Rapport de projet

Informations générales

Nom du projet : Drones Defense Hackathon

Équipe :

- Dimitri NIRPOT
- Clara LAROUDIE
- Nawel DAOUDI
- Alexandre MINVIELLE
- Matheo RAULT
- Dorian PASSARET

Encadrant : Fouad KHENFRI

Lien GitHub : https://github.com/Dorian-PSRT/TCI_AERO/tree/main

Table des matières

1. Introduction

Avec notre participation au drone défense hackathon, qui a eu lieu entre le 24 octobre et le 24 novembre (phase préliminaire incluse), nous avons dû nous pencher sur la création d'un scénario ainsi que la programmation d'un essaim de drone collaboratif capable de le réaliser. Ce scénario devait nous permettre de démontrer le côté collaboratif de notre essaim, ainsi que d'être le plus complet possible pour avoir une chance d'être sélectionné par le jury. Dans notre cas, le scénario était le suivant :

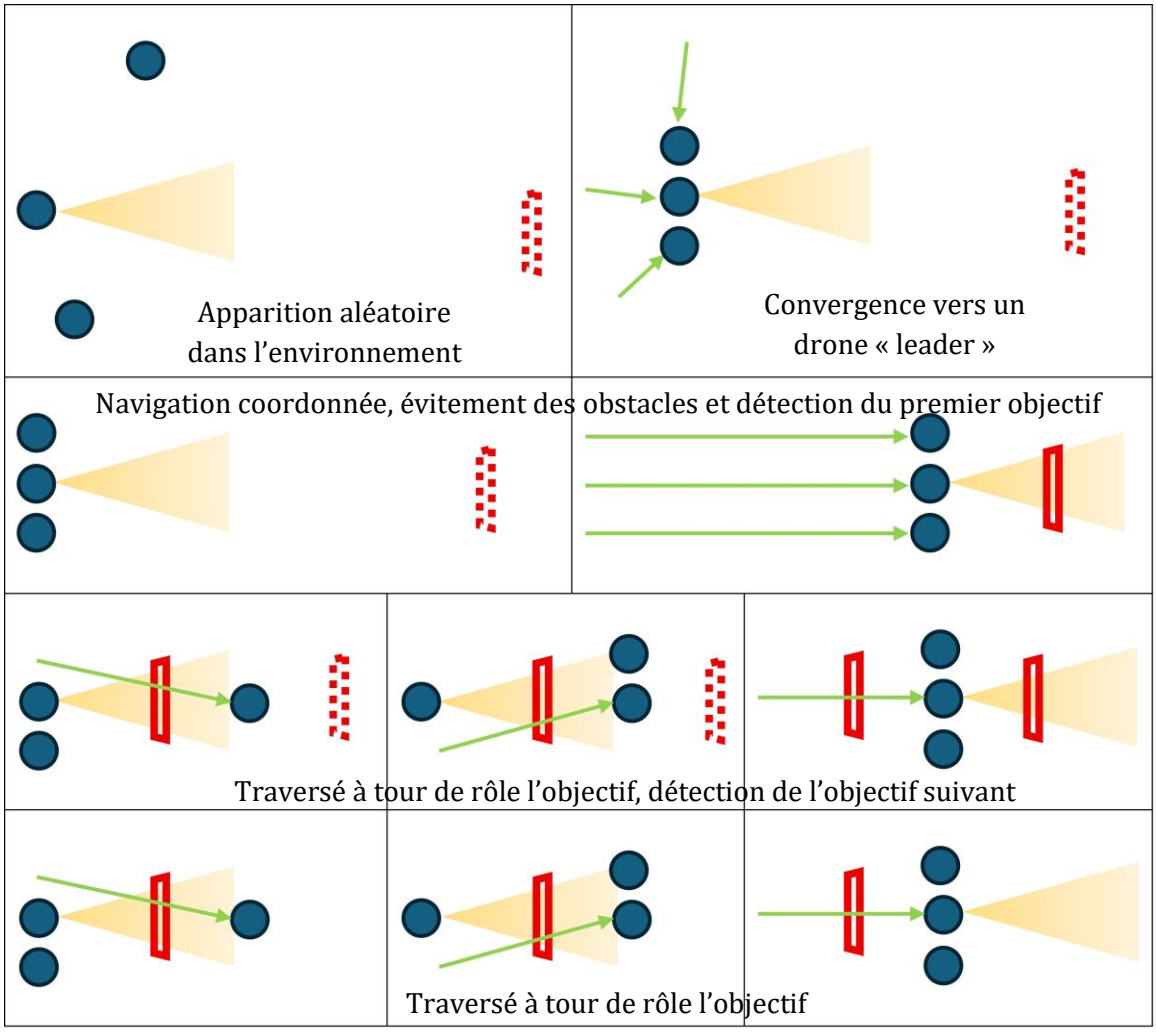

Figure 1: Scénario de la mission collaborative de l'essaim de drones

Ce scénario a été mis en place avec l'idée d'une introduction et d'une navigation au sein d'un bâtiment. Il a été sélectionné car il comporte plusieurs phases que nous trouvons

pertinentes dans un cadre opérationnel, en plus du scénario global qui peut être utile dans un cadre d'intervention militaire/policière ou de la recherche de victime en milieu sinistré. La première est la navigation coordonnée, au sein d'un champ d'obstacle et la convergence de l'essaim vers un objectif, ici une fenêtre. La seconde est la détection précise de l'objectif en question et de traverser ce dernier. Enfin, repérage de la prochaine ouverture et traversée de l'essaim de cette dernière. A noter que, à chaque ouverture, un drone doit rester en surveillance, à l'image du « petit poucet ».

Pour réaliser ce scénario, nous avons choisi d'utiliser ROS2 afin d'avoir une afin d'obtenir une architecture implémentable sur drone réel. Utiliser ROS2 nous permettait également de développer nos connaissances sur ce système méta-système d'exploitation très utilisé dans l'industrie.

2. État de l'art

2.1 Architectures de contrôle d'essaims

Les systèmes d'essaims de drones sont divisés en deux principales catégories, les essaims centralisés et décentralisés. Les architectures centralisées reposent sur une station de calcule unique recevant la position de tous les drones et envoie en retour les trajectoires à suivre pour chacun d'entre eux. Bien qu'efficaces pour des tâches de coordination complexes, elles présentent un point de défaillance unique et une scalabilité limitée due à la puissance de calcul exponentielle lié au nombre de drones.

À l'opposé, les architectures décentralisées permettent à chaque drone de prendre ses décisions basées uniquement sur des informations locales, offrant une meilleure robustesse et scalabilité. L'algorithme ISOA (Zhen et al., 2018) illustre cette approche en utilisant des cartes de phéromones et de couverture partagées implicitement entre UAVs pour coordonner des missions de recherche-attaque.

Les architectures hybrides émergent comme compromis, combinant coordination locale décentralisée et supervision centralisée légère pour les objectifs stratégiques.

2.2 Méthodes d'élection des drones

Dans le cadre de notre projet, nous avons du trouver un moyen de hiérarchiser l'essaim de manière décentralisée. Pour ce faire, nous avons choisi le max-consensus qui permet au drone de communiquer un score et de se comparer aux autres afin d'élire le plus apte à effectuer la mission

Figure 2 : Schéma simple du max consensus

2.3 Méthodes de navigation et d'évitement d'obstacles

Les méthodes réactives décentralisées permettent la navigation de drone et d'essaim de drone en milieu contraint. Les systèmes anticollision entre les drones et les obstacles et entre les drones eux-mêmes peuvent être assurée de diverses façons, avec chacune ses avantages et ses inconvénients.

La méthode des champs potentiels constitue une approche classique en modélisant l'environnement comme un champ de forces où les objectifs attirent et les obstacles repoussent le drone. Leur simplicité et réactivité les rendent populaires, mais ils souffrent de minima locaux et peinent dans les environnements encombrés. De plus, il faut bien choisir et régler les fonctions et paramètres de champs afin de trouver le meilleur compromis entre précision et sécurité du drone.

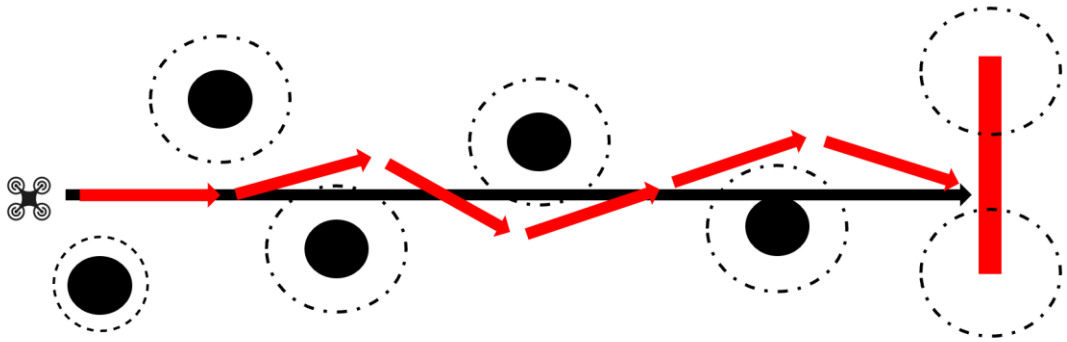

Figure 3 : Schéma simple d'une trajectoire utilisant le champ potentiel

Les méthodes ORCA (Optimal Reciprocal Collision Avoidance) garantissent l'évitement de collision décentralisé en calculant pour chaque drone l'ensemble des vitesses admissibles, permettant une navigation fluide en essaim mais nécessitant des hypothèses

sur le comportement coopératif des voisins. Cette méthode est utile pour les essaims nécessitants une navigation conflictuelle, ce qui est moins le cas pour nous dans le cadre de notre solution où l'on souhaite que tout les drones convergent vers le même point.

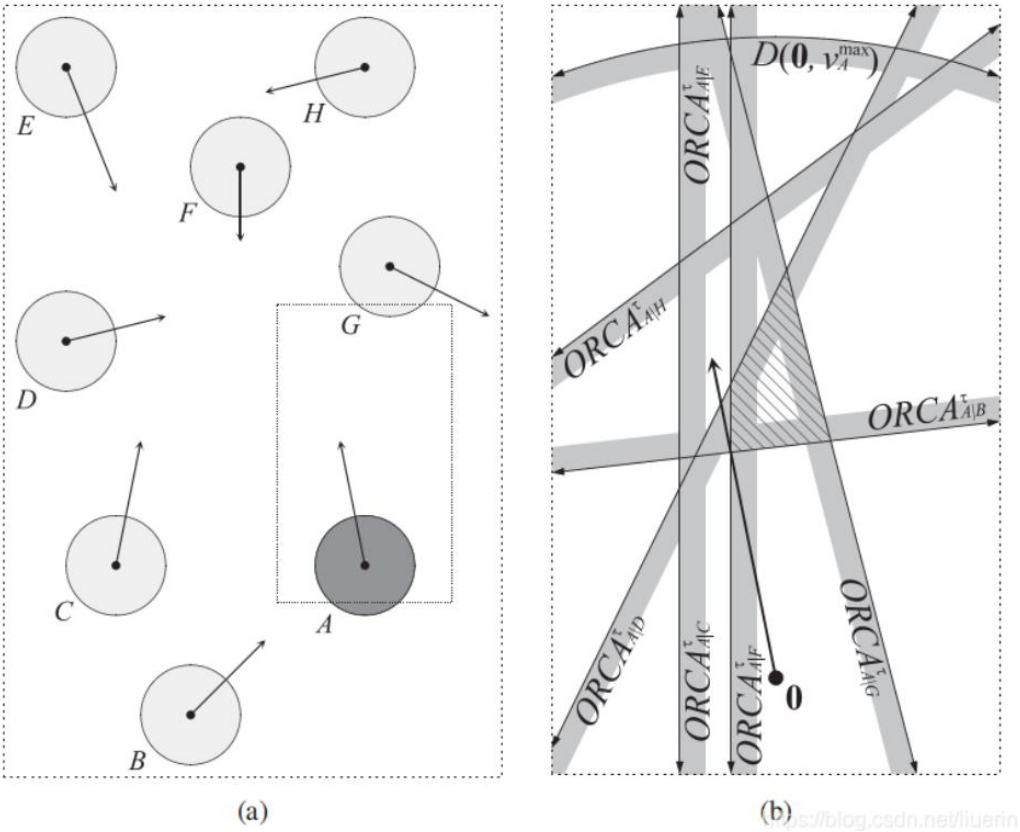

Figure 4 : Évitement d'obstacles avec la méthode ORCA

Les approches de contrôle prédictif (MPC - Model Predictive Control) optimisent les trajectoires sur un horizon glissant en intégrant explicitement les contraintes dynamiques et les prédictions de mouvement des autres agents, offrant des performances élevées au prix d'une charge computationnelle importante limitant leur application temps réel pour de grands essaims.

2.4 Approches par apprentissage

L'apprentissage par renforcement profond révolutionne le contrôle d'essaims en permettant aux politiques neuronales d'apprendre directement depuis l'expérience simulée. Batra et al. (2021) démontrent la possibilité d'entrainer un modèle puis de l'appliquer à des contrôleurs décentralisés, en l'occurrence ici sur des Bitcraze Crazyflies.

L'apprentissage par imitation depuis planificateurs centralisés experts, combiné au Reinforcement Learning pour équilibrer objectifs individuels et collectifs, offre une alternative aux méthodes de navigations réactives. Elle permet d'obtenir des modèles plus simples et légers, qui sont donc plus plausibles d'être intégrés sur des calculateurs embarqués.

Ces architectures on beaucoup de potentiel car un modèle est facilement entraînable avec des essaims de drones de taille variable ce qui le rend plus scalable. La limite principale reste cependant la demande en calcul du modèle à embarquer, vu que les calculateurs de drones possèdent une faible puissance de calcul.

2.5 Gestion de l'information et communication

La communication traditionnelle par partage d'état complet via réseau sans fil suppose un environnement idéal sans interférence, hypothèse rarement vérifiée en conditions réelles. L'observation de l'environnement incluant les voisins, obstacles et cibles, apparait comme une alternative possible.

Cela demande cependant de plus grandes capacités de calculs à chacun des drones de l'essaim, qui peut être contrebalancé par l'amélioration des ordinateurs embarqués ou des algorithmes simples, moins gourmand en puissance comparé à l'augmentation des tâches à effectuer.

3. Démarche scientifique

3.1 Hypothèses et choix méthodologiques

3.1.1 Hypothèses de départ

Ce projet a été réalisé dans le cadre d'un hackathon, ce qui implique par nature des contraintes temporelles fortes. L'ensemble de la conception, du développement et des tests devait être réalisé sur une durée limitée, estimée à environ 1 mois. Cette contrainte a fortement influencé les choix techniques et méthodologiques effectués, en privilégiant des solutions simples, robustes et rapidement implémentables plutôt que des approches plus complexes mais potentiellement plus performantes.

Les drones utilisés étaient imposés par l'organisation du hackathon : il s'agissait de Crazyflie de première génération. Ces drones sont de très petite taille, ce qui limite fortement leurs capacités embarquées, tant en termes de puissance de calcul que de capteurs disponibles et d'autonomie énergétique. En particulier, les drones ne disposent ni de caméra ni de capteurs avancés de perception de l'environnement, ce qui exclut d'emblée toute approche reposant sur la vision ou la détection embarquée complexe. En revanche,

l'ensemble des drones étant strictement identiques, nous avons pu faire l'hypothèse d'un essaim homogène, simplifiant ainsi la gestion et la coordination collective.

Le travail s'inscrit dans une logique d'essaim coordonné et coopératif. Les drones ne sont pas considérés comme des agents indépendants poursuivant des objectifs individuels, mais comme les éléments d'un système collectif poursuivant un objectif commun. Ce choix est directement lié au défi proposé par le hackathon, à savoir le défi d'« action combinée ». Dans ce cadre, un drone réalise une action principale tandis que les autres drones suivent, assistent ou complètent cette action. Nous avons ainsi fait l'hypothèse d'une coopération totale entre les drones, sans comportement compétitif ni conflit d'objectifs, le scénario étant conçu avant tout pour mettre en valeur la coordination et la collaboration au sein de l'essaim.

L'environnement de test a également été supposé contrôlé. Les expérimentations ont été réalisées en intérieur, dans une cage à drones équipée d'un système de motion capture OptiTrack. Nous avons donc posé comme hypothèse initiale que les positions des drones étaient connues avec précision et disponibles en temps réel. De même, la position de l'objectif principal, à savoir la fenêtre à traverser, a été considérée comme connue a priori. Cette hypothèse permet de se concentrer sur la problématique de la navigation et de la coordination, sans introduire la complexité supplémentaire liée à la perception et à la reconnaissance d'objets.

En cohérence avec ces hypothèses, aucune détection visuelle n'a été intégrée dans la version actuelle du projet. L'absence de caméra et de capteurs avancés a conduit à déléguer entièrement la fourniture des positions (drones et objectif) au système OptiTrack. La détection réelle de la fenêtre ou d'ouvertures dans un environnement non instrumenté a volontairement été laissée de côté et identifiée comme une piste de travail future.

Enfin, nous avons adopté une approche par jalons successifs. L'hypothèse de départ était qu'il valait mieux, dans un premier temps, faire fonctionner correctement une navigation simple et coordonnée de l'essaim, avant d'envisager toute complexification du scénario. Cette démarche incrémentale permet de valider progressivement les briques fondamentales du système, tout en restant compatible avec les contraintes de temps et de niveau d'expertise de l'équipe.

3.1.2 Contraintes du projet

Le projet a été mené sous de fortes contraintes matérielles, principalement liées à l'utilisation obligatoire des drones Crazyflie. Ces drones disposent d'une puissance de calcul embarquée très limitée, ce qui restreint considérablement la complexité des algorithmes pouvant être exécutés directement à bord. De plus, les capteurs disponibles sont réduits à l'essentiel, avec principalement une centrale inertielle (IMU), sans caméra ni capteurs de perception avancés. La charge utile très faible empêche également l'ajout de nouveaux

capteurs ou modules matériels, rendant impossible toute évolution matérielle significative dans le cadre du projet.

Le scénario choisi introduit également des contraintes opérationnelles importantes. La navigation se fait en intérieur, dans une cage à drones équipée d'un système OptiTrack, ce qui impose un espace de manœuvre limité. Le passage par une fenêtre constitue une ouverture étroite, nécessitant une coordination précise entre les drones afin d'éviter les collisions. De plus, l'ensemble de l'essaim doit franchir cette ouverture, ce qui implique une gestion collective des trajectoires et des priorités de passage, tout en respectant les contraintes de sécurité liées à la proximité entre les drones.

Il est important de préciser que, dans le cadre de ce projet, la « fenêtre » du scénario ne correspond pas à une vitre réelle, mais à un cerceau matérialisant une ouverture. En effet, les drones Crazyflie ne disposent ni de la masse, ni de la puissance, ni du matériel nécessaire pour briser une vitre ou interagir physiquement avec un obstacle solide. De plus, le cadre expérimental du hackathon ne permettait pas l'utilisation de dispositifs de destruction ou de percement. Le choix d'un cerceau permet ainsi de représenter de manière réaliste une ouverture à franchir, tout en garantissant la sécurité du matériel et des participants. Cette abstraction reste pertinente du point de vue algorithmique, car elle conserve les contraintes géométriques et de précision associées au passage par une ouverture étroite, sans introduire de risques ou de contraintes matérielles irréalistes pour ce type de drones.

Sur le plan logiciel, le choix d'un framework robotique adapté constituait une contrainte à part entière. L'équipe ayant récemment découvert ROS2, il a fallu apprendre à utiliser ce méta-système tout en développant le projet. Le système devait être à la fois modulaire, afin de séparer clairement les différentes fonctionnalités, et compatible avec une exécution en simulation comme en conditions réelles. Le choix de ROS2 s'est imposé car il s'agit d'un standard largement utilisé en robotique, bien documenté et bien adapté aux systèmes multi-robots, tout en correspondant aux connaissances récemment acquises par l'équipe.

Figure 5 : Logo ROS2

Les contraintes de simulation ont également joué un rôle important. L'installation et la prise en main des outils de simulation se sont révélées complexes pour une équipe débutante. Le

simulateur Webots a été retenu car il était déjà installé sur les postes de travail et proposait des modèles de Crazyflie ainsi qu'un contrôleur PID fonctionnel. Toutefois, il a fallu s'assurer que le comportement des drones en simulation soit cohérent avec celui observé lors des expérimentations réelles en laboratoire, ce qui a nécessité de nombreux ajustements.

Figure 6: Logo Webots

Enfin, des contraintes expérimentales sont apparues lors de l'utilisation du système OptiTrack. Des difficultés initiales liées au réglage, à la calibration, à la synchronisation et à la communication ont dû être surmontées. Ces étapes ont demandé un temps d'apprentissage non négligeable, d'autant plus que l'équipe était débutante dans ce domaine. L'intervention et l'accompagnement du tuteur de projet ont été déterminants pour résoudre ces problèmes et permettre la mise en place d'un environnement expérimental fonctionnel.

Figure 7 : Logo Optitrack

3.1.3 Choix de l'architecture de contrôle de l'essaim

Dans le cadre de ce projet, nous avons fait le choix d'une architecture de contrôle centralisée pour piloter l'essaim de drones. Ce choix est principalement motivé par le niveau d'expertise de l'équipe et par les contraintes techniques et temporelles imposées par le hackathon. La mise en place d'une architecture décentralisée, avec une prise de décision embarquée sur chaque drone, aurait nécessité une maîtrise approfondie du développement embarqué, de la synchronisation distribuée et de la communication inter-drones, ce qui dépassait le cadre du projet et le temps disponible.

L'architecture retenue repose donc sur un ordinateur central, chargé de l'ensemble des calculs de haut niveau. Chaque drone transmet en continu sa position au PC central, position fournie par le système de motion capture OptiTrack. À partir de ces informations, le PC calcule les trajectoires, les positions cibles ainsi que la logique globale de coordination de l'essaim. Les consignes de position sont ensuite envoyées aux drones, qui se contentent d'exécuter ces ordres sans prendre de décision complexe localement.

Pour la mise en place du dispositif expérimental, quatre marqueurs OptiTrack ont été fixés sur le cerceau représentant la fenêtre, disposés de manière à former un rectangle inscrit dans le cercle, afin de définir précisément la position et l'orientation de l'ouverture dans l'espace. Chaque drone a quant à lui été équipé de trois ou quatre marqueurs OptiTrack, permettant un suivi fiable de sa position et de son orientation par le système de capture de mouvement. Avant d'envisager un essaim plus important, le scénario a d'abord été testé avec trois drones, ce qui a permis de valider le principe général, le suivi OptiTrack, ainsi que les premières stratégies de coordination pour le franchissement de la fenêtre dans un environnement contrôlé.

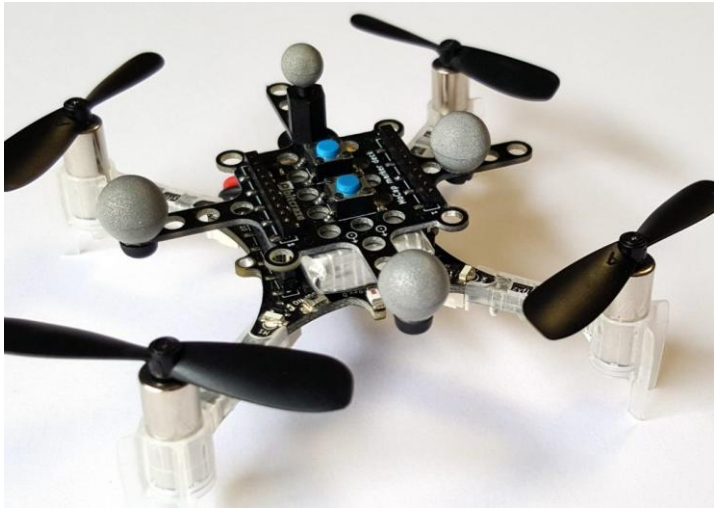

Figure 8 : Crazyflie avec marqueurs OptiTrack

Figure 9 : Cerceau faisant acte de fenêtre

Le rôle des drones est ainsi volontairement limité à l'exécution de commandes de position. Le contrôle bas niveau est assuré par des régulateurs PID déjà existants sur les Crazyflie, fournis sous forme de code standard flashé directement sur les drones. Ce code PID n'a pas été développé par l'équipe, mais il permet de garantir un suivi correct des positions cibles transmises par le PC central. Cette séparation claire entre le contrôle bas niveau embarqué et la logique de décision centralisée a permis de simplifier considérablement le développement et le débogage du système.

Cette architecture centralisée présente plusieurs avantages dans le contexte du projet. Elle offre une grande simplicité de mise en œuvre et permet un développement rapide, compatible avec les contraintes du hackathon. Elle facilite également le débogage, car l'ensemble de la logique décisionnelle est regroupé au même endroit, ce qui permet d'observer, modifier et corriger le comportement global de l'essaim de manière efficace. De plus, l'utilisation d'un PC central permet de s'affranchir des limitations de calcul embarqué des Crazyflie.

Cependant, ce choix présente également des inconvénients clairement identifiés. L'architecture centralisée introduit un point de défaillance unique : en cas de panne ou de dysfonctionnement du PC central, l'ensemble de l'essaim devient inopérant. De plus, ce type d'architecture est moins robuste et moins scalable qu'une approche décentralisée, notamment dans des scénarios réels impliquant un grand nombre de drones ou des environnements dégradés en termes de communication. Néanmoins, ces inconvénients ont été jugés acceptables dans le cadre d'un projet expérimental, avec un scénario simple, un nombre limité de drones et un environnement de test contrôlé.

En résumé, le choix d'une architecture de contrôle centralisée constitue un compromis assumé entre simplicité, faisabilité et performance. Il répond aux contraintes du hackathon et permet de démontrer efficacement la coordination d'un essaim de drones, tout en laissant la porte ouverte à des évolutions futures vers des architectures plus distribuées et robustes.

3.1.4 Choix des méthodes de navigation et d'évitement

Pour la navigation et l'évitement d'obstacles, nous avons choisi d'utiliser la méthode des champs potentiels. Ce choix a été principalement guidé par des considérations de simplicité, de rapidité d'implémentation et de maîtrise théorique par l'équipe. En effet, cette méthode avait déjà été étudiée dans le cadre des enseignements et testée lors de travaux pratiques, ce qui en faisait une solution bien comprise et immédiatement exploitable dans le temps imparti par le hackathon.

Le principe des champs potentiels repose sur la modélisation de l'environnement sous forme de forces artificielles. L'objectif à atteindre, en l'occurrence la fenêtre, génère un champ attractif qui attire les drones vers sa position, tandis que les obstacles génèrent des champs répulsifs visant à éloigner les drones et à prévenir les collisions. La combinaison de ces forces permet de produire une direction de déplacement continue, adaptée à une navigation réactive en environnement contraint. Cette approche est particulièrement bien adaptée à un contrôle centralisé, où les forces peuvent être calculées globalement à partir des positions connues des drones et des obstacles.

Dans le cadre de notre scénario, l'objectif commun de l'essaim est le passage coordonné par une ouverture étroite. La stratégie adoptée consiste à faire passer un drone en premier afin de sécuriser la trajectoire et de guider les suivants. Ce drone est désigné comme drone leader. Sa sélection repose sur une méthode de consensus simple, basée sur des critères mathématiques tels que la distance à l'objectif ou la position relative dans l'environnement. Cette approche permet d'élire automatiquement le drone le plus pertinent pour initier l'action, sans intervention manuelle.

Le choix de l'algorithme de Max-Consensus pour la désignation du drone leader a été motivé par des considérations de robustesse et de résilience de l'essaim. Une désignation fixe et prédéterminée du leader présenterait un point de défaillance unique : en cas de

dysfonctionnement du drone désigné ou de perte de communication, l'ensemble de l'essaim se retrouverait bloqué. À l'inverse, le Max-Consensus permet une élection distribuée du leader à partir d'échanges locaux, sans dépendance à un agent spécifique. Cette approche garantit que l'essaim peut toujours converger vers un leader valide, même en cas de défaillance d'un ou plusieurs drones, ce qui en fait une solution particulièrement adaptée à un système multi-agents évoluant en environnement contraint.

Une fois le drone leader sélectionné, celui-ci se dirige en priorité vers la fenêtre en suivant les champs potentiels définis. Les autres drones adaptent ensuite leur comportement pour suivre et compléter l'action du leader, tout en respectant les contraintes de sécurité liées à la proximité entre agents. Cette organisation hiérarchique temporaire permet de réduire les risques de congestion ou de collision au niveau de l'ouverture, tout en conservant une logique de coopération globale au sein de l'essaim.

D'autres méthodes de navigation et d'évitement, telles que les approches basées sur ORCA ou le contrôle prédictif (MPC), ont été envisagées lors de la phase de conception. Toutefois, elles ont été écartées en raison de leur complexité de mise en œuvre, de leur coût computationnel ou de leur inadéquation avec les objectifs du scénario. Le temps limité du hackathon ne permettait pas de garantir une implémentation fiable de ces méthodes plus avancées.

En définitive, le choix des champs potentiels s'inscrit dans une démarche pragmatique, privilégiant la fiabilité et l'adéquation au scénario plutôt que la sophistication algorithmique. Cette méthode répond efficacement aux contraintes du projet et permet de démontrer de manière claire la coordination et la navigation collective de l'essaim, tout en laissant la possibilité, à terme, de comparer ces résultats avec des approches plus avancées.

3.2 Navigation et coordination de l'essaim

Dans ce projet, nous avons implémenté une navigation hiérarchique combinant un mécanisme de consensus pour la coordination de l'essaim, une planification globale de trajectoire et une navigation locale réactive basée sur des champs potentiels. Cette section décrit les modèles mathématiques effectivement utilisés dans notre implémentation, sans entrer dans les détails logiciels. Pour en savoir plus sur le code, les liens sur la navigation se trouvent ici.

3.2.1 Coordination par Max-Consensus

La coordination de l'essaim débute par la désignation d'un drone leader à l'aide d'un algorithme de Max-Consensus. Cet algorithme permet à l'ensemble des drones de s'accorder sur un unique agent à partir de communications locales, sans décision imposée manuellement.

Dans l'implémentation réalisée, chaque drone i est caractérisé par :

$$ID_i \in \mathbb{N}$$

$$S_i \in \mathbb{R}$$

- un identifiant unique ID_i,

- un score S_i, représentant sa priorité à devenir leader.

Deux modes de calcul du score ont été intégrés dans le code.

Dans le mode principal utilisé pour les expérimentations, le score est déterminé directement à partir de l'identifiant du drone selon la relation :

$$S_i = 5 - ID_i$$

Ce choix impose un ordre de priorité fixe et reproductible au sein de l'essaim. Le drone possédant l'identifiant le plus faible obtient le score le plus élevé et est donc naturellement sélectionné comme leader. Cette approche facilite l'analyse du comportement global du système et permet de garantir une désignation stable du leader lors des tests.

Un second mode, utilisé à des fins de validation, consiste à attribuer un score aléatoire à chaque drone :

$$S_i \sim \mathcal{U}(1, 50)$$

Ce mode permet de vérifier que l'algorithme de Max-Consensus converge correctement indépendamment de la distribution initiale des scores.

L'algorithme repose sur des échanges locaux d'information. Chaque drone communique uniquement avec deux voisins logiques dans une topologie circulaire. À chaque itération, le drone compare son score avec ceux reçus de ses voisins et conserve le maximum selon la règle suivante :

Pour un drone *i*, à l'itération *k*, on note :

- $S_{best(k)}$: le meilleur score actuellement connu par le drone
- $ID_{best(k)}$: l'identifiant associé à ce score
- $S_{j(k)}$ *et* $ID_{j(k)}$: le score et l'identifiant reçus d'un drone voisin *j*

$$\text{si } S_j^{(k)} > S_{\text{best}}^{(k)} \text{ alors } \begin{cases} S_{\text{best}}^{(k+1)} = S_j^{(k)} \\ ID_{\text{best}}^{(k+1)} = ID_j^{(k)} \end{cases}$$

Sinon :

$$\begin{cases} S_{\text{best}}^{(k+1)} = S_{\text{best}}^{(k)} \\ ID_{\text{best}}^{(k+1)} = ID_{\text{best}}^{(k)} \end{cases}$$

Cette mise à jour est répétée pendant un nombre d'itérations borné, proportionnel au nombre de drones dans l'essaim. Dans l'implémentation, ce nombre est fixé à environ N/2, ce qui est suffisant pour assurer la propagation du score maximal à l'ensemble du réseau.

À l'issue du processus, tous les drones convergent vers le même couple (ID*, S*), correspondant au score maximal de l'essaim. Le drone tel que ID_i = ID* est alors désigné comme leader.

Une fois le drone leader désigné par l'algorithme de Max-Consensus, l'essaim dispose d'un agent unique chargé d'initier et de structurer l'action collective. Cette étape marque la transition entre une phase de décision distribuée et une phase de planification du déplacement. Le drone élu leader devient alors responsable de la génération de la trajectoire globale à suivre, tandis que les autres drones adoptent un comportement de suiveurs coordonnés. Cette séparation claire des rôles permet de simplifier la gestion de l'essaim tout en garantissant une cohérence globale du mouvement.

3.2.2 Planification de la trajectoire globale (Global Path)

La planification de la trajectoire globale est assurée par le drone leader, désigné au préalable par l'algorithme de Max-Consensus. Son rôle consiste à définir une succession de positions cibles permettant d'atteindre la fenêtre, puis d'organiser le passage des drones suiveurs. La trajectoire globale est représentée par une séquence ordonnée de points dans l'espace, appelés waypoints.

Chaque waypoint est défini par un triplet de coordonnées de la forme :

$$p = (x, y, z)$$

La trajectoire globale peut ainsi être notée

$$P = \{p_1, p_2, \ldots, p_k\}$$

La position de la fenêtre est fournie par le système OptiTrack et notée

$$p_w = (x_w, y_w, z_w)$$

Afin d'éviter toute collision avec le cerceau matérialisant l'ouverture, le drone leader ne vise pas directement cette position mais une position décalée située en amont de la fenêtre. Le premier waypoint est alors défini par

$$p_1 = (x_w,\ y_w - \delta,\ z_w)$$

Où δ correspond à une distance de sécurité fixée expérimentalement.

Une fois cette position atteinte, un second waypoint est défini afin de permettre au leader de se décaler latéralement et de libérer l'accès à la fenêtre pour les drones suiveurs. Ce point est défini par

$$p_2 = (x_w - \Delta x,\ y_w - \delta,\ z_w)$$

où Δx est un décalage latéral constant. Les valeurs de δ et Δx sont choisies empiriquement afin d'obtenir un positionnement stable du leader tout en garantissant un espace suffisant pour le passage des autres drones.

Au cours de son déplacement, le drone leader enregistre les positions successives qu'il atteint. Ces positions sont stockées sous forme d'un graphe de points

$$G = \{g_1, g_2, \ldots, g_n\}$$

avec

$$g_i = (x_i, y_i, z_i)$$

Ce graphe correspond à la trajectoire réellement suivie par le leader et constitue un chemin valide jusqu'à la fenêtre, déjà vérifié en conditions réelles.

Une fois le leader arrivé à destination, ce graphe est transmis aux drones suiveurs. Ces derniers utilisent directement la séquence G comme trajectoire globale de référence, sans recalcul de chemin. Cette approche garantit que tous les drones empruntent une trajectoire identique, ce qui limite les risques de divergence de trajectoire et de collision dans un environnement contraint.

La planification globale ne cherche pas à produire une trajectoire optimale au sens de l'optimisation mathématique, mais une trajectoire fiable, reproductible et compatible avec les contraintes matérielles et expérimentales du projet. Cette séparation entre planification globale par le leader et navigation locale par chaque drone permet de simplifier l'architecture logicielle tout en assurant une coordination efficace de l'essaim.

3.2.3 Navigation locale par champs potentiels (Local Path)

Le module de navigation locale a pour rôle de transformer une consigne de position globale fournie par le module *Global Path* en une succession de petites consignes locales directement exploitables par le contrôleur bas niveau du drone. Contrairement au *Global Path*, qui raisonne en termes de waypoints espacés, la navigation locale fonctionne à haute fréquence et génère un déplacement incrémental à partir de la position courante du drone.

La position courante du drone est notée

$$q = (x, y, z)$$

tandis que la position cible globale reçue est notée

$$q_{\text{goal}} = (x_g, y_g, z_g)$$

La navigation locale est déclenchée uniquement lorsque le module *Global Path* a validé un nouvel objectif. Tant que cet objectif n'est pas atteint, le module *Local Path* calcule périodiquement un vecteur de déplacement élémentaire Δq, qui est ajouté à la position courante pour former la prochaine consigne envoyée au drone.

Le déplacement dans le plan horizontal repose sur une méthode de champs potentiels. L'erreur de position horizontale est définie par le vecteur

$$\mathbf{e} = (x_g - x,\ y_g - y)$$

et sa norme :

$$\|\mathbf{e}\| = \sqrt{(x_g - x)^2 + (y_g - y)^2}$$

La force attractive associée à l'objectif est proportionnelle à ce vecteur d'erreur et s'écrit

$$\mathbf{F}_{\text{attr}} = K_{\text{attr}} \cdot \mathbf{e}$$

Les obstacles, qu'ils soient fixes ou mobiles (autres drones), génèrent une force répulsive. Pour un obstacle situé en position $o = (x_o,\ y_o)$ et de rayon r, la distance effective est

$$d = \|(x, y) - (x_o, y_o)\| - r$$

Cas 1 : drone à l'intérieur de la zone de sécurité (d ≤ 0) :

Lorsque le drone pénètre dans la zone de sécurité de l'obstacle, une force répulsive est appliquée afin d'éviter toute singularité numérique liée au terme en $1/d^2$. Dans ce cas, la force répulsive est purement exponentielle et orientée selon la direction allant de l'obstacle vers le drone :

$$\mathbf{F}_{\text{repu}} = K_{\text{repu}} \exp\left(-\frac{d^2}{2\sigma^2}\right) \frac{(x, y) - (x_o, y_o)}{\|(x, y) - (x_o, y_o)\|}.$$

Afin d'anticiper la présence des obstacles avant l'entrée dans la zone de sécurité, un rayon d'influence d_0 est défini pour chaque obstacle. Ce paramètre correspond à la distance maximale à partir de laquelle l'obstacle commence à exercer une force répulsive sur le drone. Dans l'implémentation, ce rayon d'influence est égal au rayon de l'obstacle et permet de distinguer une zone d'anticipation, où l'évitement est progressif, d'une zone critique, où une répulsion plus directe est appliquée.

Cas 2 : obstacle proche, dans le rayon d'influence ($0 < d < d_0$) :

Lorsque la distance à l'obstacle est positive mais inférieure au rayon d'influence d_0, une force répulsive combinant une dépendance en $1/d^2$ et un terme exponentiel est appliquée, conformément à l'implémentation de la classe CP :

$$\mathbf{F}_{\text{repu}} = \frac{K_{\text{repu}}}{d^2} \exp\left(-\frac{d^2}{2\sigma^2}\right) \frac{(x,y) - (x_o, y_o)}{\|(x,y) - (x_o, y_o)\|}.$$

Cette formulation permet d'assurer une répulsion suffisamment forte à courte distance tout en conservant une variation progressive de la force.

Cas 3 : obstacle hors du rayon d'influence ($d \geq d_0$) :

Lorsque l'obstacle est situé en dehors de son rayon d'influence, aucune force répulsive n'est appliquée :

$$\mathbf{F}_{\text{repu}} = \mathbf{0}.$$

La force répulsive totale exercée sur le drone est obtenue par la somme des contributions associées à l'ensemble des obstacles présents dans l'environnement. Dans l'implémentation, cette force est ensuite saturée afin de limiter son amplitude maximale et garantir un comportement stable du système.

$$\mathbf{F}_{\text{repu}} = \sum_i \mathbf{F}_{\text{repu},i}.$$

Plusieurs fonctions ont été testées pour modéliser la force répulsive des obstacles. Des lois basées sur des distributions normales et gaussiennes classiques ont d'abord été implémentées, avec une dépendance directe à la distance entre le drone et l'obstacle. Toutefois, ces formulations se sont révélées soit trop faibles à courte distance, soit trop abruptes, ce qui engendrait des oscillations ou des trajectoires instables, en particulier dans des environnements confinés ou lors du passage par la fenêtre.

La formulation finalement retenue repose sur un terme exponentiel de la forme $\exp(-d^2 / (2\sigma^2))$, combiné à une dépendance en $1/d^2$. Cette expression correspond exactement à celle utilisée dans le code et offre un compromis satisfaisant entre réactivité et stabilité. L'exponentielle permet une montée progressive de la force répulsive lorsque le drone s'approche d'un obstacle, tandis que le terme en d^2 garantit une répulsion suffisamment forte à courte distance pour éviter toute collision. Expérimentalement, cette

formulation s'est montrée la plus robuste, aussi bien en simulation que lors des essais réels, ce qui a motivé son adoption définitive dans l'implémentation finale.

La force totale dans le plan horizontal est alors donnée par la somme

$$\mathbf{F} = \mathbf{F}_{\text{attr}} + \sum \mathbf{F}_{\text{repu}} + \mathbf{F}_{\text{frontières}}$$

où le terme $F\_frontières$ représente les forces répulsives liées aux limites de l'environnement (la fonction de la force frontière se trouve dans l'annexe du code).

Le déplacement élémentaire horizontal est obtenu en normalisant cette force et en la multipliant par un pas scalaire K_pas :

$$\Delta q_{xy} = K_{\text{pas}} \cdot \frac{\mathbf{F}}{\|\mathbf{F}\|}$$

Dans le code, la valeur de K_pas n'est pas constante. Elle dépend de la distance à l'obstacle le plus proche et est bornée entre une valeur minimale et une valeur maximale :

$$K_{\text{pas}} = \text{clamp}\left(d_{\text{min}} \cdot K_{\text{pas\_err}},\ K_{\text{pas\_min}},\ K_{\text{pas\_max}}\right)$$

Ce mécanisme permet de réduire automatiquement la vitesse du drone lorsqu'il évolue à proximité d'un obstacle, améliorant ainsi la stabilité et la sécurité du déplacement.

Le mouvement vertical est traité indépendamment du champ potentiel horizontal. La variation d'altitude est calculée à partir d'une loi proportionnelle simple :

$$\Delta z = K_z \cdot (z_g - z)$$

avec une saturation afin de limiter la vitesse verticale maximale. Cette séparation entre le plan horizontal et l'axe vertical permet d'éviter les couplages instables et simplifie le franchissement de la fenêtre.

La consigne locale finale envoyée au drone correspond alors à la somme de la position courante et du déplacement élémentaire :

$$q_{\text{next}} = (x, y, z) + (\Delta x,\ \Delta y,\ \Delta z)$$

Cette consigne est publiée à une fréquence élevée, jusqu'à ce que la distance à la cible devienne inférieure à un seuil fixé expérimentalement. Lorsque

$$\|(x, y) - (x_g, y_g)\| \leq \varepsilon_{xy}$$

et

$$|z - z_g| \leq \varepsilon_z$$

Le module *Local Path* considère l'objectif atteint et déclenche un signal de fin de déplacement à destination du *Global Path*.

Ainsi, la navigation locale par champs potentiels assure un suivi continu et réactif des objectifs globaux, tout en intégrant l'évitement d'obstacles et les contraintes de sécurité. Cette approche, bien que simple, est bien adaptée aux contraintes matérielles des drones Crazyflie et permet une navigation stable dans un environnement intérieur instrumenté.

Ce découplage présente plusieurs avantages observés expérimentalement. Il améliore la stabilité du vol, en particulier lors des phases critiques comme l'approche et le franchissement de la fenêtre, où des ajustements fins en altitude sont nécessaires sans perturber la trajectoire horizontale. Il facilite également le réglage des paramètres, puisque les gains du champ potentiel et ceux du contrôle vertical peuvent être ajustés indépendamment.

Enfin, ce choix est cohérent avec l'architecture globale du système. Le *Global Path* impose des objectifs en position 3D, mais la responsabilité du *Local Path* se limite à assurer un suivi progressif et sûr de ces objectifs. En séparant clairement la gestion de l'altitude du calcul du champ potentiel, l'implémentation reste simple, robuste et bien adaptée aux contraintes matérielles et expérimentales du projet.

Les choix méthodologiques et techniques présentés dans cette section ont permis de mettre en place une architecture fonctionnelle de navigation et de coordination d'un essaim de drones, combinant sélection d'un leader, planification globale et navigation locale par champs potentiels. Toutefois, la mise en œuvre concrète de ces principes dans un environnement réel et sous contraintes fortes n'a pas été exempte de difficultés. La section

suivante revient sur les principaux problèmes rencontrés au cours du projet, qu'ils soient d'ordre technique ou organisationnel, ainsi que sur les solutions apportées pour y remédier.

4. Problèmes rencontrés et solutions apportées

Cette partie présente quels ont été les principaux problèmes rencontrés au cours du projet et les solutions mises en place pour y répondre. Les difficultés présentées sont séparées en deux parties : les problèmes techniques et les problèmes organisationnels.

4.1 Problèmes techniques

Au cours du projet, plusieurs problèmes techniques ont été rencontrés. Ceux-ci étaient principalement liés au manque de maîtrise des outils. Ces difficultés ont été surmontées, grâce à la montée en compétence de chacun et aux connaissances acquises tout au long du projet.

4.1.1 Prise en main du Crazyflie

La prise en main du drone a été difficile en début de projet, une API Python fournie par la bibliothèque « cflib » permet de commander le drone. Celui-ci ne semblait pas capable de conserver une position stable en vol et son comportement était imprévisible. La manette permettait de piloter le drone sans soucis, ce qui a écarté l'hypothèse d'un problème matériel ou d'un dysfonctionnement interne du drone.

La difficulté provenait en réalité de la manière dont l'API était utilisée. À cause des erreurs, le drone passait en mode sécurité, ce qui empêche le pilotage et nécessite de le réinitialiser. Des exemples de code avec des séquences de vol pour le drone sont disponibles sur le GitHub du Crazyflie. Le code a donc été développé en s'appuyant sur ces exemples.

Parmi les erreurs identifiées :

- Le drone doit recevoir de nouvelles coordonnées cibles à intervalles réguliers, toutes les 100 ms.

- L'estimateur doit être correctement réinitialisé avec les positions qui sont récupérées par l'Optitrack puis transmises au drone pour qu'il puisse estimer sa position.

4.1.2 Inversement du repère sur l'Optitrack

Afin de faire voler le drone en récupérant les positions avec Optitrack, il faut sélectionner les marqueurs du drone sur le logiciel de l'Optitrack, créer un « rigidbody » et le nommer (par exemple le nom du premier drone est « crazyflie_1 »). Chaque rigidbody a son propre repère, le problème était que ces repères s'inversaient fréquemment. En effet, le vol du drone était affecté si les repères étaient inversés avant le décollage.

Tant que ce souci n'était pas réglé, il fallait vérifier les axes des drones avant le décollage et s'ils étaient inversés recréer le rigidbody ou bien décocher et cocher la présence de l'asset.

La source du problème a été trouvée plus tard. Les marqueurs étaient positionnés sur le drone de manière trop symétrique. Ils ont donc été réinstallés de manière dissymétrique et ainsi éviter l'inversement des axes.

4.1.3 Formation des drones en colonne verticale

Lors du hackathon, des tests ont été réalisés afin de faire voler les drones en colonne verticale, c'est-à-dire en étant positionnés les uns au-dessus des autres, dans le but de rendre la démonstration de vol plus visuelle. Cependant, cette configuration n'est pas adaptée, le drone situé au-dessus engendre des perturbations qui affectent le drone d'en dessous. Cela empêche le drone du dessous de maintenir un vol stable.

Afin de ne pas faire face à ce problème, un espace plus important a été prévu entre les drones pendant qu'ils maintenaient la formation. Cette formation semble réaliste avec 2 drones. Mais, pour la réaliser avec plus de drone, la colonne occupera beaucoup d'espace verticalement. Cette formation ne semblerait pas adaptée pour passer une fenêtre ensuite, car tous les drones doivent se mettre à sa hauteur avant de passer. Une formation plus adaptée serait donc une formation où les drones ne se survolent pas les uns et les autres.

4.1.4 Multithreading

Les callbacks sont des fonctions appelées automatiquement lorsqu'un événement se produit. Le multithreading permet d'utiliser plusieurs callbacks en parallèle, pour cela il faut utiliser des callback groups. Cependant, leurs utilisations deviennent plus complexes avec la présence de services et topics dans ROS2 car il faut choisir le type de callback group adapté.

Deux types de callback group ont été utilisés :

- "MutuallyExclusiveCallbackGroup" permet de protéger les ressources critiques non sécurisées. Les callbacks appartenant au même groupe ne peuvent pas s'exécuter en même temps, ce qui évite les conflits et les erreurs. Il est possible de créer plusieurs groupes de ce type pour que les callbacks de ces différents groupes s'exécutent en même temps.

- "ReentrantCallbackGroup" permet l'exécution de plusieurs callbacks de ce groupe en même temps.

4.2 Problèmes organisationnels

L'une des problématiques importantes lors d'un travail en groupe est la gestion de projet. Elle implique l'organisation, la planification et la répartition des tâches. Une gestion de projet mal structurée peut impacter son avancement global.

4.2.1 Contrainte de temps

Le délai pour réaliser le projet était d'un mois. Le défi était de définir l'objectif du projet, celui-ci devait être suffisamment intéressant, tout en restant réalisable avec un délai relativement court. De plus, au lancement du projet, la difficulté était d'estimer la quantité de travail qui pourrait être accomplie dans les délais.

Cette contrainte de temps a conduit à définir plusieurs objectifs sous forme de scénario à plusieurs étapes, avec un objectif minimal à atteindre. Cette approche a permis de garantir l'atteinte d'au moins un scénario, tout en conservant la possibilité d'en réaliser davantage en fonction de l'avancée du projet.

4.2.2 Réunions régulières

Afin d'anticiper des difficultés liées à une mauvaise communication et à un mauvais suivi du projet, des réunions ont été régulièrement organisées. Ces réunions permettaient de prendre connaissance des avancées de chacun, de leurs difficultés et de s'entraider en proposant des solutions. De plus, la suite du projet et la répartition des tâches étaient déterminées à l'issue de ces réunions.

Cette organisation a contribué à maintenir de la coordination au sein de l'équipe et à éviter des problèmes de communication ou de répartition des tâches. La mise en place de réunions régulières était un élément clé pour la réussite du projet.

5. Perspectives et améliorations futures

Le projet est loin d'être terminé et en 3 mois on a eu bien plus d'idées que de temps pour les implémenter. Dans l'immédiat, malheureusement l'essaim n'accomplie pas la mission désirée sur simulateur malgré un fonctionnement correct avec les drones réels. Cela serait donc la première chose à rectifier à l'avenir.

La dernière feature que l'on a essayé d'implémenter est la possibilité de repli. Il existe déjà des éléments en lien dans certaines node. Cette feature nous a été très demandé par les mentors durant le hackathon. Le but serait de permettre à l'utilisateur de l'essaim de pouvoir ordonner un repli général de l'essaim à tout moment. Cette méthode se baserait sur la map dessinée, puis partagée par le leader de l'essaim. En effet, le chemin qui a été tracé serait la route à prendre pour le repli. Le repli est donc indépendant de la réception des position GPS. Si les obstacles n'ont pas bougé, les drones peuvent se replier sans être endommagés en prenant cette route. La complétion de cette feature implique aussi de réaliser une interface. Ce qui nous mène au prochain point.

En effet, malgré l'envie et le nombre de fois que l'on nous l'a répété, nous n'avons pas eu le temps d'implémenter une interface opérationnelle. Nous n'avons réalisé qu'une

interface sommaire avec le minimum des interactions que l'on aurait voulu avoir : paramétrer le test en modifiant utils.json en remplissant des cases de l'interface, un bouton launch… et un bouton Repli. Nous avons commencé cette interface avec la bibliothèque tkinter, mais d'autres bibliothèques seraient plus adaptées si l'objectif est d'avoir une « jolie » interface. Il faut avouer que l'embellissement des interfaces tkinter est assez limité.

D'un point de vue plus global, nous avions imaginé à terme utiliser un deck IA pour détecter les entrées puis définir laquelle serait la plus approprié pour rentrer, ou encore se poster à plusieurs entrés et rentrer simultanément pour créer un effet de blast... Mais de façon plus proche dans le développement, il serait intéressant d'arriver à implémenter un scénario plus long avec l'idée suivante qui a été validée par les mentors : effectuer une recherche dans un bâtiment. A la suite du scénario actuel, une fois que le drone leader est à côté de la fenêtre et que les drones suiveurs sont tous passés par la fenêtre, on définit un nouveau drone leader qui va mapper un chemin vers la prochaines entrée (de la même manière qu'au début) : une porte par exemple ; puis faire suivre les autres drones sauf 1, puis recommencer. En effet, l'idée est de parcourir le bâtiment en laissant un drone dans chaque pièce (le premier drone leader peut rester dehors et « monter la garde » si besoin). Le but étant alors de chercher efficacement dans le bâtiment. Ainsi, répartir l'essaim dans le bâtiment pour chercher apporte un vrai intérêt à l'utilisation d'un essaim pour cette mission en utilisant le fameux principe de « diviser pour mieux régner ».

En parallèle de tout cela il est possible de continuer à améliorer les outils de tests (util_essaim.py, utils.json…) en liant le fichier my_world.wbt à utils.json par exemple. En effet, on définit des positions d'obstacles dans utils.json pour les nodes, alors qu'il existe déjà des définitions d'obstacles dans my_world.wbt pour le simulateur. Il faudra peut-être aussi essayer de tout rassembler dans un seul workspace ROS2, car actuellement des nodes sont lancées dans ws_drones et dans ws_simu.

6. Conclusion

Ce projet a permis de mettre en œuvre un essaim de drones collaboratif capable de réaliser un scénario de navigation coordonnée en environnement intérieur, en s'appuyant sur une architecture centralisée et des outils standards de la robotique moderne. L'utilisation de ROS2, du système de motion capture OptiTrack et des drones Crazyflie a rendu possible la démonstration d'une coordination multi-drones fiable, malgré des ressources embarquées très limitées. Le choix des champs potentiels pour la navigation et l'évitement d'obstacles s'est révélé adapté aux contraintes du projet, offrant une solution simple, réactive et rapidement déployable dans le cadre du hackathon.

D'un point de vue technique, le projet a nécessité la maîtrise de nombreuses briques logicielles et matérielles, allant de la communication entre ROS2 et les drones, à l'intégration d'un système de capture de mouvement, en passant par la gestion de la synchronisation et du contrôle en temps réel. Les difficultés rencontrées, notamment lors de la prise en main des Crazyflie, de l'OptiTrack ou du multithreading sous ROS2, ont

conduit à une compréhension approfondie de ces outils et à l'adoption de bonnes pratiques de développement et de débogage en robotique.

Enfin, ce projet a constitué une expérience d'apprentissage particulièrement riche pour l'ensemble de l'équipe. Il a permis de consolider des compétences en systèmes multi-robots, en architecture logicielle et en gestion de projet sous contrainte, tout en mettant en évidence les compromis nécessaires entre complexité algorithmique et faisabilité expérimentale. Les résultats obtenus, bien qu'améliorables, posent des bases solides pour des développements futurs plus avancés, notamment vers des architectures distribuées et des scénarios plus réalistes.

7. Références bibliographiques

Algorithme de fusion des données décentralisé : https://www.sciencedirect.com/science/article/pii/S0957417423019462?ref=pdf_download&fr=RR-2&rr=9959d4e4dd7dd652

Apprentissage par reforcement du controle d'un essaim de drone : https://arxiv.org/pdf/2109.07735

Algorithme robuste d'atribution des taches : https://dspace.mit.edu/handle/1721.1/42177

Code de controle d'essaim de drone basé sur la dépense d'energie https://www.sciencedirect.com/science/article/pii/S0968090X23003777#sec4Code de controle d'essaim de drone basé sur la dépense d'energie : https://www.sciencedirect.com/science/article/pii/S0968090X23003777#sec4

Thèse sur l'utilisation de Consensus dans un essaim de drones : https://theses.hal.science/tel-02529658/document

Les différents types de callback group :

https://discourse.openrobotics.org/t/how-to-use-callback-groups-in-ros2/25255

Liens du code GitHub sur la navigation :

- Max Consensus :

https://github.com/DorianPSRT/TCI_AERO/blob/simulateur/ws_drones/src/fusion_CP_Consensus/fusion_CP_Consensus/decision_node1.py

- Global Path:

https://github.com/Dorian-PSRT/TCI_AERO/blob/simulateur/ws_drones/src/fusion_CP_Consensus/fusion_CP_Consensus/global_path_node1.py

- Champ Potentiel :

https://github.com/Dorian-PSRT/TCI_AERO/blob/simulateur/ws_drones/src/fusion_CP_Consensus/fusion_CP_Consensus/champs_pot_class.py

- Local Path :

https://github.com/Dorian-PSRT/TCI_AERO/blob/simulateur/ws_drones/src/fusion_CP_Consensus/fusion_CP_Consensus/local_path_node1.py