sighline.

The LPS is composed of an amovible deck that can be plugged on the GPIO connector of the Crazyflie, and some anchors that must be placed and configured before flying. Each anchor must have a known position in order to have a working positioning system.

Swarm

A very famous way of moving a multi-agent system is to take inspiration from nature, an example would be the displacement of the bird moving in a flock. C. Reynold[3] described a model that could implemented by giving only three rules to each individual :

- collision avoidance : each individual will avoid every other that would be to close
- staying close to each other : each individual will not go too far from the other one
- alignement : each individual tends to go in the same direction as the other one.

This kind of behaviour can easily be set up once the communication between each individual is organised.

Effective Work

Hardware and available tools

For my internship I was given some material to achieve my research :

- 3 Crazyflies
- spare motor, landing legs and propellers
- 3 batteries and charging module
- 3 LPS deck
- 6 LPS anchors
- 2 CrazyRadio

Unfortunately, the university could not provide me with a computer, so I had to work on my own laptop.

Once the Crazyflie is assembled, one of the easiest ways to start it is by using the nRF51822 chip which allows the control of the drone with any smartphone equipped with bluetooth and the Crazyflie Client application. This application acts as a controller and the drone can fly using the “joysticks”. It is very simple, but it does not allow programming or positioning estimation.

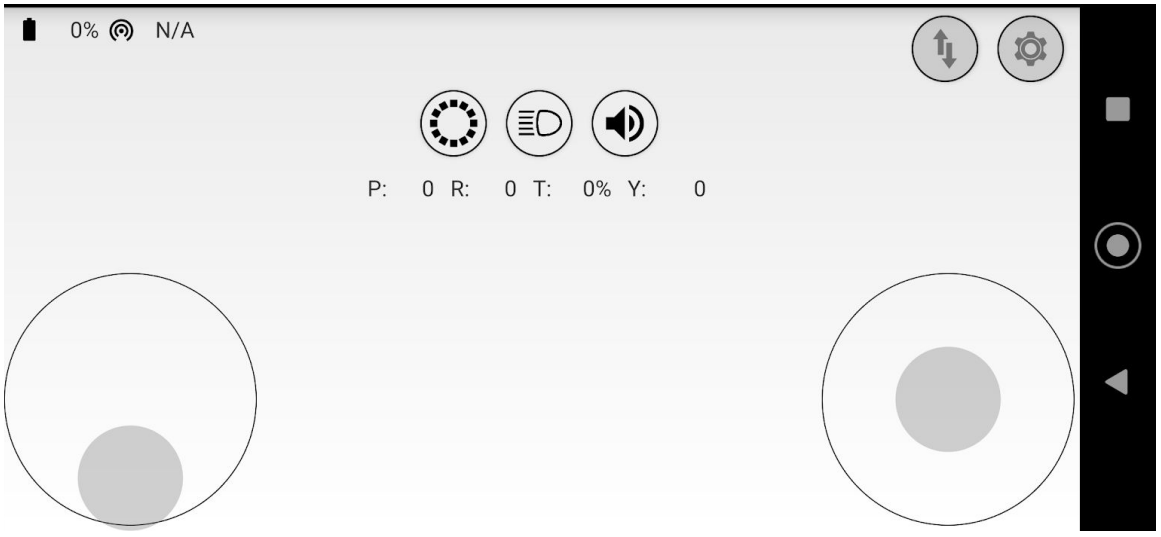


Fig 4 , screenShot of the android application Crazyflie Client

The Bitcraze company provides a complete virtual machine with all the developpement tools to start developing using the software VirtualBox. Even though it was very complete, I decided to install all the development tools on my personal linux distribution to optimise the performance of my software.

In order to start developing you have to install some tools :

- The last version of Python3 and pip3. Since most of the scripts use Python3, you have to make sure you use the updated version.

- The Crazyflie Client. It's a software developed by Bitcraze that enables you to monitor and configure the Crazyflies. You can update the firmware of the onboard chip of the drone using this client. You can also configure each drone's radio address.

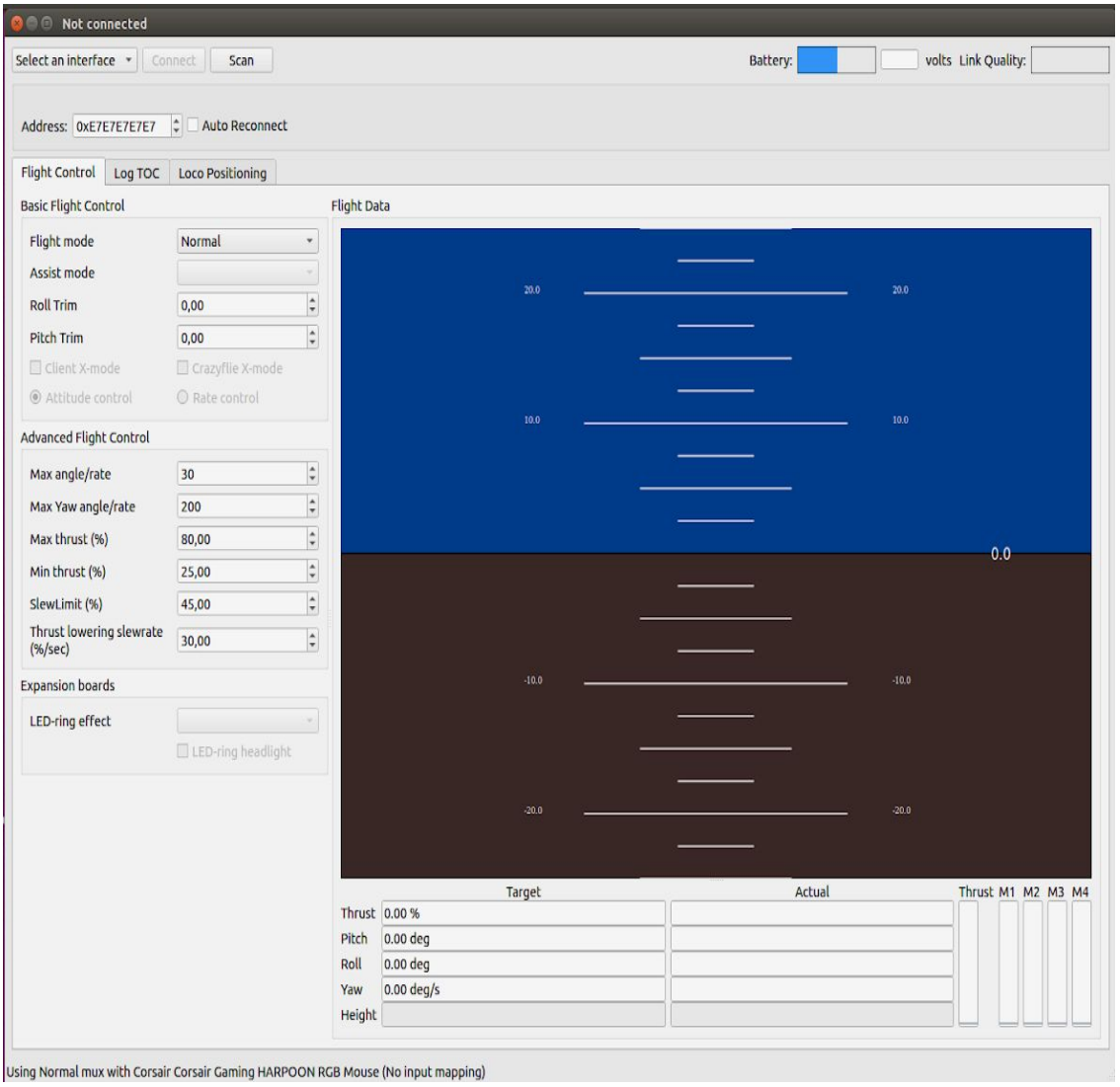


Fig 5 : Screenshot of the Crazyflie Client (linux)

- The Crazyflie library : the instructions to install this library are available on the Bitcraze Github repository. If you have installed the version of python3 and pip3, you should not encounter troubles while installing this library.

This library gives you all the tools to start developing using python scripting, you just have to include the library in you script :

```
import cflib
from cflib.crazyflie import Crazyflie
```

Even if there are many examples provided in the Github repository, Bitcraze does not give instructions or documentation on the library. So you have to look into the description of each function and read every example to understand how to use crazyflieLib. This takes a long time and many tests to understand how to create a working script.

UWB with the Loco Positioning System

Once you start to understand how to program the drone you have to look into the Loco Positioning System since making the drone fly without a working positioning system will most likely end in a crash…

As for the Crazyflie onboard firmware, you can update the anchor firmware through the Crazyflie Client. All the instructions are given on the Bitcraze website.

Once each anchor is updated, you have to set them in your room at known positions. The more precise you are with the anchor positioning, the more precise the drone position will be. The anchors must be positioned as far as possible from each other.

Once the position of the anchors are measured, you have to enter them using the Crazyflie Client through the Crazy radio.

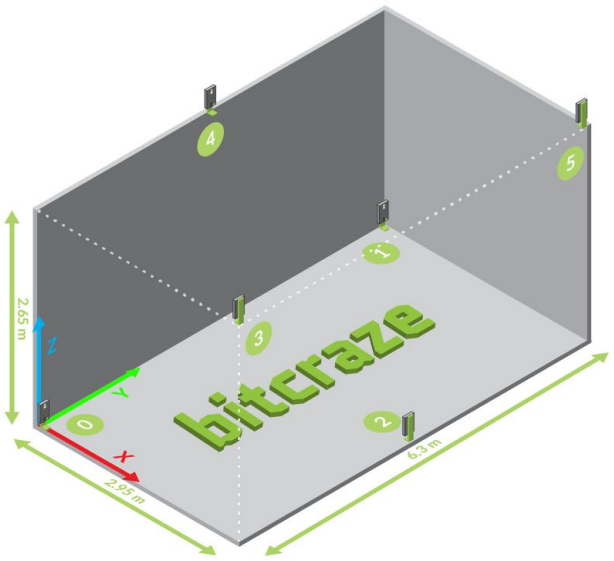


Fig 6 : example of anchor positioning on Bitcraze website

If everything is done well, positioning with the LPS should be available. you can check every position using the Crazyflie Client, and access the “Loco Positioning” tab. The Crazyflies automatically include the LPS data to the Extended Kalman Filter. The anchors are shown in green, and the drones in blue.

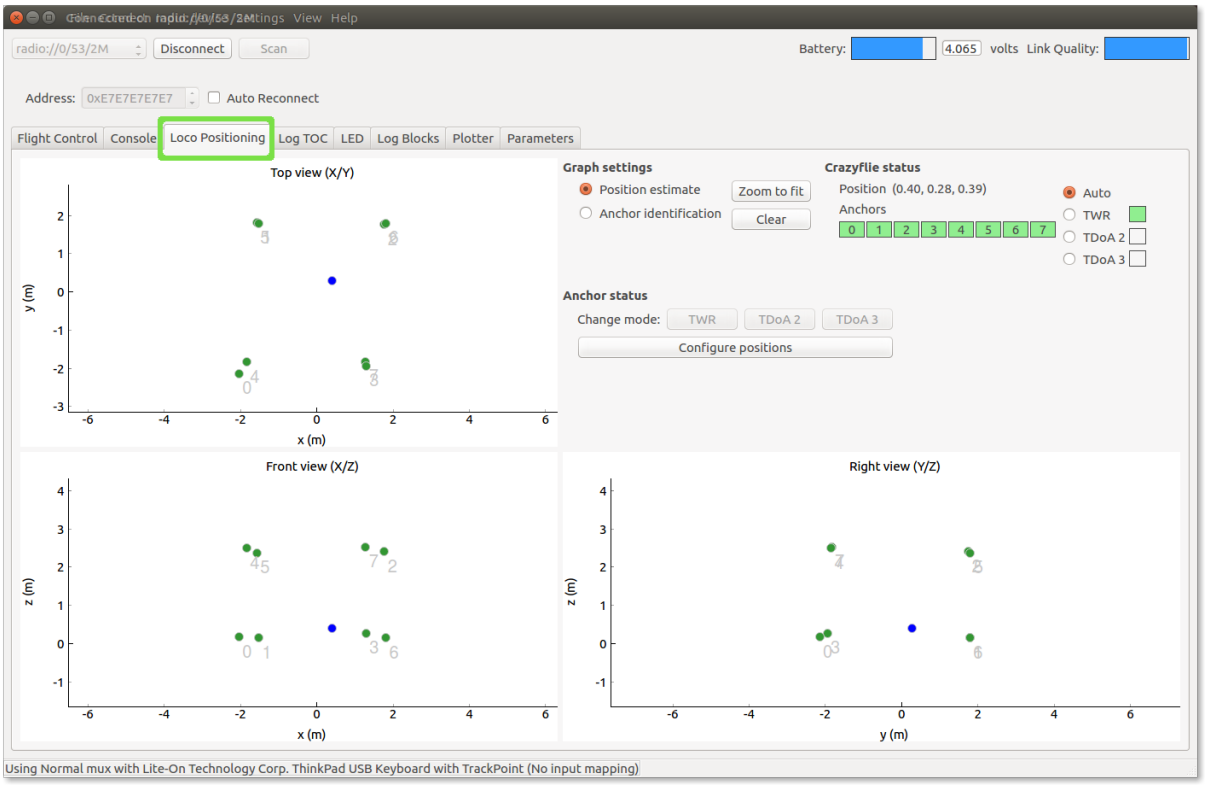


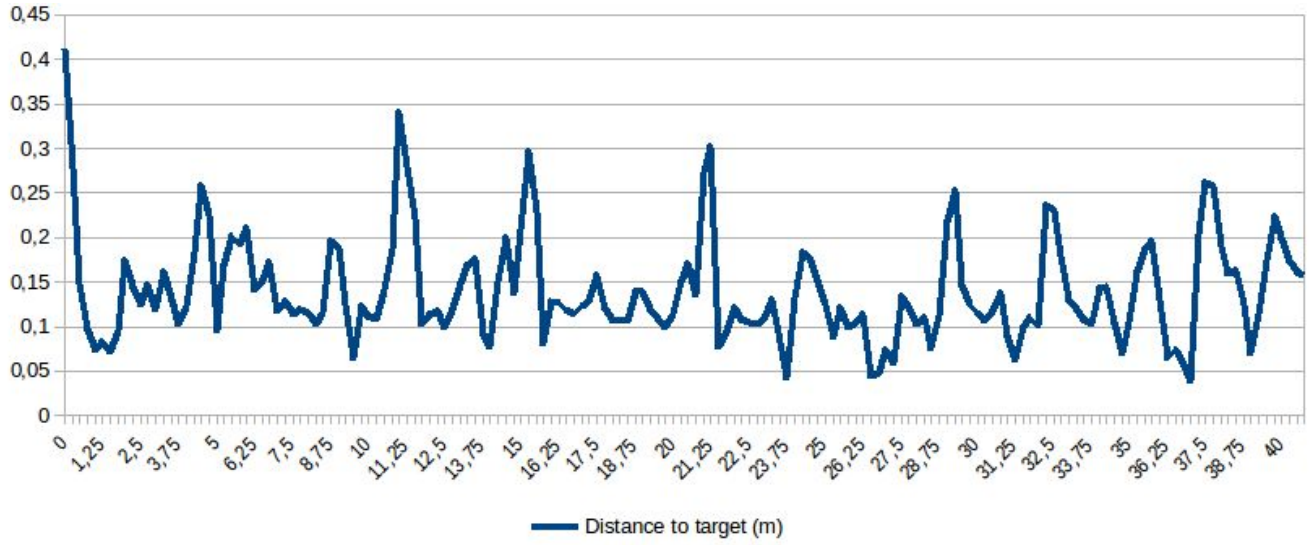
Fig 7 : Loco Positioning Tab with drone and anchors

The LPS can be configured in three mode :

- Two-Way Ranging (TWR): the most precise but it only allows one drone to fly at the time.
- Time-Distance-of-Arrival 2 (TDoA2): This mode allows multiple drones to fly at the same time, but has less precision than the TWR.
- Time-Distance-of-Arrival 3 (TDoA3): Improvement of the TDoA2 but is still experimental.

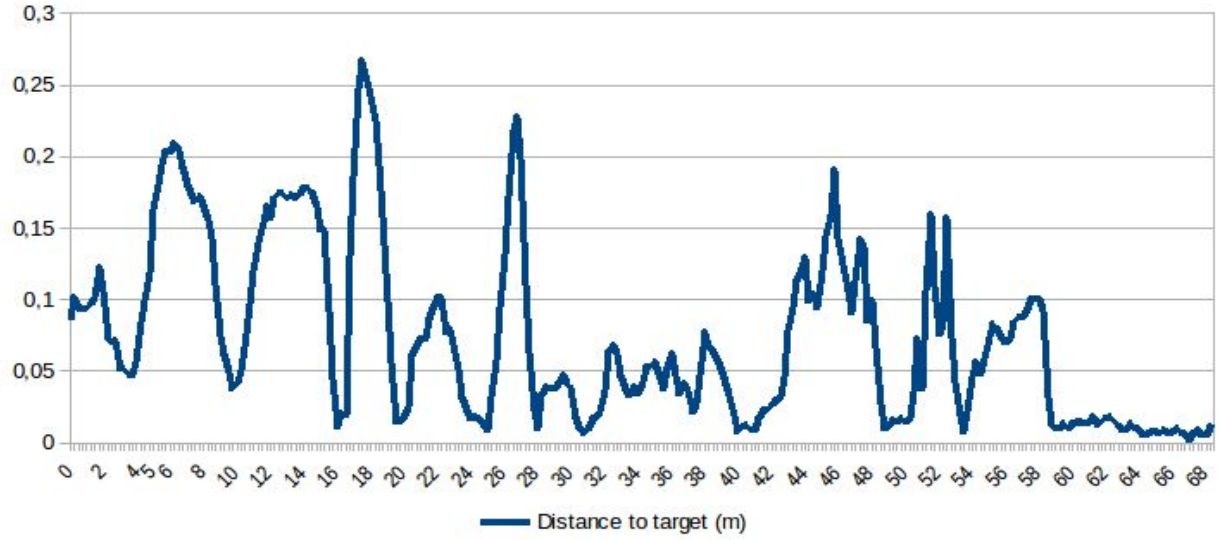
Once I found a flat I could stay and work in I started to install a setup with the UWB anchors I had. I installed the 6 anchors I had in my room following as close as possible the instruction given by Bitcraze. I place 4 of them in high places and 2 others in lower positions.
When I started to make the drone fly, it could not hold precisely its position, it was constantly drifting even though I was using the most precise method of the LPS : TWR. I checked many times the position of the anchors but the result was the same.

I logged the position of the drone while it was flying and this is the graph I obtained :



This graph shows the euclidean distance of the drone from the target I assigned it to fly to. A difference of 10cm would be acceptable for flying a swarm but with so much error in the positioning, collisions would occur between the drones.
On this graph, the first pike is due to the drone taking off. More data from these logs can be found in Appendix 1.

I also logged the position of the drone while it was just standing on the ground :



In this graph the drone is standing and the ground and the “target” is actually the position the drone was standing on. More data from these logs can be found in Appendix 2.

I repeated this process many times to check and the results were identical everytime.

So in view of the results my conclusion was the LPS was suffering from disturbance I could not identify that would not allow me to make the drone fly safely. So I started looking for other ways to continue my work.

Crazyswarm

During my initial research about the Crazyflies, I found a project called the “CrazySwarm”[4] that was working with the Robot Operating System (ROS). This project has an option to simulate the flight of the Crazyflies. The ROS project is very efficient and makes an easy transition between real flight and simulation. It supports different positioning systems such as Vicon, OptiTrack, Qualisys and UWB and could switch from one project to another by just changing one parameter.

This project gives very clear instructions of the installation and a complete documentation on how to use the python script. While very well explained, it took me many days to install the ROS project with all its components working all together and another few days to understand how to use it.

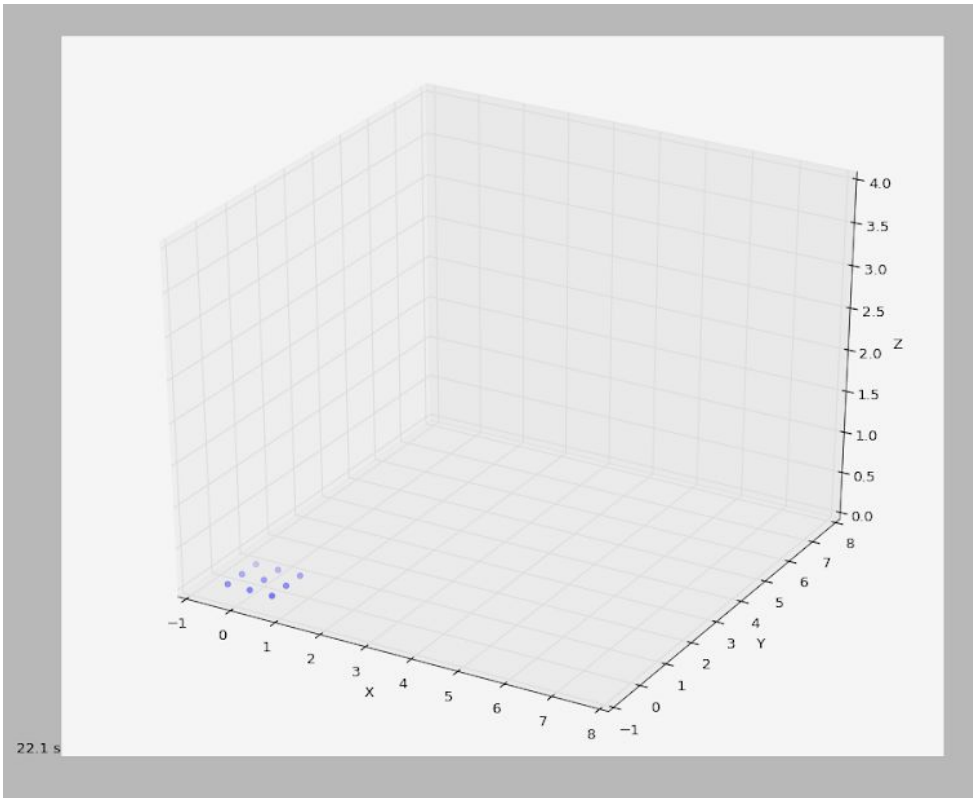


Fig 8 : Example of simulation using matplotlib. Each blue dot is a drone

Avoidance Algorithm

Once I got my hands on the Crazyswarm project, I started moving the drone in the simulation. Starting with one drone, I made it go from waypoint to waypoint in straight lines. I started to add one obstacle and when the drone was close to the obstacle, it would change its trajectory from a straight line to a circular trajectory around the obstacle.

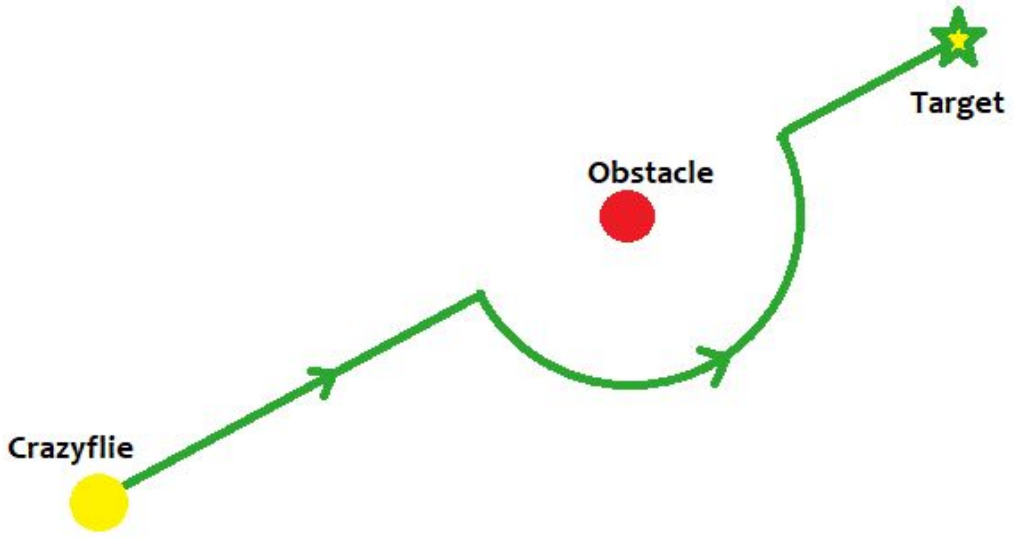


Fig 9 : example of circular trajectory around one obstacle.

This simple script worked fine with one drone and one Obstacle, but would not be efficient when adding more obstacles so I started working on a different avoidance mechanism. I took inspiration from Reynold's work [3]and the Boids algorithm and started working with vectors for my script. So each timestep :

- the drone would be attracted to the position of the target
- if close to the obstacle, it would try to go in the opposite direction in which the obstacle is.

The results were quite similar to the previous trajectory, but adding more obstacles would be easier.

Swarm Control

I then added more Crazyflies in the simulation. Doing so, I created a Planner class that would manage the swarm and contain every Crazyflies class of the simulation. The swarm would have a target position and each drone a relative position in the swarm. So the absolute target position of the drone would aim for would be the swarm target position plus its relative position. When a drone is close enough to its target position, it will stop to move and wait for the other drones to arrive. When all drones have arrived, the swarm will go to the next waypoint, and if there are no more, the drones land.

Since I added more drones I had to avoid collisions with the other members of the swarm. So I again took inspiration in the Boids algorithm and added a repulsion factor if drones were to close from each other. I also added the alignment of the trajectories of the swarm's member, and a tendency to keep his previous direction to avoid abrupt direction changes.
Finally, since this script is supposed to be executed for indoor flights, I added boundaries in the simulation to simulate the walls.

```
movementVector =\
   self._prevDirectionFactor       *   self._prevDirection +\
   self._neighboursDirectionFactor *   neighboursDirectionVector +\
   self._targetAttractionFactor    *   targetAttractionVector +\
   self._neighboursAttractionFactor*   neighboursAttractionVector -\
   self._neighboursAvoidanceFactor *   neighboursAvoidanceVector -\
   self._obstaclesAvoidanceFactor  *   obstaclesAvoidanceVector  -\
   self._wallAvoidanceFactor       *   wallAvoidanceVector
```

Fig 10 : vector calculated for each drone at a frequency of 100Hz

Now that I had a good working base I added more features to the python script :

- A simple control panel from which I could change the factor of the different vectors, even during the swarm flight
- A trace of each drone's path in the MatplotLib simulation
- A more realistic simulation using Vispy instead of MatplotLib allowing collision visualisation
- A csv reading of the mission waypoints, obstacles, wallpositions, swarm formations. This allows us to test different mission configurations using the same script.

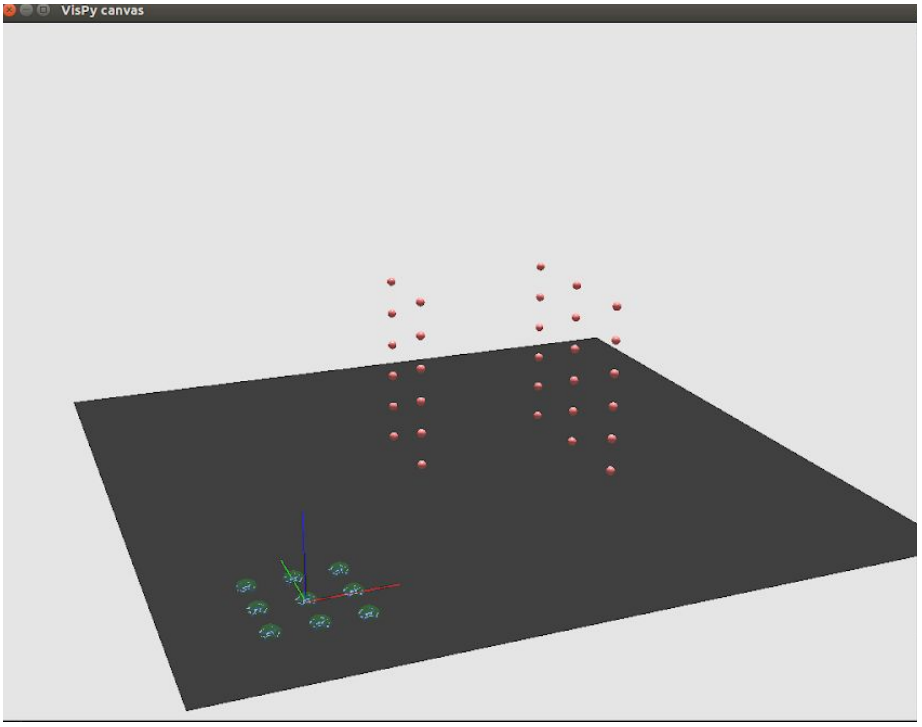


Fig 10 : Simulation environment using Vispy

Using this new feature I could evaluate the importance of each factor in the comportement of the swarm is general.

If the factors are not balanced well, the swarm will not behave well, or can collide with something.

Factor	Too weak	Too strong
Previous Direction	The drone can change direction abruptly	The drone can take too much time to turn and never achieve its goal
Neighbours Direction	The drone can change direction abruptly	The swarm can take too much time to turn and never achieve its goal
Target Attraction	The drone will nor try to go to the target	The drone may collide with some obstacles one the way to the target
Neighbours Attraction	The swarm will most likely split	Each drone may collide with the others
Neighbours Avoidance	Each drone may collide with the others	The swarm may have difficulties to move
Obstacles Avoidance	The drone may collide with obstacles	The swarm may not arrive to the target
Wall Avoidance	The drone may collide with the walls	The swarm may not arrive to the target

Table 1 : Risks of having factors too strong of too weak.

Synthesis

Problems encountered

During this internship, most of the difficulties I encountered were due to the global pandemic situation and all the consequences that came with is.
I could only stay in the university for one week before I was sent working from the Airbnb I was staying at. It was not a great place to work and I had a lot of difficulties focusing in this place because of the noise and small desk I had in the room. Finding another place to stay took me a long time and was very harrasing.
One of the difficulties I faced was the lack of programming documentation for the Crazyflie python librairie. It took me sometimes an hour to find the function I was looking for, just by reading exemples and the source code.
The main problem I faced was not being able to work with the LPS. Even after hours of research and many tries I could not manage to make the swarm fly in my room. I was really frustrated at not being able to test the script I wrote in real conditions.

Possible improvements

Even though my script won't work with the UWB positioning system, I'm convinced that if the localisation was precise enough, it would work. I haven't had the opportunity to try it with a tracking system, but since the positioning of my script is handled by a ROS node accepting other positioning hardware such as Vicon or OptiTrack, it would theoretically be possible to add tracking devices on the Crazyflies and make them fly using my work.

Another way to improve the script would be to handle the situation where the swarm can be blocked because of the obstacle layout. This could be possible using a pathfinding algorithm like the A*. It could also be interesting to train the swarm, for example with a neural network, to find the optimal factors for the swarm, or to get out of blocking situations.

Conclusion

This internship at QUT was a great opportunity for me to start working in the research field. The subject that was assigned to me was really interesting and I learned a lot while working on it. It allowed me to improve my knowledge on the quadcopters and develop new ones on the positioning systems and on many other robotics fields. It was also my first experience in a company where French is not the main language. Even though I had difficulties from time to time, I always managed to understand and be understood.

Even though I did not achieve making the drones fly in my room using the UWB system, I was still able to find a solution in order to keep working on the algorithms to control the swarm and with the simulation tools I used, I achieved some results. There is still room for improvement for this project with the possibility to extend the project to artificial intelligence for example. I am frustrated that the global situation restrained the possibilities during this internship, I would have liked to do much more work during my time at QUT. Nevertheless I hope my contribution will help others advance the research state in this field where there is still plenty to do.

Bibliography

[1] M. C. P. Santos, L. V. Santana, M. M. Martins, A. S. Brandao, et M. Sarcinelli-Filho, « Estimating and controlling UAV position using RGB-D/IMU data fusion with decentralized information/Kalman filter », in *2015 IEEE International Conference on Industrial Technology (ICIT)*, Seville, mars 2015, p. 232‑239, doi: 10.1109/ICIT.2015.7125104.

[2] « Fabrice et al. - Control of the quadcopter Crazyflie.pdf ».

[3] C. W. Reynolds, « Flocks, Herds, and Schools: A Distributed Behavioral Model », p. 21.

[4] J. A. Preiss, W. Honig, G. S. Sukhatme, et N. Ayanian, « Crazyswarm: A large nano-quadcopter swarm », in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, Singapore, Singapore, mai 2017, p. 3299‑3304, doi: 10.1109/ICRA.2017.7989376.

Appendices

Appendix 1 :

	X (m)	Y (m)	Z (m)	Deviation X (m)	Deviation Y (m)	Deviation Z (m)	Distance (m)
Target	**1,8**	**1,3**	**1,3**				
average	**1,7951686832**	**1,328842962**	**1,1840315585**	**-0,0048313168**	**0,028842962**	**-0,1159684415**	**0,2312982365**
min	**1,6070295572**	**1,0364097357**	**-0,4592695832**	**-0,1929704428**	**-0,2635902643**	**-1,7592695832**	**0,0081442927**
max	**2,3196315765**	**1,4888936281**	**1,4223293066**	**0,5196315765**	**0,1888936281**	**0,1223293066**	**1,7744320255**
delta	**0,7126020193**	**0,4524838924**	**1,8815988898**	**0,7126020193**	**0,4524838924**	**1,8815988898**	**1,7662877328**
stdev	**0,1121416736**	**0,0934529974**	**0,4123845687**	**0,1121416736**	**0,0934529974**	**0,4123845687**	**0,3896517107**

data analysis for one flying drone using UWB positioning

Appendix 2 :

	X (m)	Y (m)	Z (m)	Deviation X (m)	Deviation Y (m)	Deviation Z (m)	Distance (m)
Target	**1,810018969**	**1,0437290534**	**0,7380662229**				
average	**1,810018969**	**1,0437290534**	**0,7380662229**	**1,69561334670024E-17**	**-2,74527875180039E-17**	**0,0042360828**	**0,0964519339**
min	**1,7322118282**	**0,789244473**	**0,7200139761**	**-0,0778071408**	**-0,2544845804**	**-0,0151005056**	**0,0060398155**
max	**1,8684304953**	**1,2473769188**	**0,7605122924**	**0,0584115262**	**0,2036478654**	**0,0224460695**	**0,2666135113**
delta	**0,136218667**	**0,4581324458**	**0,040498316 3**	**0,136218667**	**0,4581324458**	**0,0375465751**	**0,2605736958**
stdev	**0,0251899633**	**0,0926523217**	**0,0084236588**	**0,0251899633**	**0,0926523217**	**0,0084236588**	**0,0671031892**

data analysis for one drone not moving using UWB positioning