Découverte, mise en œuvre et manipulation de drones et de robots

RAPPORT DE STAGE DE FIN D'ÉTUDES À LA PRÉPA DES INP

Axel MEMIN

Laboratoire d'accueil : Laboratoire Bordelais de Recherche en Informatique

Encadré par : **Serge CHAUMETTE**

Enseignant référent : **Patrick LANUSSE**

Période du stage : du 2 mai au 9 juin 2023

TABLE DES MATIERES

I. Remerciements

Je tiens à adresser mes remerciements à toutes les personnes qui ont participé à la réalisation et au bon déroulement de ce stage.

Tout d'abord, j'aimerais adresser mes sincères remerciements à Mr Serge CHAUMETTE pour le précieux temps qu'il m'a accordé, la confiance et la liberté d'action qu'il m'a donnée avec le matériel. Il a su me confier des tâches me permettant d'appliquer mes connaissances tout en découvrant de nombreux nouveaux concepts.

Je tiens également à remercier l'ensemble des doctorants, ingénieurs et stagiaires de l'équipe pour leur aide et leurs conseils durant la réalisation de ce stage.

Enfin, je remercie Mr Patrick LANUSSE pour son aide au cours de ce stage et son suivi sur son bon déroulement.

II. Glossaire

- **Adresse IPv4 =** nombre de 32 bits identifiant de manière unique une interface réseau sur un système.
- **Bibliothèque Python =** ensemble logiciel de modules ajoutant des possibilités étendues à Python.
- **Carte réseau =** interface entre l'équipement ou la machine dans lesquels elle est montée et les machines connectées sur le même réseau.
- **Code Opensource =** code conçu pour être accessible au public : l'utilisateur peut voir et modifier le code à sa convenance.
- **Dictionnaire Python =** collection qui associe une clé à une valeur.
- **Espace de Couleur RGB =** espace colorimétrique visible par l'œil humain qui est produit par la synthèse additive de trois couleurs primaires : Red, Green, Blue.
- **Essaim de drone =** ensemble coordonné de drones ayant pour but d'effectuer une tâche commune dans divers types d'applications, civiles ou militaires.
- **FPV =** First Person View.
- **LaBRI =** Laboratoire Bordelais de Recherche en Informatique.
- **Pare-feu informatique =** dispositif de protection du réseau qui surveille le trafic entrant et sortant.
- **Python =** langage de programmation.
- **Raspberry Pi =** nano ordinateur monocarte pouvant se connecter à un moniteur, à un clavier et disposant d'une connectivité WiFi et Bluetooth.
- **Socket =** interface de connexion qui permet d'utiliser facilement les protocoles de transport tels que TCP et UDP.

III. Listes des figures

IV. Introduction

Placé au cœur de l'innovation, le développement des drones va prendre une grande part dans le monde de demain. En effet, leurs domaines d'applications touchant de nombreuses disciplines, l'optimisation et l'automatisation de leur utilisation est un sujet d'avenir. On a pu voir leur utilité pour l'assistance des opérateurs humains, notamment dans la surveillance, la lutte contre les feux de forêts, etc.

L'intelligence artificielle prend également de l'ampleur dans notre monde qui dépend de plus en plus du numérique. Celle-ci, intégrant des bases de données toujours plus nombreuses, acquiert une efficacité incontournable, la rendant indispensable dans certains usages (médecine, droit…).

L'alliance de ces deux concepts permet d'ouvrir un champ des possibles hors du commun dans les applications industrielles et de recherches, militaires ou civiles.

Très intéressé par cette branche et ses domaines d'applications, j'ai décidé d'effectuer mon stage de fin de classe préparatoire intégrée à La Prépa des INP Bordeaux au sein d'un laboratoire de recherche pour pouvoir découvrir le métier d'ingénieur mêlé à celui de chercheur. Je réalise donc mon stage au LaBRI (Laboratoire Bordelais de Recherche en Informatique), laboratoire dont l'activité se situe au confluent de ces deux phénomènes émergents.

Le « Game Fest' » est un évènement organisé du 28 au 30 novembre 2023 dans le but de faire découvrir aux enfants et aux adolescents les multiples activités possibles autour du numérique et du jeu. Dans le cadre de cet évènement, le LaBRI est amené à présenter des drones automatisés pour vulgariser de manière ludique et démontrer les possibilités d'action d'un drone.

Ainsi, j'ai eu pour mission durant ce stage de faciliter l'interaction entre un utilisateur et un drone dans le but de lui faire réaliser plusieurs tâches ludiques afin de le présenter au salon de « Game Fest' ».

Le présent rapport présentera dans un premier temps le laboratoire dans lequel j'ai réalisé ce stage, le LaBRI, ses domaines d'intervention et sa structure. Dans un second temps, j'aborderai le déroulement de ma mission, le cheminement de pensée et de programmation pour pouvoir rendre le drone confié le plus ludique et interactif possible.

V. *L'organisme d'accueil*

A. LaBRI : Laboratoire Bordelais de Recherche en Informatique

Le Laboratoire Bordelais de Recherche en Informatique (LaBRI) est une unité mixte de recherche du CNRS fondée en 1988. Rattaché à l'Université Bordeaux, le CNRS et à Bordeaux INP, il dépend du département des sciences et technologies de l'information et ingénierie (ST2I). Il est également partenaire de l'Institut National de Recherche en Informatique et en Automatique (INRIA) depuis 2002.

Le laboratoire réunit environ 300 personnes, dont 110 enseignants chercheurs, 40 chercheurs et plus de 100 doctorants, post-doctorants et ingénieurs contractuels. Cet effectif est divisé en 13 équipes de recherche, structurées en 5 départements [1] :

- Méthodes et Modèles Formels – M2F : Recherche dans la théorie des automates, la logique et l'algèbre. Les différents objectifs :
    - créer des programmes et des protocoles informatiques dignes de confiance (équipe MTV)
    - améliorer les requêtes dans les bases de données et de connaissance (équipe RATIO)
    - contrôler les robots et les systèmes de transport intelligents (équipe DART)
- Support et AlgoriThmes pour les Applications Numériques hAutes performanceS - SATANAS : Recherche dans la réalisation efficace de simulations frontières. Les différents objectifs :
    - contribuer à la réalisation d'outils indispensables à la mise en œuvre efficace de simulations frontières (équipe AAHP)
    - recherche dans l'exploitation efficace de machines parallèles (équipe SEHP)
- Systèmes et Données - SeD : Recherche à simplifier la mise en œuvre des outils numériques, améliorer leurs performances, assurer un degré élevé de confiance sur leur bon fonctionnement et proposer une utilisation simple pour leurs utilisateurs finaux. Les différents objectifs :
    - Recherche sur le cycle de vie des données, de leur production à leur visualisation pour leur exploitation mais aussi leur stockage, leur modélisation ou encore leur analyse (équipe BKB)
    - développement de systèmes logiciels qui évoluent dans des environnements complexes (équipe PROGRESS)
- Image et Son – I&S : Recherches en acquisition, traitement, analyse, modélisation, synthèse et interaction de médias audiovisuels. Les différents objectifs :
    - conception de systèmes d'acquisition, de restitution et d'interaction multimodale, et étude des processus d'interaction humain-machine basés sur de tels systèmes (équipe MANIOC)
    - Recherche sur les thématiques du traitement de l'image, de la vidéo et du son (équipe TAD)

- Combinatoire et Algorithme - COMBALGO : Recherche dans la manipulation efficace des données. Les différents objectifs :
    - Recherche dans l'organisation, la communication ou le travail ensemble efficace d'entités communicantes dans un environnement non centralisé (équipe AD)
    - Recherche dans l'obtention de théorèmes décrivant des structures discrètes, leurs relations, et leur comportement (équipe CI)
    - Développement de la théorie des graphes et sur ses applications, notamment dans le domaine des communications dans les réseaux (équipe GO)

Le LaBRI est placé en plein cœur de l'innovation en informatique. Son activité de recherche conduit à la soutenance de nombreuses thèses, la publication d'ouvrages et d'articles scientifiques. Il participe également de manière directe à la création de start-ups. Son financement provient de contrats régionaux, de projets européens, de l'Agence National de Recherche (ANR) et de contrats bipartites avec des entreprises tel que Thales et Ubisoft.

Le laboratoire a également reçu de nombreux prix et distinctions pour ses différentes activités dans la recherche comme par exemple l'attribution de plusieurs Palmes Académiques ou l'insigne de Chevalier de la Légion d'Honneur, reçue par une enseignante-chercheuse du LaBRI.

B. Equipe PROGRESS

La réalisation de mon stage s'est déroulée au sein d'une partie de l'équipe PROGRESS du département Systèmes et Données du LaBRI.

Comme dit précédemment l'équipe Progress contribue au développement de systèmes logiciels qui évoluent dans des environnements complexes. De tels systèmes peuvent avoir une existence à une échelle globale (e.g. web ou les grands réseaux du futur), locale (e.g. système d'exploitation) ou intermédiaire (e.g. essaims de drones ou véhicules intelligents). La section dans laquelle j'ai évolué est quant à elle concentrée sur les drones et leurs automatisations en essaims.

Une particularité de l'équipe Progress est qu'elle s'intéresse à la mise en œuvre de systèmes en conditions réelles. Cet objectif nécessite bien souvent, au-delà des aspects théoriques, l'exploitation d'une démarche scientifique incluant des méthodologies de simulation et d'émulation, de test, et même d'évaluation en conditions réelles. Les domaines applicatifs abordés par l'équipe sont nombreux et incluent par exemple les systèmes de surveillance aérienne ou le transport intelligent.

VI. Mes missions

A. Prise en main du Drone Mambo Parrot

Lors de mon arrivée au LaBRI, il m'a été confié un drone Mambo Parrot FSV dans le but de le prendre en main avec Python pour ensuite pouvoir l'automatiser pour des tâches particulières.

Le drone Mambo est petit et léger, mesurant seulement 18 cm x 18 cm avec une forme de drone standard et équipé d'une caméra 720p permettant une retransmission vidéo live.

Figure 1 : Mini-drone Mambo Parrot FPV

Pour pouvoir contrôler le drone avec l'interface de Python j'ai utilisé la bibliothèque « Pyparrot » [2] créée par Dr. Amy McGovern, une professeure de l'université d'Oklahoma en intelligence artificielle, pour programmer les Mini-drones Parrot. Cette bibliothèque permet d'accéder aux commandes élémentaires du drone et à sa retransmission vidéo en direct.

Pour faire réaliser des mouvements élémentaires au drone, j'utilise la classe fly_direct(roll, pitch, yaw, vertical_movement, duration) de pyparrot permettant d'agir sur le roulis, le tangage, le lacet ainsi que la commande de puissance. La bibliothèque comporte également des commandes basiques tel que le décollage, l'atterrissage, la récupération des états des capteurs, etc.

```
def avant(mambo):
    mambo.fly_direct(roll=0, pitch=20, yaw=0, vertical_movement=0, duration=0.1)

def arriere(mambo):
    mambo.fly_direct(roll=0, pitch=-20, yaw=0, vertical_movement=0, duration=0.1)

def gauche(mambo):
    mambo.fly_direct(roll=-20, pitch=0, yaw=0, vertical_movement=0, duration=0.1)

def droite(mambo):
    mambo.fly_direct(roll=20, pitch=0, yaw=0, vertical_movement=0, duration=0.1)

def haut(mambo):
    mambo.fly_direct(roll=0, pitch=0, yaw=0, vertical_movement=20, duration=0.1)

def bas(mambo):
    mambo.fly_direct(roll=0, pitch=0, yaw=0, vertical_movement=-20, duration=0.1)

def rotation(mambo, deg):
        mambo.turn_degrees(deg)
```

Figure 2 : Code des mouvements élémentaires du drone

B. Interprétation Vidéo

1. Récupération Stream Vidéo de la caméra

Tout d'abord, le drone retransmet la vidéo de sa caméra par une connexion wifi au format RTSP. La RTSP, Real Time Streaming Protocol, est un format permettant de contrôler un serveur de média à distance. Mon but est de pouvoir interpréter la retransmission grâce à la bibliothèque python « OpenCV » (Open Computer Vision).

OpenCV est une bibliothèque libre, initialement développée par Intel, spécialisée dans le traitement d'images en temps réel et la lecture de contenu vidéo ou image. Mon usage de cette bibliothèque se tournera principalement sur l'interprétation vidéo pour la détection de couleurs, de marqueurs Aruco ainsi que la capture d'écran en temps réel.

Cependant cette bibliothèque présente des lacunes dans le traitement de vidéos sous format RTSP. Pour pouvoir contourner ce problème, j'ai décidé de retransmettre le Stream de la caméra grâce au lecteur multimédia « VLC Player » et d'interpréter la vidéo en réalisant une capture vidéo en direct de mon écran grâce à OpenCV.

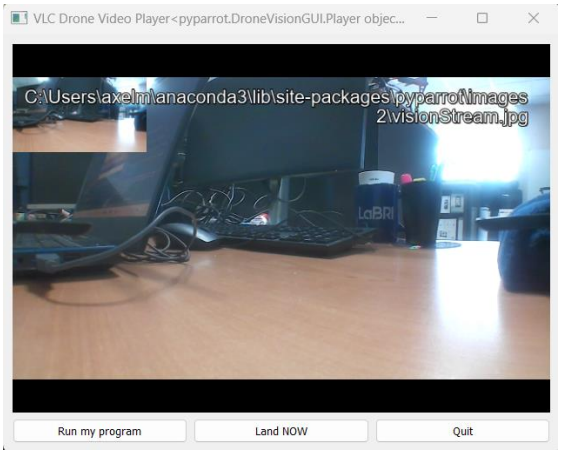

Figure 3 : fenêtre VLC de retransmission de la vision du drone

Une fois le lecteur VLC ouvert, on peut observer 3 boutons permettant de communiquer des informations particulières au drone. Le bouton « Run my program» permet de communiquer au drone une suite de commandes prédéfinies dans le programme. Le bouton « Land NOW » permet de faire atterrir d'urgence le drone. Le bouton « Quit » permet de quitter l'interface de VLC et de se déconnecter du drone.

2. Détection des Couleurs

Pour détecter une couleur particulière sur une image, on procède à un filtrage de l'image. Pour cela, on utilise la fonction « inRange() » d'OpenCV qui prend en compte 3 arguments : l'image à traiter, la limite la plus basse et la plus haute de la couleur voulue (dans le schéma de couleur RGB). Cette dernière retourne un masque dans lequel les pixels de couleur compris entre les limites sont en blanc et le reste en noir. En appliquant finalement le masque obtenu sur l'image originale, on obtient la zone de couleur voulue issue de l'image.

On peut voir l'application de ce processus ci-dessous dans un exemple de détection de la couleur rouge sur une palette de couleurs. En réalisant cette détection sur chaque frame, on obtient la détection sur la vidéo live du drone.

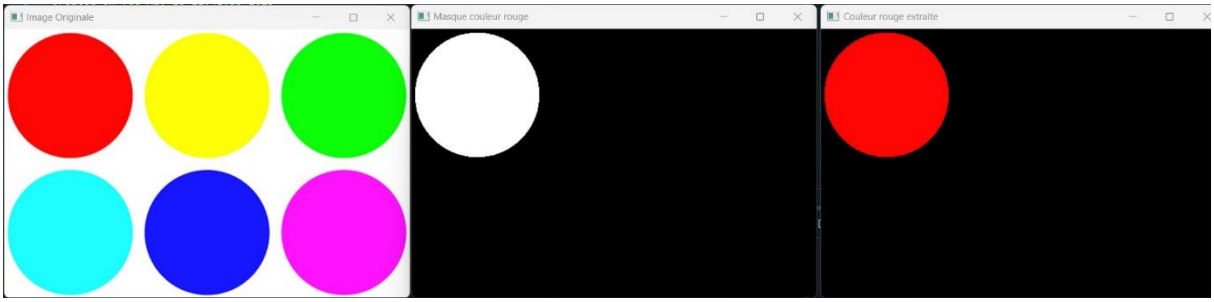

Figure 4 : Exemple de traitement d'image pour une détection de la couleur rouge. Dans l'ordre, la palette, le masque obtenu et le résultat de la superposition du masque sur la palette [3]

3. Détection des marqueurs Aruco

Un marqueur Aruco est une grille pouvant avoir des dimensions de 4x4 jusqu'à 7x7, composée de cases noires et blanches représentant une valeur. Il est similaire à un QR code mais son usage me sera plus facile pour mon application car il est bien moins complexe. C'est un système performant réalisé par un algorithme optimisant au maximum les différences entre chaque valeur pour éviter des erreurs de détection.

Pour pouvoir les détecter, on réalise tout d'abord une présélection de possibles candidats marqueurs avec un passage de l'image étudié à travers différents filtres.

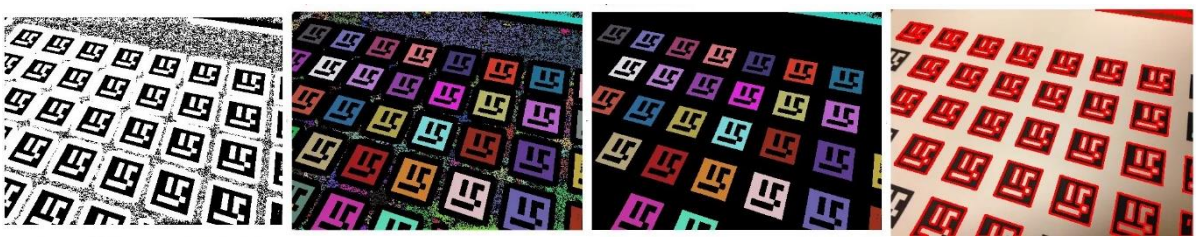

Figure 5 : Traitement d'image pour détection de marqueur Aruco [4]

On recherche ensuite les angles des rectangles des marqueurs pour pouvoir ensuite les étudier et obtenir leurs valeurs. Pour cela, on utilise l'algorithme Douglas-Peucker qui sert à simplifier un polygone ou une ligne brisée en supprimant des points.

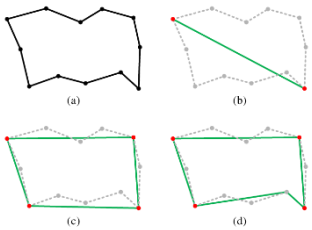

Figure 6 : Exemple d'application de l'algorithme Douglas-Peucker

Cela nous permet également de retirer les possibles candidats aux formes non rectangulaires. On remet finalement en perspective les derniers candidats (figure 7) et on les binarise pour obtenir une grille binaire aux mêmes dimensions que celle du marqueur (figure 8), 1 pour blanc et 0 pour noir, représentant le marqueur pour le comparer avec la bibliothèque des marqueurs Aruco.

Figure 7 : Remise en perspective du marqueur [4]

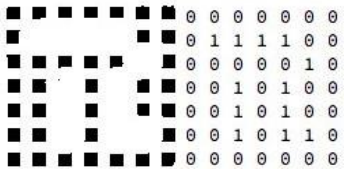

Figure 8 : Cadrillage et binarisation du marqueur [4]

La connaissance de la position des coins des marqueurs permet l'accès à sa localisation et son orientation. Ces données permettent un grand usage pratique du marqueur Aruco notamment pour calibrer des outils, orienter, etc.

4. Détection du nombre de doigts levés

Afin de communiquer de manière simple et visuelle avec le drone, j'ai décidé d'utiliser ma main pour lui assigner différentes tâches.

Pour réaliser la détection du nombre de doigts levés d'une main, il est nécessaire d'utiliser un logiciel d'intelligence artificielle permettant de détecter les différentes parties de la main. Pour cela, on fait appel à la bibliothèque python opensource « Mediapipe » [5], développée par Google.

Mediapipe est une suite de bibliothèques et d'outils permettant d'intégrer rapidement des techniques d'intelligence artificielle (IA) et d'apprentissage automatique (ML) au sein de programmes. Cet apprentissage par le traitement de grandes banques de données permet à un programme informatique de classifier, prédire des phénomènes avec une précision dépendant de la qualité et la quantité des données utilisées. Elle possède notamment une bibliothèque se concentrant sur la détection de différents points de repère de la main. Cette intelligence artificielle reconnait les emplacements de 21 points caractéristiques de la main (figure 9) et peut restituer leurs localisations, donnant accès aux principales informations des mouvements de la main étudiée. Elle a été fondée sur la base du « machine learning », entrainée avec environ 30 mille différentes images de mains.

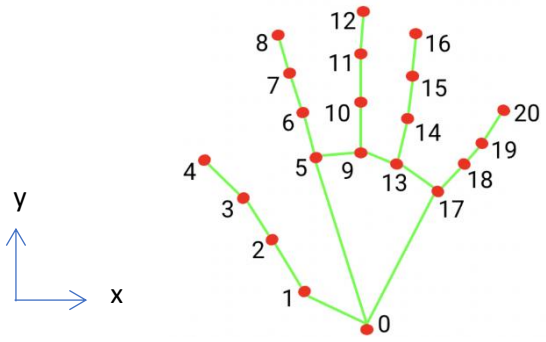

Figure 9 : Les différents points de repère de la main et leurs valeurs au sein de la bibliothèque Mediapipe [5]

Pour pouvoir déterminer si un doigt est levé, on fixe une distance caractéristique comme la moitié de la distance entre le point 0 et 9. On considère que le doigt est levé lorsque la distance entre le point de repère placé à la base et à la fin du doigt est supérieur à la distance caractéristique définie précédemment (fonction Annexe A). Par exemple, pour déterminer si l'index est levé on compare la différence entre les coordonnées en y des points 8 et 5 à la différence entre les coordonnées en y des points 9 et 0. On réalise la même opération pour les restes des doigts sauf pour le pouce pour lequel on compare la différence entre les coordonnées en x des points 5 et 4.

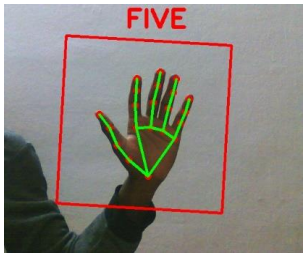

Figure 10 : Exemple de reconnaissance du chiffre 5 montré avec une main

C. Réalisation de tâches précises à des endroits désignés

Dans le cadre d'une présentation du drone dans un salon, mon objectif était d'automatiser le drone pour le rendre le plus interactif possible ainsi que de lui permettre de réaliser des tâches pouvant avoir une application concrète. Pour cela, j'ai cherché à l'automatiser dans le but qu'il puisse déterminer un trajet à partir d'une carte et réaliser des tâches assignées, grâce à l'interprétation vidéo vue précédemment, à différents points donnés.

1. Grille interactive modélisant l'environnement

Pour pouvoir obtenir une modélisation de l'espace dans lequel se situe le drone, j'ai programmé une grille interactive, grâce à la bibliothèque Python « Pygame », permettant à l'utilisateur d'indiquer l'emplacement des obstacles, du point de départ et d'arrivée ainsi que les endroits où réaliser une tâche donnée.

Pygame est une bibliothèque libre multiplateforme qui facilite le développement de jeux vidéo temps réel avec le langage de programmation Python.
J'ai choisi cette bibliothèque car elle facilite la programmation multimédia: interface graphique, entrées claviers, etc. Grâce à ces caractéristiques, lorsque le programme est lancé, une grille apparait aux dimensions données et ajoute des objets suivant les touches appuyées (ex : clic gauche, appui clavier de la touche 'D'…).

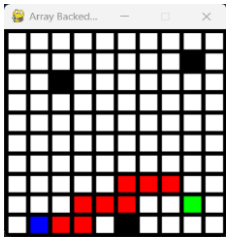

Figure 11 : Exemple d'une grille remplie. En rouge les obstacles, en bleu le point de départ, en vert le point d'arrivée et en noir les points où réaliser les tâches assignées.

Cette grille remplie par l'utilisateur permet l'obtention des coordonnées nécessaires à la détermination du trajet à emprunter par le drone. Pour cela, je récupère les coordonnées des obstacles et je recherche les trajets les plus courts du départ au premier point, du premier point au deuxième point, etc, jusqu'à l'arrivée grâce à un algorithme.

2. Algorithme de détermination du trajet le plus court

Afin de déterminer le trajet le plus court à emprunter pour le drone, une adaptation de l'algorithme A* pour Python semble la plus adaptée [6].

A* est un algorithme de recherche de chemin dans un graphe entre un nœud initial et un nœud final, inspiré de l'algorithme de Dijsktra. Il utilise une évaluation heuristique sur chaque nœud pour estimer le meilleur chemin y passant et visite ensuite les nœuds par ordre de cette évaluation heuristique. Pour cela, l'algorithme tente de minimiser la somme F = G + H avec G la distance parcourue depuis le point de départ et H une estimation de la distance à parcourir restante déterminée grâce au théorème de Pythagore. Il choisit donc à chaque étape la case lui permettant de se rapprocher le plus du point d'arrivée, tout en conservant dans une liste secondaire les autres possibilités paraissant moins adaptées au premier abord. Cette liste permet de choisir une autre option dans le cas où le chemin choisi ne serait pas optimal.

L'algorithme Python de A* renvoie sous forme d'une liste de coordonnées le trajet déterminé pour le drone. Pour pouvoir faire réaliser le trajet à mon drone, j'ai créé un programme (Annexe B) comparant les coordonnées des différents axes. Je prends pour origine des axes des abscisses et des ordonnées le coin supérieur gauche de la grille. Si la coordonnée en x au rang i est égale à la coordonnée en x au rang i+1 cela signifie que le drone reste sur la même colonne. On compare donc les coordonnées en y de deux rangs, si elle augmente du rang i au rang i+1 cela signifie que le drone doit avancer vers le bas et au contraire si elle diminue cela signifie que le drone doit avancer vers le haut. On réalise le même processus si la coordonnée en y au rang i est égale à la coordonnée en y au rang i+1. On compare les coordonnées en x de deux rangs, si elle augmente du rang i au rang i+1 cela signifie que le drone doit avancer vers la droite et au contraire si elle diminue cela signifie que le drone doit avancer vers la gauche. En retranscrivant ensuite cela sous forme de commandes de déplacement, je peux faire réaliser au drone le trajet demandé. Un extrait du résultat obtenu est visible ici : vidéo.

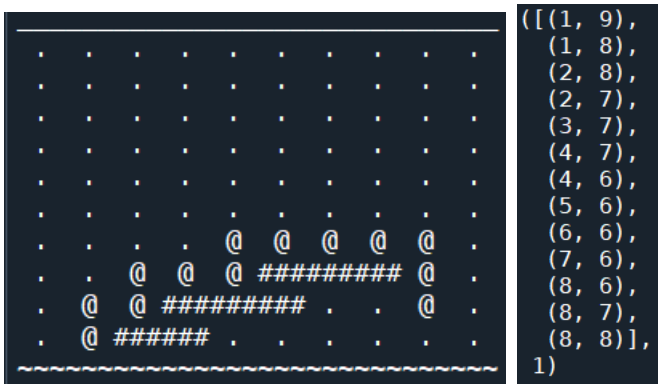

Figure 12 : Exemple de trajet de la figure 11 calculé à l'aide de l'algorithme A* du point de départ au point d'arrivée avec les '#' désignant les obstacles et les '@' désignant le trajet du drone, ainsi que la liste des coordonnées du trajet évoqué précédemment

En associant les différents processus vus précédemment, on peut assigner au drone différentes opérations à réaliser grâce à des couleurs, des signes de mains ou des marqueurs Aruco. Après avoir ensuite rempli une grille modélisant l'environnement du drone et des localisations précises où une action préprogrammée doit être réalisée, le drone pourra choisir le chemin le plus court pour réaliser les tâches demandées, telles qu'une pirouette ou modifier son altitude, et se poser au point d'arrivée désigné sur la grille.

D. Chifoumi

1. Drone contre humain

Afin de pouvoir démontrer les capacités du drone automatisé par Python, mon tuteur de stage m'a confié la mission de programmer mon drone pour pouvoir jouer au jeu du Chifoumi avec ce dernier. Le jeu du Chifoumi, aussi appelé pierre-papier-ciseaux, oppose deux joueurs montrant en même temps avec leurs mains soit une pierre, un papier ou des ciseaux. La pierre bat les ciseaux, les ciseaux battent le papier et le papier bat la feuille.

Le drone montre son choix en se positionnant au-dessus d'une des trois feuilles, représentant chacune un élément du jeu (pierre, feuille ou ciseaux), situées devant lui et sur ses côtés. Cependant ce mouvement d'avancée et de retour n'est pas extrêmement précis. A cause des variations de mouvement du drone pendant son vol stationnaire et de l'irrégularité de ses commandes de mouvements, le drone ne retourne pas exactement à sa position initiale, ce qui est source de problèmes pour les futurs déplacements lors des parties suivantes. Afin de pallier à ce problème, j'ai positionné un marqueur Aruco en face du drone car je n'ai accès qu'à la caméra frontale. Le drone, après son envol, stocke la longueur initiale d'un côté du marqueur pour ensuite s'en servir de référence de distance. Durant son vol, le drone compare en permanence cette longueur initiale avec la longueur du côté en temps réel pour pouvoir déterminer si il est trop proche, trop loin ou à la bonne distance. De même, pour pouvoir se recentrer, le drone compare la position initiale du centre du marqueur avec celle mesurée en temps réel afin de déterminer si il est trop à gauche, trop à droite ou au centre.

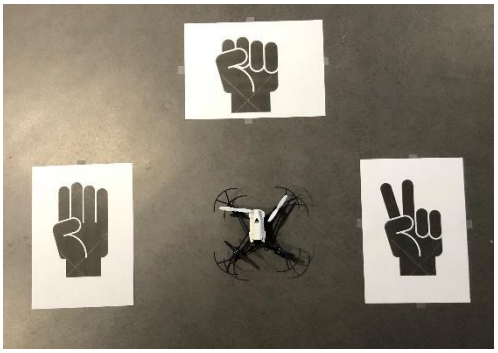

Figure 13 : installation du jeu chifoumi drone contre humain

Une fois positionné au-dessus de son signe joué, le drone détermine le signe de main réalisé par l'adversaire. Pour cela, j'ai ré-utilisé la bibliothèque de Mediapipe de reconnaissance des mains vue précédemment. Les ciseaux prennent donc le nombre de doigts montrés de 2, la pierre de 0 et la feuille de 5. Le signe interprété par le drone est ensuite comparé grâce à un dictionnaire contenant chaque signe en clé et le signe le battant en valeur. Le drone réalise une figure suivant l'issue de la manche : salto si il a gagné, descends et remonte si perdu, tourne la tête si égalité.

Un extrait du résultat obtenu est visible ici : vidéo.

2. Drone contre drone

Pour pousser encore plus loin l'activité dérivée du jeu du Chifoumi, nous avons, avec mon tuteur, décidé de complexifier encore un peu plus la tâche en tentant de faire jouer deux drones entre eux. L'objectif serait que chaque drone ait trois emplacements désignant chacun un symbole différent du jeu. Les drones se déplaceraient, en se tournant le dos, à l'endroit désignant leur symbole choisi puis se retourneraient en même temps, interpréteraient le symbole joué par le drone adverse et réaliseraient chacun une figure célébrant soit la victoire, la défaite ou l'égalité.

a) Connexion à un essaim

Pour pouvoir réaliser cette tâche, la première étape est de pouvoir connecter les deux drones Mambo à mon ordinateur. Le code source du module « Pyparrot » utilise la bibliothèque Python « Socket » [7] servant d'interface logicielle avec les services du système d'exploitation d'un ordinateur, plus particulièrement avec les services réseaux. Le programme récupère ensuite tous les réseaux connectés à l'ordinateur et se connecte à celui avec l'adresse IPv4 d'un drone Mambo, « 192.168.99.3 ». Muni d'une carte réseau, je me suis connecté aux deux drones par wifi pour essayer de pouvoir contrôler les deux en même temps.

Cependant, pour faciliter leur usage, le fabricant « Parrot » a donné la même adresse IPv4 à tous les drones Mambo. Les deux drones ne peuvent pas être différenciés pour la connexion et le programme se connecte donc à chaque fois sur le drone connecté sur la carte réseau principale de l'ordinateur. J'ai tout d'abord tenté de modifier l'adresse IPv4 locale d'un des deux drones mais le système du drone interdisait ensuite toute connexion. J'ai alors essayé de mettre un pare-feu sur mon application, bloquant ainsi l'accès au réseau d'un des drones pour pouvoir contrôler celui sur lequel le programme se connectait. En activant et désactivant cette contrainte, je pensais pouvoir établir une connexion avec les deux drones. Malheureusement, cette méthode n'a pas fonctionné pour des raisons que je ne maitrise pas. Je suppose que, même en bloquant l'application utilisée pour Python, le programme passe par les commandes fondamentales de l'ordinateur esquivant donc ce pare-feu.

J'ai donc utilisé la fonction « bind(host, port)» de la bibliothèque Python Socket qui affecte l'adresse spécifiée dans le paramètre « host » à la socket référencée ainsi que le port d'écoute. Cependant cette méthode fixe l'accès au wifi qui va être utilisé et ne peut ensuite être à nouveau modifié pour la seconde connexion. Les deux drones ne peuvent donc pas être connectés en même temps. Pour contourner ce problème, je connecte donc tout d'abord le premier drone en spécifiant l'accès au premier port wifi, lance son programme puis connecte le second drone et lance son programme en spécifiant l'accès au second port wifi.

Une autre solution plus viable, serait d'utiliser une Raspberry Pi pour connecter un drone à l'ordinateur et le second à cette dernière et de lancer les deux programmes indépendamment.

b) Synchronisation des deux drones

Le chifoumi opposant les deux drones est réalisé en lançant deux programmes indépendants chacun connecté à un drone. Cependant, il est nécessaire pour le bon déroulement de la partie que les drones soient synchronisés dans leurs mouvements. Ma meilleure option accessible était de les accorder grâce à un clap de mains. Pour cela, j'ai adapté un programme pour permettre aux drones de pouvoir détecter un son du type d'un clappement de mains via le micro de mon ordinateur.

Un son de clap de main est caractérisé par sa grande amplitude et sa courte durée. Grâce à la bibliothèque « Pyaudio » [8], facilitant l'accès aux ports audio de l'ordinateur par Python, il est possible de récupérer le son enregistré en live par le micro afin de pouvoir le traiter. Pour pouvoir filtrer la piste audio récupérée, je calcule la valeur efficace de son amplitude définie par la racine carrée de la

moyenne de son carré. Je la compare ensuite avec une valeur seuil que j'ai définie initialement. Lorsqu'elle est au-dessus de cette valeur, je rajoute plus un à une variable servant de compteur de la longueur du bruit entendu. Une fois le son fini, la condition sur l'amplitude n'est plus remplie, je compare donc la valeur de la longueur du son avec une longueur définie au préalable à partir de laquelle je considère que le son est aussi court que celui d'un clap. Si la longueur est inférieure à cette valeur, je considère que le son entendu est un clap autrement je ne considère pas ce son.

Une fois le clap de main interprété par l'ordinateur (fonction Annexe C), les drones se déplacent en face du marqueur Aruco représentant leur symbole choisi de manière aléatoire. Ils se retournent ensuite pour pouvoir connaitre le signe joué par le drone adverse.

c) Comprendre le symbole joué par l'adversaire

Pour pouvoir comprendre le symbole joué par le drone adverse, je me suis tout d'abord heurté à la problématique de comment pouvoir connaitre l'emplacement du drone grâce à une interprétation vidéo afin de pouvoir connaitre le symbole joué. La reconnaissance de forme grâce à une IA similaire à celle utilisée pour la reconnaissance de la main fut ma première option. Cependant, il n'en existait pas en particulier pour le drone que j'utilise et créer ma propre banque de données pour entrainer l'IA aurait été trop long. L'interprétation vidéo des couleurs pour repérer le drone m'est également venue à l'esprit mais pour cela il était nécessaire d'avoir des couleurs apparaissant uniquement sur le drone et non dans l'environnement l'entourant. N'ayant aucunes informations sur le milieu dans lequel le drone allait évoluer dans le salon « Game Fest' », j'ai finalement opté pour l'usage de marqueurs Aruco.

En plaçant un marqueur Aruco de valeur 1 sur chaque drone, on peut obtenir la position du drone adverse pour pouvoir ensuite déterminer son symbole joué. Lorsque le drone se retourne, il détecte le marqueur de valeur 1 et obtient la position de chacun de ses coins. On sait que ce dernier est situé juste au-dessus du marqueur qui va nous permettre de connaitre le signe joué. Je réalise ensuite une nouvelle capture vidéo de mon écran mais cette fois-ci avec des dimensions particulières pour limiter le champ d'interprétation et pouvoir détecter uniquement le marqueur placé en dessous du drone qui indique son jeu.

Les dimensions des marqueurs Aruco désignant les symboles du jeu sont 3 fois plus grandes que celles du marqueur placé sur le drone pour l'identification de son adversaire.

Je définis donc un moniteur à capturer d'une largeur de 4 fois la longueur d'un côté du marqueur de valeur 1 car je souhaite avoir un peu de marge dans la fenêtre d'acquisition vidéo pour faciliter l'interprétation et éviter les erreurs ; un départ selon l'axe x (en rappelant que l'axe x commence à gauche de l'écran) à la localisation du milieu du marqueur 1 moins l'épaisseur divisée par 2 afin de centrer dans la fenêtre le marqueur étudié ; un départ selon l'axe y (en rappelant que l'axe y commence en haut de l'écran) au maximum des coordonnées en y des coins du bas du marqueur pour isoler l'objet à détecter. En réalisant une détection uniquement sur cette partie d'écran, on obtient le signe joué par le drone adverse et on peut donc déterminer l'issue du round.

Figure 14 : à gauche fenêtre de retransmission vidéo de la caméra et à droite fenêtre restreinte obtenue avec la méthode décrite dans le but d'interpréter uniquement ce qui se situe en dessous du drone.

Malheureusement limité par le temps, je n'ai pas pu finaliser cette partie du projet pour obtenir un résultat montrable en vidéo.

VII. Bilan

Ce stage a été une expérience très enrichissante pour moi car il m'a permis de découvrir le monde de la recherche qui m'était encore inconnu et de me confronter à des problématiques d'ingénieur.

Mon travail sur les drones m'a permis d'approfondir mes connaissances en Python, notamment sur son usage avec les réseaux ainsi que sa possible utilisation en lien avec l'intelligence artificielle.

Mon travail était principalement individuel, j'ai donc appris à utiliser mes connaissances et à savoir trouver seul des solutions réalisables face aux différents problèmes que j'ai rencontrés.

Ce stage m'a également permis de toucher du doigt les possibilités très vastes de la programmation informatique (reconnaissance de la main, de marqueurs, etc …) et ne fait qu'augmenter mon envie de découvrir davantage cette discipline.

J'espère enfin que les démonstrations (chifoumi, parcours avec obstacles), résultant de mon stage, créeront des vocations et feront découvrir au jeune public du salon « Game Fest' » les capacités de l'alliance de la mobilité du drone et la puissance d'interprétation de l'intelligence artificielle.

J'ai beaucoup apprécié ce stage et cela me conforte dans mon orientation vers la robotique et l'informatique.

VIII. Bibliographie

[1] LaBRI, Plaquette LaBRI 2022

[2] Dr. Amy McGovern, Bibliothèque Python Pyparrot, https://pyparrot.readthedocs.io/en/latest/

[3] OpenCV color detection and filtering with Python, https://www.bluetin.io/opencv/opencv-color-detection-filtering-python/

[4] Aruco marker detection, https://medium.com/@calle_4729/using-mathematica-to-detect-aruco-markers-197410223f62

[5] Google, Bibliothèque Python Mediapipe, https://mediapipe-studio.webapps.google.com/home

[6] Implementation of A*, https://www.redblobgames.com/pathfinding/a-star/implementation.html

[7] Bibliothèque Python Socket, https://docs.python.org/3/library/socket.html

[8] Bibliothèque Python Pyaudio, https://pypi.org/project/PyAudio/

IX. Annexe

Annexe A : Fonction permettant de détecter le nombre de doigts levés, avec « .landmark[point] » une fonction permettant d'obtenir les coordonnées du point désigné

```python
def compte_doigts(points_main):
    cnt = 0

    thresh = (points_main.landmark[0].y*100 - points_main.landmark[9].y*100)/2

    if (points_main.landmark[5].y*100 - points_main.landmark[8].y*100) > thresh:
        cnt += 1

    if (points_main.landmark[9].y*100 - points_main.landmark[12].y*100) > thresh:
        cnt += 1

    if (points_main.landmark[13].y*100 - points_main.landmark[16].y*100) > thresh:
        cnt += 1

    if (points_main.landmark[17].y*100 - points_main.landmark[20].y*100) > thresh:
        cnt += 1

    if (points_main.landmark[5].x*100 - points_main.landmark[4].x*100) > thresh/2:
        cnt += 1

    return cnt
```

Annexe B : Fonction permettant de faire réaliser au drone un trajet sous forme de coordonnées

```python
def carte(path, mambo):
    a=0
    b=0
    c=1
    for i in range(len(path)-1):
        #cas meme ligne
        if path[i][0]==path[i+1][0]:
            #comparaison coordonnée colonne pour savoir rotation gauche ou droite
            if path[i][1]==path[i+1][1]+1:
                if b==0:
                    mambo.smart_sleep(1)
                    avant(mambo, 20)
                elif b==1 and c==1:
                    mambo.smart_sleep(1)
                    mambo.turn_degrees(-90)
                    angle=angle-90
                    mambo.smart_sleep(1)
                    avant(mambo, 20)
                elif b==1 and c==0:
                    mambo.smart_sleep(1)
                    mambo.turn_degrees(90)
                    angle=angle+90
                    mambo.smart_sleep(1)
                    avant(mambo, 20)
                a=0
            else:
                if b==0:
                    mambo.smart_sleep(1)
                    avant(mambo, 20)
                elif b==1 and c==0:
                    mambo.smart_sleep(1)
                    mambo.turn_degrees(90)
                    angle=angle+90
                    mambo.smart_sleep(1)
                    avant(mambo, 20)
                elif b==1 and c==1:
                    mambo.smart_sleep(1)
                    mambo.turn_degrees(-90)
                    angle=angle-90
                    mambo.smart_sleep(1)
                    avant(mambo, 20)
                a=1
            b=0
        #cas ligne différente
        else:
            #comparaison coordonnée ligne pour savoir rotation gauche ou droite
            if path[i][0]==path[i+1][0]+1:
                if b==1:
                    mambo.smart_sleep(1)
                    avant(mambo, 20)
                elif b==0 and a==1:
                    mambo.smart_sleep(1)
                    mambo.turn_degrees(-90)
                    angle=angle-90
                    mambo.smart_sleep(1)
                    avant(mambo, 20)
                elif b==0 and a==0:
                    mambo.smart_sleep(1)
                    mambo.turn_degrees(90)
                    angle=angle+90
                    mambo.smart_sleep(1)
                    avant(mambo, 20)
                c=1
            else:
                if b==1:
                    mambo.smart_sleep(1)
                    avant(mambo, 20)
                elif b==0 and a==0:
                    mambo.smart_sleep(1)
                    mambo.turn_degrees(90)
                    angle=angle+90
                    mambo.smart_sleep(1)
                    avant(mambo, 20)
                elif b==0 and a==1:
                    mambo.smart_sleep(1)
                    mambo.turn_degrees(-90)
                    angle=angle-90
                    mambo.smart_sleep(1)
                    avant(mambo, 20)
                c=0
            b=1
```

Annexe C : Fonction permettant de détecter un clappement de main

```
def listen(self):
    try:
        #récuperation live du microphone de l'ordinateur
        block = self.stream.read(INPUT_FRAMES_PER_BLOCK)
    except IOError as e:
        self.errorcount += 1
        print( "(%d) Error recording: %s"%(self.errorcount,e) )
        self.noisycount = 1
        return
    #obtention de la valeur efficace de l'amplitude
    amplitude = get_rms( block )
    #comparaison de l'amplitude avec une valeur prédefinie
    if amplitude > self.tap_threshold:
        #gestion du cas d'un son qui pourrait potentiellement être un clap
        self.quietcount = 0
        #variable permettant d'avoir une notion de la longueur en temps du son
        self.noisycount += 1
        #gestion d'un environnement bruyant
        if self.noisycount > OVERSENSITIVE:
            #baisse la sensibilité
            self.tap_threshold *= 1.1
    else:
        #cas d'un silence
        #comparaison avec valeur prédefinie pour savoir si on considère le son
        #comme assez court pour être un clap
        if 1 <= self.noisycount <= MAX_TAP_BLOCKS:
            #retourne qu'un clappement a été détecté
            return True
            self.tapDetected()
        #remise à 0 de la variable pour le prochain son détecté
        self.noisycount = 0
        self.quietcount += 1
```