REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Mouloud MAMMERI de Tizi-Ouzou

Faculté de Génie Electrique et d'Informatique

Département d'informatique

MEMOIRE

DE FIN D'ETUDE

En vue de l'obtention d'un diplôme de Master Académique en Informatique.

Option : conduite de projet informatique.

Planification de trajectoires pour un véhicule aérien autonome (UAV).

Encadré par :

- *Mr H.Radja.*
- *Mr S.Yahiaoui.*

Réalisé par :

- *Mr Sadallah Nassim.*
- *Mlle Babou Ouahiba.*

Promotion 2017-2018

Remerciements et dédicaces

Nous remercions Dieu ALLAH tous puissant qui nous a donné le courage de confronter ce nouveau domaine, et la force de réaliser ce travail.

Nous tiendrons à remercier les personnes qui grâce à eux, nous ne pourrions jamais être là, nos parents et ainsi que toutes nos familles.

Nous tiendrons à remercier ***Dr YAHIAOUI****, chercheur au centre de recherche scientifique et technique CERIST, et* ***Mr RADJA****, enseignant à l'université UMMTO de Tizi Ouzou, d'avoir acceptés d'encadrer ce travail. Nous les remercions vivement pour leurs aides, leurs soutiens, leurs conseils et leurs générosités.*

Nous tiendrons à remercier ***Mr EMBAREK Mohammed, Mr DAOUI Mehammed et Mr SADALLAH Madjid****, pour leur aide précieuse, sans oublier les autres personnes qui travaillent au niveau du CERIST pour leur générosité.*

Nous voudrions remercier très spécialement ***les membres de jury*** *d'avoir acceptés de participer à ce jury de mémoire.*

Enfin, nous tiendrons à remercier toute personne ayant contribué de près ou de loin à l'élaboration de ce travail notamment nos ami(e)s et nos proches.

Remerciements et dédicaces

Nous dédions ce modeste travail à la mémoire de nos grands-parents, que dieu bénisse leurs âmes.

A ce qui ont éclairé notre vie, nos chère parents, pour leurs aides et leurs soutiens toute au long de nos études, et qui ont fait de nous ce que nous sommes aujourd'hui et nous espérons qu'un jour nous serons capables de leurs donner au moins le minimum car quoi qu'on fasse on arrivera jamais à leurs rendre tous.

A toutes nos familles et nos proches.

A tous nos collègues et nos ami(e)s.

Résumé

Ce projet traite le sujet de la planification de trajectoires pour un drone sous contraintes d'évitement d'obstacles statiques et dynamiques. Cette trajectoire doit être optimale vis-à-vis la sécurité et le temps de parcours, et tout ça à l'aide d'un algorithme de recherche qui utilise la recherche hors ligne et en ligne conjointement.

D'une part, un état de l'art a été présenté dans ce mémoire sur les drones et sur les méthodes de planification de trajectoires, dont nous avons cité brièvement quelques méthodes et algorithmes qui existent concernant la planification de chemin pour les robots en générale et pour les UAVs en particulier.

Un drone quadri rotor a été réalisé d'une côté, à base de quatre moteurs électriques (Brushless), fixé sur l'extrémité de châssis en plastique dure. Du l'autre côté, un environnement de simulation pour le système de drone a été réalisé sous UBUNTU 16.04 LTS à l'aide ….

D'une autre part, une méthode de planification de trajectoire Multi Objectifs a été implémentée d'abord dans un environnement de simulation, puis sur notre drone quadri rotor. Cette méthode tienne compte de deux objectifs qui sont le temps de parcours et le niveau de sécurité. Elle est basée sur la recherche hors ligne OFPS en utilisant la méthode FM^2 pour éviter les obstacles statiques, et sur la recherche en ligne ONPS en utilisant une carte SIM dynamique pour contourner les obstacles inattendus.

Enfin, des tests sont établis sur l'expérimentation synthétique et réaliste de la planification de trajectoires pour notre drone accompagnée des résultats et discutions.

Mots clés : UAVs, Planification de trajectoires, Recherche hors ligne, Recherche en ligne, FM^2, A^*.

Sommaire

Liste des figures

Liste des tableaux

Introduction générale

Les dernières années ont vu émerger au sein des sciences de l'automatique le domaine de la robotique aérienne, qui est un domaine de recherche très actif principalement grâce aux grandes potentialités qu'offrent les véhicules aériens autonomes (UAVs en anglais) pour aider, voire remplacer l'être humain dans plusieurs domaines d'application. Grâce aux récentes avancées technologiques dans le domaine de la robotique, de la miniaturisation et de l'électronique embarquée, la conception et la commercialisation de dispositifs UAV efficaces, à coût raisonnable, et dotés d'importantes capacités de navigation autonome est donc devenue possible. Ces dispositifs promettent de nouvelles possibilités tant pour des applications civiles que militaires, en particulier pour des tâches difficiles ou dangereuses, par exemple dans le secteur manufacturier ou la recherche et sauvetage. Pour ce faire, les UAVs sont dotés de capacité de perception et de recueil d'information sur leur environnement afin de pouvoir accomplir leurs différentes tâches de localisation, planification et de navigation. Ils doivent ainsi se déplacer de façon autonome et être capables d'éviter les obstacles éventuels.

L'une des problématiques dans le domaine de la robotique aérienne relatives à la planification de trajectoire est la navigation du véhicule aérien tout en évitant les obstacles. Le problème devient plus complexe avec la présence d'obstacles dynamiques car différentes variables doivent être prises en considération simultanément : la position de l'obstacle, la vitesse de mouvement de l'obstacle, l'état de l'environnement, etc.

Nous nous intéressons dans ce travail à la navigation d'un UAV dans un environnement complexe contenant des obstacles statiques et dynamiques. Plus précisément, nous abordons la problématique de l'exécution de trajectoires planifiées pour les UAVs, lors de la présence des obstacles que soient statiques ou dynamiques. En se basant sur l'étude des différentes approches utilisées dans ce contexte, nous élaborons une nouvelle méthode de planification de trajectoire qui répond non seulement aux besoins fondamentaux de sécurité et d'efficacité, mais aussi à celui d'optimisation.

La méthode proposée a pour objectif d'assurer deux objectifs essentiels, à savoir la sécurité vis-à-vis des obstacles statiques ou dynamiques, et le temps de parcours. Par conséquent, elle utilise conjointement deux types de recherche : la recherche hors ligne avec l'algorithme OFPS (Offline Path Search) et la recherche en ligne avec l'algorithme ONPS (Online Path Search).

Notre projet est réalisé au sein de l'équipe UbiSys (Ubiquitous Systems) dans la division Théorie et Ingénierie des Systèmes Informatique DTISI du centre de recherche scientifique et technique CERIST à Ben Aknoun, Alger.

Le présent mémoire s'articule autour de cinq chapitres :

Dans le premier chapitre, nous présentons un état de l'art sur les drones. Nous commençons par définir le dispositif appelé drone, puis nous décrivons les différentes composantes de son système. Par la suite, nous retraçons brièvement l'historique de ces dispositifs, leur évolution dans le temps, ainsi que les différents critères de leur classification. Nous énumérons enfin quelques domaines d'utilisation des drones et nous présentons la réglementation relative à leur utilisation dans les applications civiles.

Le deuxième chapitre est consacré à la présentation de l'état de l'art relatif à la planification de trajectoires dédiée aux robots en générale. Nous détaillons quelques-unes des méthodes les plus utilisées. Nous proposons une classification de ces méthodes en trois grandes approches : l'approche géométrique, l'approche de l'espace de vitesse et l'approche de recherche dans un graphe. Cette classification nous permet de discuter des forces et des limites de chacune de ces approches.

Le troisième chapitre est dédié à la description de quelques méthodes de planifications de chemin dédiées aux UAVs. Nous regroupons ces méthodes en trois groupes : les méthodes en ligne, les méthodes hors ligne et les méthodes qui combinent les deux types précédents. Le principe de chaque méthode est détaillé puis discuté en termes d'avantages et d'inconvénients.

Le quatrième chapitre porte sur la présentation de notre contribution. Plus précisément, nous décrivons la conception et l'implémentation d'une variante améliorée de l'une des méthodes présentées dans le chapitre trois, à savoir Multi-Objective Path Planning (MOPP). Dans un premier temps, nous décrivons le principe de cette méthode. Par la suite, nous introduisons notre contribution, et nous détaillons les méthodes et algorithmes utilisées (principalement la méthode FM^2 et l'algorithme A*), etc. Le principe algorithmique et les différents algorithmes sont développés et expliqués en dernier lieu.

Le cinquième chapitre a pour objectif de présenter l'ensemble de tests de validation que nous avons réalisés, ainsi que la discussion de leurs résultats. Nous commençons d'abord par présenter l'environnement de l'expérimentation. Par la suite, nous décrivons les différentes expérimentations que nous avons réalisées pour la planification de chemin d'un drone quadri rotor avec notre méthode. Deux types d'expérimentations sont décrites : synthétiques (avec

simulation) et réelles (sur le drone en environnement réel). Nous commentons et discutons les résultats de chacun de ces tests de validation.

Ce mémoire s'achève avec une conclusion générale et une présentation de nos perspectives.

I.1. Introduction

Les recherches et les développements liés aux drones ont été très actives au cours des dernières années dans la robotique aérienne, reposent sur la capacité de contrôle de leur vol dans des environnements complexes, cette capacité passe par la maîtrise des lois de contrôle associées à ces engins, de la cartographie de l'environnement, et la prise en compte de leur interaction avec l'environnement. Ces recherches sont motivées par les récentes avancées technologiques, où les drones sont devenus capables de transporter des caméras, des capteurs, des équipements de communication ou d'autres dispositifs. C'est ce qui rend ces engins utilisables dans diverses missions complexes et pénibles pour un être humain tel que la surveillance, la recherche, le sauvetage, le renseignement, les opérations de combat, etc.

Dans ce chapitre nous allons présenter dans un premier temps le drone, montrant ses différentes parties le constituant ainsi que l'historique (le parcours) d'évolution des drones. Dans un deuxième temps, nous allons énumérer les différentes classifications des drones, les capteurs et méthodes de navigation. Dans un dernier temps, nous mentionnerons les différents domaines d'utilisation et les lois concernant leur usage civil, on termine par une conclusion qui argumente notre choix.

I.2. Définition d'un Drone

Un drone est un aéronef [1] capable de voler et d'effectuer des missions diverses sans la présence humaine à bord en embarquant une charge utile, souvent contrôlé à partir d'une station terrestre. La taille d'un drone peut aller de quelques centimètres à plusieurs mètres et son endurance varie de quelques minutes jusqu'à plus de 40 heures et peut atteindre jusqu'à 20 000 m d'altitude [1] .

Le terme « Drone » est à l'origine un mot anglais apparu en Angleterre en 1935, il signifie un « faux bourdon » qui est une abeille mâle, utilisé en langue française en référence au fonctionnement bruyant et paresseux que font certains d'entre eux qui ressemble à cet insecte. D'autres appellations telles que UAV (Unmanned Aerial Vehicle) qui se traduit par « Véhicules

[1] **Aéronef** : tout appareil capable de s'élever ou de circuler dans l'air.

aériens autonomes », UAS (Unmanned Aerial System) ou encore RPAS (Remotely Piloted Aerial System) par l'ICAO[2], sont souvent utilisés.

Les drones ont plusieurs formes et configurations selon la mission qu'ils réalisent. En revanche, ils ont les mêmes parties constituant leur structure qu'on peut représenter comme suit :

- ***Les moteurs***

Ils sont responsables de créer l'énergie nécessaire à la portance du drone dans l'air en considérant la masse totale, l'altitude et l'endurance du vol. On trouve plusieurs types de moteurs (voir la figure I-1) tels que les turboréacteurs, les turbines à hélices, moteurs à pistons ou encore des moteurs électriques.

Figure I-1: Quelques types de moteurs.

- ***Les voilures***

Assurent la sustentation du drone (i.e. le maintien du drone dans l'air), ils peuvent êtres fixes (des ailes) ou tournantes (des hélices) pour l'exploitation en vol stationnaire. La figure I-2 illustre un exemple des voilures.

Figure I-2: Drone à voilure tournante (à gauche) et, à voilure fixe (à droite).

[2] **ICAO** : International Civil Aviation Organization qui est une organisation qui dépend des nations unies qui a pour rôle de participer à l'élaboration des normes pour la standardisation du transport aérien international. Le lien du site Web : http://www.icao.int.

- ***La cellule***

La cellule est l'ossature qui peut avoir plusieurs formes et qui porte les moteurs, les systèmes de bord et la charge utile ainsi que le carburant ou la batterie. La figure I-3 montre un exemple d'une cellule.

Figure I-3 : Cellule d'un drone.

- ***Les systèmes de bord***

Constitués de plusieurs équipements qui sont asservis entre eux, tels que les capteurs pour mesurer les paramètres du vol, une mémoire contenant la partie programmable du vol, des calculateurs qui sont dédiés au pilotage et la navigation, ainsi que des actionneurs qui agissent sur les commandes de vol.

- ***La charge utile***

Un ensemble d'équipements (capteurs, processeurs, calculateurs, etc.) spécifiques à la réalisation de la mission, généralement placés au-dessous de la structure comme le montre la figure I-4, dans le but d'assurer trois fonctions qui sont :

1. L'acquisition des données de l'environnement de la mission par des capteurs spécifiques aux besoins de cette dernière qui peuvent êtres de type électromagnétique (Radar), électro-optique (caméras), etc.
2. Traitement des données à bord par des calculateurs en vue de leur exploitation en vol ou au sol.
3. La sélection des informations utiles à bord pour les transmettre à la base de contrôle en temps réel ou bien les enregistrer pour un déploiement ultérieur.

Figure I-4 : Une charge utile (caméra) d'un drone.

- ***Le Système de transmission***

Il assure l'acheminement des informations envoyées par le drone et la réception des ordres émis par la station de contrôle. la transmission est assurée par la télécommunication soit en relais, qui peut être un satellite ou un autre vecteur aérien (drone ou avion), soit assurée par la portée optique sur des courtes distances (jusqu'à 150 km) qui peut être une diode LED de forte puissance ou un Laser.

- ***Intelligence embarquée***

Qu'on appelle aussi IA (Intelligence Artificielle), un ou plusieurs algorithmes embarqués sur le drone accompagnés des calculateurs dédiés associés avec les informations des capteurs. Ils offrent un certain degré d'autonomie au drone vis-à-vis du pilotage et de la navigation d'une part, et la réalisation de la mission d'une autre part. Ces algorithmes sont caractérisés par leur stabilité et optimisation vu leur usage sensible qui nécessite une interaction du drone en temps réel avec l'environnement d'évolution.

➢ ***Le Système drones***

L'ensemble des éléments nécessaires pour mettre en œuvre un drone, tel que présenté sur la figure I-5, il est composé de deux (02) segments :

- ***Le segment air***

Constitué du drone, sa charge utile ainsi que le système de transmission.

- ***Le segment sol***

Composé d'un ou plusieurs hommes qui ont un degré d'intervention plus au moins élevé et un ensemble de matériels qui est composé de :

- Matériel dédiés au lancement et la récupération du drone tels que les filets et les rails de lancement.

- Matériel dédié à la maintenance et à la réparation.
- Matériel destiné à la conduite de la mission à partir du sol en assurant la gestion du vol et la navigation, la réception des données, leur décryptage, leur analyse et leur transmission au centre de calcul.

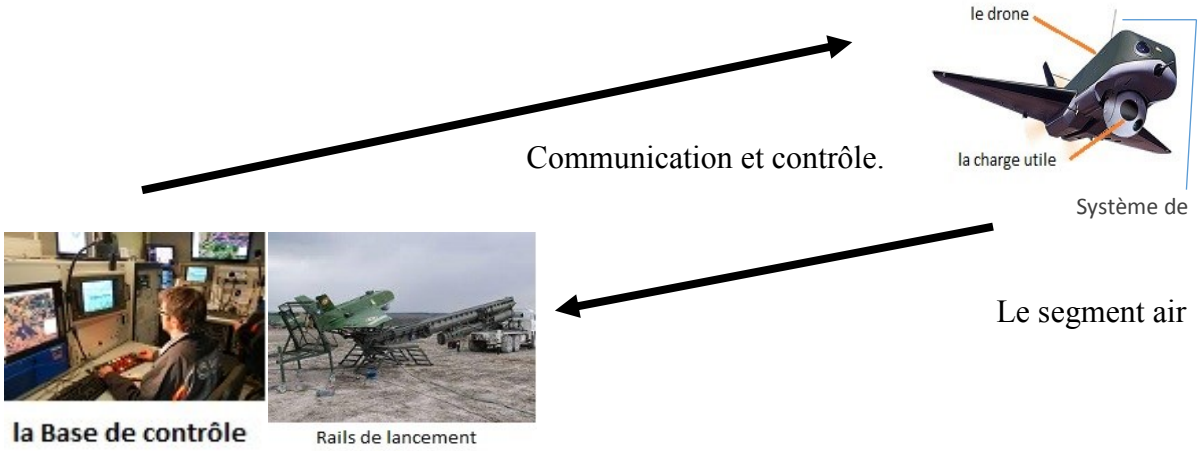


Figure I-5 : Le système drone.

I.3. Evolution des drones

Les drones d'aujourd'hui qui sont hyper évolués [2], ne sont qu'une conséquence de plusieurs recherches et évolutions dans le domaine de l'aérodynamique, partant en premier lieu d'une inspiration qui engendre une idée, donnant naissance ensuite à des réalisations primaires en vol et qui, aujourd'hui, évoluent en exponentiel.

I.3.1. L'inspiration

L'homme s'est inspiré de la nature pour construire les premiers objets volants, (voir la figure I-6) en étudiant les différents comportements d'oiseaux, insectes, etc.

Figure I-6 : Une libellule.

I.3.2. L'idée de vol vertical

La première idée était celle de Léonard De Vinci entre 1487 et 1490, il était question d'une « vis aérienne » qui consiste en vol vertical, d'ailleurs il est interprété par certains comme le précurseur des hélicoptères modernes.

La figure I-7 montre le dessin de Léonard De Vinci :

Figure I-7 : Modèle de la vis aérienne de Léonard De Vinci.

I.3.3. Premier principe d'avion sans pilote

Le premier drone a vu le jour en France avant la première guerre mondiale

- **Entre 1894 et 1897 :** Monsieur Octave Détable qui est un chercheur français en aviation a réussi à faire voler un planeur de 19 m² sans un pilote à bord en utilisant un stabilisateur (des cônes divergents). Ayant réussi sur un planeur, il voulait l'appliquer sur un avion pour assurer la contrainte de la sécurité. La figure I-8 est une image correspondante au planeur de Mr Détable :

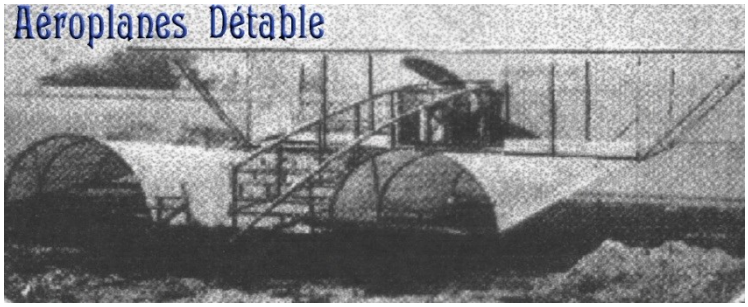


Figure I-8 : Planeur réalisé par Octave Détable.

- **En 1912** : le lieutenant Max Boucher et Mr Détable, qui sont présentés sur la figure I-9, s'engagent à la réalisation d'un avion sans pilote dont le premier essai eut lieu en 1914 résultant un échec dû aux capacités insuffisantes du moteur utilisé. Le projet fût abandonné à cause de la première guerre mondiale qui survient.

Figure I-9 : Octave Détable (à gauche) et Max Boucher (à droite).

I.3.4. L'évolution des systèmes de drones

A partir de la première guerre mondiale [3], les chercheurs dans le secteur militaire, particulièrement dans le secteur de l'aviation ont amélioré les systèmes de drones.

- **En 1916** : Archibald Low qui est un chercheur en physique et inventeur lance un projet d'avion-cible commandé à distance par des ondes de télégraphie sans fil (TSF) en Grande Bretagne qui s'intitule « Aerial Target », cet avion-cible autonome servait de cible pour entrainer les apprentis-pilotes aux tirs sans risquer d'abattre l'avion remorquant la cible. La figure I-10 montre l'Aerial Target et son inventeur :

Figure I-10 : UK, 1916, Aerial Target et son inventeur Archibald Low.

- **En 1917 :** les trois ingénieurs américains Elmer Ambrose Sperry, Lawrence Sperry et Peter Cooper Hewitt ont construit un avion radiocommandé qui porte le nom « Hewitt-Sperry Automatic Airplane » (voir la figure I-11), il héberge un gyroscope pour sa stabilisation et la direction, un baromètre pour le contrôle de l'altitude, un servomoteur et d'autres équipements pour mesurer la distance. sa mission était de porter des bombes à larguer une fois à la destination.

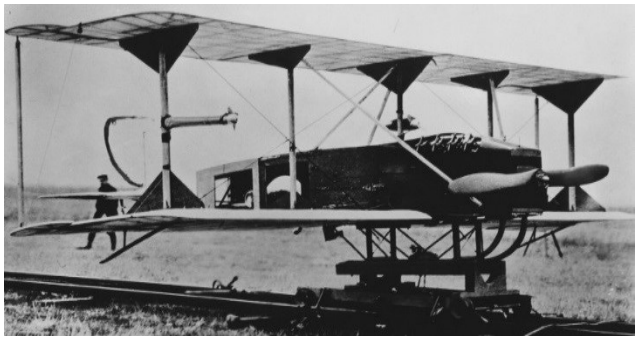


Figure I-11 : Le Hewitt-Sperry Automatic Airplane sur sa voie de lancement.

La France n'est pas exclue puisque le 2 Juillet 1917 le colonel Max Boucher reprit ses travaux abandonnés et réussi à faire voler un avion sans pilote à bord, sur 1 Km à Avord, toute en gardant le principe de Mr Détable.

- **En 1935 :** la société britannique De Havilland produit l'avion-cible « DH.82 Queen Bee » (voir la figure I-12) qui a la particularité de revenir à la base de contrôle d'une façon autonome. DH.82 était le premier avion qui porte le nom « drone » justement [2].

Figure I-12 : DH.82 Queen Bee.

D'autres drones ont été réalisés dans les années 30 tel que « Radioplane OQ-2 » des américains.

- **Entre 1935 et 1994 :** cette période a connu l'apparition des drones caractérisés par leur endurance longue et haute altitude et pouvant acquérir les informations, les traiter et les envoyer à la station de contrôle en temps réel, conçus principalement pour des missions militaires (utilisés pendant la guerre de Corée, celle de Vietnam et aussi celle de Kippour entre les nations arabes et Israël), [1], connues sous le nom ISTAR (Intelligence, Surveillance, Target Acquisition and Reconnaissance) [4]. La figure I-13 illustre quelques drones construits.

Figure I-13 : Rayen FireBe (à gauche), Lockheed D-21 (au centre) et MQ 1-A Predator (à droite).

Par ailleurs, des modèles réduits d'hélicoptère radiocommandés ont vu le jour, en 1969 le premier hélicoptère radiocommandé fût réalisé par Dieter Schlüter et qui a volé pendant 5 secondes à 3 mètres de hauteur. En 1970, le modèle « Bell Huet cobra Schlüter » été le premier produit commercialisé, il a enregistré un record de vol de 10 minutes environs.

- **Entre 1994 et 2012 :** les drones deviennent des guerriers ! En effet, les drones n'effectuent pas seulement des missions ISTAR mais peuvent aussi mener des combats (voir a figure I-14), ils hébergent une technologie révolutionnaire leur permettant une autonomie assez avancée telle que l'endurance qui peut atteindre les 40 heures

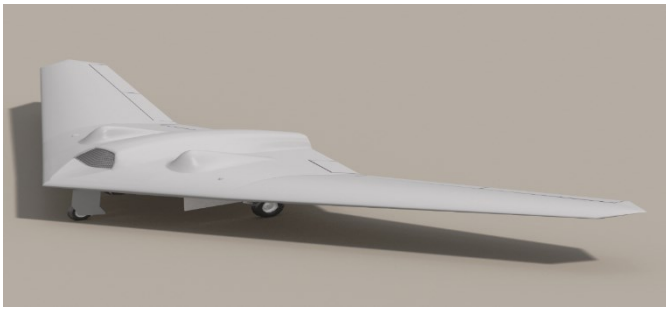


Figure I-14 : Modèle 3D de RQ-170 Sentinel.

- **Aujourd'hui :** on trouve plusieurs catégories de drones (militaire et civil) partant de quelques grammes à plusieurs tonnes en différentes formes selon les utilisations et missions pour lesquelles ils sont destinés. Quelques exemples sur la figure I-15.

Figure I-15 : Les différents types de drones civils et militaires.

I.4. Classification des Drones

La classification des drones varie selon les pays. Cependant les drones peuvent être classés en plusieurs catégories selon l'altitude, l'endurance qui est le temps que peut passer l'aéronef en vol, la taille ou encore leur voilure. [5] Dans ce cadre, les drones peuvent se décomposer en trois familles qui sont :

I.4.1. Les drones à voilure fixe

Les drones de cette famille sont constitués d'une paire d'ailes assurant la sustentation (voir la figure I-16), la propulsion est assurée par une ou plusieurs hélices, [6] cette famille est composée de catégories suivantes :

- **Les drones volant à moyenne altitude et de langue endurance (MALE) :** qui utilisent une charge utile qui peut atteindre 100kg et peuvent voler à une altitude de 5 000 à 15 000 mètres avec une autonomie d'une trentaine d'heures
- **Les drones volant à haute altitude et longue endurance (HALE) :** qui peuvent voler à une altitude de 20 000 mètre avec une autonomie d'une trentaine d'heures, ils possèdent une charge utile importante et ils sont équipés de caméras (visible et infrarouge) et des radars divers
- **Les drones de combat UCAV [3] :** Ils sont équipés de systèmes d'armes ou de recueils de renseignements et destinés à accomplir des missions de reconnaissance, d'attaque et de tir, ils peuvent embarquer une charge utile létale.
- **Les drones tactiques TUAV [4] :** qui peuvent voler à une altitude de 200 à 5 000 mètres avec une autonomie d'une dizaine d'heures, ils se décomposent en deux catégories qui sont :
  - **Les micro-drones (Micro Air Vehicle ou MAV) :** Ils ont une taille inférieure à 15 cm, de quelques dizaines à quelques centaines de grammes, ils peuvent voler jusqu'à dizaine de kilomètres pendant une vingtaine de minutes et accomplir des taches dont des drones plus gros sont incapables, ils sont généralement dotés d'hélices entrainées par des moteurs électriques
  - **Les mini-drones (Mini Air Vehicle ou MAV aussi) :** Ils ont une dimension de l'ordre d'un mètre et peuvent voler jusqu'à un plafond de 300 mètres avec une autonomie de quelques heures et une charge utile très légère.

[3] **UCAV :** unmanned combat air vehicles

[4] **TUAV** : tactical unmanned aerial vehicle

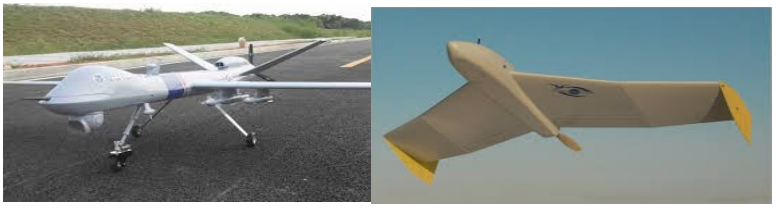


Figure I-16 : Drones à voilure fixe.

I.4.2. Les drones à voilure tournante

Cette famille de drones est caractérisée par le décollage et l'atterrissage vertical, capables de faire du vol stationnaire, à basse vitesse et à basse altitude donc n'ont pas besoin de piste de décollage ou d'atterrissage, ils utilisent un ou plusieurs rotors tel que présenté à la figure I-17, ils peuvent réaliser des missions irréalisables par les véhicules à voilure fixe [6]. Cette famille englobe plusieurs catégories :

- **Mono rotor :** se compose d'un rotor principal avec barre stabilisatrice qui permet la sustentation et la propulsion
- **Birotors contrarotatifs :** se composent de deux rotors tournant en sens opposés et à même vitesse qui permettent la sustentation, la translation.
- **Les quadri rotors :** se compose d'une armature en croix symétrique avec des moteurs et des rotors aux extrémités de chaque tige
- **Les convertibles :** sont des machines munies d'un mécanisme de basculement des rotors qui permet à la fois le vol stationnaire, le décollage et l'atterrissage vertical dans des zones restreintes et difficiles, Le principal inconvénient de ces appareils c'est l'instabilité lors de la phase de transition entre le vol en mode avion et en mode hélicoptère. [5]

Figure I-17 : Drones à voilure tournante.

I.4.3. Les drones à ailes battantes

La sustentation et le pilotage des drones de cette famille sont assurés par des ailes battantes, menés d'une capacité de vol avec des trajectoires similaires à celles des insectes et de vol stationnaire à basses vitesses, comme les voilures tournantes, ce qui est intéressant pour des missions dans des espaces réduits. Ces engins permettent des manœuvres agiles, ce qui

représente un autre avantage, certain pour les missions de reconnaissance ou de surveillance, [6], la figure I-18 montre des exemples des drones de cette famille :

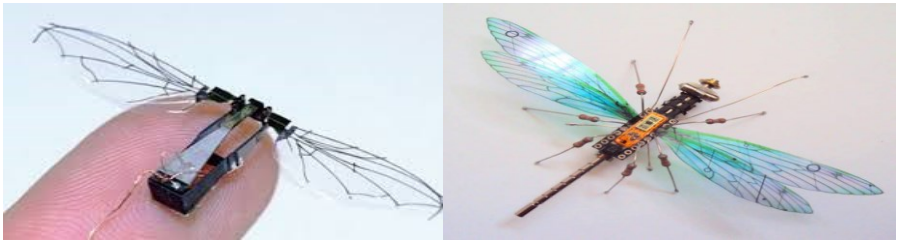


Figure I-18 : Drones à ailes battantes.

Le tableau I-1 résume les avantages et les inconvénients de chaque famille de drones :

Famille	Avantages	Inconvénients
Drones à voilures fixes	- Capacité à porter plus de poids - Capacité de voler avec plus d'autonomie et d'altitudes	- Pas de vol stationnaire - Pas de vol à basse vitesse et à basse altitude - Besoin de piste de décollage et d'atterrissage
Drones à voilures tournantes	- Permet le vol à décollage et à atterrissage verticale - Permet le vol stationnaire - Permet le vol à basse vitesse et à basse altitude - Permet le manœuvre en air - Plus de stabilité et facile à contrôler	- Complexité en maintenance et réparation - Moins d'autonomie, de vitesse et d'altitude en vol
Drones à ailes battantes	- Permet le vol stationnaire à basse vitesse - Permet de faire des missions dans des espaces réduits - permet le manœuvre agile en air	- complexité de réalisation - mémoire trop limitée - ressources d'énergie limitées

Tableau I-1 : Avantages/Inconvénients des différentes familles de drones.

I.5. Les capteurs pour la navigation et la localisation

Les drones embarquent différents capteurs assurant une navigation, une stabilité et une localisation dans l'environnement d'évolution, on distingue principalement deux catégories :

I.5.1. Capteurs proprioceptifs

Les capteurs de ce type permettent de mesurer la vitesse, la position et l'accélération de l'engin par rapport à un état de référence. Leur fonctionnement ne nécessite pas une connaissance de l'environnement dans lequel l'engin évolue. Cependant, ces informations se dégradent au cours du temps ce qui les rend inutilisable à une référence long terme [7].

Les capteurs proprioceptifs les plus utilisés sont :

- **Les accéléromètres :** des capteurs qui calculent l'accélération linéaire [5] de l'engin sur les trois axes orthogonaux [7].
- **Les Gyroscopes et Gyromètres :** les deux capteurs permettent de définir la vitesse et la position angulaire[6] de l'engin dans l'air tel que le gyroscope mesure la position angulaire sur l'axe où il a été monté, par contre un gyromètre permet de mesurer la vitesse instantanée de rotation autour des axes du repère mobile (la vitesse de roulis, de tangage et lacet) [8].

Centrales inertielles (IMU : Inertiel Measurment Unit) : C'est un système de navigation fournissant l'altitude, la vitesse et la position de l'engin composée principalement de trois accéléromètres et trois gyromètres. La technologie MEMS (Micro-Electro-Mechanical Systems) est la plus utilisée actuellement puisqu'elle intègre des capteurs simples de petite taille tels que les accéléromètres, gyromètres ou encore les magnétomètres dans un circuit électronique comme illustré à la figure I-19 [6].

[5] **Accélération linéaire :** Le changement d'accélération d'un objet mobile selon une ligne orthogonale, le résultat est donné en m/s².

[6] **Vitesse angulaire :** c'est la vitesse de rotation d'un engin exprimée en Radians par secondes ($rad.s^{-1}$), plus souvent en tours par minute (tr/min).

Figure I-19 : Une carte IMU.

Comme présenté précédemment, les informations acquises par les capteurs proprioceptifs ne sont pas utilisables à une référence de long terme, c'est pourquoi il est indispensable de les associer à un système permettant de recaler périodiquement la position absolue de l'engin et qui sont les capteurs extéroceptifs.

I.5.2. Capteurs extéroceptifs

Ce type de capteurs utilise les informations de l'environnement extérieur à l'engin (drone) dans le but de connaître sa situation par rapport à certains repères tel que le repère visuel. Ils sont nécessaires pour la localisation ainsi que pour la détermination de l'environnement dans lequel le drone évolue. Les informations acquises par ces derniers sont indépendantes du déplacement de l'engin et l'erreur acquise par les capteurs proprioceptifs ce qui les rend utilisables dans des références à long terme, [7]. Les plus utilisés dans cette catégorie sont :

- **Le compas magnétique (magnétomètre) :** Donne la référence de la direction (le nord) magnétique, fonctionne sur le principe de la boussole dont l'orientation dépend des lignes du champ magnétique terrestre.
- **Le gyrocompas :** Donne la référence de de la direction nord géographique qui est toujours stable par rapport à celui magnétique.
- **Le compas satellitaire :** Basé sur le système de positionnement global (GPS) dont le principe consiste à avoir deux antennes aux extrémités de l'engin équipées des récepteurs GPS, ainsi, il permet de définir l'orientation du vecteur aérien.
- **Les capteurs télémétriques :** Permettent de connaitre l'environnement directe du drone, ils mesurent la distance entre le plus proche objet et l'engin et cela par mesure du temps entre l'émission d'une onde telle que l'onde acoustique (utilisation des ondes ultrasonore sur une courte distance), les ondes optiques (télémètres lasers qui utilisent des ondes lumineuses) et sa réception.

La figure I-20 illustre un exemple d'un capteur extéroceptif.

Figure I-20 : Capteur télémétrique ultrason pour mesurer la distance avec un objet distant.

I.5.3. Le système de géolocalisation

C'est le GPS (Global Positioning System) qui est un système de positionnement par balises actives (les satellites), composé de 24 satellites en orbite autour de la terre. Le système de géolocalisation est utilisé pour acquérir la position absolue de l'engin hébergeant un récepteur GPS à la surface du globe ou dans une région délimitée [9].

I.6. Méthodes pour la navigation

La navigation autonome est un problème important de la commande des drones, elle fait référence à un ensemble de techniques localisant le véhicule dans son environnement d'évolution dans l'objectif de lui permettre de maîtriser ses déplacements et cela d'une façon autonome. On déduit donc qu'elle est essentielle à la réussite d'une mission [10].

Plusieurs approches de navigation autonome peuvent être employées :

I.6.1. La navigation par suivi de terrain

Cette approche consiste à maintenir le véhicule à une altitude constante par rapport au sol sur le long de la mission comme le montre la figure I-21. Dans le domaine militaire, un RADAR[7] est embarqué à bord du véhicule ce qui lui permet de s'ajuster et suivre le relief au plus près. Pour les véhicules de plus petite taille, une caméra remplace le RADAR généralement lui permettant la détection et l'ajustement aux différentes variations du terrain grâce aux flux optiques.

[7] **RADAR :** Appareil qui permet de déterminer la position et la distance d'un objet par l'émission d'ondes radioélectriques et la détection des ondes réfléchies à sa surface.

Figure I-21 : Navigation par suivi de terrain par flux optiques.

I.6.2. La navigation par points de passage

Appelée aussi navigation proportionnelle, cette approche consiste à faire passer le véhicule par un ensemble de points de références dont la position sur les trois dimensions est définie à priori [10]. Cette méthode est simple à mettre en œuvre mais n'est efficace que lorsque la distance entre l'engin et la cible est suffisamment grande [11]. La figure I-22 montre un exemple.

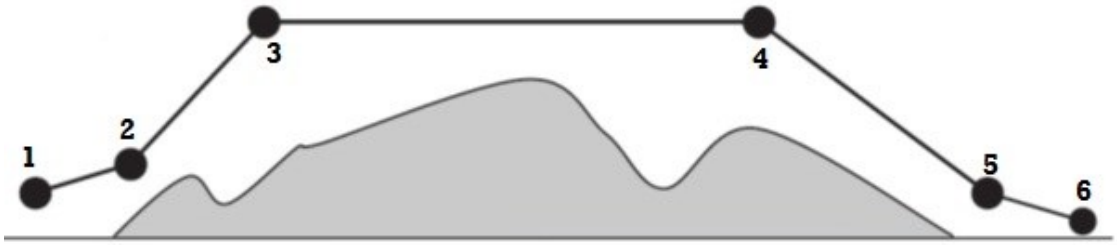


Figure I-22 : Navigation par points de passage.

Une autre approche issue de cette méthode qui est la navigation par suivi de trajectoire, le principe et de suivre une cible repérée jusqu'à atteindre l'objectif. Dans cette méthode on ne parle pas alors d'un ensemble définit de points de passage mais un nombre qui peut être infini [10].

I.6.3. L'évitement d'obstacles

La navigation autonome sans collision requière un mécanisme de détection et d'évitement d'obstacles. Les premières stratégies de navigation consistaient à définir une liberté de mouvement selon l'axe vertical pour prévenir un risque de collision. Aujourd'hui, les drones sont au milieu urbain ce qui implique des stratégies de contournement efficaces dans le plan horizontal ainsi que la détection d'obstacles imprévus et la réaction en temps réel tel que présenté sur la figure I-23 [11]. Pour que le véhicule aérien puisse répondre à ces critères il doit disposer d'un ensemble d'équipement électronique et des algorithmes performants [10].

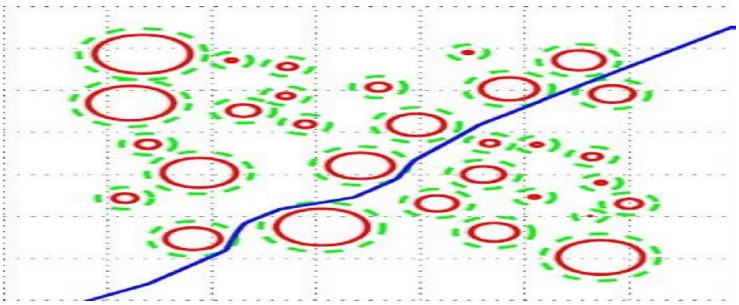


Figure I-23 : Navigation avec évitement d'obstacles.

I.7. Les domaines d'utilisation

On retrouve le drone aujourd'hui dans de nombreux domaines d'activité car son utilisation ne cesse de se rallonger. On distingue principalement deux types d'utilisations, utilisation militaire et utilisation civile.

I.7.1. Utilisation militaire des drones

Les drones sont utilisés dans l'armée depuis 1994 et ça afin de faire l'observation, le renseignement, la reconnaissance du terrain pour les troupes terrestres et aériennes et comme une arme de combat. Le pilotage d'un drone peut s'effectuer à partir de sites distants de plusieurs milliers de kilomètres de l'avion.

La figure I-24 illustre ce dire :

Figure I-24 : Utilisation des drones dans le secteur militaire.

I.7.2. Utilisation civile des drones :

Le transfert des drones vers le secteur civil a permis d'imaginer un nombre important d'applications civiles comme présenté sur la figure I-25, ce qui ouvre une croissance potentielle et considérable dans des secteurs variés et innovants tels que :

- Etude de l'atmosphère, des sols (géologie) et des océans.
- Incendie de forêts, avalanches.
- Surveillance des cultures et épandage agricole.

- Recherche et sauvetage (mer, montagnes, désert…).
- Largage de vivres et d'équipements de sauvetage en zone hostiles.
- Surveillance du trafic routier et du transport de matières dangereuses.



Figure I-25 : Quelques domaines d'utilisation civile des drones.

I.8. Règlementation sur l'utilisation des drones

L'utilisation d'un aéronef qui circule sans pilote à bord est soumise à de nombreuses règlementations qui sont fixés dans des articles de loi telles que les arrêtes de 11 Avril 2012, les arrêtes de 2016, celles de 2017 et qui se différencient d'un pays à l'autre. On prend à titre d'exemple les règles primordiales mises par la DGAC [8] Française et qu'il faut respecter lors de l'usage d'un drone civil :

- L'utilisation d'un drone doit se faire en respectant la sécurité des personnes.
- Il est interdit de faire survoler un drone juste en dessus d'une personne car les hélices du drone sont dangereuses et elles peuvent blesser.
- Il ne faut pas qu'un drone atteigne une hauteur supérieure à 150m.
- Il ne faut jamais survoler un drone la nuit, et la personne qui pilote le drone ne doit jamais perdre de vue son appareil.
- Il est interdit de survoler un drone en dessus d'un espace public en agglomération, près des aérodromes et sur des sites sensibles tels que les zones militaires, les centres nucléaires, les répartiteurs électriques, les voies ferrées, etc.
- Il faut toujours demander l'autorisation des personnes se trouvant sur les prises de vue avant de pouvoir les utiliser et il est interdit de faire diffuser les prises de vue à titre commerciales.

[8] **DGAC** : Direction Générale de l'Aviation Civile, est une administration française chargée de réglementer et de superviser la sécurité aérienne, le transport aérien et les activités de l'aviation civile en général. Le lien de site web : https://www.ecologique-solidaire.gouv.fr/direction-generale-laviation-civile-dgac

D'autres nouvelles règles entreront en vigueur concernant l'obligation de déclarer les drones dont leurs poids dépassent 800 grammes et la nécessité qu'ils devront être équipés de signaux lumineux et sonores afin d'être facilement identifiables dans le territoire aérien.

La FAA [9] a mis également une nouvelle réglementation aux Etats-Unis le 21 Juin 2016 concernant les catégories de drones civils et, spécialement, ceux qui ont moins de 25 Kg et qui volent à une altitude inférieure à 120 mètres, parmi ces règles on cite :

- Il est interdit de dépasser les 122 mètres d'altitudes et les 160 Km/h de vitesse.
- L'utilisateur du drone doit avoir plus de 16 ans et passer un examen de 3 heures dans un centre agréé.
- Le drone doit être enregistré auprès de la FAA avant de pouvoir décoller.
- Il est interdit de voler la nuit, mais il est possible de voler soit 30 min avant le lever du jour et 30 min après le coucher du soleil.
- Il est interdit de piloter depuis un avion, mais il est possible de le faire depuis une voiture en mouvement.
- Il est interdit d'attacher une charge utile au drone qui diminue ses capacités [12].

[9] **FAA** : Federal Aviation Administration, est une agence gouvernementale chargée des réglementations et des contrôles concernant l'aviation civile aux États-Unis, créée le 23 août 1958. Le lien de site web : http:3//www.faa.gov/.

I.9. Conclusion

Au cours de ce chapitre, nous avons brièvement présenté les drones et leur architecture, l'historique associé à l'apparition des premiers drones ainsi leur étapes d'évolution. Différentes configurations de drones ont été aussi mentionnées telles que la famille des drones à voilure fixe, à ailes battantes et à voilure tournante. Nous avons également cité les capteurs et les méthodes pour la navigation, ensuite les domaines d'applications (civil et militaire) en mettant en évidence quelques lois existantes par rapport à leur utilisation civile.

Dans le chapitre suivant, nous allons aborder la notion de la planification de trajectoires, dont nous allons présenter quelques méthodes et algorithmes reliées à ce contexte. Ce rapport va porter par la suite sur la famille des drones à voilures tournante et, particulièrement, les Quad Rotors, suite à leur multiples avantages tels que le non besoin d'une piste de décollage et d'atterrissage, la capabilité de voler à basse vitesse et à basse altitude, et ils sont également plus stables, faciles à contrôler et à gérer.

II.1. Introduction

La planification automatique de la trajectoire des robots sans collision dans des environnements dynamiques a fait l'objet de plusieurs travaux de recherche. Leur application dans les UAVs (Unmanned Aerial Vehicle) a connue récemment plusieurs évolutions et améliorations, leur permettant de jouer un rôle important dans multiples domaines comme la livraison de différents types de produits, la communication en situation d'urgence, la reconnaissance, la lutte contre les incendies, la détection, le sauvetage et bien d'autres domaines d'application.

La mise en œuvre des techniques de planification de trajets dans le contexte des robots en général et des drones en particulier revient à leur fournir le moyen de gérer leur déplacement dans la zone cible en toute sécurité et à temps opportun. Autrement dit, il faut trouver et déterminer un chemin optimal et sans collision qui permet à un robot de se déplacer et de changer la direction tout en évitant les obstacles existants dans l'environnement en partant d'une position de départ et arrivant à celle de destination. C'est ce que l'on appelle le problème de la planification de trajectoire.

Dans ce chapitre, nous allons définir dans un premier temps la notion de la planification et présenter les différents aspects de la planification. Dans un deuxième temps, nous présenterons quelques méthodes qui existent pour planifier une trajectoire, dédiées aux robots en générale et qui sont les plus utilisées, en expliquant brièvement leurs principes, ainsi que leurs avantages et leurs inconvénients.

II.2. Définition de la planification de trajectoire

- ***La planification***

Elle sert à donner les moyens au robot pour trouver une suite d'actions à appliquer sur le monde, pour le faire passer d'un état initial à un état qui satisfait le but à atteindre.

- ***L'état***

Représente la position et l'orientation d'un robot à un moment donné.

- ***La trajectoire ou le plan***

C'est une séquence d'actions qui s'exécute à travers le temps et qui mène au but. Une fois le plan est déterminé, il peut être exécuté soit dans une simulation soit dans un robot.

La détermination de cette trajectoire se fait en respectant un certain nombre de contraintes et de critères qui sont [13]:

- ***Contraintes relatives au robot :*** Concernant sa géométrie, sa cinématique (des grandeurs dépendant du temps et du l'espace) et sa dynamique (des forces qui s'exercent sur le robot) et même l'architecture (type) du robot.
- ***Contraintes de l'environnement :*** Concernant essentiellement la non-collision avec les obstacles.

D'autres critères peuvent être également considérés tels que la prise en compte de la distance de sécurité aux obstacles pour un robot manipulateur[10] ou robot mobile, ou encore la qualité et la stabilité.

II.3. Les différents aspects de la planification

La planification présente plusieurs aspects importants [14] :

- ***La prévision***

Dans la cadre de la planification de trajectoire, la trajectoire ou le plan est une prévision d'une séquence d'actes à accomplir. Lors de l'exécution, la surveillance du monde permet de vérifier que cette séquence d'actions est bien exécutée et qu'elle mène du point de départ vers le but.

- ***La création d'une nouvelle connaissance***

Dans la planification de trajectoire, une fois cette dernière est élaborée, elle peut être conservée pour la réutiliser lorsqu'un problème similaire se posera.

- ***La résolution automatique du problème***

La solution est l'assemblage d'un ensemble d'actions élémentaires élaboré lors de la planification et qui permet au robot d'atteindre l'objectif.

II.4. Types de méthodes de planification de trajectoires

Il existe différentes manières pour classifier les méthodes de planification, on trouve une classification (Globale/Locale), dont les méthodes globales permettent de retourner une solution complète en considérant la totalité de l'environnement, par contre les méthodes locales permettent de trouver des solutions partielles à fur et au mesure que le drone avance dans l'environnement. Dans la classification (Offline/Online), une méthode est Offline si elle permet de trouver le trajet complet du point de départ vers le point d'arriver avant de décoller. Elle est

[10] **Robot manipulateur :** c'est un robot programmable avec des fonctions similaires à des humains, peut être autonome ou contrôlé manuellement, souvent conçu pour des applications industrielles (ex : bras manipulateur).

Online si elle est exécutée en plein vol pour guider l'UAV. La classification que nous avons présentée dans cette partie permet de regrouper les méthodes sur trois approches (Géometric method, Space Velocity et Graph Search) :

II.4.1. L'approche géométrique

Cette approche utilise l'information géométrique des obstacles comme entrée pour rechercher un chemin. Parmi les méthodes de ce type, nous citons :

II.4.1.1. Champs de potentiel

Les champs de potentiel [15] est une méthode réactive qui consiste à calculer à chaque pas de temps le mouvement à appliquer par le robot afin d'obtenir un chemin entre deux configurations[11] **q0** et **qf,** dans un espace contenant des obstacles, après récupération des informations sur l'environnement fournies par les capteurs du système. La méthode considère l'environnement du robot comme étant un champ artificiel qui applique des forces différentes sur le robot, elle le traite comme étant une particule soumise à divers champs de potentiel attractifs et répulsifs, régissant son mouvement comme suit :

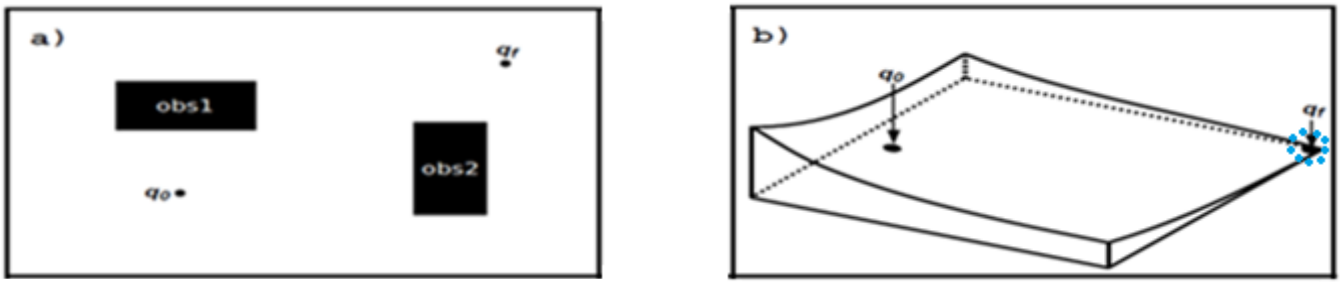
- Le point d'arrivé **pf** est une force d'attraction pour le robot pour le guider vers ce dernier.

- Les différents obstacles qui se trouvent dans l'espace de configuration[12] sont des forces répulsives pour le robot dans le but d'éviter les différentes collisions.

- Le chemin à suivre est donc calculé en faisant sommer les différentes forces attractives et répulsives. La figure II-1 suivante illustre cette méthode [16]

[11] **Une configuration :** est une spécification complète de la position de chaque point du robot dans l'espace physique.

[12] **Espace de configuration** : représente l'ensemble de toutes les configurations possibles du robot.



Espace de configuration du robot et présence des obstacles dans celui-ci

Force attractive générer par la position

Force répulsive exercée par les obstacles

Combinaisons des deux

Figure II-1 : Calcul d'un chemin entre deux configurations q0 et qf.

- ***Les avantages***
  - La méthode des champs de potentiel est totalement réactive.
  - Elle peut être très facilement implémentée en temps réel.

- ***Inconvénients***
  - Présence de problèmes d'oscillation du mouvement du robot en présence d'obstacles et principalement dans des passages étroits.
  - Il n'y a pas de décision de la direction à prendre par le robot.

II.4.1.2. Histogramme de champs de vecteurs (VFH)

Histogramme de champs de vecteurs (en anglais : Vector Field Histogram VFH) [17] est une méthode d'évitement d'obstacles en temps réel, qui permet la détection des obstacles inattendus et l'évitement des collisions avec eux tout en dirigeant simultanément le robot mobile vers la cible. Cette méthode suit les étapes suivantes :

- Construction d'une grille d'occupation bidimensionnelle locale en discrétisant (fragmentant) l'espace de configuration autour du robot en cellules.
- Une valeur de certitude est mise pour chaque cellule : haute si souvent perçue contenant un obstacle, faible sinon. Ces valeurs sont obtenues par un échantillonnage continu et

rapide des capteurs pendant que le robot se déplace et sont mise à jour en fonction de ce déplacement à l'aide des capteurs de distance embarqués.

- Chaque cellule de la grille exerce une force de répulsion virtuelle vers le robot, cette force est proportionnelle à la valeur de certitude et inversement proportionnelle à la distance entre la cellule et le centre du robot. A chaque itération, toutes les forces de répulsion sont totalisées pour donner la force répulsive résultante.

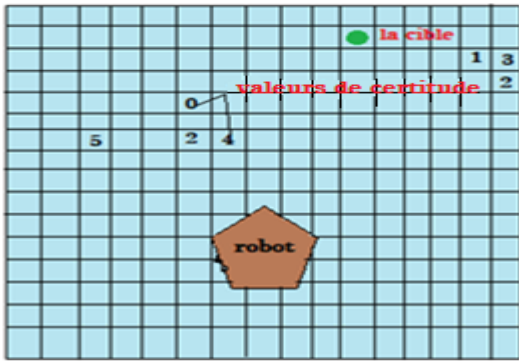
- Simultanément, une force d'attraction virtuelle de magnitude constante est appliquée au robot, le tirant vers la cible. Le vecteur de force résultant est calculé par la sommation des deux forces répulsive et attractive.

- Le contenu de chaque cellule est traité comme un vecteur d'obstacle dont la direction est déterminée depuis la cellule jusqu'au centre du robot. La grandeur du vecteur à chaque cellule est donnée par la suite.

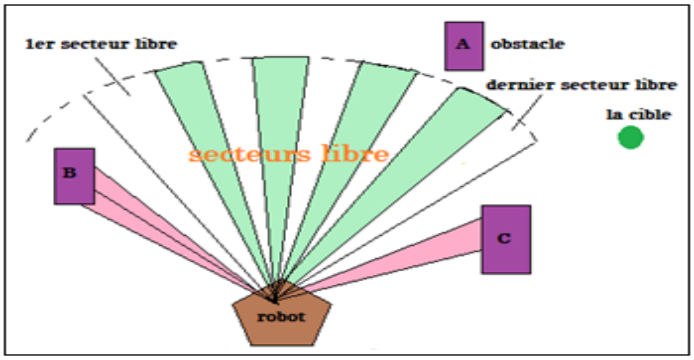
- Ensuite, un histogramme polaire[13] est construit autour de l'emplacement momentané du robot à partir de la grille d'occupation en discrétisant les différentes directions possibles du robot en **n** secteurs angulaires, cet histogramme est construit en pondérant pour chaque secteur de la discrétisation polaire les cellules traversées de la grille d'occupation contenant des obstacles.

- Une fois cet histogramme est construit, des vallées candidates sont déterminées comme les suites de secteurs contigus d'histogramme polaire libres d'obstacles. La direction à prendre par le robot est déterminée par le milieu de la vallée menant le plus directement au but et le contrôle nécessaire est calculé afin de se diriger vers celle-ci. La figure II-2 résume cette approche.

[13] **Un histogramme polaire :** est un histogramme obtenu en discrétisant les directions possibles autour du robot en secteurs.



Grille d'occupation : la présence d'obstacle e mise à jour au fur et à mesure de déplacement du rob



Histogramme polaire calculé à partir de la grille d'occupation : les vallées libres d'obstacles sont déterminés afin de choisir une direction à suivre

Figure II-2 : Histogramme de champs de vecteurs

II.4.1.3. Graphe de Voronoi

Le diagramme de Voronoi [16], [18] représente une décomposition particulière d'un espace métrique déterminée par les distances à un ensemble discret d'objets de l'espace, en général un ensemble discret de points. Ce diagramme est utilisés dans différentes domaines, tels que la biologie, la géographie, la reconnaissance des formes, etc. et pour de nombreuses applications comme la localisation de nouvelles installations en recherche opérationnelle et le contrôle de couverture en robotique mobile. En planification de trajectoire, un tel diagramme permet de calculer la distance la plus éloignée des obstacles, dans le but de générer un chemin qui mène le robot d'un point de départ vers un but et qui présente une bonne marge de sécurité vis-à-vis des obstacles en suivant ces étapes :

- Partitionnement de l'espace de configuration du robot en un ensemble de régions (cellules) adjacentes et chacune a un centre et une frontière partagée. Tous les points situés dans une cellule de Voronoi sont plus proches de son cellule que de tous les autres.

- Création d'une connexion en reliant les centres de ces cellules et le point de départ et d'arrivé sont connectés à tous les sommets de leurs cellules Voronoi.

- Identifier les bordures des cellules (appelés arêtes de Voronoi) de sorte que chaque point des bordures soit à égale distance de chaque centre des deux cellules adjacentes.

- Le robot est représenté par un point qui perçoit les obstacles (les centres des cellules Voronoi). Ensuite, il détecte les obstacles les plus proches de lui.

- Quand c'est deux obstacles proches, il suit la ligne médiane entre ces deux derniers sur une arête du graphe de Voronoi. Et quand c'est trois obstacles proches ou plus, il définit un lieu comme suit :

- Le robot déplace, en maintenant une distance égale entre les deux premiers obstacles détectés A et B tel que : d (A) = d (B).

- Le robot sélectionne un obstacle C tel que : d (C) = d (A) = d (B).

- Le robot s'arrête et définie un lieu et dès qu'il est sur ce lieu, il sélectionne une arête de sortie et il effectue une rotation pour faire face à cette arête.

- Le robot lance sur une nouvelle arête et répète la procédure précédente.

La figure II-3 représente l'utilisation de diagramme de Voronoi pour déterminer un chemin loin des obstacles entre deux situations.

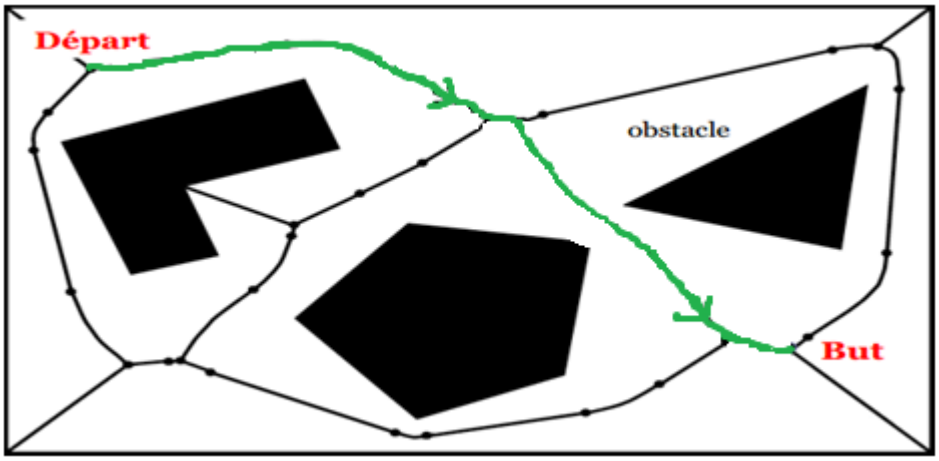


Figure II-3 : Planification de trajectoire par le diagramme de Voronoi

➢ ***Les avantages des méthodes VFH et Voronoi***

- VFH et Voronoi sont faciles à implémenter et à représenter.

➢ ***Inconvénients des méthodes VFH et Voronoi***

- Pas de prises en compte des contraintes cinématiques du robot, tel que l'accélération, la vitesse, etc.
- Limitation de nombre de trajectoires possibles due à la discrétisation de l'espace de recherche.

II.4.2. L'approche de l'espace de vitesse

Cette approche travaille dans l'espace de vitesse du robot. Elle prend en compte la forme du véhicule ainsi que les contraintes cinématiques et dynamiques. Parmi les méthodes de ce type, nous citons :

II.4.2.1. Vitesse de courbure

La méthode de vitesse de courbure (en anglais : Curvature Velocity Method CVM) [19] est une méthode d'évitement d'obstacles locale dans un environnement inconnus ou partiellement connus, elle est basée sur l'évaluation des différentes trajectoires possibles dans l'espace des vitesses, puis de sélectionner, depuis cet espace, un mouvement à exécuter. Autrement dit, il s'agit de sélectionner un chemin à suivre permettant d'éviter les obstacles et de se rapprocher du but, son principe est le suivant :

- La méthode suppose que le robot parcourt toujours des arcs de cercle, appelés courbes **c**, ce sont les différentes trajectoires possibles dans l'espace des vitesses.

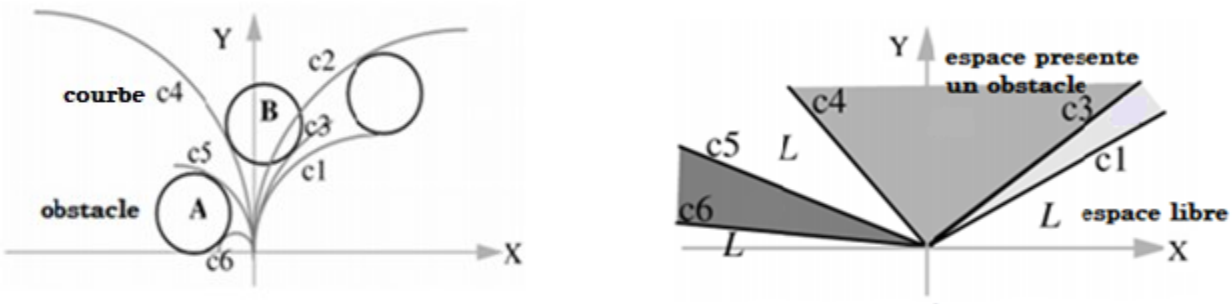
- Calculer la distance que le robot parcourrait avant de heurter l'obstacle.

- Le robot calcule tous les couples de vitesse linéaire / vitesse angulaire **(V,W)** possibles.

- Ensuite, à chaque couple **(V,W)**, est associée une fonction de coût objective basée sur la distance aux obstacles, la modification de l'orientation par rapport au but et sur le temps nécessaire pour rejoindre le but.

- Le calcul de cette fonction peut être très complexe pour des obstacles de forme arbitraire, pour résoudre ce problème, nous considérons le robot et même les obstacles comme des cercles.

- Le couple de vitesse maximisant cette fonction est alors sélectionné et envoyée au robot lors du prochain pas du temps. A l'aide de la fonction objective, le robot peut présenter différents comportements selon les poids et la répartition des obstacles allant de ralentir pour tourner et éviter un obstacle et à voyager à pleine vitesse. La figure II-4 représente la caractérisation des mouvements possibles dans l'espace des vitesses linéaires et angulaires : sélection du chemin à suivre maximisant une fonction de coût.



Trajectoires possibles représentées dans l'espace des vitesses linéaire et angulaire

Contrôle des trajectoires dans l'espace de vitesse

Figure II-4 : Méthode de vitesse de courbure CVM

➢ ***Les avantages***

- La méthode permet de prendre en compte les contraintes cinétiques[14] du robot.

➢ ***Inconvénients***

- La méthode n'est pas toujours apte à garantir la sécurité de chemin.
- La méthode mène le robot à passer relativement proche des obstacles et entrer en collision avec ces derniers.

II.4.2.2. Fenêtre dynamique

La méthode dite de fenêtre dynamique (en anglais : Dynamic Window DW) [20] est développée pour des robots qui se déplacent avec des vitesses élevées. Cette méthode n'est pas une méthode de déplacement globale car elle ne fournit pas de manière directe un moyen d'aller d'un point A à un point B. Elle s'applique de manière locale. A chaque pas de temps, elle fournit au robot une commande (couple) de vitesse **(V, W)** vitesses linéaire **V** et vitesses angulaire **W** pour que celui-ci ne rentre en collision avec aucun obstacle et cela jusqu'au pas du temps suivant. L'algorithme de la méthode se déroule en quatre étapes essentielles :

- Au départ, la méthode considère uniquement les trajectoires circulaires (courbures) déterminées par les couples **(V, W)**, elle considère exclusivement le premier pas de temps et suppose que les vitesses dans les pas du temps restants sont constantes et que les accélérations sont nulles.

- Le robot calcule tous les couples de vitesses possibles. Ensuite il filtre parmi ces couples de vitesses ceux qui respectent les contraintes cinématiques (où v et w ne doivent pas

[14] **La cinétique :** c'est l'étude des mouvements du robot en intégrant les masses.

dépasser les vitesses maximales $\mathbf{V_{max}}$ et $\mathbf{W_{max}}$), les contraintes dynamiques (où les accélérations linéaire et angulaire appliquées entre chaque pas de temps doivent être bornées) et la contrainte de sécurité dont le robot doit être certain de pouvoir s'arrêter avant d'entrer en collision avec un obstacle. Le robot ne garde plus que certains couples de vitesses admissibles.

- Puis, le robot génère un ensemble des couples candidats notées $\mathbf{U_r}$ comme étant l'intersection entre l'ensemble des vitesses admissibles respectant les contraintes cinématiques notées $\mathbf{U_s}$ et les vitesses respectant les contraintes dynamiques notées $\mathbf{U_d}$ et les vitesses notées $\mathbf{U_a}$ respectant la contrainte de sécurité :

$$\mathbf{Ur = Us \cap Ud \cap Ua}$$

- Une fois cet ensemble est définit, une fonction de coût similaire au CVM est associée à chaque couple de vitesse de cet ensemble, le couple de vitesses choisi est celle qui maximise cette fonction. La figure II-5 résume cette méthode.

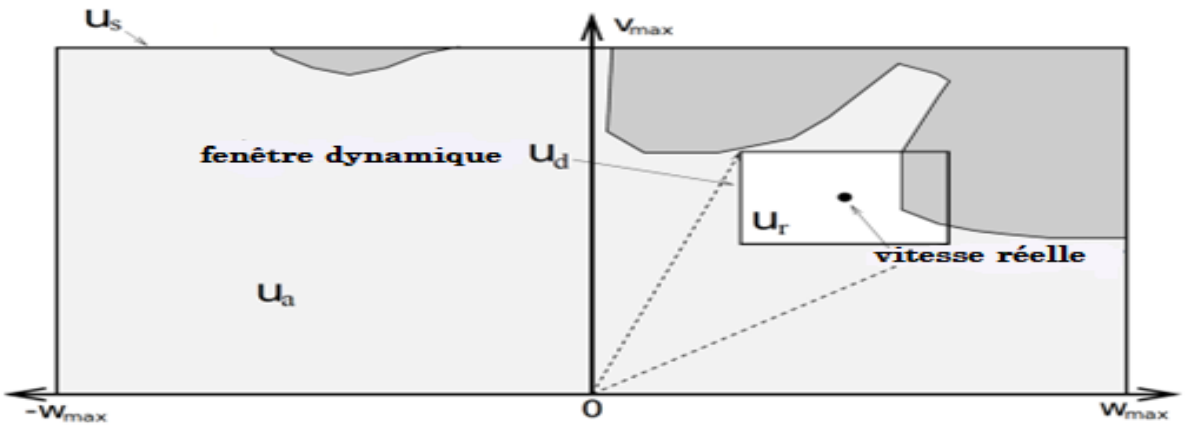


Figure II-5 : Méthode de la fenêtre dynamique DW.

➢ ***Les avantages***

- Son utilisation est très intéressante pour des robots se déplaçant rapidement ou ayant des capacités d'accélération et de ralentissement limitées.

➢ ***Inconvénients***

- Manque de flexibilité.
- Implémentation difficile dans un cadre multi-robot.

II.4.3. Approche de recherche dans un graphe

Cette approche capture la topologie de l'espace de recherche dans le but de simplifier le problème à une recherche dans un graphe en se basant sur deux étapes [16] :

- La construction du graphe dans l'espace de recherche approprié.
- Le parcours du graphe dans le but de déterminer un chemin entre deux situations initiale et finale.

Pour simplifier, un graphe est un ensemble de nœuds reliés par des arcs, chaque arc possède une certaine valeur, qui est un coût. Tous les nœuds ne sont pas forcément reliés entre eux. Un nœud représente un état. Les graphes s'appliquent à de nombreux problèmes, comme le routage des données sur Internet, l'apprentissage automatique et bien sûr la recherche de chemin. La figure II-6 illustre une représentation d'un graphe avec 5 nœuds et 6 arêtes évaluées.

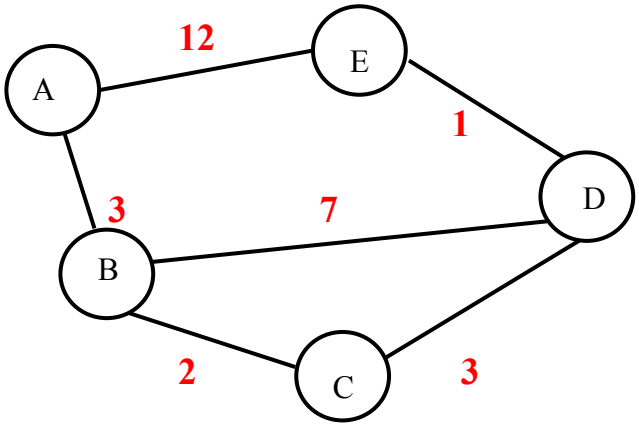


Figure II-6 : Exemple d'un graphe à 5 nœuds et 6 arcs.

Dans le cas de la recherche de chemin, un nœud est une position et la valeur d'un arc peut être la distance entre les deux nœuds qu'il relie. Ainsi, en plaçant des nœuds à certains endroits, des algorithmes peuvent établir un itinéraire (qui est en fait une suite de nœuds qui forment un chemin). Dans ce cas, il y a un arc entre deux nœuds s'il n'y a pas d'obstacle entre eux. Le plus court chemin entre deux positions est le chemin dont la somme des valeurs des arcs est minimum. Ainsi dans l'exemple précèdent, le plus court chemin entre A et E c'est A-B-C-D-E avec une valeur totale de 9 et non A-E avec une valeur de 12.

Cette approche inclut les algorithmes destinés pour la recherche dans un graphe. Parmi ces algorithmes, on cite les suivants :

II.4.3.1. L'algorithme de Dijkstra

Cet algorithme [21] est l'ancêtre des algorithmes de recherche de chemin modernes. Le principe de l'algorithme de Dijkstra est de parcourir un graphe afin de trouver un chemin ayant

la distance la plus faible entre un nœud de départ et un autre d'arrivé (la somme des valeurs des arêtes reliant ces deux nœuds soit minimum). Pour cela, l'algorithme suit les étapes suivant :

- Prendre le nœud N de distance minimale : au début ce sera le nœud de départ.
- Affecter à chaque nœud successeur V_i de N la distance minimale de N plus la valeur de l'arête entre N et V_i.
- Refaire l'étape précédente et à chaque fois le nœud courant pris c'est celui qui a une distance minimale.
- Si un nœud est revisité, et sa distance a diminué alors, sa valeur de distance sera mise à jour.
- Continuer de la même façon jusqu'au parcourt de tous les nœuds de graphe ou atteinte du but.
- A la fin de l'algorithme, on dispose du plus court chemin depuis le nœud de départ vers le nœud d'arrivé ou vers n'importe quel nœud du graphe.

Exemple : cherchons les plus courts chemins d'origine **A** dans le graphe de la figure II-7.

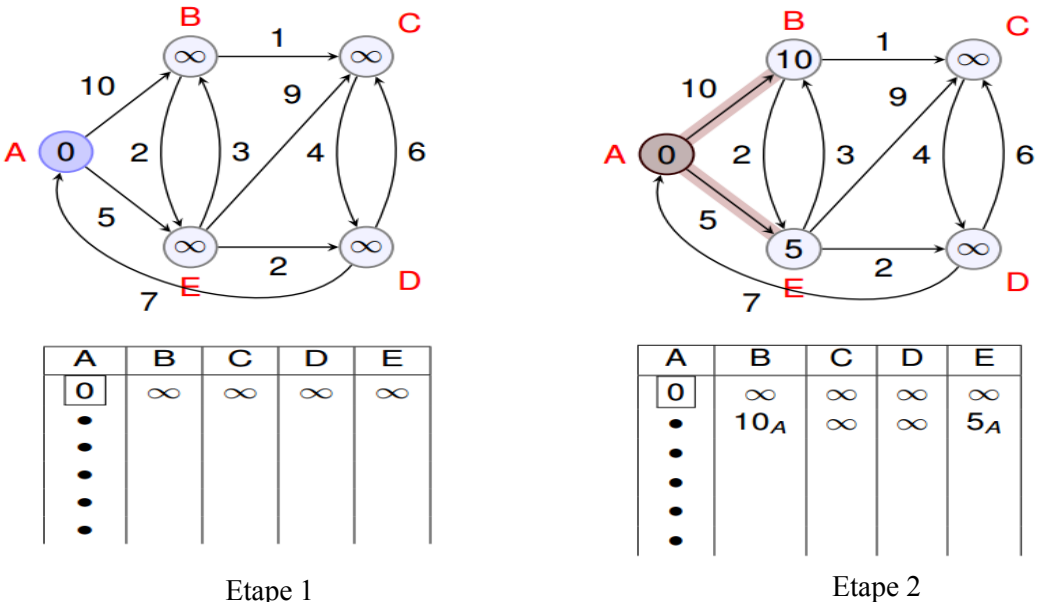
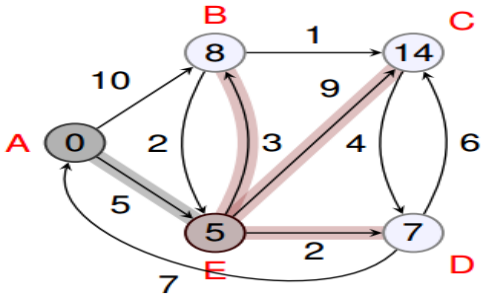


Figure II-7 : Dijkstra-étapes 1 et 2.

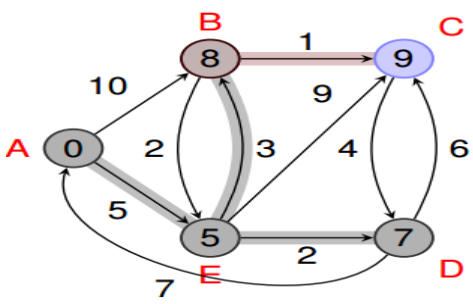
Nous nous plaçons au nœud de distance minimale égale à **0**, (ici le nœud de départ **A)** et nous plaçons dans un tableau tous les nœuds du graphe et nous les affectons des valeurs de distance initiale à l'infinie ∞.

Nous étudions chacun des nœuds successeurs partant du nœud de départ choisi. Dans les colonnes du tableau, nous mettons la distance de ***A*** + la valeur de l'arête entre ***A***, le nœud successeur et le nom du nœud parent, par exemple pour le nœud **B**, nous mettons $\mathbf{10_A}$.



A	B	C	D	E
0	∞	∞	∞	∞
•	10_A	∞	∞	5_A
•	8_E	14_E	7_E	•
•				•
•				•

Etape 3



A	B	C	D	E
0	∞	∞	∞	∞
•	10_A	∞	∞	5_A
•	8_E	14_E	7_E	•
•	8_E	13_D	•	•
•	•	9_B	•	•
•	•	•	•	•

Etape 4

Figure II-8 : Dijkstra-étapes 3 et 4

Nous nous plaçons de nouveau au nœud de distance minimale, ici **E** et nous refaisons la même itération, remarquant que la distance de **B** est mise à jour **8** au lieu de **10**. Et ainsi de suite, jusqu'au parcourt de tous le graphe (voir la figure II-8).

Nous nous arrêtons au nœud **C** avec une distance minimale **9** car si on continue vers **D**, sa distance va augmenter et ne vas pas diminuer.

Nous obtenons du plus court chemin depuis le nœud de départ **A** vers n'importe quel nœud.

- ***Avantage***
  - Simplicité de compréhension et d'application.
- ***Inconvénient***
  - Pas assez performant pour des graphes de grande taille.

II.4.3.2. L'algorithme A*

Cet algorithme qu'on prononce A star ou A étoile [22] est une extension de l'algorithme de Dijkstra, sa différence est qu'il va chercher à se diriger vers le nœud d'arrivée, à l'aide d'une heuristique (une heuristique est une fonction qui, pour des nœuds, renvoie une valeur. Cette

fonction peut être quelconque mais doit satisfaire une contrainte : elle ne doit jamais surestimer la distance réelle).

Plus concrètement, la différence de cet algorithme par rapport à l'algorithme précèdent est dans le calcul de la distance à affecter à un nœud. Au lieu de lui affecter la distance au nœud de départ, on affecte la distance minimale entre le nœud de départ et d'arrivée qu'on pourrait obtenir si on passait par ce nœud. Pour cela, l'algorithme suit les étapes suivantes :

- Utiliser deux listes : open (qui est une liste prioritaire des nœuds qui sont candidats mais pas développés dont leurs successeurs n'ont pas encore été explorés), et close (qui est une liste des nœuds qui ont été visités et développés dont leurs successeurs ont déjà été explorés).
- Initialiser la liste open avec le nœud de départ sous le forme (N_0, f, N_{parent}) où f est une fonction du coût égale à g + h tel que : g est la distance entre le nœud de départ et le nœud courant, h est l'heuristique qui renvoie la distance entre le nœud courant et le nœud d'arrivé.
- Une fois le nœud courant est visité, on le dépile de open et on l'empile dans close et on empile ses fils (successeurs) dans open selon l'ordre croissant de la valeur de *f*.
- Dépiler le premier nœud depuis open et empiler ses fils (s'ils n'existent pas déjà dans close) et l'empiler dans close.
- Continuer de la même manière le reste des nœuds du graphe et si un nœud apparait deux fois avec une valeur différente de f, on garde uniquement là où se valeur est minimum.
- A la fin de l'algorithme, on obtient le plus court chemin en suivant les parents des nœuds présents dans la liste close, on remonte le fil jusqu'à arriver au point de départ.

➢ **Exemple :** cherchons le plus court chemin entre le nœud n0 de départ et le nœud n6 d'arrivé dans le graphe de la figure II-9.

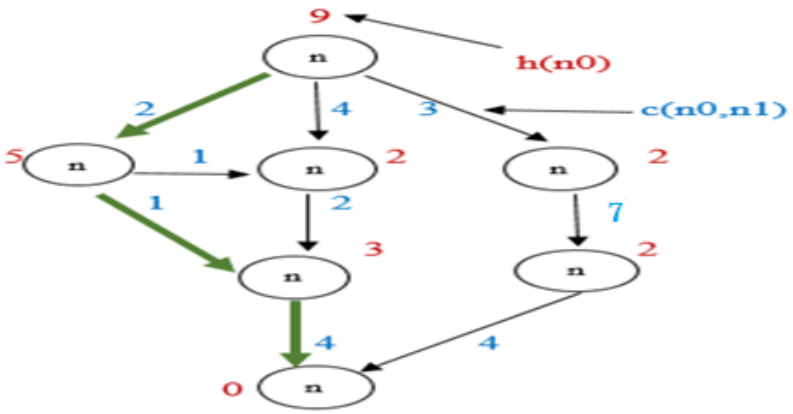


Figure II-9 : Exemple de A.*

Le tableau 0II-1 suivant résume le processus de recherche :

Contenu d'open à chaque itération	Contenu de close à chaque itération
1- (n0, 9, void)	1- Vide
2- (n1, 5, n0) ; (n2, 6, n0) ; (n3, 7, n0)	2- (n0, 9, void)
3- (n2, 6, n0) ; (n3, 7, n0) ; (n5, 12, n1)	3- (n0, 9, void) ; (n1, 5, n0)
4- (n3, 7, n0) ; (n4, 9, n2) ; (n5, 12, n1)	4- (n0, 9, void) ; (n1, 5, n0) ; (n2, 6, n0)
5- (n2, 5, n3) ; (n4, 6, n3) ; (n5, 12, n1)	5- (n0, 9, void) ; (n1, 5, n0) ; (n3, 7, n0)
6- (n4, 6, n3) ; (n5, 12, n1)	6- (n0, 9, void) ; (n1, 5, n0) ;(n3, 7, n0) ; (n2, 5, n3)
7- (n6, 7, n4) ; (n5, 12, n1)	7- (n0, 9, void) ; (n1, 5, n0) ;(n3, 7, n0) ; (n2, 5, n3) ; (n4, 6, n3)
8- solution : **n0, n3, n4, n6**	8- (n0, 9, void) ; (n1, 5, n0) ; (n3, 7,n0) ; (n2, 5, n3) ;(n4, 6, n3) ;(n6, 7, n4)

Tableau II-1 : Processus de recherche avec A.*

- ***Avantage***
  - A* est facilement réalisé et calculé en temps réduit.
- ***Inconvénient***
  - A* dépend fortement de la fonction heuristique, ce qui influence sur la performance de l'algorithme.

II.5. Conclusion

Nous avons abordé dans ce chapitre la notion de la planification de trajectoire afin de générer un chemin optimal et sans collision avec les obstacles pour les robots en général, et pour les UAVs en particulier, et nous avons présenté par la suite les principes de quelques méthodes et algorithmes de planification de trajectoire et d'optimisation de chemins apparus dans ces dernières années.

Dans le chapitre qui suit, nous allons présenter quelques méthodes de planification de trajectoires qui sont optimisées, plus récentes et dédiées pour les UAVs.

III.1. Introduction

Les véhicules aériens sans pilote (UAVs) attirent de plus en plus l'attention ces dernières années en raison de leur contribution importante et application rentable dans nombreuses tâches. Pour se déplacer en toute sécurité d'une position à une autre, l'UAV doit éviter les obstacles qui se présentent sur son chemin, tout en assurant l'optimalité de sa trajectoire empruntée parmi celles disponibles.

Des techniques et de nombreuses méthodes de planification de trajectoires importantes ont été proposées (cf. chapitre 2). Ces dernières années, ces méthodes ont été améliorées et appliquées à la planification de trajectoires pour les UAVs. Il existe encore d'autres plus récentes, plus performantes et mieux adaptées pour les drones. La majorité de ces méthodes s'appuient sur des algorithmes évolutionnaires inspirés des phénomènes naturels (les attaques des fourmilions pour les fourmis, la recherche du chemin optimal par les fourmis, etc).

Dans ce chapitre, nous allons présenter quelques méthodes établies pour la planification de trajectoires multi objectives (le temps de parcours, la sécurité de vol, la consommation du carburant, etc.) pour les UAVs selon trois approches principales :

- en ligne,
- hors ligne, et
- combinaison hors ligne et en ligne.

Une explication brève est abordée pour chaque méthode, ainsi que les avantages et inconvénients. Une étude comparative entre différentes méthodes sera établie par la suite.

I.4. Les méthodes de planification en ligne

Dans cette approche, la recherche du chemin et faite durant le vol de l'UAV en temps réel, où les obstacles sont dynamiques et inattendus. Les solutions retournées sont optimales par rapport à la recherche locale.

III.1.1. Méthode de planification de trajectoire géométrique

En anglais (A Geometrical Path Planning Method) [23], caractérisée par l'habilité de planifier en temps réel. Elle permet au drone de trouver un chemin sans collision vers une destination dans un environnement complexe 2D et 3D en se basant sur les aspects géométriques. Dans un premier temps, les obstacles de l'environnement qui bloquent la ligne qui relie le point de départ et le point d'arrivé sont représentés sous différentes formes de polyèdres convexes (Cas 2D : la forme rectangulaire, trapézoïdale, triangulaire, circulaire et elliptique. Cas 3D : la forme sphérique, cuboïde et elliptique) et stockés dans une liste. Dans un

deuxième temps, un sous objectif est calculé pour l'obstacle le plus proche au point de départ selon sa forme. Dans un dernier temps, Le chemin complet est constitué en rassemblant les différents sous chemins trouvés. Les étapes que suit cette méthode sont :

a. Représenter les obstacles sous les différentes formes géométriques et stocker leurs informations dans une liste.
b. Calculer une ligne entre le point de départ et le point d'arriver.
c. Requêter tous les obstacles qui se croisent avec la ligne calculée et retrouver l'obstacle le plus proche du point de départ.
d. Choisir un sous-objectif parmi les bords des obstacles qui est le plus proche du départ.
e. Prendre le sous but comme le nouveau point de départ.

Et on répète les étapes précédentes (b-e) jusqu'à atteindre le but.

➢ ***Exemple***

Cherchons le plus court chemin entre le point de départ *S* et le point d'arrivé *F* dans l'environnement contenant des obstacles 2D et 3D (O_1 et O_2) respectivement, comme montré à la figure III-1.

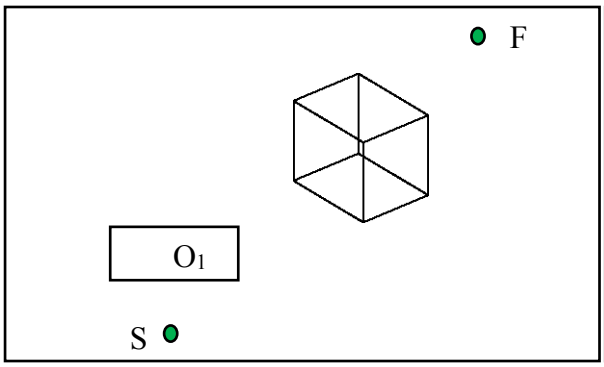


Figure III-1 : Environnement contenant des obstacles 2D/3D représentés sous forme géométrique

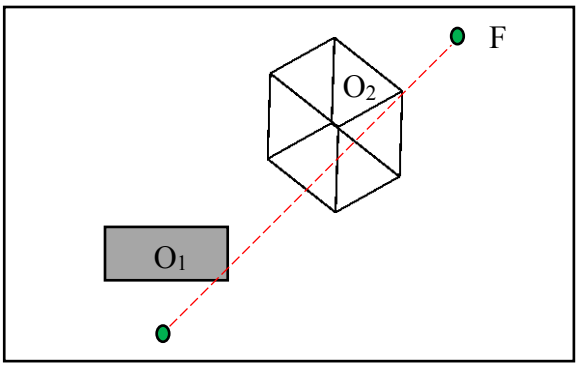


Figure III-2 : Calcul de ligne entre S et F et retrouve de l'obstacle le plus proche du S.

Une ligne est calculée entre le point de départ *S* et celui d'arrivé *F*, et choix du l'obstacle le plus prêt du point de départ qui est O_1 tel que présenté au Figure III-2.

Une fois l'obstacle est déterminé on procède au calcul du meilleur contournement possible au niveau de O_1 qui est rectangulaire en 2D. Pour cela on recherche le point *P* qui est l'intersection entre $O_{1a}\,O_{1b}$ et la ligne *SF*. Dans ce cas, $O_{1a}\,O_{1b}$ est le bord de O_1 le plus proche de *S*, on trouve le min $\{\| O_{1a}P \|, \|P\, O_{1b} \|\}$. Le sous-objectif est O_{1b} vu qu'il est le plus prêt de *P* comme le montre la figure III-3.

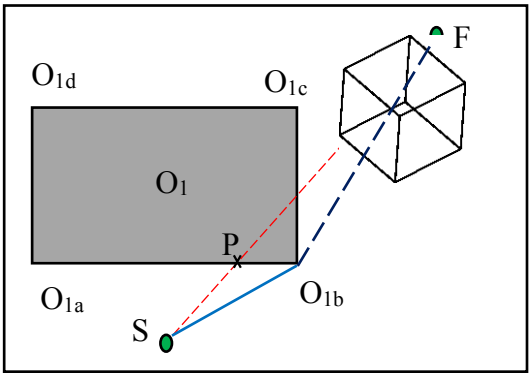


Figure III-3 : Résolution du premier sous objectif de l'obstacle en 2D.

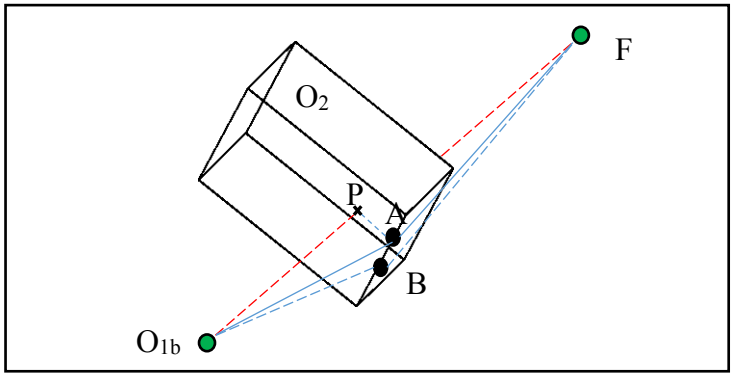


Figure III-4 : Contournement de l'obstacle en 3D.

S'il n'y avait pas l'obstacle cuboïde (O_2), le chemin serait $SO_{1b}F$, mais lorsqu'on prend O_{1b} comme point de départ, la ligne $O_{1b}F$ se croise avec l'obstacle O_2 donc il faut contourner cet obstacle qui est en 3 dimensions.

Calculer le sous objectif O_2 consiste à trouver la face qui croise SF, ensuite, le point P. Lorsque le point P est trouvé on calcule la ligne verticale sur tous les bords de la face sauf celui du bas.

On détermine le bord de la face le plus prêt du P qu'on nomme AB tel que le point A est le point de départ de P vers AB, et B est un point arbitraire sur le bord AB comme montré sur la figure III-4. Le résultat final de la trajectoire est $\boldsymbol{SO_{1b}AF}$.

- ***Les avantages :***
  - court délai pour retrouver le chemin.
  - traitement avec plusieurs formes d'obstacles convexes.
  - Pas besoin d'un modèle de grille pour trouver le chemin.

- ***Les inconvénients :***
  - Le chemin global n'est pas garanti en optimalité.

III.1.2. Improved Artificial Potential Field Method

La méthode citée dans [24] est appliquée au problème de planification de trajectoire 3D d'UAV en temps réel dans des environnements dynamiques où les obstacles sont modélisés par des cylindres. Cette méthode est basée sur le champ de potentiel artificiel (APF), avec l'amélioration de la fonction du potentiel répulsif. En effet, l'application de l'APF pour éviter les obstacles bien qu'elle soit facile, efficace et largement utilisée, un problème majeur en

utilisant cette approche peut se présenter, à savoir les minimas locaux[15] qui peuvent piéger le robot avant d'atteindre son objectif. *Improved* APF, contrairement à APF originale, elle peut faire en sorte que l'UAV évite de se coincer dans les minimas locaux en ajustant l'accélération de l'UAV à la rencontre de ces derniers (voir la figure III-5).

Dans la méthode APF originale, la fonction du potentiel répulsif est basée sur la distance entre l'UAV (représenté par un point) et l'obstacle notée *P*. La sphère d'influence où le répulsif devient actif est notée par *P0*. La méthode retourne une valeur non nulle si *P* <= *P0*, et une valeur nulle si *P* > *P0*.

Par contre dans la méthode *Improved APF*, la fonction du potentiel répulsif est définie comme la somme de deux fonctions $\boldsymbol{F_{rep1}}$ et $\boldsymbol{F_{rep2}}$ si *q(j)* <= *P0*, et retourne une valeur nulle si *q(j)* >= *P0*, où :

- $\boldsymbol{Frep_1}$: est la fonction du potentiel répulsif délimitée de l'obstacle à l'UAV.
- $\boldsymbol{Frep_2}$: est la fonction du potentiel répulsif délimitée de l'UAV au but.
- ***q(j)*** *:* est la distance entre l'UAV et l'obstacle à l'itération *j*

C'est cette fonction du potentiel répulsif $\boldsymbol{Frep_2}$ qui permet à l'UAV d'éviter de tomber dans des minimas locaux. La méthode *Improved* APF est résumée en six étapes qui sont :

- Initialisation des paramètres : le point de départ *S*, le point cible *T*, le rayon de l'obstacle *R*, la distance d'influence de l'obstacle *P0*, le numéro d'itération *j*, et le paramètre *X(x, y)* qui représente la position de chaque objet de l'espace.
- Si *xS* >= *xT* ou *yS* >= *yT* alors on s'arrête.
- sinon :
  - Calculer l'angle-attractive (c'est l'angle entre le potentiel attractif $\boldsymbol{F_{att}}$ et l'axe ***X***, l'angle-répulsive (c'est l'angle entre le potentiel répulsif $\boldsymbol{F_{rep}}$ et l'axe *X*) et le nombre d'itérations ***j***.
  - Calculer l'angle-somme qui est l'angle entre la force de potentiel total $\boldsymbol{F_{sum}}$ et l'axe ***X***.
  - Passer au point suivant ***H*** en fonction de la distance et l'*angle_somme*.
- Répéter les mêmes itérations lorsque ***S*** = ***H*** et s'arrêter dès que *xS* >= *xT* ou *yS* >= *yT*. La figure11 suivante illustre cette méthode (le potentiel total $\boldsymbol{F_{sum}} = \boldsymbol{F_{rep1}} + \boldsymbol{F_{rep2}}$).

[15] **Les minimas locaux :** ce sont des configurations de l'espace où l'UAV est sous l'influence de deux forces attractive et répulsive dont la somme est nulle.

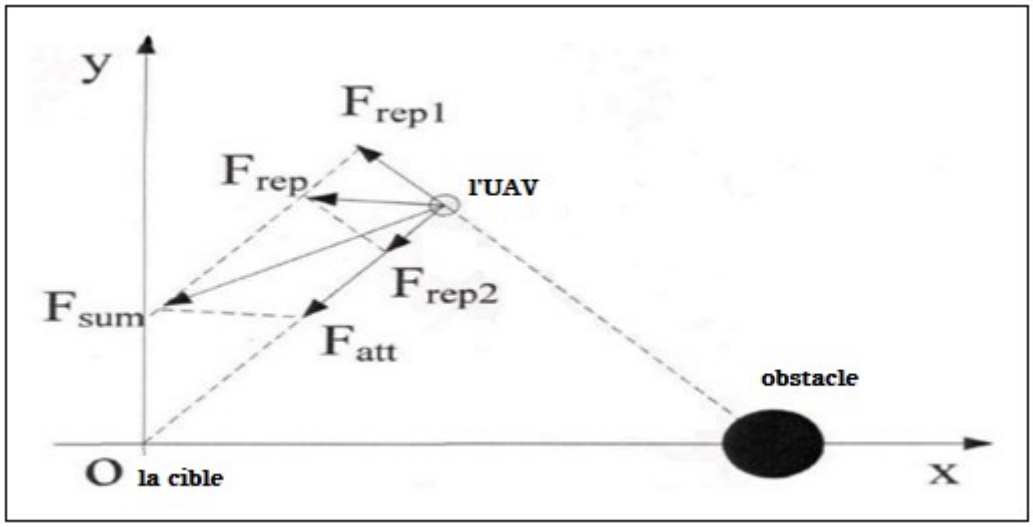


Figure III-5: Improved Artificial Potential Field Method.

- ***Les avantages***
  - L'évitement du problème des minimas locaux.
  - Maximiser l'évitement d'obstacles par rapport à l'approche traditionnelle.
- ***Les inconvénients***
  - La méthode ne prend pas en compte la cinématique ou la dynamique de l'UAV.
  - Des obstacles se trouvent dans la vie réelle qu'on ne peut pas représenter avec des cylindres.
  - La méthode ne prend pas en compte la taille et l'altitude de l'UAV.

I.5. Les méthodes de planification hors ligne

Dans cette approche, la recherche du chemin se fait avant la mission du vol de l'UAV, où les obstacles sont statiques et détecté dans la carte géographique.

III.1.3. Algorithme d'optimisation de colonies de fourmis ACO

La planification de chemin d'un UAV basée sur l'algorithme ACO (en anglais : Ant Colony Optimization Algorithm) [25] permet de générer un chemin optimal dans un espace de navigation contenant des obstacles. La méthode s'inspire du comportement des fourmis recherchant un chemin entre leur colonie et une source de nourriture, dont le point de départ est représenté en tant que une colonie et le point cible en tant que source de nourriture. ACO repose sur trois comportements intelligents qui sont : la communication des fourmis basée sur les phéromones[16], le comportement de la mémoire des fourmis, et les activités des groupes de fourmis. Le principe de l'algorithme est résumé en sept étapes qui sont :

- Placer *M* fourmis aléatoirement dans *N* villes (états) et définir la valeur initiale de la variable piste de *phéromone* (le chemin entre deux fourmis voisines) qui est une valeur epsilon positive.

- Pour chaque fourmi, la liste qui modélise sa mémoire contient initialement son état de départ où s'était placé. Puis ajoutant des états on sélectionnant à chaque fois l'état à visiter pour chaque fourmi.

- S'il y a un obstacle dans la direction de déplacement de la fourmi, elle choisira aléatoirement une autre direction. L'espace de recherche de chaque fourmi est un monde carré, dont chacune ne peut se déplacer qu'à l'intérieur de cet espace.

- La fourmi trace une piste de *phéromone* pendant son déplacement et calcule sa valeur. Elle disperse la plus grande valeur lorsqu'elle trouve la nourriture ou le nid.

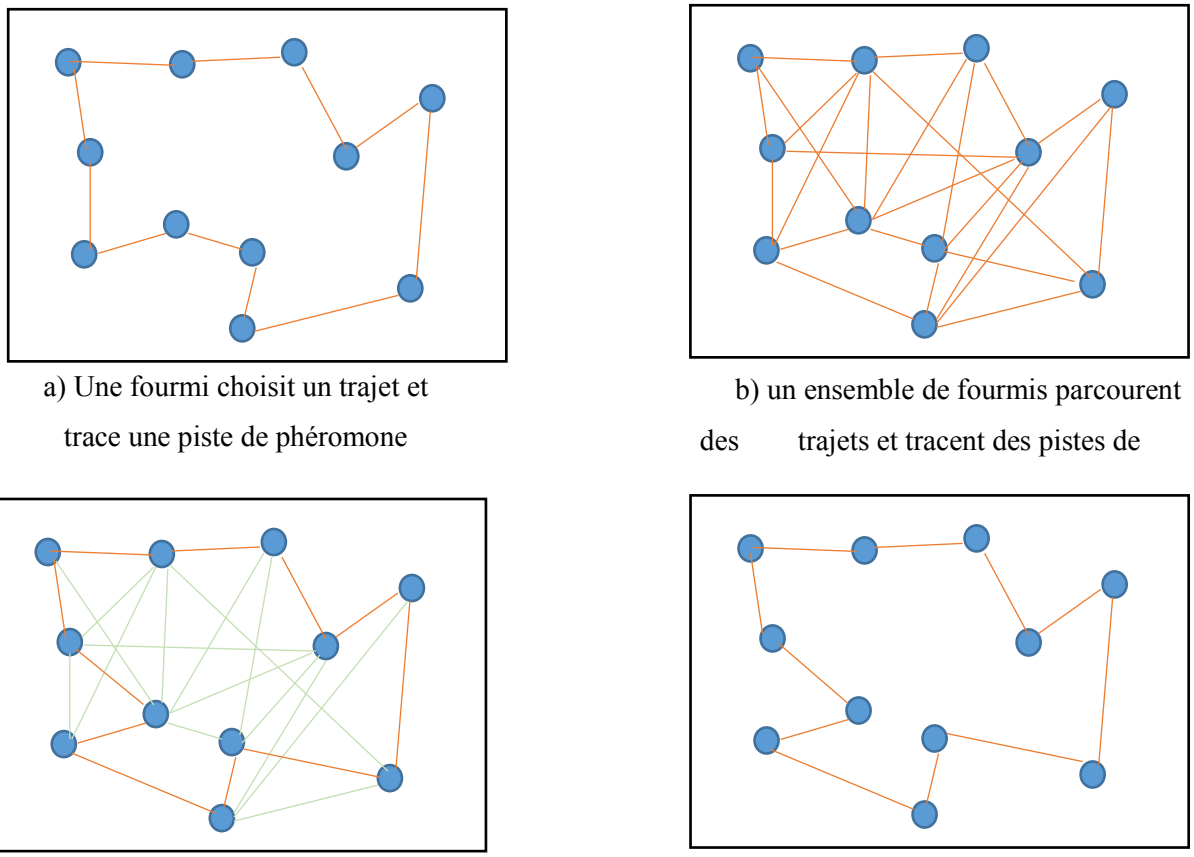
- la piste s'évapore progressivement à chaque itération (déplacement de fourmi d'un état à l'autre), la valeur de la variable *phéromone* de chaque piste est mise à jour en fonction de cette évaporation et de l'amélioration des phéromones. Plus l'intensité de la piste de

[16] **Les phéromones :** ce sont des substances chimiques comparables aux hormones émissent par la plupart des animaux et qui agissent comme un message entre les individus d'une même espèce.

phéromone disposée sur le chemin entre deux états est grande, plus le trajet aura de chance d'être choisi dans les itérations suivantes.

- Calculer le trajet de chaque fourmi une fois que toutes les fourmis ont terminé leur mission de visite, puis sauvegarder le trajet le plus court (le chemin qui contient plus de phéromones).

- L'évaporation est une simple décrémentation de la variable *phéromone*, elle fait disparaitre les mauvaises solutions et garde que le chemin le plus court. La figure III-6 résume le principe de cet algorithme.



a) Une fourmi choisit un trajet et trace une piste de phéromone

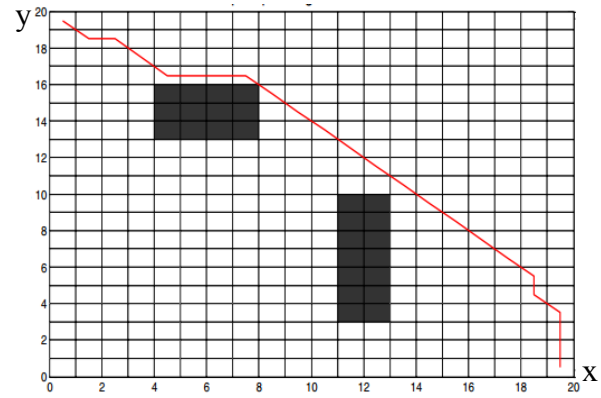
b) un ensemble de fourmis parcourent des trajets et tracent des pistes de

c) Les arêtes du meilleur chemin sont plu renforcées que les autres

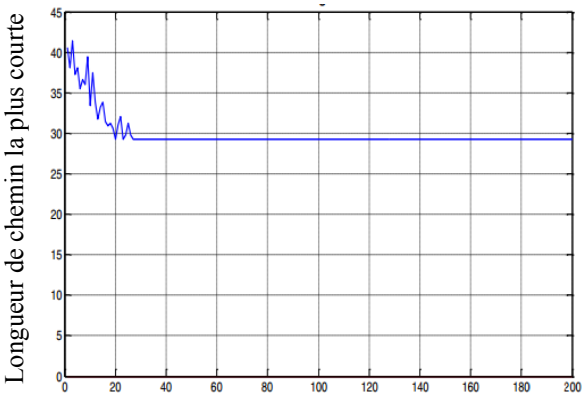
d) l'évaporation fait disparaitre les mauvaises solutions

Figure III-6: Principes de l'algorithme ACO.

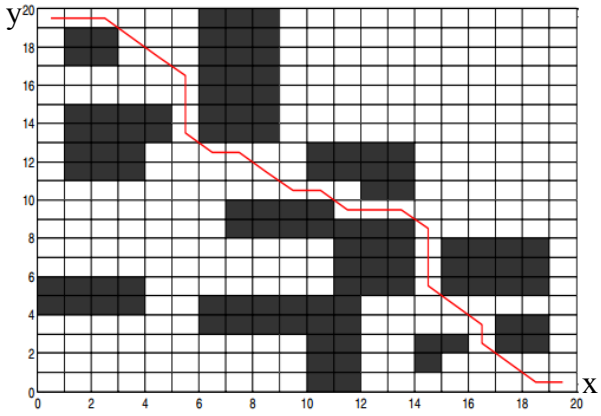
La figure III-7 montre que l'algorithme ACO peut toujours obtenir le plus court chemin pour les UAV même avec l'augmentation du nombre d'obstacles, et il a toujours une vitesse de convergence rapide pour la planification du chemin des UAV [26].



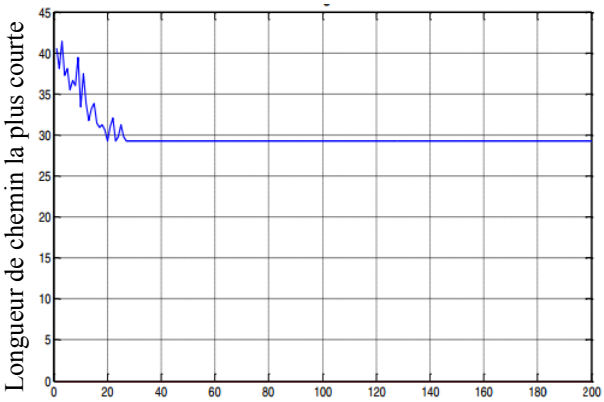
Résultat de l'itinéraire de l'UAV



Courbe de convergence ACO



Résultat de l'itinéraire de l'UAV



Courbe de convergence ACO

Figure III-7 : Comparaison entre la planification de chemin UAV avec deux obstacles et avec plusieurs obstacles.

➢ ***Avantages de l'algorithme ACO***

- ACO a une grande efficacité et une grande robustesse.
- ACO a une grande adaptabilité.

➢ ***Inconvénients de l'algorithme ACO***

- Coût parfois élevé lord de la génération de solution.
- Temps d'exécution parfois long.

III.1.4. Hybrid genetic Algorithm (HGA)

La méthode HGA proposée dans [27] permet de planifier une trajectoire pour un drone en dépend des régions non convexes (zones d'exclusion aériennes, des aéroports, des zones militaires, etc.), dont le drone doit garder une distance de sécurité, et, les incertitudes qui sont liées à la position du drone et des facteurs externes tel que la turbulence du vent. Ceci est fait en combinant les deux méthodes Muti-Population Genetic Algorithm (MPGA) [28], Visibility

Graph (VG) [29] pour trouver tous les chemins possibles en résolvant la non-convexité associée. La route la plus sécurisée pour le drone est définie en utilisant un modèle de résolution linéaire.

Dans la méthode VG, les nœuds sont les coins des obstacles et le point de destination, où ceux qui sont connectés entre eux sont dits «visibles » l'un pour l'autre. Pour diminuer le temps de calcul les connexions qui ne seront pas sélectionnées (des nœuds qui sont attachés à des obstacles) seront éliminées.

La MPGA permet de représenter les régions non-convexes par des zones peuplées, et les solutions (chemins partiels) de chaque population sont représentées par des individuels où sont structurés dans des arbres ternaires dont le meilleur individu est positionné sur la racine de l'arbre. Un opérateur de migration est utilisé pour échanger les individuels (chemins partiels) entre les différentes populations.

Les différentes étapes de cette méthode peuvent être résumées comme suit :

- générer tous les individus pour chaque population d'une façon aléatoire dans un premier temps.
- Evaluer chaque solution partielle en la transformant en une solution complète avec le modèle de résolution proposé, et retourner une valeur d'aptitude.
- Structurer chaque population dans un arbre ternaire avec les individus disposés hiérarchiquement dans différents clusters (groupes). chaque cluster est composé de quatre individus sur deux niveaux dont l'individu le plus apte sera à la racine du cluster. Le cluster chef (leader) est positionné dans la racine de la hiérarchie de l'arbre comme le montre la figure III-8 :

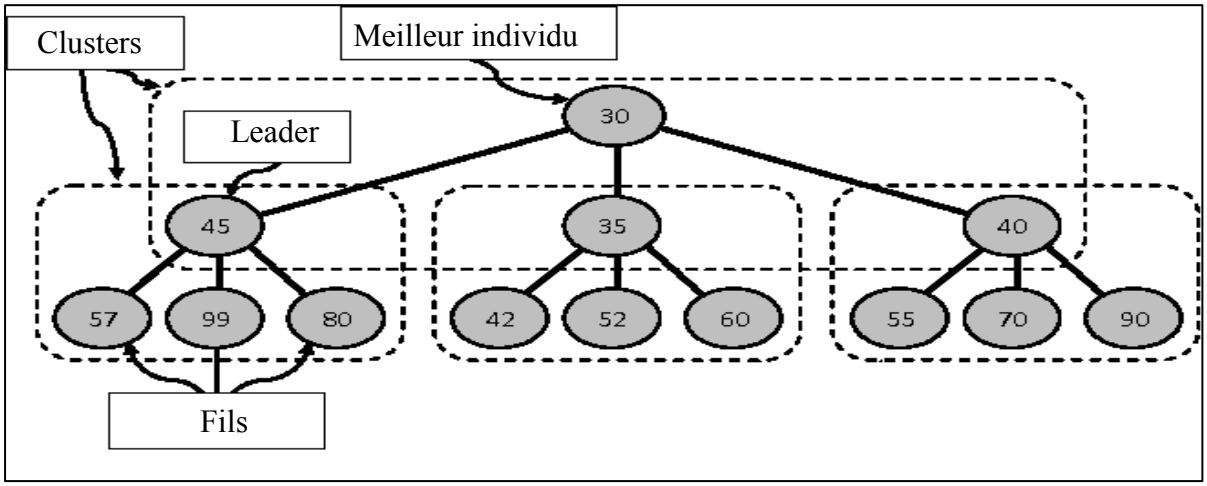


Figure III-8 : Individus structurés dans un arbre ternaire.

- Sélectionner deux individus aléatoirement (parent qui est le leader du cluster et l'autre choisi au hasard parmi les fils) pour reproduire un nouvel individu par le croisement de ces derniers et éliminer les sommets répliqués. une évaluation est appliquée pour le fils pour définir la solution optimale et son aptitude.
- Remplacer les parents précédant avec les nouveaux individus si l'aptitude de ces derniers est meilleure.
- S'il n y a pas de nouveaux individus générés pour la population après un nombre d'exécution alors les meilleurs individus seront migrés vers une autre population.
- Une fois la limite du temps est atteinte la meilleure route trouvée est retournée.

La figure III-9 montre un exemple d'exécution de cette méthode :

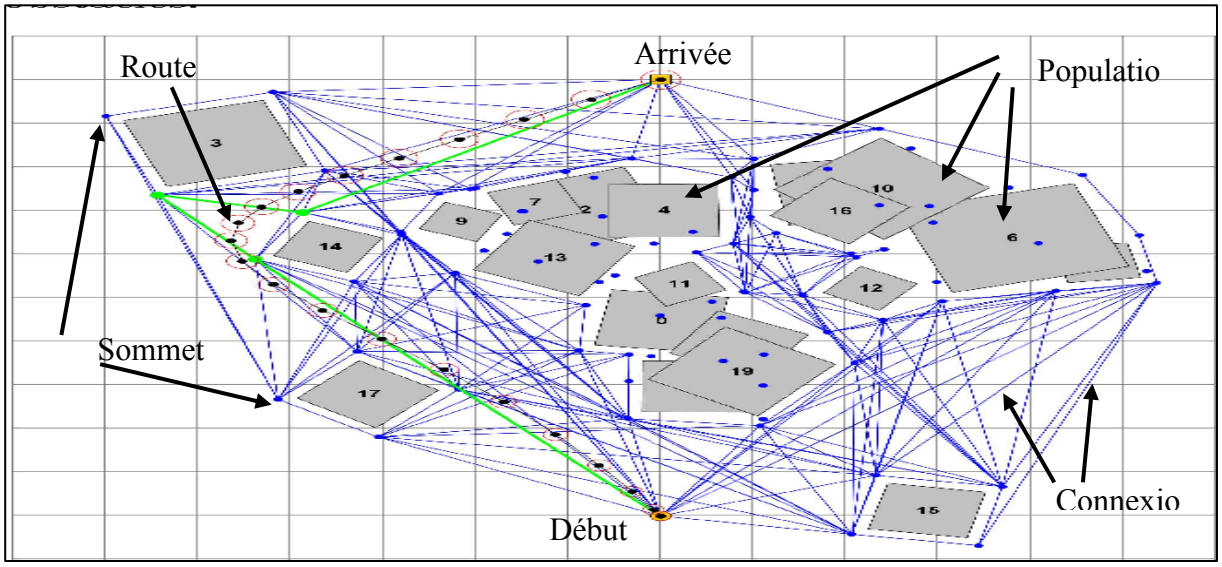


Figure III-9 : Exemple d'exécution de HGA.

➢ ***Les avantages***

- Temps d'exécution pour trouver une trajectoire très réduit.

➢ ***Les inconvénients***

- la HGA est pire en termes de fonction objective comme la consommation du carburant.

III.1.5. Dynamic Adaptive Ant Lion Optimizer (DAALO)

L'algorithme évolutionnaire DAALO qui est proposé dans [30] est utilisé pour la planification de trajectoires pour les UAVs dans un environnement 2D. Il traduit le phénomène naturel d'attaques des fourmis par le fourmilion (insecte qui se nourrit des fourmis. Elle creuse un trou pour piéger ses victimes). Le fonctionnement de cette méthode comprend le mouvement aléatoire des fourmis, la construction des pièges des fourmilions, le piégeage des fourmis dans les pièges, le rattrapage des fourmis et la reconstruction des pièges en prenant en considération la taille du trou (plus le fourmilion mange des fourmis, plus il devient fort et plus la taille des pièges est grande, ce qui implique une plus grande probabilité de rattraper des fourmis).

- ***Modélisation de la route***

L'environnement est modélisé dans un espace orthogonal *o_xy*, où les obstacles sont des cercles ou des rectangles. La détermination de la route consiste à représenter le point de départ P_s et le point d'arrivée P_t, les connecter par une ligne droite. Ensuite, diviser la droite P_sP_t en (D+1) segments équivalentes et transformer l'axe ***(x, y)*** dans un nouveau axe ***o'_x'y'*** dont ***x'*** = P_sP_t, et P_s est l'origine. Cela diminue le calcul puisque les coordonnées sur l'axe ***x'*** sont connues (le calcul se fait seulement pour la coordonnée ***y'***). Pour chaque point d'un segment, une ligne perpendiculaire est tracée. Enfin, un point qui répond le mieux aux objectifs dans chaque segment ($P_1, P_2, \cdots P_D$) est choisi pour contribuer à la route.

La figure III-10 suivante illustre cette modélisation :

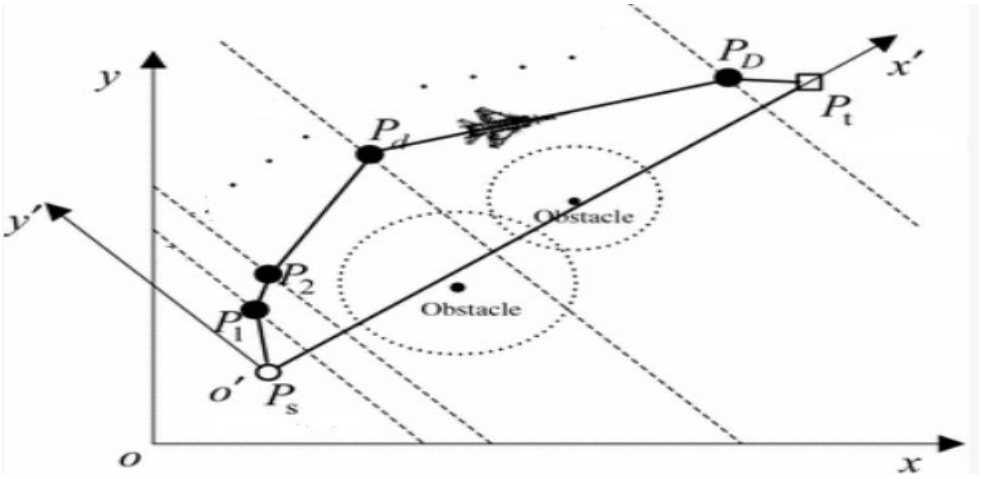


Figure III-10 : Modèle de la route planifiée.

Les contraintes physiques de l'UAV sont prises en considération pour planifier la trajectoire, on trouve principalement le degré de rotation toléré et l'endurance maximale, dont la fonction du coût est basée sur ces deux contraintes.

- ***Principe de fonctionnement du DAALO***

Le processus simplifié est comme suit :

- Le fourmilion creuse le piège.
- Se cache dedans et attends une fourmi.
- La fourmi arrive et glisse dans le trou.
- Le fourmilion la rattrape et la mange.
- Le fourmilion reconstruit le piège (il est noté que la taille du piège est proportionnelle au nombre de fourmis mangées, si le piège est plus grand alors la probabilité de rattraper plus de fourmis est plus grande).

Pour modéliser mathématiquement ce processus deux espèces à prendre en considération les fourmis et les fourmilions. Les fourmis se déplacent dans l'espace de recherche et les fourmilions les piègent avec le trou le moins coûteux. Pour cela, deux populations seront créées : une pour les fourmis et l'autre pour les fourmilions. Pour chaque individu d'une population sa position et sa valeur d'aptitude sont associées.

- ***Le mouvement aléatoire***

Une génération aléatoire des déplacements de fourmis d'une sorte à ce qu'elles auront 3 mouvements possible (fluctuation autour de la position initiale, comportement incrémental ou tendance décroissante) autour des fourmilions. Ces mouvements sont bornés entre deux valeurs (minimale et maximale) de saut à faire pour chaque fourmilion.

- ***Construction des pièges***

Plus le fourmilion est fort (la fonction du coût est réduite) la probabilité de rattraper des fourmis est grande. Pour cela, sélectionner le fourmilion le plus apte (***P_{sel}***) à partir de la population courante des fourmilions, et enregistrer la meilleure solution obtenue jusqu'à présent comme élue (***P_{elite}***). Les pièges des ***P_{sel}*** et ***P_{elite}*** doivent être considérés en même temps vu qu'ils influencent le mouvement des fourmis durant les itérations. D'où le mouvement des fourmis est défini autour des ***P_{sel}*** et ***P_{elite}***.

- ***Le piégeage des fourmis et ajustement dynamique des pièges***

Définir le saut minimal et maximal des fourmis pour chaque itération pour avoir un meilleur espace de recherche pour chaque itération et assurer la convergence. Pour cela les recherches locales et globales doivent être équilibrées. La taille des trous de chaque itération doit être ajustée dynamiquement en récupérant le taux d'avancement de la population précédente :

- Si le taux est inférieur à 20% alors l'espace de recherche est large ce qui permet l'évitement des minimas locaux mais qui manque de précision, où l'espace de la population courante est réduit.
- Si le taux est supérieur à 20% alors les solutions sont focalisées sur la recherche locale, par conséquent augmenter l'espace de recherche pour améliorer la recherche globale.
- Si le taux est 20 % alors les recherches locale et globale sont équilibrées.

▪ ***Rattrapage des fourmis et reconstruction des pièges***

Le rattrapage d'une fourmi est fait lorsque sa valeur d'aptitude est meilleure que celle du fourmilion, donc le fourmilion mis à jour sa position vers celle de la fourmi chassée. Pour achever le processus, toutes les valeurs d'aptitudes des fourmis et fourmilions sont concaténées et classées selon un ordre croissant. Les *N* premières valeurs sont déterminés comme la mise à jour des coûts des fourmilions et les positions correspondantes.

- ***Etapes de l'exécution du DAALO***

1. A partir des informations de l'environnement (l'espace, la taille, la position et le type d'obstacles) et les performances de l'UAV on détermine les contraintes de l'environnement (la partie éloignée des obstacles et les parties qui sont dans les obstacles) et contraintes dynamiques de l'UAV.
2. Initialisation des paramètres tel que le nombre de fourmis et fourmilions, leurs dimension de positions *D*, le nombre maximum d'itérations, définir les bornes *min* et *max* initiaux des sauts. Ensuite, choisir les points de passages initiaux (seules les coordonnées *y* seront prises) et calculer le coût de la route générée par la première population des fourmilions. Choisir le $\boldsymbol{P_{sel}}$ avec le modèle de sélection ainsi que le $\boldsymbol{P_{elite}}$ enregistré.
3. Adapter la taille des trous dynamiquement par la mise à jour du saut minimum et maximum autorisés autour des $\boldsymbol{P_{sel}}$ et $\boldsymbol{P_{elite}}$.
4. Déterminer le mouvement aléatoire des fourmis autours des fourmilions $\boldsymbol{P_{sel}}$ et $\boldsymbol{P_{elite}}$.
5. Obtenir la position de toutes les fourmis et calculer le coût de la route correspondante.
6. Mettre à jour les positions de tous les fourmilions et choisir un $\boldsymbol{P_{sel}}$ et enregistrer le $\boldsymbol{P_{elite}}$.
7. Si le nombre d'itérations maximal est atteint, on fait sortir la meilleure solution et la route correspondante. Sinon, aller à l'étape 3.

➢ ***Les avantages***

- Une convergence rapide vers le but.
- Moins de calculs effectués et moins de ressources consommées.

➢ ***Les inconvénients***

- Absence de la considération des obstacles dynamiques ce qui la rend inadéquate dans un environnement dynamique.
- Le nombre d'itérations pour un bon résultat est limité.
- Des obstacles qu'on ne peut pas représenter avec les cercles ou rectangle.

I.6. Méthodes de planification hors ligne / en ligne

Dans cette approche, la planification de trajectoire combine les deux approches précédentes, où l'environnement contient des obstacles statiques et dynamiques et le chemin établi est basé sur la recherche hors ligne et en ligne.

III.1.6. Multi-Objective path planning method

La méthode présentée dans [31] est une méthode de planification de trajectoire multi objective pour les UAVs dans un environnement urbain à basse altitude. Son objectif est de planifier une trajectoire optimale en garantissant la sécurité du drone durant le vol. Pour cela elle traite le problème de la pré-planification statique (hors ligne) qui consiste à trouver la meilleure route en termes de temps abusé et l'évitement d'obstacles statiques, et la re-planification dynamique qui est en temps réel (en ligne) qui permet de contourner les obstacles imprévus détectés par des capteurs embarqués.

Dans cette méthode, l'espace de recherche est modélisé avec un graphe spatial bidimensionnel *G (N, A, c)*. *N* est l'ensemble des nœuds du graphe dont chacun représente une cellule de la grille de la carte géographique. *A* est l'ensemble des arcs qui relient les différents nœuds. Une fonction du coût ***c (c1, c2)*** est associée pour chaque arc, ***c1*** qui est la durée nécessaire pour passer d'un nœud à un autre, pendant que ***c2*** représente la sécurité de ce parcourt vis-à-vis des obstacles.

Dans ce cadre, l'auteur a proposé le concept suivant :

➢ Introduction du concept d'indice de sécurité avec lequel une carte d'indice de sécurité *SIM* (Safety Index Map) est établie pour l'environnement. Précisément, deux types de cartes *SIM* sont construites, à savoir :

- ***Une carte SIM statique :*** Construite hors ligne à partir de la grille de la carte géographique, elle incorpore les principaux obstacles dans la carte géographique correspondante en considérant l'erreur de positionnement GPS.
- ***Une carte SIM dynamique :*** Construite en ligne (en plein vol), elle inclut les obstacles inattendus qui ne sont pas disponibles dans la carte géographique,

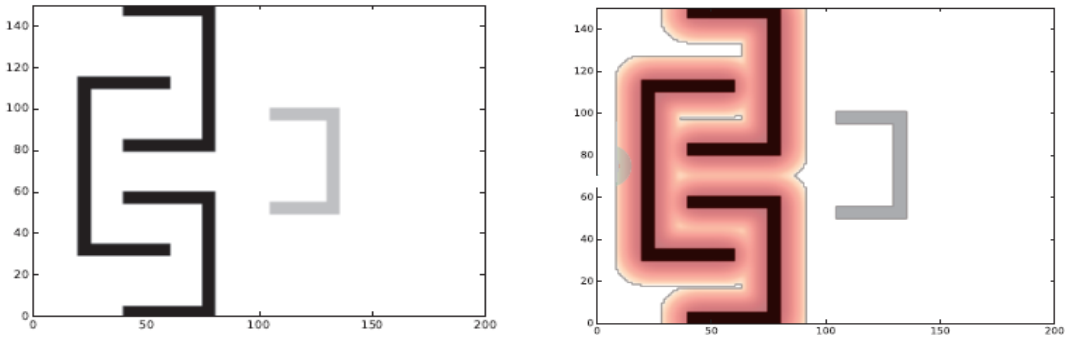
concernant à la fois la portée de détection des capteurs et la distance de sécurité de l'UAV.

- Démarrer le processus de la recherche du chemin optimal (en considérant deux objectifs qui sont : **la durée du vol** et **sa sécurité**) en exploitant conjointement la recherche hors ligne et en ligne qui sont basées sur la *SIM* statique et la *SIM* dynamique respectivement. Un algorithme de recherche hors ligne (*OFSA* : Offline Search Algorithm) est utilisé pour trouver le chemin de durée courte et qui permet d'éviter les obstacles statiques à base de la *SIM* statique, pendant qu'un algorithme de recherche en ligne (*ONSA* : Online Search Algorithm) permet de trouver le chemin optimal pour contourner rapidement les obstacles imprévus en utilisant la *SIM* dynamique avec une complexité de calcul faible.

- ***Les étapes d'exécutions de la méthode***

La méthode utilise 5 algorithmes différents qui contribuent à la recherche du chemin, nous allons décrire d'une manière récapitulative les principales étapes :

- **Construction de la carte SIM statique :** l'algorithme dédié à la construction de la *SIM* prend en entrée la carte géographique contenant les obstacles statiques comme l'illustre la figure III-11(a). Pour chaque cellule (représente un nœud) de la grille elle calcule son indice de sécurité *P* par rapport à la marge de sécurité avec les obstacles comme le montre la figure III-11(b), la valeur de *P* est entre 0 et 1. *P* se rapproche du 1 s'il respecte la marge de sécurité par rapport aux obstacles. Pour chaque cellule contenant un obstacle elle sera marquée comme interdite.



(a) (b)

Figure III-11 : Map avec obstacles sur la gauche, et la SIM correspondante sur la droite.

- **Construction du coût pondéré des nœuds :** l'algorithme *CWCDM* (Construction of a Weighted Cost Distribution Map) est responsable d'affecter un coût pour chaque nœud de la *SIM*. Pour cela, il prend comme entrée le point de départ et d'arrivée, la SIM, un facteur de pondération α (pondérer par rapport à la sécurité du chemin et sa

durée du temps) et la vitesse de l'UAV ***V***. Ensuite, pour chaque nœud (qui n'est pas un obstacle) dans la *SIM* un coût lui serait attribué. Le coût est égal à la distance entre un nœud et son voisin divisée par la vitesse pondérée (***V / (SIM** [nœud]*V*α** + *(1-α))*).

- **Construction du chemin optimal :** l'algorithme *OFSA* prend en entrée le point de départ, d'arrivée et le coût des nœuds. A partir du point de départ, *OFSA* empile le voisin qui a le meilleur coût pour chaque nœud courant jusqu'à atteindre le point d'arrivée.
- **exploitation du chemin :** tant que le nœud courant n'est pas le but, on exploite le chemin fourni par *OFSA*. Pour chaque nœud atteint, Si un obstacle imprévu est détecté la *SIM* dynamique affectera une étiquette ***NOUVEAU_INTERDIT*** si la distance de ce dernier est inférieure à la marge de sécurité définie. Une liste des nouveaux obstacles sera retournée, ensuite, on appelle l'algorithme *ONSA*(*) pour recalculer le chemin à partir du nœud courant, et on exploite le chemin fourni par *ONSA*. Si le but n'est pas encore atteint alors on recalcule le chemin à partir du nœud courant avec l'algorithme *OFSA*.

(*) **ONSA :** il s'est inspiré de *A**, la différence est que l'heuristique est basée sur le coût fourni par l'algorithme *CWCDM* et pas la ligne droite vers le but comme celle de *A** ce qui permet de réduire l'espace de la recherche. Aussi, une autre différence est qu'il peut retourner le chemin retrouvé dès que la ligne qui part du nœud courant au but ne se croise pas avec la zone qui représente les nouveaux obstacles. Cet algorithme permet de réduire le temps d'exécution et garantir une réaction en temps réel.

La figure III-12 représente un exemple d'exécution de l'algorithme : les images (a)-(d) montrent que le drone suit le chemin optimal généré par *OFSA*. Par la suite, dès qu'il détecte l'obstacle inattendu qui rentre dans le cercle de détection, *ONSA* recalcule le chemin pour l'éviter (les images (e)-(i)).

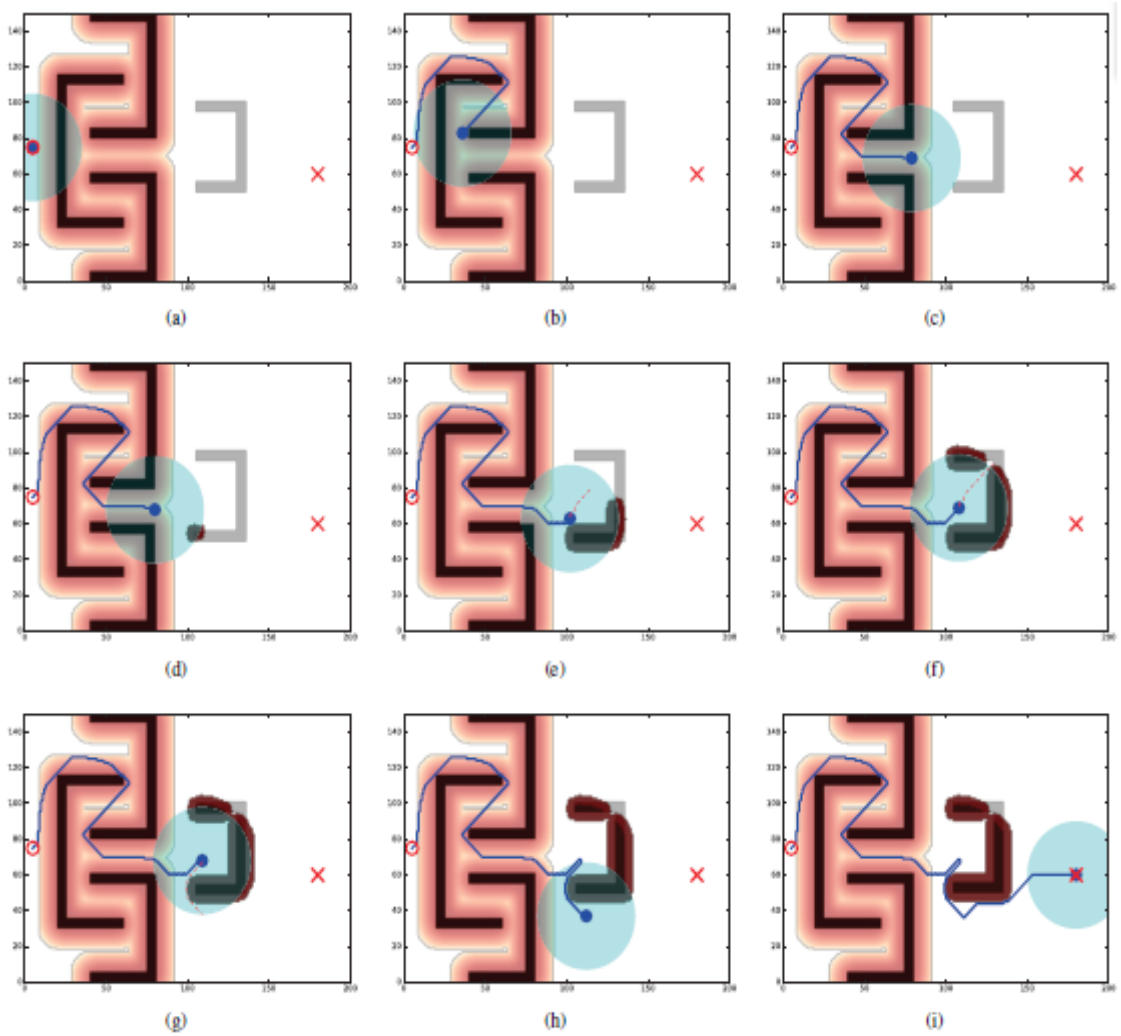


Figure III-12 : Exemple d'exécution de la MOPP.

- ***Les avantages***
  - Agiles dans les environnements à basse altitude
  - Retourne des chemins adaptables aux préférences des utilisateurs concernant la sécurité et la durée des chemins.

- ***Les inconvénients***
  - L'impossibilité d'obtenir un chemin hautement sécurisé et de durée trop courte puisque les deux objectifs se contredisent (la sécurité et la durée du temps).

I.7. Etude comparative

Nom de la méthode	Type	Nombre de dimensions	Temps d'exécution	L'optimalité	L'idée de base
Méthode de planification de trajectoire géométrique	En ligne	2D/3D	Court délai	Non optimale	Aspects géométriques
Algorithme d'optimisation de colonies de fourmis ACO	Hors ligne	2D	Délai parfois long	optimal	Comportement des fourmis
Hybrid Genetic Algorithm HGA	Hors ligne	2D	Court délai	Non optimal	Algorithme MVGA et VG
Multi-Objective Path Planning method	En ligne/hors ligne	2D	Délai parfois long	Non optimale	Recherche en ligne et hors ligne
Dynamic Adaptive Ant Lion Optimizer DAALO	Hors ligne	2D	Court délai	optimal	Attaque des fourmis
Improved Arificial Potential Field APF	En ligne	3D	Court délai	Non optimale	Méthode APF

Tableau III-1: Comparaison d'algorithmes pour la planification de trajectoires UAV.

- ***Méthode de planification de trajectoire géométrique***

Cette méthode est utilisée pour planifier une trajectoire d'un UAV dans un environnement bidimensionnel et tridimensionnel. Efficace en termes de calcul en temps réel, elle est basée sur les aspects géométriques et elle ne nécessite pas un modèle de grille pour trouver le chemin. Malgré son fonctionnement complet elle reste non-optimale par rapport aux objectifs multiples.

- ***Algorithme d'optimisation de colonies de fourmis ACO***

Cet algorithme est utilisé pour établir un plan efficace de planification de chemin UAV, c'est un algorithme bidimensionnel hors ligne, basé sur le comportement des fourmis. Il est adaptable pour tous les plans de planification. Le chemin retourné est optimal, mais le coût et le temps d'exécution sont parfois coûteux.

- ***Hybrid Genetic Algorithm HGA***

Cet algorithme est utilisé pour planifier un chemin dans un environnement non convexe, c'est un algorithme bidimensionnel hors ligne basé sur un algorithme génétique multi populations avec un graphe de visibilité. Il trouve des solutions dans un délai du temps réduit. Il reste non-optimal vu qu'Il y a toujours un risque de collision avec les obstacles.

- ***Multi-Objective Path Planning method***

Cette méthode est utilisée pour planifier une trajectoire d'un UAV dans un environnement bidimensionnel urbain dynamique à basse altitude. Elle est à la fois en ligne et hors ligne, où la recherche hors ligne est exploitée pour trouver le chemin le plus court avec évitement d'obstacles statique basé sur la carte *SIM* statique, pendant que la carte *SIM* dynamique est construite lors de la détection de nouveaux obstacles afin de les éviter avec l'algorithme *ONSA*. MOPP est une méthode efficace qui répond à l'exigence de sécurité, et elle a un temps de parcours court mais elle n'est pas hautement optimale en ce qui concerne la continuité de trajet et sa sécurité.

- ***Dynamic Adaptive Ant Lion Optimizer DAALO***

Cet algorithme est utilisé pour planifier une trajectoire pour un UAV, c'est un algorithme bidimensionnel hors ligne basé sur le phénomène naturel d'attaques des fourmis. Il est efficace en termes de calcul et d'objectivité. Il a une vitesse de convergence et de précision rapide, et il peut planifier un itinéraire plus lisse et plus court. Il ne prend pas en considération les obstacles dynamiques.

- ***Improved Artificial Potential Field APF***

Cette méthode est utilisée pour planifier une trajectoire d'un UAV dans un environnement tridimensionnel, elle est basée sur APF, elle est efficace et s'exécute en temps réel. Le problème des minimas locaux est éliminé avec cette méthode, ainsi que l'évitement d'obstacles est assuré. La méthode ne prend pas en considération la cinématique de l'UAV, sa taille et son altitude.

I.8. Conclusion

Nous avons vu dans ce chapitre quelques méthodes de planification récentes qui sont dédiées aux UAVs, nous avons présenté leurs principes de traiter avec l'environnement, dont les algorithmes génétiques pour aboutir à une solution. Nous avons aussi mentionné quelques avantages et inconvénients de chaque méthode. Une étude comparative est ensuite établie pour comparer les différents algorithmes que nous avons présentés durant ce chapitre.

Dans le chapitre suivant, nous allons présenter une technique de planification de trajectoire pour notre drone quadri rotors basée sur la méthode Multi-Objective Path Planning (MOPP) en raison de ses multiples avantages tel que sa possibilité d'établir une planification en combinant la recherche hors ligne et en ligne (considération complète de l'environnement). Son efficacité en termes de calcul et de sécurité, ainsi que son adaptabilité aux différents environnements.

I.9. Introduction

La planification de trajectoire d'un UAV consiste à déterminer un chemin sans collision entre sa position de départ et celle d'arrivée dans un environnement contenant des obstacles. Plusieurs méthodes et algorithmes ont été proposés pour résoudre le problème de planification (cf. chapitre 3). MOPP [31] est l'une des méthodes dont l'objectif est de trouver un chemin optimal, tout en respectant les exigences de ce type d'environnement, particulièrement en termes de sécurité et de temps de parcours. La particularité de cette méthode est son adaptation à la recherche de chemin aussi bien en mode hors ligne qu'en celui en ligne. Les résultats obtenus lors de différentes expérimentations et simulations démontrent l'efficacité de cette méthode pour la détection de trajectoire. Néanmoins, son grand inconvénient est qu'elle n'assure pas une continuité et une sécurité cohérente lorsque le drone passe à proximité des obstacles. Dans le présent travail, nous proposons un ensemble d'améliorations ayant pour objectif d'améliorer la prise en charge de cette méthode des deux besoins de continuité et de sécurité, qui sont des aspects critiques et primordiaux pour la planification de trajectoires.

Dans ce chapitre, nous allons d'abord introduire notre méthode ImpMOPP (Improved MOPP), qui est une version améliorée de la méthode MOPP. Dans un deuxième temps, nous allons détailler le mécanisme de cette méthode ainsi que les différentes étapes de sa conception et sa réalisation.

I.10. Aperçu global de la méthode ImpMOPP

La méthode ImpMOPP (Improved Multi Objectif Path Planning) est une version améliorée de MOPP, ayant pour objectif de planifier un chemin optimal pour un drone afin que celui-là puisse éviter les obstacles éventuels pouvant se trouver dans l'environnement, que ces derniers soient statiques ou dynamiques. Les améliorations proposées portent sur deux aspects fondamentaux :

1. Assurer la continuité lors de la navigation
2. Garantir un niveau de sécurité optimal pendant le vol.

Comme la méthode de base MOPP, notre méthode est basée sur la recherche hors ligne et en ligne, en tenant compte de deux objectives, à savoir le temps de vol et le niveau de sécurité. Dans cette méthode, deux algorithmes sont introduits pour les deux mécanismes de recherche (hors ligne et en ligne).

L'algorithme de recherche hors ligne OFPS (Offline Path Search) est exploité pour planifier un chemin optimal permettant d'éviter les obstacles statiques. Pour cela, l'algorithme

applique la méthode FM² (Fast Marching Square) dont un des objectifs est la construction de la map de vitesses à partir d'une grille binaire dans laquelle les obstacles sont évalués à 0 et les espaces libres à 1. L'algorithme de recherche en ligne ONPS (Online Path Search) est quant à lui exploité lorsque des obstacles inattendus sont détectés pour les contourner, et cela en construisant une carte de traitement de nouveaux obstacles dynamiques PNO (Processing New Obstacles).

I.11. Description détaillée de ImpMOPP

La méthode ImpMOPP construit d'abord une map des vitesses. C'est sur cette base que nous construisons une nouvelle méthode de planification de trajectoire permettant de résoudre le problème de continuité et de sécurité. Pour ce faire, cette map est utilisée lors de l'exécution des mécanismes de recherche hors ligne et en ligne. La carte de définitions de nouveaux obstacles PNO est en ligne lorsqu'on détecte un obstacle inattendu concernant la portée et la distance de sécurité du drone.

IV.1.1. Le problème de planification de chemin multi-objectif

La planification de trajectoire d'UAV dans un environnement urbain complexe à basse altitude est limitée par un ensemble de contraintes internes (heure d'arrivée, consommation d'énergie…) et contraintes externes (d'autres UAVs, lignes à haute puissance…). Dans le cadre de planification de trajets, il existe deux objectifs principaux de décision, à savoir la sécurité et le temps de parcours.

- ***La sécurité***

L'objectif de la sécurité est modélisé avec un critère d'évitement des collisions qui nécessite une distance de séparation des obstacles. De plus, il y a une incertitude dans la position, la vitesse et la direction des obstacles dynamiques en raison d'une erreur du capteur. Cette incertitude peut augmenter le risque de collision : plus les cellules sont proches des obstacles, plus le coût de croisement est élevé, ce qui représente un danger des cellules.

- ***Le temps de parcours***

Le plan de mission doit être aussi optimisé pour répondre à un objectif important qui est le temps de déplacement. Ce dernier est généralement limité pas la vitesse de croisière et la trajectoire de vol.

La solution pour le problème d'optimisation multi-objective (le temps de parcours et la sécurité) consiste à trouver un chemin optimal p entre le point source et le point cible dans un graphe $G\ (N, A, c)$, où N représente un ensemble de nœuds, A désigne un ensemble d'arcs,

$c : A \rightarrow R^k$ représente une fonction de coût du trajet tel que K est le nombre de fonctions objectives. Chaque arc de A a deux coûts non négatifs notés par $c_{i,1}$ et $c_{i,2}$ représentant respectivement le temps de parcours et la sécurité. Soit $g_j(p)$ le coût de tous les arcs dans un chemin p pour le *j-ième* objectif, et soit $g(p) = (g1(p), g2(p))$ le vecteur du coût pour un chemin p. Par conséquent, par rapport à tout autre chemin q, le chemin optimal p doit satisfaire les conditions suivantes :

$$gj(p) \leq gj(q), \forall\, j \in \{1,2\}, et\ \exists i \in \{1,2\}, gi(p) < gi(q).$$

IV.1.2. Construction de la map des coûts des nœuds

La méthode ImpMOPP est décrite par un graphe spatial, représenté par un triplet *(G, S_{start}, S_{target})*, *où G = (N, A, c)* représente un graphe dont N est un ensemble de points, A désigne un ensemble d'arcs, $c : A \rightarrow R^K$ représente une fonction de coût de trajet tel que K est le nombre de fonctions objectives, et le vecteur de coût de dimension K *c (m, n)* indique le coût de la planification d'un chemin commençant au point S_m et se terminant au point S_n. De plus, $S_{start} \epsilon N$ et $S_{target} \epsilon N$ désigne respectivement l'ensemble de points de départ et l'ensemble de points cibles.

Un schéma d'échantillonnage cartésien uniforme est utilisé pour la construction du graphe G où chaque point S est mappé uniquement à une grille ou une cellule dans le graphique, et S fait référence simultanément à la cellule et au point situé au centre de chaque cellule. Pour modéliser le mouvement du drone, un modèle de mouvement à huit connectivités est utilisé, ce qui signifie qu'un UAV est supposé avoir huit directions de déplacement possibles comme montré sur figure IV-1.

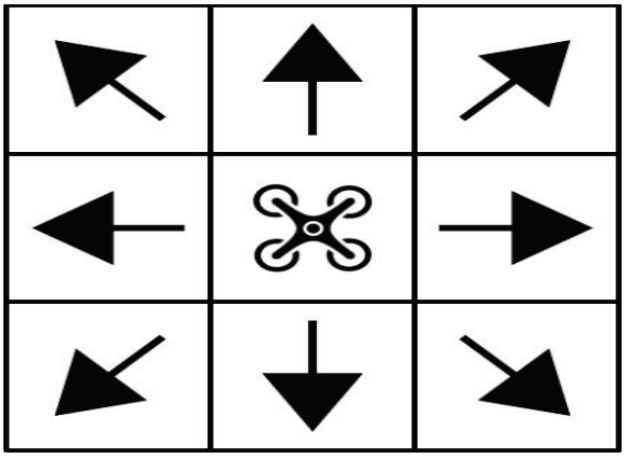


Figure IV-1 : Modèle de mouvement de drone à huit connectivités.

IV.1.3. La recherche hors ligne (OFPS)

Dans cette phase, un algorithme de planification de trajectoire basé sur une recherche hors ligne OFPS (Offline Path Search) est utilisé pour trouver un moyen d'éviter les obstacles statiques. Dans la recherche hors ligne, plusieurs méthodes existantes permettant de trouver un chemin plus court dans un environnement avec des obstacles. Parmi ces méthodes, nous avons utilisé la méthode FM² (Fast Marching Square), basée sur la FMM (Fast Marching Method) qui est elle-même un cas particulier de la méthode *LS*[17] (Level Set) [32].

- ***La méthode FM***

La méthode FM [33] est un algorithme numérique de calcul très efficace pour suivre le mouvement d'une interface d'ondes physiques (appelée front et notée Γ). Cette méthode a été appliquée à différents domaines de l'informatique, à l'exemple de l'imagerie médicale, du calcul des trajectoires, etc.

Dans cette méthode, le front d'onde est appelé *interface*. Cette dernière peut être un plan bidimensionnel (2D) ou tridimensionnel (3D) ; néanmoins, le modèle mathématique peut être généralisé à n dimensions. La méthode FM calcule le temps T requis par une onde pour atteindre chaque point de l'espace. L'onde peut provenir d'un ou plusieurs points, et chaque point source génère une onde, les points sources ont un temps associé $T=0$.

Dans le contexte de la méthode FM, le front Γ est supposé évoluer par un mouvement dans la direction normale, la vitesse F est non négative et ne doit pas forcement être la même partout. A tout instant, ce mouvement de front est décrit par une équation appelée *Eikonal* sous la forme :

$$1 = F(x)|\nabla T(x)| \qquad (1)$$

Où x est la position, $F(x)$ est la vitesse de l'onde à cette position et $T(x)$ est le temps nécessaire pour que l'interface d'onde atteigne x. L'amplitude du gradient de la fonction d'arrivée $T(x)$ est inversement proportionnelle à la vitesse, ce qui implique l'équation modélisant une expansion d'onde suivant :

$$\frac{1}{F} = |\nabla T| \qquad (2)$$

[17] **LSM :** est une technique numérique d'analyse de surfaces et de formes. Elle permet par exemple d'étudier le mouvement d'un objet en contact avec un objet déformable. Pour plus de détail sur cette méthode, voir la référence 2.

La fonction *T(x)* créée par une onde qui se développe à partir d'un seul point ne présente qu'un minimum global à la source. Comme *F>0*, l'onde ne fait que croître, et donc les points les plus éloignés de la source ont un *T* plus grand.

Etant donné que le front ne peut que s'élargir (*F>0*), le temps d'arrivé *T* est à valeur unique. Une solution discrète pour l'équation Eikonal a été proposée par Osher et Sethian [34]. En 2D, la zone est discrétisée à l'aide d'une grille. L'intersection d'une ligne *i* et d'une colonne *j* de la grille correspond à un point *p* (x_i, y_j) dans le monde réel. La discrétisation du gradient ∇T selon [34] conduit à l'équation suivante :

$$\begin{pmatrix} \max\left(D_{ij}^{-x}T, 0\right)^2 + \min\left(D_{ij}^{+x}T, 0\right)^2 + \\ \max\left(D_{ij}^{-y}T, 0\right)^2 + \min\left(D_{ij}^{+y}T, 0\right)^2 \end{pmatrix} = \frac{1}{F_{ij}^2} \qquad (3)$$

Une solution plus simple et moins précise pour l'équation précédente proposée par Osher et Sethian peut être exprimée comme suit :

$$\begin{pmatrix} \max\left(D_{ij}^{-x}T, -D_{ij}^{+x}, 0\right)^2 + \\ \max\left(D_{ij}^{-y}T, -D_{ij}^{+y}, 0\right)^2 \end{pmatrix} = \frac{1}{F_{ij}^2} \qquad (4)$$

Où

$$D_{ij}^{-x} = \frac{Ti,j-Ti-1,j}{\Delta x}$$

$$D_{ij}^{+x} = \frac{Ti+1,j-Ti,j}{\Delta x} \qquad (*)$$

$$D_{ij}^{-y} = \frac{Ti,j-Ti,j-1}{\Delta y}$$

$$D_{ij}^{+y} = \frac{Ti,j+1-Ti,j}{\Delta y}$$

Δx et Δy sont l'emplacement de la grille dans les directions *x* et *y*, en remplaçant (*) dans l'Equation (4) précédente, et soit :

$$T = T_{i,j}$$

$$T_x = \min(T_{i-1j}, T_{i+1j}) \qquad (5)$$

$$T_y = \min(T_{ij-1}, T_{ij+1})$$

Nous pouvons réécrire l'équation d'Eikonal pour un espace discret 2D comme suit :

$$\max\left(\frac{T-T_x}{\Delta x},0\right)^2+\max\left(\frac{T-T_y}{\Delta y},0\right)^2=\frac{1}{F_{ij}^2} \qquad (6)$$

Comme on suppose que la vitesse du front est positive ($F>0$), T doit être supérieur à T_x et T_y à chaque fois que le front de l'onde n'a pas encore atteint le point aux coordonnées (i, j), nous pouvons résoudre cette équation en trois étapes, tout d'abord, nous résolvons le quadratique suivant :

$$\left(\frac{T-T_x}{\Delta x}\right)^2+\left(\frac{T-T_y}{\Delta y}\right)^2=\frac{1}{F_{ij}^2} \qquad (7)$$

Si $T>T_x$ et $T>T_y$ (en prenant la plus grande valeur de T en résolvant l'équation (7) précédente), la valeur obtenue est la solution correcte pour l'équation (3). Sinon, si $T<T_x$ ou $T<T_y$, à partir de l'équation (6), le nombre correspondant de $(T\text{-}T_1/\Delta x, 0)$ est 0,et par conséquent, l'équation (7) est une équation régulière de la forme $aT^2 + bT + c = 0$, où:

$$a=\Delta_x^2+\Delta_y^2$$

$$b=-2(\Delta_y^2T_x+\Delta_x^2T_y) \qquad (8)$$

$$c=\Delta_y^2T_x^2+\Delta_x^2T_y^2-\frac{\Delta_x^2\Delta_y^2}{F_{ij}^2}$$

Afin de simplifier la notation pour le cas à n dimensions, supposons que la grille est composée de cellules cubiques, c'est-à-dire que $\Delta x = \Delta y = \Delta z = \cdots = h$. Notons T_d comme la généralisation de T_x ou T_y pour la dimension d, jusqu'à N dimensions. F désigne la vitesse de propagation pour un point avec des coordonnées $(i, j, k, \ldots)$. Après simplification des termes, la discrétisation de l'Eikonal est une équation quadratique avec des paramètres :

$$a=N$$

$$b=-2\sum_{d=1}^{N}T_d \qquad (9)$$

$$c=\left(\sum_{d=1}^{N}T_d^2\right)-\frac{h^2}{F^2}$$

- ***Mise en œuvre de la méthode FM***

L'équation d'Eikonal discrète peut être résolue de manière itérative sur une grille. Pour ce faire, les cellules de la grille doivent être étiquetées comme l'un des types suivants :

- UNKNOWN : les cellules dont la valeur *T* n'est pas encore connue (le front d'onde n'a pas atteint la cellule).
- NARROW BAND (bande étroite) : les cellules doivent faire partie de la première vague lors de la prochaine itération ; on leur attribue une valeur *T* qui peut être changé dans les futures itérations de l'algorithme.
- FROZEN (congelé) : les cellules qui ont déjà été traitées et donc ayant une valeur *T* qui est fixe.

L'algorithme de la méthode comprend trois étapes : l'initialisation, la boucle principale et la finalisation.

- L'initialisation : l'algorithme commence par définir *T=0* dans la ou les cellules d'où provient l'onde, ces cellules sont étiquetées comme *FROZEN*. Par la suite, il identifie tous leurs voisins comme étant à bande étroite et il calcule *T* pour chacun d'eux.
- La boucle principale : dans chaque itération, l'algorithme résout l'équation Eikonal pour tous les voisins non gelés de la cellule à bande étroite ayant la valeur *T* la plus faible. Cette cellule est alors étiquetée comme *FROZEN*. La bande étroite conserve une liste ordonnée de ses cellules, classées d'une façon croissante de leur valeur *T*.
- La finalisation : lorsque toutes les cellules sont gelées (c'est-à-dire que la bande étroite est vide), l'algorithme se termine.

Le pseudo code de la méthode FM est le suivant :

Algorithme : FM method.

1. Input : gridmap G of size n*m ; initial wave propagation Ori cells

2. Output : the gridmap G with the T defined for all the cells

3. Initialisation

4. for $g_{ij} \in$ Ori **do**

5. g_{ij} .T ← 0 ;

6. g_{ij} .state ← FROZEN ;

7. **for** $g_{kl} \in g_{ij}$ .neighbors **do**

8. **if** g_{kl} = FROZEN **then** jump to the next point **else**

9. g_{kl} .T← Eikonal_Solver (g_{kl}) ;

10. **if** g_{kl}.state = NARROW BAND **then**

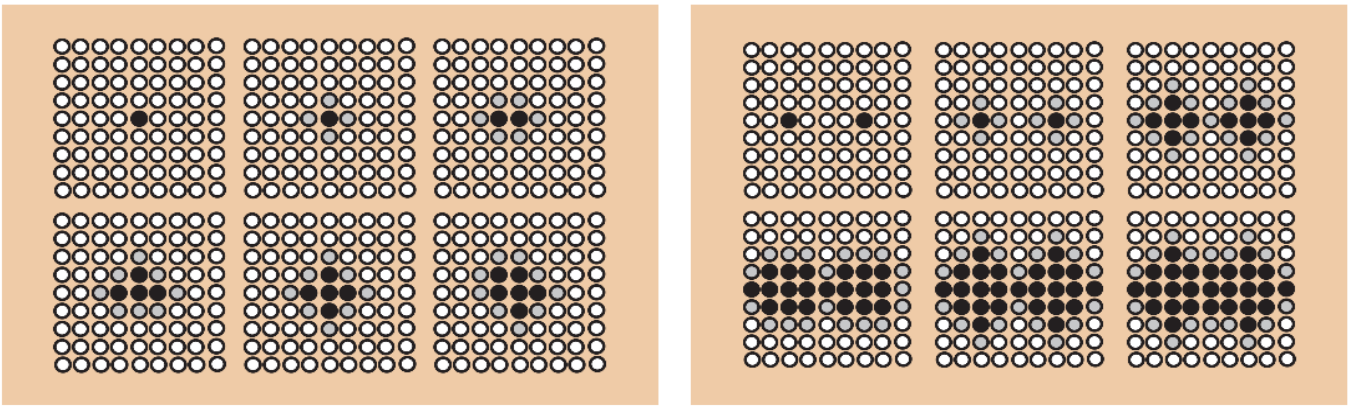
11. NARROW_BANDE.Update_position (g_{kl}) ;

12. **else if** g_{kl}.state = UNKNOWN **then**

13. g_{kl}.state ← NARROW BAND ;

```
14.                     NARROW_BAND.insert_in_position (g_kl) ;
15.                 end if
16.             end if
17.         end for
18.         Loop ;
19.         while NARROW BAND ≠ empty do
20.             g_kl ← NARROW_BAND.popfirst () ;
21.             for g_kl ϵ g_ij .neighbors do
22.                 if g_kl .state = FROZEN then jump ; else
23.                     g_kl .T← Eikonal_Solver (g_kl) ;
24.                     if g_kl.state = NARROW BAND then
25.                     NARROW_BANDE.Update_position (g_kl) ;
26.                     if g_kl.state = UNKNOWN then
27.                       g_kl.state ← NARROW BAND ;
28.                       NARROW_BAND.insert_in_position (g_kl) ;
29.                     end if
30.                 end if
31.             end for
32.         end while
33. end for
```

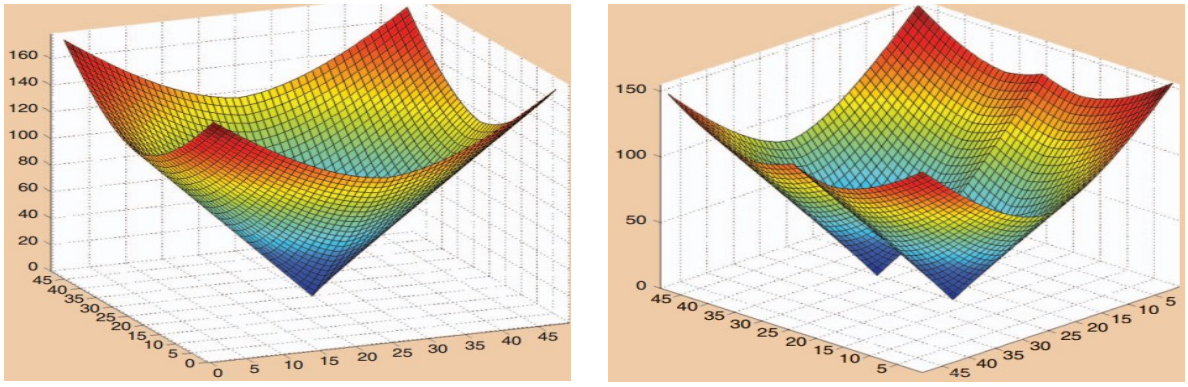
La figure IV-2 illustre le processus implémenté par cet algorithme. Sur la figure IV-2.a, la vague provient d'un point, tandis que sur la figure IV-2.b, deux sources d'onde existent. Les points noirs représentent les nœuds gelés, ayant leur valeur *T* est fixe. Les points gris quant à eux représentent la bande étroite, tandis que les points blancs sont inconnus. Sur la figure IV-2.a, les vagues se développent concentriquement à la source alors que, sur la figure IV-2.b, ils se rejoignent au fur et à mesure que les vagues se développent individuellement et grandissent ensemble. Les cellules avec *T* inferieur sont développées en premier.



a : expansion des ondes itératives avec un point source. b : expansion des ondes itératives avec deux point

Figure IV-2: Processus d'expansion des ondes itératives.

Si on considère T comme la troisième dimension sur l'axe des z, le résultat de l'achèvement de l'expansion des ondes de la figure IV-2 se traduit par la figure IV-3. Puisque $F>0$, et à mesure que l'on s'éloigne des ressources, le temps T nécessaire pour atteindre le point est plus grand (plus haut sur l'axe z). avec une seule source, il n'y a qu'un seul minimum situé à la source, avec plusieurs sources, il y aura autant de minima que de points de départ, chacun situé aux sources d'ondes.



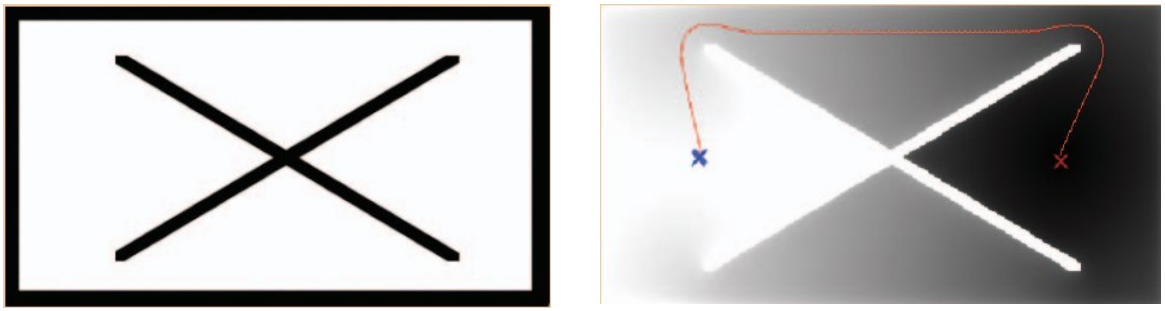
a : un point source b : deux point source

*Figure IV-3 : Méthode FM appliquée sur une grille de 50*50.*

➢ *La méthode FM et la planification de trajectoire*

Dans la planification de trajectoire à l'aide de la méthode FM, une grille binaire dans laquelle les obstacles sont évalués à 0 et les espaces libres à 1 est utilisée. Ces valeurs peuvent être considérées comme la vitesse d'expansion des ondes F sur la grille. Au niveau des obstacles, la vitesse d'expansion de la vague est nulle, car cette dernière ne peut les traverser. Par contre, dans l'espace libre, la vitesse d'expansion des vagues est constante et est égale à 1.

Afin de calculer le chemin entre deux points *P0* et *P1*, nous pouvons développer une onde de *P0* jusqu'à ce qu'elle atteigne *P1* ou vice versa. Le chemin suivi par l'interface entre la cible et la source sera toujours la trajectoire la plus courte dans le temps, et comme la vitesse d'expansion de la vague est constante, cette trajectoire est également la solution la plus courte en distance. La figue IV-4 montre l'obtention du chemin le plus court de *P0* à *P1* par la méthode FM dont la figure IV-4.b montre l'expansion de l'onde en échelle de gris et l'onde provient du point initial marqué d'une croix rouge.



a : carte originale

b : expansion du chemin et des vagues à l'échelle de gris

Figure IV-4 : Exemple d'obtention de chemin par la méthode FM.

La grille obtenue enregistre à tout pixel le temps *T* requis par le front d'onde pour atteindre ce pixel.

Il existe de nombreuses autres méthodes rapides et efficaces sur le plan informatique, basées sur une grille et elles se sont toutes inspirées de la méthode FM. La majorité de ces algorithmes fournissent la même solution. Parmi ces méthodes, on trouve :

- **Simplified Fast Marching Method (SFMM) :** c'est une variante relativement inconnue du FM (FMM) standard, mais ayant de bonnes performances. SFMM est une version réduite de FMM où la bande étroite est implémentée en tant que file d'attente prioritaire simple. Cette file peut contenir différentes instances de la même cellule avec des valeurs différentes. De plus, il peut arriver qu'une cellule soit gelée tout en appartenant à la bande étroite. Chaque fois qu'une cellule ait sa valeur mise à jour, elle est placée dans la file d'attente prioritaire. Une fois qu'elle est ouverte et insérée dans *Frozen*, les instances restantes dans la file d'attente sont simplement ignorées.

- **Fast Sweeping Method (FSM) :** c'est un algorithme itératif qui calcule la carte d'heure d'arrivée en traversant successivement l'ensemble de la grille suivant un ordre spécifique. FSM effectue des itérations dans des directions alternées. Ces directions sont

choisies de sorte que toutes les courbes caractéristiques possibles de la solution Eikonal soient divisées en quadrants possibles ou octants en environnement 3D. Par exemple, une grille bidimensionnelle a 4 itérations possibles (les combinaisons des dimensions *x* et *y* traversant vers l'avant et vers l'arrière sont : le *nord-est*, le *nord-ouest*, le *sud-est* et le *sud-ouest*, comme le montre la figure IV-5, où la cellule la plus sombre est le point initial et les cellules ombrées sont celles analysées par la FSM en cours.

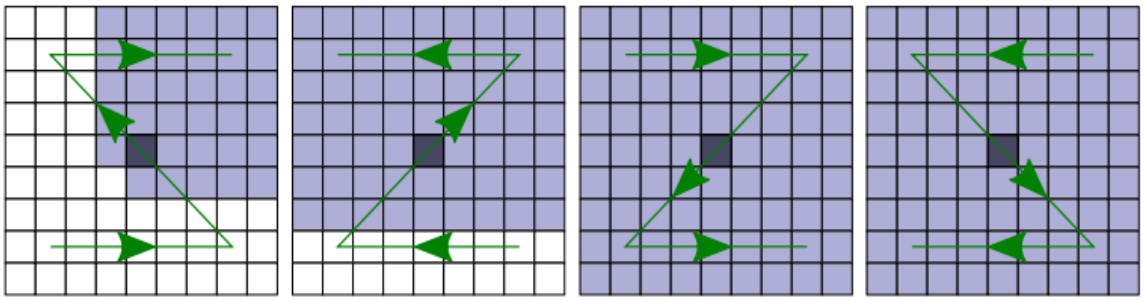


Figure IV-5 : Les directions de FSM en 2D représentées par des flèches.

La FSM effectue des balayages (traversées) des cellules et améliore la valeur de ces dernières jusqu'à ce qu'à atteindre des valeurs constantes. À chaque balayage, l'équation Eikonal est résolue pour chaque cellule. Dans l'algorithme à *N*-dimensions, les directions de balayage sont désignées comme un tableau binaire avec les éléments 1 (sens de traversée en avant) ou −1 (sens de traversée en arrière) dans cette dimension. Ce tableau est initialisé à 1 (*Nord-Est* dans le cas 2D ou *Nord-Est-Haut* en 3D) et la grille est initialisée comme dans FMM. Le FSM effectue autant de parcours de grille que nécessaire jusqu'à ce que la valeur T_i de chaque cellule soit constante.

- **Lock Sweeping Method (LSM) :** c'est une version fondamentalement identique à FSM à laquelle est ajoutée l'option de suivi de l'état de chaque point : verrouillé (*Frozen*) ou déverrouillé (*Narrow*). LSM nomme une cellule comme étant déverrouillée si l'un de ses voisins a changé et que sa valeur peut donc être améliorée. Sinon, la cellule est étiquetée comme verrouillée et sera ignorée. Dans la pratique, il s'avère que la plupart des cellules sont bloquées lors d'un balayage. Par conséquent, le temps de calcul économisé est important.

- **Fast Iterative Method (FIM) :** c'est une méthode inspirée de la méthode FMM qui évalue itérativement chaque point dans une liste *Narrow* jusqu'à ce que sa valeur soit constante. Une fois qu'un nœud converge (sa valeur devient constante), ses voisins sont insérés dans *Narrow* et le processus se poursuit. *Narrow* est implémenté comme une liste

non triée. FIM est conçu pour être efficace pour le calcul parallèle, car tous les éléments de *Narrow* peuvent être évalués simultanément. L'initialisation de FIM est la même que la méthode FM. Ensuite, la valeur de chaque élément dans *Narrow* est mise à jour. Étant donné que *Narrow* est une liste, les nouveaux éléments doivent être insérés juste avant le point en cours d'évaluation. Enfin, ce point est retiré de *Narrow* et étiqueté comme *Frozen*. Un nœud peut être ajouté plusieurs fois à *Narrow* pendant l'exécution de FIM, car chaque fois qu'un voisin parent est mis à jour, le nœud peut améliorer sa valeur. Dans le pire des cas, *Narrow* contient toute la grille et la boucle passe tous les points à plusieurs reprises.

Le chemin calculé en utilisant la méthode FM est le plus court, mais il peut ne pas être sûr en raison de sa proximité par rapport aux obstacles. Pour cette raison, ce chemin ne correspond pas forcément à celui avec le temps le plus court, car l'UAV doit se déplacer lentement lorsqu'il est proche des obstacles afin d'éviter les collisions ou les mouvements risqués. La solution habituelle consiste à élargir les obstacles avant de calculer le chemin, ou bien de calculer des potentiels répulsifs artificiels. Mais quand l'UAV se déplace à travers une porte par exemple, il est préférable, du point de vue de sa sécurité, de passer à un point équidistant des deux extrémités de la porte ; il en va de même pour les coins, etc. Cependant, ces méthodes classiques génèrent d'autres problèmes tels que les minima locaux ou la suppression de chemins réellement réalisables. Pour cela, la méthode FM² est souvent utilisée. Cette dernière est basée sur FM mais avec la particularité d'être :

- robuste : il est facile de prédire sa sortie,
- déterministe : elle ne produit jamais des résultats inattendus ou des échecs,
- efficace : elle analyse l'environnement et renvoie le chemin avec un algorithme relativement léger, et
- sûre : elle définit un chemin qui soit le plus loin possible des obstacles.

➢ *La méthode FM²*

La méthode FM² (Fast Marching Square) citée dans [35] est une variante de la méthode FM, appliquée à une carte discrétisée dans laquelle les obstacles sont étiquetés par 0 et les espaces libres par 1. Son principe de fonctionnement réside dans le fait qu'elle applique deux fois la FM. La première application est pour objectif de construire la carte de vitesses en considérant les obstacles comme source de propagation de l'onde. La deuxième application, quant à elle, sur la carte de vitesses générée, elle permet d'attribuer différents coûts pour les

différents nœuds en dépends de leur vitesse, le point source de propagation est le point cible de la trajectoire, elle s'arrête lorsque l'onde atteint le point de départ de la trajectoire.

- ***La carte de vitesses***

FM² considère tous les obstacles comme des sources d'onde ; ceci signifie que plusieurs ondes se développent en même temps. La figure IV-6.a montre une carte résultante de l'application de cette expansion d'ondes à une carte avec un seul obstacle en forme de croix. Cette carte représente un champ potentiel de la carte originale. A mesure que les cellules s'éloignent des obstacles, la valeur de T_i calculée s'accroît. Cette carte peut être vue comme une carte de vitesses : la valeur T_i est proportionnelle à la vitesse maximale autorisée du drone à chaque point. Comme on peut le constater sur la figure, la vitesse de chaque point diminue à l'approche d'une cellule contenant ou faisant partie de l'obstacle, et augmente lorsqu'il s'en éloigne. Une normalisation des valeurs des cellules vers les vitesses autorisées du robot est effectuée, et une carte de vitesses est générée. Cette dernière définit la vitesse du robot en tout point de l'environnement, de telle sorte à assurer sa sécurité. La représentation en 3D des vitesses des cellules est illustrée sur la figure IV-6.b, et le principe de fonctionnement est définit dans l'algorithme *MAPVelocity*.

- ***Calcul des coûts***

La FM^2 réapplique la FM sur la carte de vitesses générée en considérant le point cible de la trajectoire comme source de propagation d'onde dont le coût est $T_i=0$. Le coût (temps) T_i, pour chaque nœud i, à partir de la cible est calculé en fonction de sa vitesse. Le processus s'achève lorsque le point de départ de la trajectoire est traité. Le résultat obtenu de l'application de FM^2 est une carte dont les cellules ont un coût définit comme temps de parcours entre deux points de la grille.

Pour tracer le chemin optimal entre deux points on calcule le gradient à partir du point initial vers le point d'arrivée et cela en suivant la direction qui maximise la valeur du gradient entre ces deux points. Cette direction est calculée en appliquant le filtre de *Sobel* pour chaque point de la carte comme suit :

$$gradx = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} * T \qquad grady = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * T$$

Où T est le coût (temps de parcours), $grad_x$ et $grad_y$ sont la valeur du gradient sur les deux dimensions x et y respectivement.

Le chemin est calculé d'une manière itérative où $grad_{ix}$ et $grad_{iy}$ sont calculés pour chaque point p_i. Le point p_{i+1} est calculé depuis p_i suivant une des deux équations 10 et 11, jusqu'à ce que le minimum global (le $T = 0$) soit atteint.

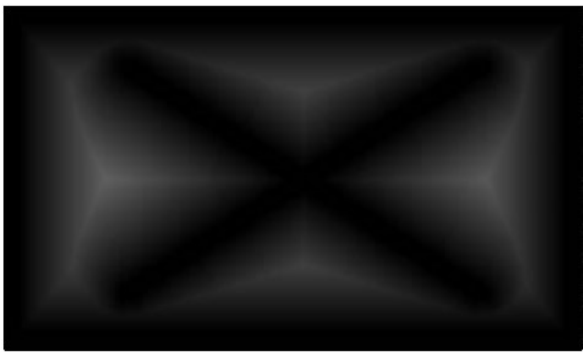
$$mod_i = \sqrt{\mathbf{grad}_{ix}^2 + \mathbf{grad}_{iy}^2} \tag{10}$$

$$alpha_i = arctan\left(\frac{grad_{iy}}{grad_{ix}}\right)$$

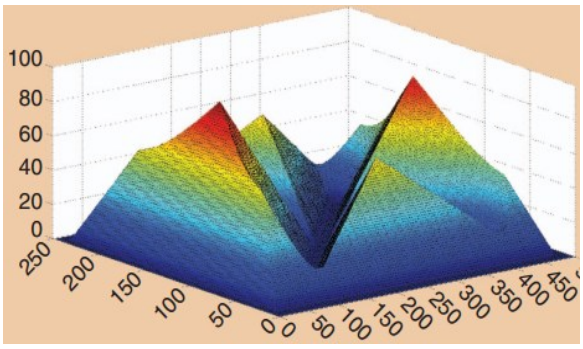
$$p(i+1)x = p_{ix} + step.\cos(alpha_i) \tag{11}$$

$$p(i+1)y = p_{iy} + step.\sin(alpha_i)$$

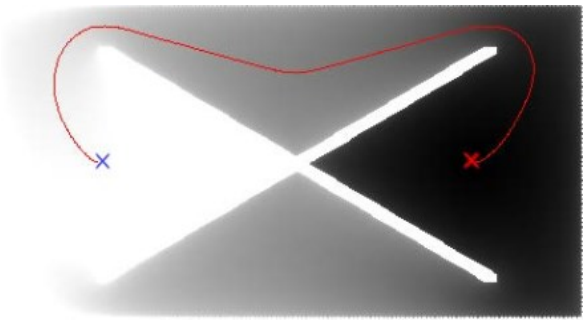
La figure IV-6.c montre le tracé du chemin et la figure IV-6.d donne le profil de vitesse. Nous pouvons voir que la vitesse devient plus grande en s'éloignant des obstacles.



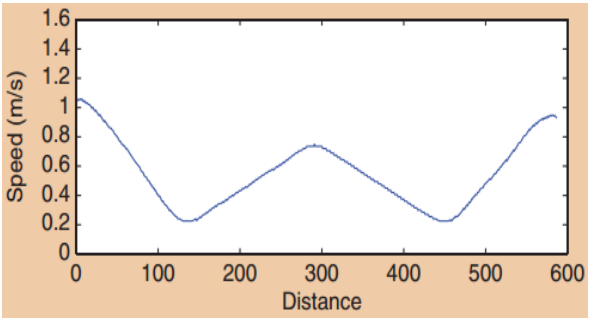
a : Carte de vitesses



b : représentation 3D



c : chemin de FM



d : profil de vitesse

Figure IV-6 : Etapes de FM².

Le chemin obtenu est calculé comme dans FM, de sorte que le drone soit toujours considéré comme un point holonome. Mais au lieu de prendre une valeur constante pour la vitesse de propagation F, la valeur de la carte de vitesses est utilisée. Soit une onde développée à partir d'un point de la grille i. Nous savons que la vitesse d'expansion $F_i = T_i$. Puisque T_i est la valeur de la carte de vitesses à la cellule i, il en résulte que la vitesse d'expansion F_i va donc

dépendre de la position. Si l'on suppose que le drone se déplace à la vitesse maximale autorisée à chaque point, et selon les propriétés de Fast Marching, la trajectoire obtenue est la plus optimale en termes de temps.

- ***FM² : la variation saturée***

Selon la figure IV-6.a, nous constatons que le chemin calculé n'est pas optimal. Ce chemin tente de garder la trajectoire le loin possible des obstacles. Mais il existe des environnements dans lesquels il n'est pas nécessaire de suivre une trajectoire trop éloignée des obstacles (une petite valeur de la distance peut suffire pour assurer une navigation sécurisée). Pour ce genre de contexte, une version saturée de la carte de vitesses est implémentée (la valeur de saturation est comprise entre 0 et 1). Lorsque la première application de FM est achevée, la carte de vitesses est d'abord normalisée puis saturée.

La carte est normalisée en fonction de deux paramètres de configuration :

- Vitesse maximale autorisée, qui est la vitesse de contrôle maximale que le robot peut recevoir.
- Distance de sécurité, qui est la distance de l'obstacle le plus proche à laquelle la vitesse maximale peut être atteinte.

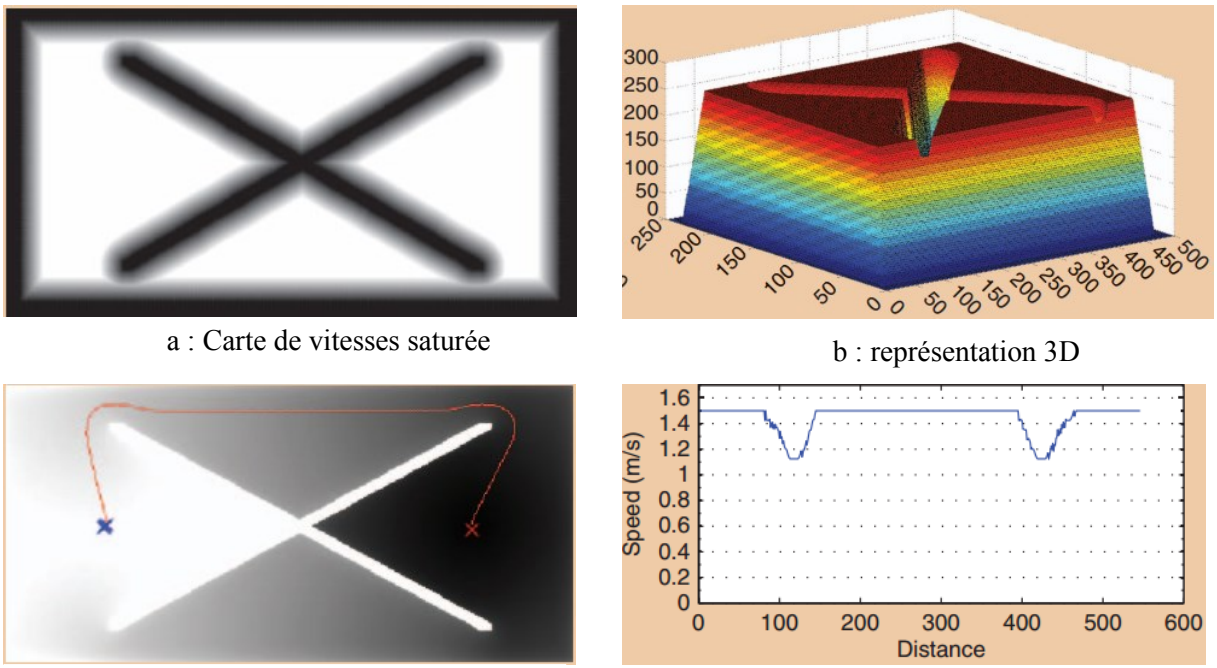
Enfin, la carte est saturée à la vitesse maximale autorisée.

Après ce processus de normalisation et de saturation (voir l'algorithme *MAPVelocity*), les points de la carte qui ont une distance aux obstacles supérieure à celle de la saturation auront la vitesse maximale. Les points qui se trouvent à une distance inférieure à celle de la saturation auront une vitesse nulle au niveau des obstacles, et qui change d'une manière proportionnelle à la distance de saturation. La figure IV-7 montre la variation saturée de la figure IV-6 précédente avec la nouvelle trajectoire calculée. On peut voir que le chemin et le profil de vitesse soient maintenant comme prévus.

Algorithme : MAPVelocity.

```
1.  Input : VELOCITY, SATURATION
2.  Source_obstacles = set containing the obstacles
3.  FM (Source_obstacles, -1, MAP)
4.  maxDistance = SATURATION
5.  maxValue = the maximum T value in the map
6.  maxVelocity = maxDistance / the grid spacing unit
7.  for node in the map do
8.      velocity = node.T / maxValue
9.      If velocity < maxVelocity then
10.         node.velocity = (velocity / maxVelocity) * VELOCITY
```

11. **Else** node.velocity = VELOCITY
12. **End if**
13. node.T = INFINITY
14. node.state = UNKNOWN
15. **End for**



a : Carte de vitesses saturée

b : représentation 3D

c : chemin de FM saturée

d : profil de vitesse

Figure IV-7: Etapes de FM² saturée.

IV.1.4. La recherche en ligne (ONPS)

La recherche en ligne est faite lorsqu'un obstacle inattendu est détecté. Dans ce cas-là, une carte de traitement de nouveaux obstacles dynamique (PNO) est construite pour l'endroit où se trouve cet obstacle. On utilise l'algorithme de recherche en ligne ONPS (Online Path Search) pour contourner cet obstacle, puis on poursuit le parcours à partir du nœud courant.

- ***La carte PNO (Processing New Obstacles)***

La carte PNO est construite pour représenter les zones sûres et celles dangereuses (contenant des obstacles inattendus) en se basant sur deux critères : la portée de perception et la marge de sécurité.

- **Portée de perception**

Les UAVs doivent récolter toutes les informations nécessaires relatives à l'environnement de leurs missions. Notons R la portée de détection (*detection range*) des capteurs embarqués. La portée perceptive d'un UAV est définie comme un cercle centré sur

l'UAV, et R est le rayon dans l'espace bidimensionnel. Cette portée peut prendre la forme suivante (formule 12) :

$$(xc - x)^2 + (yc - y)^2 = R \qquad (12)$$

Où (x, y) est un point arbitraire dans la gamme perceptuelle, et (x_c, y_c) est la position courante de l'UAV.

Si un UAV détecte des obstacles inconnus dans sa gamme de perception, il va les considérer comme inattendus et par conséquent va construire la carte PNO.

- **La marge de sécurité**

Lorsqu'un UAV trouve un obstacle inattendu dans son champ de perception, il prend une mesure d'évitement d'obstacle qui est le freinage d'urgence associé à sa vitesse. Nous appelons la distance de freinage *marge de sécurité* (notée par d_{sm}). Ensuite, la carte PNO pour ces obstacles imprévus et construite en utilisant la forme suivante :

$$I = \begin{cases} +\infty & dc \leq dsm \\ 0 & else \end{cases} \qquad (13)$$

Où dc représente la distance verticale entre l'UAV et l'obstacle inattendu et I est l'indice de sécurité (les points qui sont à une distance $d_c < d_{sm}$ auront un indice élevé).

- ***L'algorithme de recherche en ligne***

La recherche en ligne est réalisée uniquement si l'UAV rencontre un obstacle inattendu. Elle se fait à l'aide de l'algorithme ONPS qui lui-même utilise l'algorithme A*. La différence avec A* classique est dans le faite que ONPS exploite les connaissances fournies par l'OFPS pour guider la direction de la recherche au lieu d'adopter une distance étroite dans l'algorithme A*. De plus, un mécanisme de réduction de l'espace de recherche est exploité dans l'ONPS pour accélérer l'efficacité de la recherche.

ONPS utilise une fonction de coût notée $f(s)$ pour identifier l'ordre dans lequel la recherche traite des points dans un arbre. Elle est calculée comme suit :

$$f(s) = k(s) + h(s)$$

Où $k(s)$ est le coût du trajet du point de départ au point courant et $h(s)$ représente une heuristique du coût du trajet du point actuel au point cible. L'algorithme A* classique utilise le coût linéaire du point cible pour calculer $h(s)$, car il représente le coût le plus petit possible entre deux points. Dans le cas de l'ONPS, on attribue à $h(s)$ les coûts de nœuds générés par l'OFPS pour tous les autres points du graphe spatial en se déplaçant vers le point cible. Par

conséquent, si le chemin droit entre le point actuel et le point cible n'apparaît pas dans les zones de menaces dynamiques générées par des obstacles inattendus, alors le processus de recherche en ligne se termine. Lorsque le point cible est loin du point de départ, l'espace de recherche est réduit et l'efficacité d'exécution d'ONPS est améliorée. Le pseudo code de l'algorithme de recherche en ligne est le suivant :

Algorithme : AStar.

```
Input : START, GOAL, CostNodeMap, Extend_New_Obstacle
CLOSED = empty set
OPENED = priority queue containing START
Came_from = empty set
G [START] = 0
F [START] = START.T
While OPEN is not empty do
    current = pop lowest rank item from OPEN
    if current == GOAL or line from current to GOAL does not intersect with area
represented as Extend_New_Obstacle then
        return inversed came_from
    End if
    Add current to CLOSED
    For n in neighbors of current do
        if n is not an obstacle then
            Tentative_g = current.T + cost (current, n)
            If n in OPENED and tentative_g < G[n] then
                Remove n from OPENED
            Else if n in CLOSED and tentative_g < G[n] then
                Remove n from CLOSED
            Else if n not in OPENED and n not in CLOSED then
                G[n] = tentative_g
                F[n] = G[n] + n.T
                Add n to OPENED
                came_from[n] = current
            End if
        End if
    End for
End while
Return failure
```

Algorithme : ONPS.

```
Input : START, GOAL, CostNodeMap, Extend_New_Obstacle, Is_detected
Output : Current position
Path_online = []
current = START
while Extend_New_Obstacle != [] do
    if path_online ≠empty do
        current = pop the first element from Path_online
        Add current to global_path
        if is_detected do
            Path_online = AStar (current, GOAL, Extend_New_Obstacle)
```

```
40.     Else
41.         Path_online = AStar (current, GOAL, Extend_New_Obstacle)
42.     If Path_online is empty do
43.         Extend_New_Obstacle, is_detected = [], False
44.         Break
45.     Extend_New_Obstacle, is_detected = TheDetectedAreaOf (current)
46. Return the current position
```

L'exploitation des différents algorithmes proposés est résumé dans le pseudo code suivant :

Algorithme : ImpMOPP.

```
1.  Input : START, GOAL, binary map MAP, velocity of UAV VELOCITY, STARURATION,
    SECURITY_MARGIN
2.  map_velocity = MAPVelocity (VELOCITY, SATURATION, MAP)
3.  Cost_distribution_map = FM (GOAL, START)
4.  Plan_path = OFPS (START, GOAL, Cost_distribution_map)
5.  Global_path = queue containing START
6.  Current = START
7.  Is_replanning = False
8.  While current ≠GOAL do
9.      If Is_replanning ==False then
10.         Current = pop from Plan_path
11.     Else
12.         Is_replanning = False
13.         Plan_path = OFPS (current, GOAL, Cost_distribution_map)
14.         Continue
15.     End if
16.     Append current to Global_path
17.     Extend_New_Obstacle = empty set
18.     Is_detected = False
19.     Extend_New_Obstacle, Is_detected = TheDetectedAreaOf (current)
20.     If Is_detected then
21.         Current = ONPS (current, GOAL, Cost_distribution_map,  Extend_New_Obstacle)
22.         Is_replanning = True
23.     End if
24. End while
```

Dans ce qui suit, quelques figures illustrant une partie du code source de la méthode que nous avons implémentée :

MOPPFrame (JusteAvantDronekit) | MOPPFrame (sansinterface) | MOPPFrame (InterfacescriptMOPP)

```
913
914 def ImpMOPP(START, GOAL, MAP, sgmx, sgmy, VELOCITY, ds):
915     global actual_path,id1,extend_new_ob,is_detected
916     plan_path,replan_path,actual_path,init,prev,id1 =[],[],[],[],[-1,-1],-1
917     alpha = input("la valeur de la SATURATION  ]0, 1[ :")
918     tSim1 = time()
919     tFMM = computeVelocityMap(VELOCITY,alpha)#map des vitesses
920     tSim2 = time()-tSim1
921     init.append(GOAL.indice)
922     g = START.indice
923     tFMM1 = time()
924     distribution_cout = FMM(init,g,MAP)
925     tFMM2 =  time() - tFMM1
926     tGradientDescent1 = time()
927     plan_path = OFSA(START, GOAL, distribution_cout)
928     tGradientDescent2 = time() - tGradientDescent1
929     DessineDegrade()
930     dessiner(START, 'red')
931     dessiner(GOAL, 'red')
932     fenetre.update()
933     courant = plan_path[0]
934     plan_path.remove(courant)
935     prev = courant
936     is_replanning = False
937     ind = -1
938     ind = coord2indx([int(floor(courant[0])),int(floor(courant[1]))], ind)
939     while ind !=GOAL.indice:
940         if not is_replanning:
941             prev = courant
942             courant = plan_path[0]
943             n_courant = getNode(coord2indx([int(floor(courant[0])),int(floor(courant[1]))], ind))
944             plan_path.remove(courant)
945         else:
946             prev = courant
947             n_courant = getNode(coord2indx([int(floor(courant[0])),int(floor(courant[1]))], ind))
948             is_replanning = False
949             plan_path = OFSA(n_courant, GOAL,distribution_cout)
950         dessinerPath(prev, courant)
951         actual_path.append(courant)#ordonne au drone de partir au point courant
952         extend_new_ob,is_detected = DetectedAreaAndNewObstacles(n_courant,SENSING, SECURITY_DISTANCE)
953         if is_detected:
954             n_courant = getNode(coord2indx([int(floor(courant[0])),int(floor(courant[1]))], ind))
955             replan_path,courant = ONSA(n_courant, GOAL, MAP,extend_new_ob,is_detected)
956             is_replanning = True
957         ind = coord2indx([int(floor(courant[0])),int(floor(courant[1]))], ind)
958     print 'le temps de map velocity = ',tSim2
959     print 'le temps de FMM = ',tFMM2
960     print 'le temps Gradient Descent = ',tGradientDescent2
961     return True
```

Figure IV-8 : Code source de la main méthode ImpMOPP.

MOPPFrame (JusteAvantDronekit) | MOPPFrame (sansinterface) | *MOPPFrame (InterfacescriptMOPP)

```
860 def ONSA(sta, GOAL, COST,extended_new_ob,is_detected):
861
862     global extend_new_ob,actual_path
863     current = sta
864     path_OnLine = []
865     indexes = -1
866     longueur = -1
867     id2 = -1
868     prev = [-1,-1]
869
870
871     while extended_new_ob !=[]:
872         sleep(0.2)
873         if path_OnLine!=[]:
874                 grille.delete(id2)
875                 coords = path_OnLine[0]
876                 prev = indx2coord(current.indice, prev)
877                 dessinerPath(prev, coords)
878                 indexes = coord2indx(coords,indexes)
879                 current = getNode(indexes)
880                 path_OnLine.remove(coords)
881                 actual_path.append(coords)
882                 if is_detected:
883                     path_OnLine,longueur = AStar(current, GOAL, extended_new_ob)
884         else:
885                 path_OnLine,longueur = AStar(current, GOAL, extended_new_ob)
886         if longueur ==0:
887             extend_new_ob,is_detected = [],False
888             break
889
890         extend_new_ob,is_detected=DetectedAreaAndNewObstacles(current, SENSING, SECURITY_DISTANCE)
891
892     return [],actual_path[-1]
893
```

Figure IV-9 : Code source de la méthode ONSA.

MOPPFrame (JusteAvantDronekit) | MOPPFrame (sansinterface) | *MOPPFrame (InterfacescriptMOPP)

```
318 def FMM(init_points, goal, SIM):
319
320     global cwcdm,neighbors,n_neighs,minneighbors,Tvalues,n,sourceObs
321
322     NarrowBand =[]
323     n_neighs,j,n,itera,idxMin = 0,0,0,-1
324     neighbors = [0 for l in range(4)]
325     minneighbors = [0 for i in range(2)]
326     Tvalues = []
327     stopWaveProp = False
328
329     def getneighbors(idx,ne):
340
341     def getNeighborsInDim(idx,ne,i):
378     def getMinValueInDim(idx,dim):
391
392     def solveEikonalNDims(idx,dim):
412
413     def SolveEikonal(idx):
447
448     for i in init_points:
449         getNode(i).cost = 0
450         heappush(NarrowBand,(getNode(i).cost,getNode(i) ))
451     while NarrowBand != [] and not stopWaveProp:#the main loop of FMM
452         neighbors = [0,0,0,0]
453         n_neighs = 0
454         itera+=1
455         best = heappop(NarrowBand)
456         idxMin = best[1].indice
457         n_neighs = getneighbors(idxMin,neighbors)
458         getNode(idxMin).type ='A'
459         for s in range(n_neighs):
460             j = neighbors[s]
461             if  (getNode(j).type =='A'  or getNode(j).TAG==FORBIDDEN or getNode(j).TAG==NEW_FORBIDDEN):
462                 continue
463             else:
464                 cost =SolveEikonal(j)
465                 if NarrowBand.__contains__((getNode(j).cost,getNode(j))):
466                     if cost<getNode(j).cost:
467                         indexe = NarrowBand.index((getNode(j).cost,getNode(j)))
468                         getNode(j).cost = cost
469                         _siftdown(NarrowBand,0,indexe)
470                 else:
471                     getNode(j).cost = cost
472                     getNode(j).type = 'N'
473                     heappush(NarrowBand,(getNode(j).cost,getNode(j)))
474         if idxMin == goal:
475             stopWaveProp = True
476     return SIM
```

Figure IV-10 : Code source de la méthode FMM.

I.12. Conclusion

Dans ce chapitre, nous avons présenté la méthode que nous avons utilisée et qui nous a permis de planifier un chemin pour notre drone. Cette méthode est basée sur la recherche hors ligne et en ligne. Elle garantit deux objectifs essentiels qui sont la sécurité vis-à-vis des obstacles et la minimisation du temps de parcours. Nous avons également citée quelques méthodes utilisées dans la recherche en ligne, et qui sont inspirées de la méthode FMM.

Dans le chapitre qui suit, nous allons exposer d'une manière détaillée les différentes expérimentations appliquées à notre méthode MOOP dans l'environnement de simulation et dans un environnement réel en utilisant un drone quadri rotor. Enfin, nous allons tester et évaluer nos propositions, et commenter les résultats.

I.13. Introduction

Notre travail consiste à trouver un chemin qui permet à un drone de se déplacer d'un point initial à un autre final tout en évitant la collision avec des obstacles et aussi d'optimiser la fonction du coût exprimée en termes de la distance parcourue par le drone durant son déplacement. Pour cela, nous avons réussi à résoudre ce type de problème (problème de planification) en utilisant une méthode basée sur la recherche hors ligne et en ligne et qui vérifie deux objectifs essentiels qui sont la sécurité vis-à-vis des obstacles et le temps de parcourt, et que nous avons implémenté sur l'environnement de simulation et sur notre drone réel.

Dans les expérimentations suivantes nous évaluons les performances de notre méthode avec quatre indicateurs qui sont la vitesse de la trajectoire, la sécurité de la trajectoire, le temps d'exécution de A* qui comprend les durées minimales, maximales et moyennes, et le nombre d'exécution de A* en mode ONPS.

Dans ce chapitre, nous allons décrire dans un premier temps notre environnement de simulation et les différents outils utilisés. Dans un deuxième temps, nous allons présenter les résultats des expérimentations de notre méthode MOPP, que ce soit l'expérimentation synthétique, ou l'expérimentation sur le drone.

I.14. Préparation de l'environnement

Nous avons implémenté notre méthode ImpMOPP en utilisant le langage de programmation Python sur un seul ordinateur avec le système d'exploitation UBUNTU 16.04 LTS installé à l'intérieur, qui est une version supportée à long terme. Les principaux paramètres de configuration matérielle de cet ordinateur sont un processus Intel(R) Celeron(R) CPU à 1.90GHz et 4 GB de mémoire.

Nous avons conçu différentes expériences pour différents scénarios, à savoir la simulation synthétique sur l'interface graphique, et la simulation réaliste sur le drone de type quadri rotors.

Simulation réaliste sur le drone

Pour simuler le système drone nous avons installé différents outils :

- **Le drone :** Nous avons utilisé un drone simulé quadri rotor (forme de X), avec son programme de contrôle qui est un firmware quad frame Ardupilot[18] avec l'API

[18] **Ardupilot :** c'est une suite logicielle Autopilote de véhicule libre et sans pilote, développé par des amateurs pour contrôler des modèles réduits d'avion et rover. Pour plus de détails, voir ce site : www.ArduPilot.org.

DroneKit[19] qui est responsable de générer des messages de type MavLink[20], et ainsi, connecter notre script vers le drone simulé.

- **La station de contrôle :** Nous avons installé une station de contrôle qui est Mission Planner (version 1.3.56), la figure V-1 montre l'interface graphique de cette station.

Figure V-1: Interface de Mission Planner.

- **Protocole de communication :** Pour la communication entre la station, le drone et le DroneKit nous avons installé le progiciel MAVProxy qui permet de transporter les messages MavLink.

Le schéma représentatif de notre environnement simulé est donné sur la figure V-2.

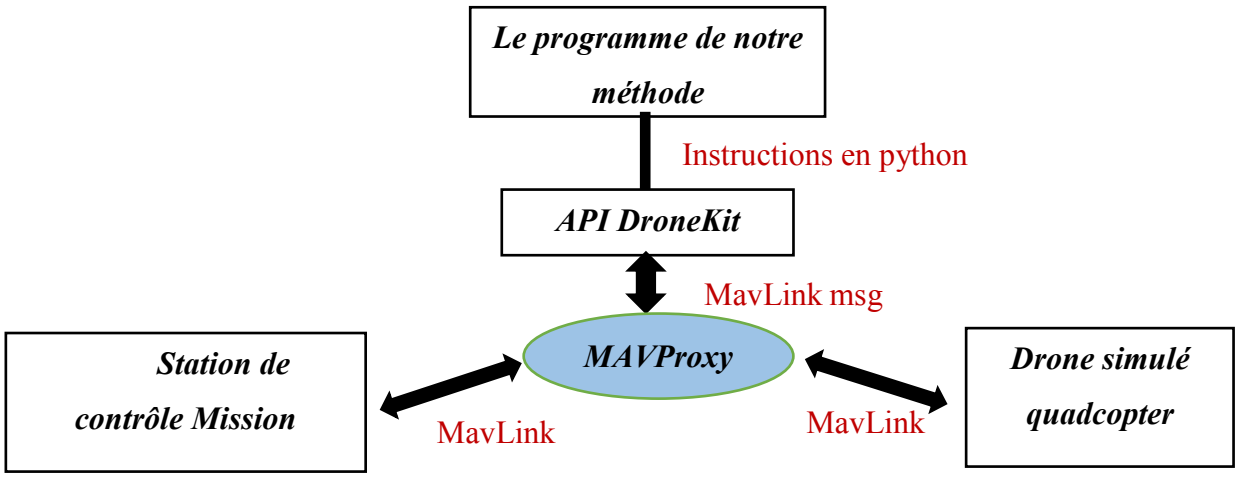


Figure V-2: Système drone.

[19] **DroneKit :** C'est une API qui permet aux développeurs de créer des applications, qui s'exécutent sur un ordinateur compagnon intégré sur le drone et communiquent avec le contrôleur du vol. Utile lors des calculs intenses.

[20] **MavLink :** C'est un protocole de communication destiné pour les véhicules aériens de taille réduite.

I.15. Expérimentation sur l'interface graphique

V.1.1. Carte avec obstacles statiques

Dans cette section, nous avons généré une carte de structure de plan (une map binaire) à l'échelle 200 treillis * 150 treillis, la résolution d'un treillis est 1 :1 mètre. La figure V-3 montre une carte avec trois obstacles statiques connus, les coordonnées de tous les sommets de chaque obstacle sur la carte ainsi que le point de départ de l'UAV et son point cible, sont indiquées dans le tableau V-1.

Obstacle connu 1	(0,40), (0,80), (60,80), (60,40), (55,40), (55,75), (5,75), (5,40)
Obstacle connu 2	(80,40), (80,80), (149,80), (149,40), (145,40), (145,75), (85,75), (85,40)
Obstacle connu 3	(30,60), (30,20), (115,20), (115,60), (110,60), (110,25), (35,25), (35,60)
Obstacle imprévu	(55,130), (55,135), (95,130), (95,135)
Point départ	(75,5)
Point cible	(85,185)

Tableau V-1 : Coordonnées d'obstacles, de point départ et

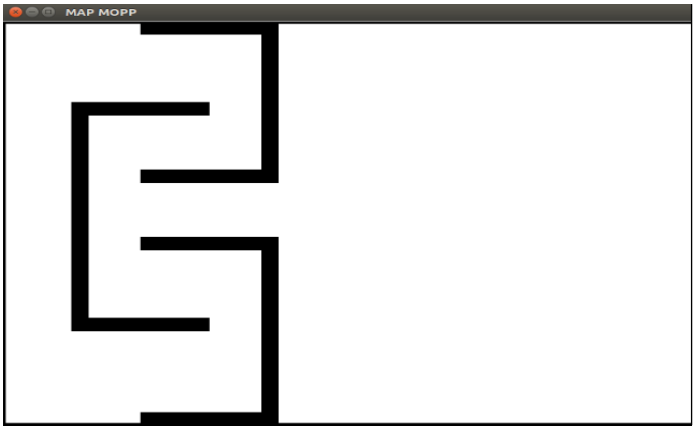
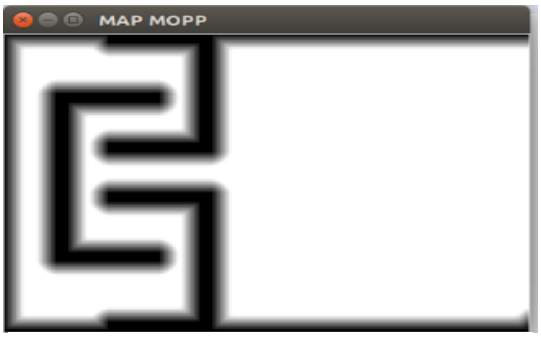


Figure V-3 : Carte graphique avec trois obstacles statiques.

Nous avons construit par la suite une carte de vitesses à partir de la carte d'interface illustrée sur la figure V-3, avec deux valeurs de saturations différentes 0.1 et 0.5 comme montre la figure V-4.

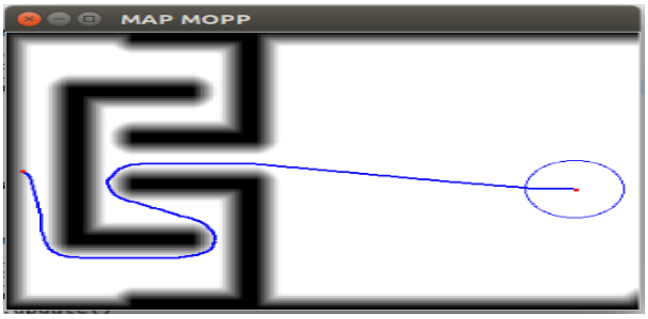


a : map de vitesses saturée à 0.1.

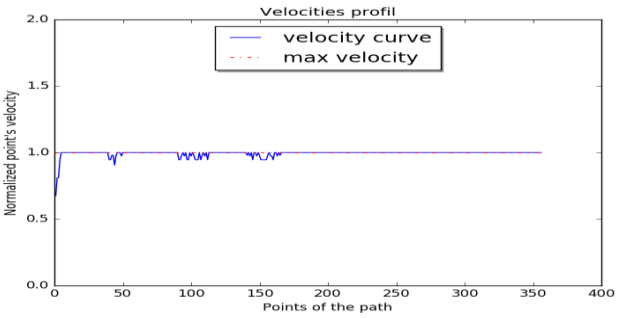
b : map de vitesses saturée à 0.5.

Figure V-4 : Carte de vitesses avec deux valeurs de saturations différentes.

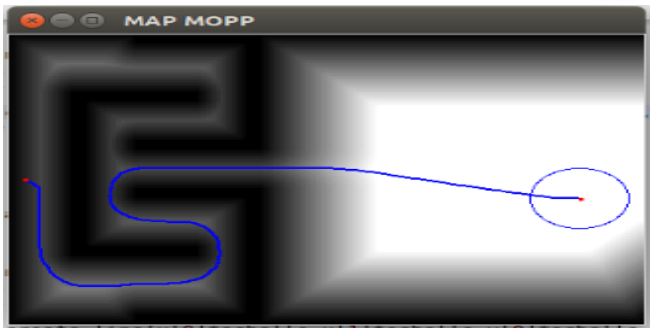
La figure V-5 donne des résultats de trajectoire de l'UAV suite à l'application de la méthode de recherche hors ligne FM² sur les cartes de la figure V-3, avec les vitesses du chemin obtenu dans les deux cas, où la vitesse de l'UAV est de 1m/s. La courbe bleue représente la trajectoire, le cercle bleu ciel c'est le rang de détection de l'UAV, et les deux points rouges sont les points de départ et d'arrivée.



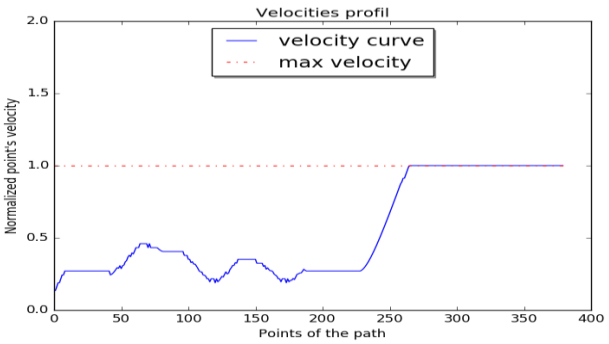
a : chemin obtenu avec saturation de 0.1.



b : profil de vitesse.



c : chemin obtenu avec saturation à 0.5.



d : profil de vitesse.

Figure V-5 : Résultats d'application de FM² sur la carte de vitesses saturée à 0.1 et 0.5.

À partir des figures V-5.a et V-5.b on peut constater que la vitesse des points de passage de l'UAV sont majoritairement à la valeur maximale mais la sécurité est faible puisque la trajectoire est proche des obstacles. En comparaison avec le résultat des figure V-5.c et V-5.d

on remarque que la vitesse dans la majorité des points de passage est inférieure à la valeur maximale mais le chemin est plus sécurisé.

Pour conclure, un temps de trajet minimal implique un risque élevé, cela est dû à la contradiction des deux objectifs et ainsi, nous pouvons dire qu'une solution avec un temps optimal et une sécurité optimale ne peut pas être obtenue.

V.1.2. Carte avec obstacles statiques et dynamiques

Dans cette section, nous avons vérifié que la méthode ImpMOPP peut contourner efficacement les obstacles inattendus, la configuration de cette expérience est la même que l'expérience précédente, sauf qu'un obstacle inattendu est impliqué (voir la figure V-6). Les coordonnées de tous les sommets de l'obstacle imprévu sur la carte sont indiquées dans le tableau V-1 précédent.

Figure V-6 : Carte de vitesses saturée à 0.5 avec un obstacle imprévu

Les figures V-7 et V-8 décrivent les résultats intermédiaires du parcourt du drone en évitant les obstacles connus et inattendus ainsi que les vitesses du chemin obtenu dans les deux cas. sur la figure V-7, le rang de détection qui est représenté par un cercle (bleu ciel) est de 15 treillis (15 m), et la marge de sécurité égale à 4 treillis (4 m). sur la figure V-8, le rang de détection est égal à 30 treillis (30 m), et la distance de sécurité égale à 4 treillis (4 m). Sur ces deux figures, la partie noire indique l'existence d'obstacles, notamment d'obstacles connus et imprévus détectés pendant le vol.

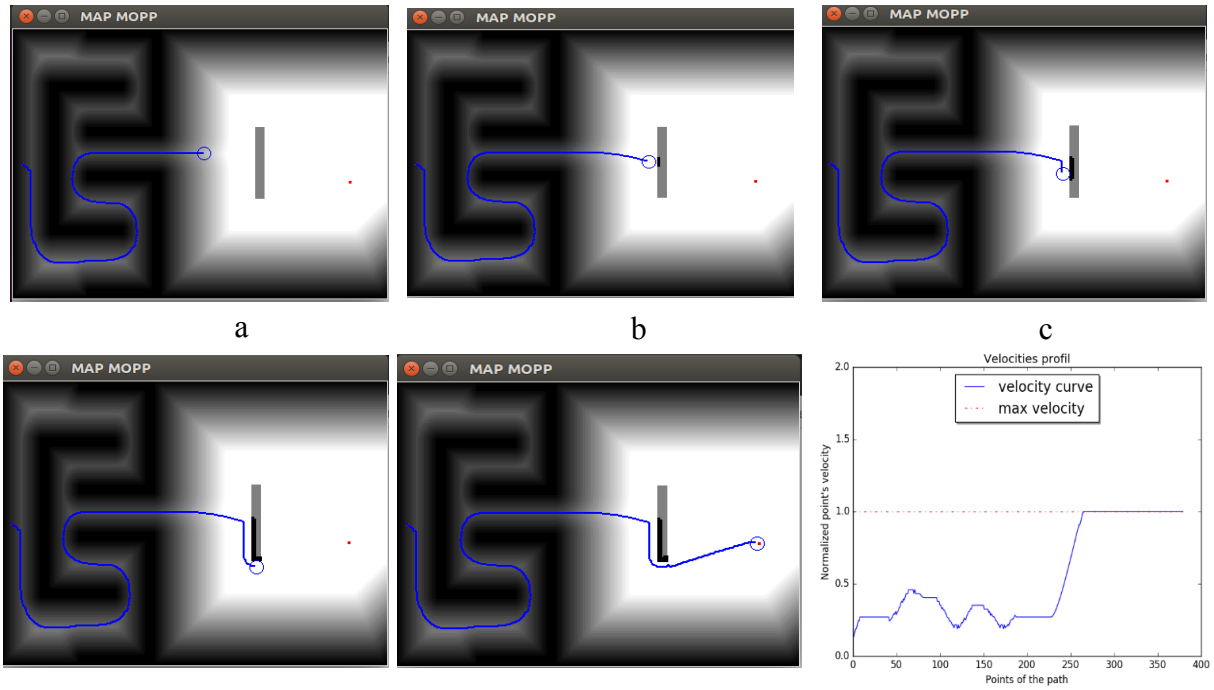


Figure V-7 : Processus d'exploration de chemin (Rang de détection 15).

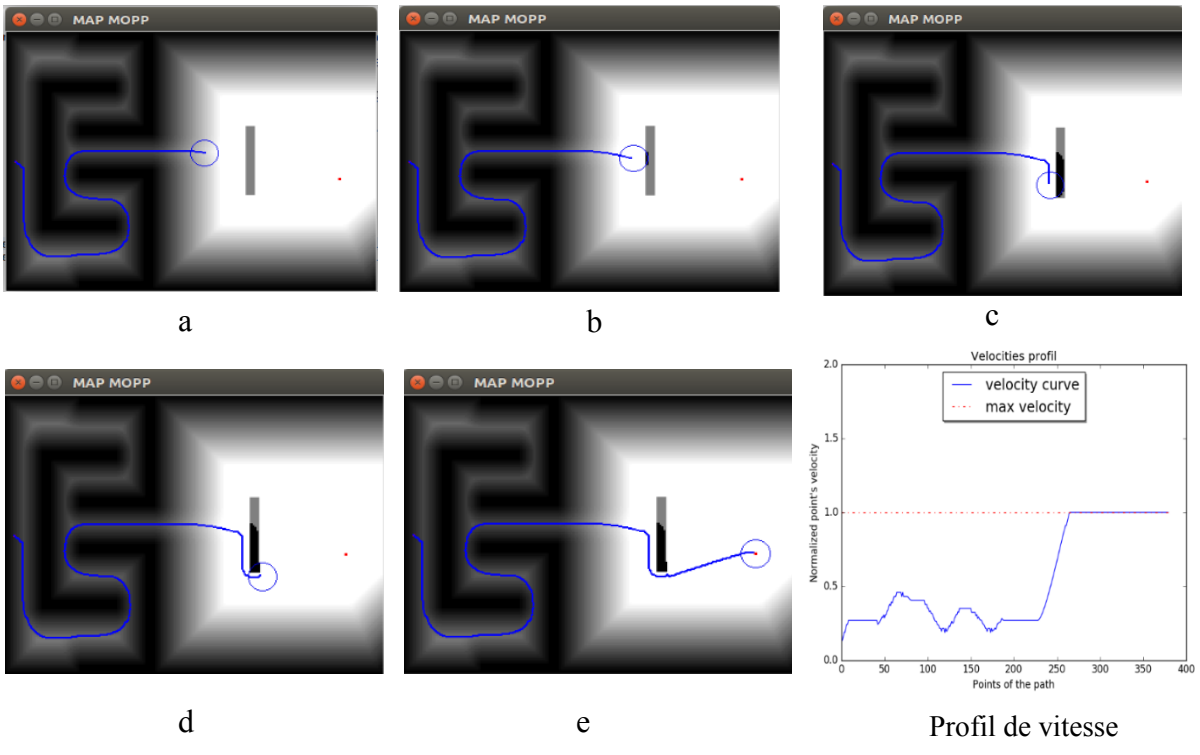


Figure V-8 : Processus d'exploration de chemin (Rang de détection 30).

Les résultats illustrés dans les figures V-7 et V-8 montrent bien que notre méthode contourne efficacement les obstacles non mentionnés sur la carte en garantissant une marge de sécurité par rapport à ces obstacles et la vitesse maximale.

Cependant, le nombre et le temps d'exécution de l'algorithme A* en mode ONPS diffère en dépends du rang de détection des capteurs intégrés sur le drone, le résumé dans le tableau V-2 montre une différence lorsque le rang de détection change. Pour un rang de 15 mètres le nombre de fois que ONPS a fait appel pour A* est plus grand que lorsque le rang égal à 30 mètres. En contrepartie, la durée moyenne d'exécution est faible lorsque le rang de détection est restreint, ce qui est logique vu que la surface détectée sur un rayon de 15 mètres est plus petite que la surface sur un rayon de 30 mètres. La durée maximale d'exécution de A* est inférieure à 30 millisecondes ce qui montre bien son efficacité et rapidité pour contourner un obstacle avec le mécanisme d'arrêt de processus de recherche.

Rang de détection	Distance de sécurité	Nombre d'appels de A^*	Meilleure durée d'exécution	Durée d'exécution maximale	Durée d'exécution moyenne
15 m	4 m	24	0.0012 s	0.0285 s	0.0147 s
30 m	4 m	18	0.0018 s	0.0288 s	0.0190 s

Tableau V-2 : Temps d'exécution de A par ONPS*

Devrons-nous choisir un rang de perception large ou restreint ? Cela dépend de la situation réelle vu que les capteurs avec une large perception peuvent coûter cher.

I.16. Expérimentation sur le drone

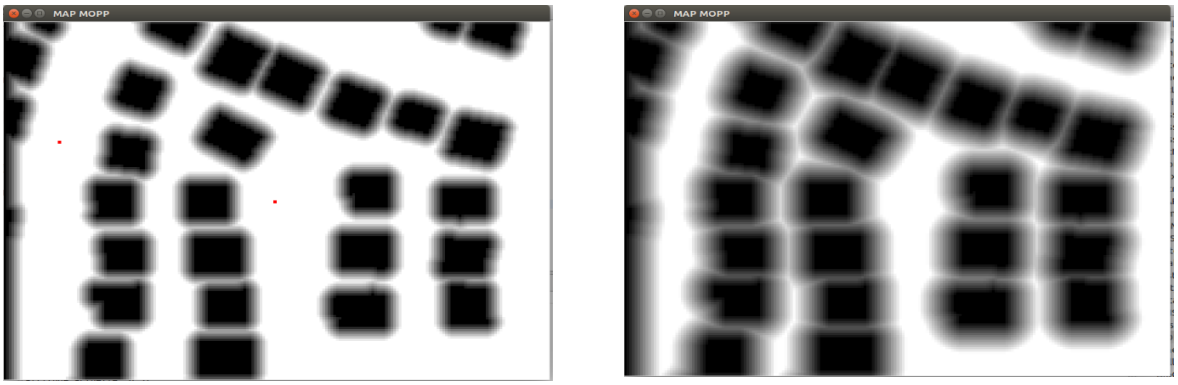
Dans cette section, nous avons récupéré une carte réelle de Google map à l'échelle 200*150 mètres, contenant des obstacles statiques (des bâtiments), ensuite nous avons extrait à partir de cette carte un modèle en utilisant Photoshop, dont tous les contours des bâtiments sont extraits et marqués avec des carrées noirs. Cette carte représente une région de la Californie comme illustré dans la figue V-9.

a : carte map obtenu par Google

b : modèle de la carte.

Figure V-9 : Carte map réelle avec des obstacles statiques.

Toutes les expériences faites dans cette section ne concernent que le cadre de la recherche hors ligne. Nous avons appliqué la méthode d'OFPS sur la carte avec deux valeurs de saturation différentes (voir figure V-10), l'une à 0.2, et l'autre à 0.4. Les coordonnées du point de départ de l'UAV est de (74,99), et le point final se situe à (50,20).



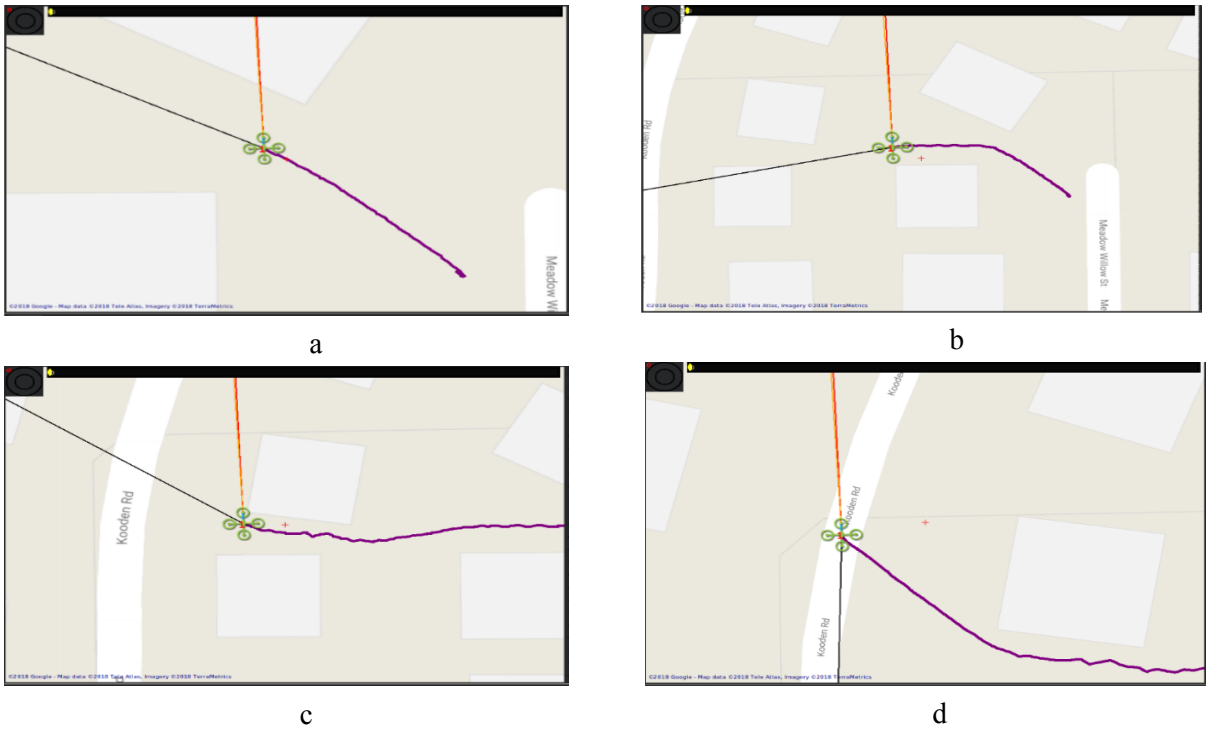
a : modèle de carte saturé à 0.2

b : modèle de carte saturé à 0.4

Figure V-10 : Modèle du map avec deux valeurs de saturation.

La figure V-11 représente des résultats intermédiaires de l'application de la méthode FM^2 sur le drone illustré dans la station de contrôle Mission Planner, le chemin et obtenu avec une

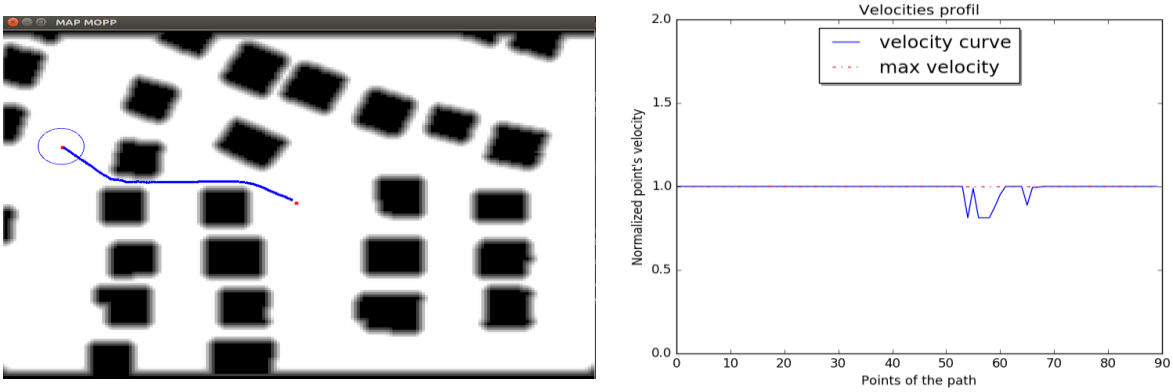
saturation de 0.2. Le résultat du chemin final avec la vitesse de parcourt sont montrés sur la figure V-12.



a b

c d

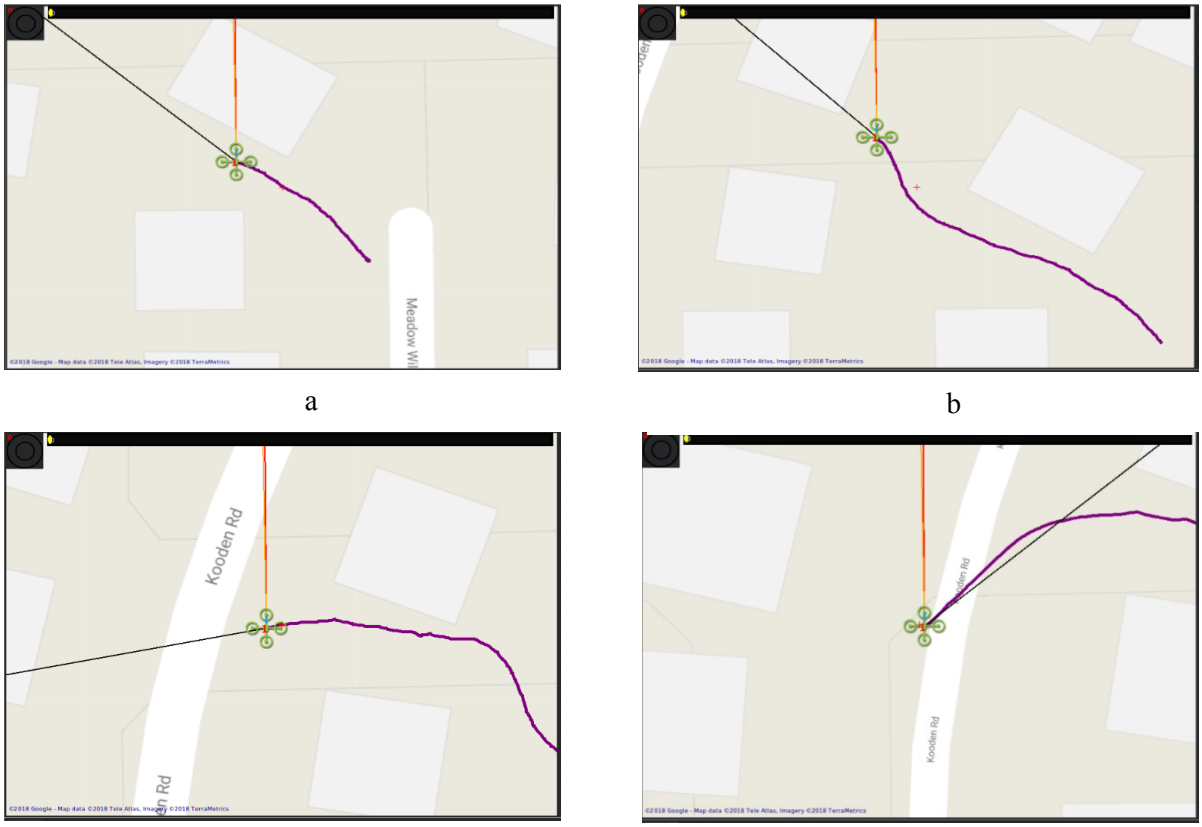
Figure V-11 : Visualisation du processus d'exploration de chemin sur Mission Planner



a : chemin final avec saturation de b : profil de vitesses

Figure V-12 : Résultats de chemin final (saturation de 0.2) avec le profil de vitesses.

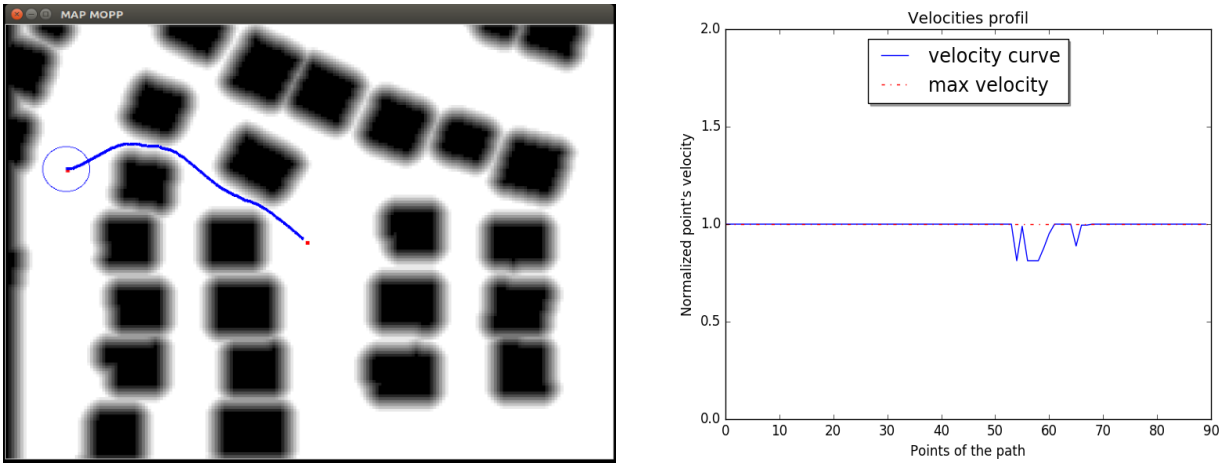
La figure V-13 représente des résultats intermédiaires de l'application de la méthode FM^2 sur le drone avec une saturation de 0.4. Le résultat du chemin final de l'UAV avec sa vitesse de parcourt sont montrés sur la figure V-14.



a b

c d

Figure V-13 : Processus d'exploration du chemin dans Mission Planner.



a : chemin final avec saturation de b : profil de vitesses

Figure V-14 : Résultat du chemin final (saturation de 0.4) avec le profil de vitesse.

A partir des figures V-12 et V-14 nous pouvant observer l'impact du facteur de la saturation qui résulte en deux chemins différents tel que, pour une saturation de 0.2 les points de chemin obtenus ont majoritairement la vitesse maximale par contre le trajet est proche des bâtiments ce qui augmente le niveau de risque. Pour la saturation de 0.4, les vitesses des point de trajet obtenues ont une vitesse inférieure à la vitesse maximale ce qui augmente le temps de parcourt, d'autre part, le chemin est plus sécurisé vu qu'il s'éloigne largement des bâtiments

I.17. Observations

En récapitulation de ces différentes expériences réalisées, nous pouvons donner les observations suivantes :

- La méthode ImpMOPP garantit un chemin optimal et trouve toujours une solution quel que soit le type d'obstacle.
- Le chemin optimal vis-à-vis du temps de parcourt et la sécurité est paramétrable selon les besoins et les préférences des utilisateurs.
- La méthode permet le contournement des obstacles inattendus en dépends du rang de détection des capteurs intégrés sur le drone.
- Les figures V-12 et V-14 illustrent le fait qu'un utilisateur préfère une sécurité élevée de chemin, la méthode génère un chemin éloigné des obstacles. Bien que la méthode peut aussi générer un chemin plus rapide avec un niveau de sécurité plus faible.
- En général, ces résultats prouvent que la méthode est utile dans les zones urbaines à basse altitude avec des obstacles statiques et imprévus.

I.18. Conclusion

Dans ce chapitre, nous avons présenté d'une façon simple et abstraite, les différentes étapes par lesquelles la méthode a été testée. Dans un premier temps, nous avons présenté l'environnement de la mise en épreuve de la méthode. Deuxièmement, nous avons présenté les résultats de l'application de la méthode ImpMOPP sur une interface graphique expliquant les différentes valeurs obtenues par rapport aux quatre indicateurs mentionnés précédemment. Finalement, les résultats de l'adaptation de la partie OFPS sur un drone sont donnés suivis par un ensemble d'observations concernant les caractéristiques de la méthode.

Les résultats obtenus sont satisfaisants, néanmoins des améliorations concernant l'optimisation de l'algorithme restent à ajouter (contournement des obstacles imprévus par le drone réel) afin d'obtenir un fonctionnement intégral de notre méthode dans la réalité.

Conclusion générale et prespectives

Dans ce travail réalisé nous avons décrit et implémenté une méthode améliorée destinée à la planification de trajectoires multi objective pour les véhicules aériens autonomes dans un environnement 2D complexe et dynamique. Cette méthode répond à deux objectifs qui sont le temps de parcours et le maintien du niveau de sécurité. Le processus de recherche d'une solution est basé sur les deux mécanismes de recherche, à savoir la recherche hors ligne (OFPS) qui utilise la méthode FM² et la recherche en ligne (ONPS) qui utilise l'algorithme A* pour contourner les obstacles imprévus. Pour parvenir à cette implémentation, nous avons introduit des généralités sur les drones (définition, composants, évolution, classification, capteurs et méthodes pour la navigation, domaines d'utilisation, réglementation) dans le premier chapitre. La notion de la planification de trajectoire a été abordée dans le second chapitre, dont quelques méthodes de planification dédiées aux robots ont été détaillées en montrant leurs avantages et limites. Dans le troisième chapitre, nous avons présenté des méthodes récentes, spécifiques aux UAVs, accompagnées des explications de leur principe, ainsi que leurs avantages et inconvénients. Le chapitre quatre est dédié à la présentation de notre méthode MOPP, dont le modèle de son système est développé, aussi, les différents algorithmes et leurs organisation ont été mentionnés. Le chapitre cinq résume les différentes expérimentations qui ont été réalisées, que ce soit dans un environnement simulé ou sur un UAV micro, de type quadri rotor, dont plusieurs scénarios ont été élaborés où l'environnement contient des obstacles statiques et dynamiques. Les résultats obtenus des différents tests et les observations sont donnés dans le même chapitre.

Les résultats des différentes expérimentations montrent que notre méthode ImpMOPP à la capacité de guider l'UAV vers sa destination en assurant un niveau de sécurité plus élevé et un temps de parcours plus optimal, comparativement à d'autres solutions (par exemple la version originale de MOPP). En effet, devant différentes situations, le drone réussit à naviguer correctement tout en évitant les obstacles statiques et dynamiques présents dans son environnement.

Les résultats obtenus de l'application de notre méthode MOPP dans l'environnement 2D ouvrent plusieurs perspectives pour d'autres possibilités d'amélioration, notamment, l'adaptation de cette méthode pour un environnement 3D. La puissance de la méthode FM² a amené les chercheurs à l'améliorer constamment. Ceci constitue une opportunité pour nous de faire évoluer la méthode de planification de trajectoires proposée, en optimisant les temps de calcul grâce aux nouvelles variantes de FM². La recherche en ligne basée sur A* reste limitée

dans le cas des obstacles non convexes, à cause des chemins non-optimaux qu'elle peut retourner. Une éventuelle amélioration de l'ONPS devrait contribuer à résoudre ce problème.

L'application des algorithmes de planification de trajectoires sur les UAVs est de nos jours en constante croissance, et leur utilisation se propage et se généralise dans différents domaines d'application dans le monde, que ça soit pour des besoins militaires que civils (la livraison des colliers, recherche et sauvetage, inspection des endroits dangereux,…). En Algérie, nous assistons actuellement à l'introduction de ces dispositifs. Ceci ouvre des champs d'application très intéressants pour l'implémentation à grande échelle des algorithmes que nous proposons. À titre d'exemple, l'implémentation de la méthode ImpMOPP pour les drones d'un hôpital menés de sacs destinés aux premiers soins, comme charge utile, améliorent le taux de réussite des missions de sauvetage. Un autre cas illustratif est la rescousse d'un patient en détresse (par exemple après une attaque cardiaque) se trouvant à une certaine distance de l'hôpital (avec conditions de circulation défavorables). L'utilisation du drone permet de joindre rapidement la personne en détresse et lui acheminer les équipements nécessaires (par exemple un défibrillateur accompagné d'un guide d'utilisation), en attendant une ambulance ; ceci augmentera considérablement la chance de sauver le patient en détresse.

Bibliographie

[1] M. M. Rida, «Observation et Commande de Drones Miniatures à voilures tournantes,» l'Université de Aboubekr Belkaid Tlemcen, 2015.

[2] C. Levaux, «De mesures. Une histoire du drone des 1960 à nos jours,» *Intervalles,* vol. 7, pp. 62-75, 2015.

[3] C. Antoine, D. Benjamin, H. Audouard, N. T. Nam et S. P. Libasse, «les Drones,» INSA, 2014.

[4] KALOUL Chabane, iderabdellah fateh et hamdad ghiles, «conception et réalisation d'un drone quadrirotor».

[5] D. Poinsot, «Commande d'un drone en vue de la conversion vol rapide - vol stationnaire,» Ecole nationale superieure de l'aeronautique et de l'espace, 2008.

[6] M. M. Rida, «Observation et Commande de Drones Miniatures à voilures tournantes,» Tlemcen, 2015.

[7] A. KOEHL, «Modélisation, Observation et Commande d'un Drone Miniature Birotor Coaxial,» 2012.

[8] M. M. Rida, «Observation et commande de Drones Miniatures à voilures tournantes,» 2015.

[9] A. KOEHL, «Modélisation, Observation et Commande d'un Drone Miniature Birotor Coaxial,» 2012.

[10] D. Adrien, «Stratégies de commande pour la navigation autonome d'un drone projectile miniature,» 2013.

[11] J. M. Pflimlin, «Commande d'un minidrone a hélice carénée : De la stabilisation dans le vent à la navigation autonome,» 2006.

[12] «la FAA publie sa nouvelle réglementation sur les vols de drones,» Newsroom, 23 06 2016.

[13] J. Ahuactzin, «Le fil d'Ariane : Une méthode de planification générale,» PhD Thesis, INP, Grenoble, 1994.

[14] B. Loubna, «Planification de trajectoire d'un robot basée sur les réseaux de neurones et les algorithmes génétiques,» université des sciences et de technologie d'Oran, 2010.

[15] B. Ismail, «Planification du trajectoire et évitement d'obstacle par lés réseaux de neurones pour les robots mobile autonome,» Université Abou Bakr Belkaid– Tlemcen, 2011.

[16] V. Delsart, «Navigation autonome en environnement dynamique :une approche par déformation de trajectoire,» Université de Grenoble, 2010.

[17] J. Borenstein et Y. Koren, «The vector field histogram-fast obstacle avoidance for mobile robots,» *IEEE transactions on robotics and automation,* vol. 7, pp. 278-288, 1991.

[18] I. R. Nourbakhsh and R. Siegwart, "Introduction to autonomous mobile robots," *The MIT Press,* vol. 262, no. 19502, pp. 142-150, 2004.

[19] R. Simmons, «The curvature-velocity method for local obstacle avoidance,» chez *Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on*, 1996.

[20] D. Foxa, . W. Burgard et . S. Thrun, «The dynamic window approach to collision avoidance,» *IEEE Robotics and Automation Magazine,* vol. 4, pp. 23-33, 1997.

[21] . M. Barbehenn et S. Hutchinson, «Efficient search and hierarchical motion planning by dynamically maintaining single-source shortest paths trees,» *IEEE transactions on robotics and automation,* vol. 11, n° 12, pp. 198-214, 1995.

[22] Z. Mengying , W. Hua et C. Feng, «Online path planning algorithms for unmanned air vehicle,» chez *Unmanned Systems (ICUS), 2017 IEEE International Conference on*, 2017.

[23] X. Liang, G. Meng, Y. Xu and H. Luo, "A geometrical path planning method for unmanned aerial vehicle in," *Intelligent Service Robotics,* pp. 1-12, 2018.

[24] . L. Lifen , S. Ruoxin , L. Shuandao et W. Jiang, «Path planning for UAVS based on improved artificial potential field method through changing the repulsive potential function,» chez *Guidance, Navigation and Control Conference (CGNCC), 2016 IEEE Chinese*, 2016.

[25] U. Cekmez, . M. Ozsiginan et O. K. Sahingoz, Multi Colony Ant Optimization for UAV Path Planning with Obstacle Avoidance, IEEE, Éd., 2016, pp. 47-52.

[26] J. Chen, Y. F. et T. Jiang, Path Planning under Obstacle-Avoidance Constraints Based on Ant Colony Optimization Algorithm, IEEE, Éd., Harbin Engineering University, 2017, pp. 1434-1438.

[27] M. d. S. Arantes, J. d. S. Arantes, C. F. M. Toledo et B. C. Williams, «A hybrid multi-population genetic algorithm for uav path planning,» chez *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, 2016.

[28] C. F. M. Toledo, M. França, R. Morabito et A. Kimms, «Multi-population genetic algorithm to solve the synchronized and integrated two-level lot sizing and scheduling problem,» *International Journal of Production Research,* vol. 47, n° 11, pp. 3097-3119, 2009.

[29] Y. Kuwata, "Real-time trajectory design for unmanned aerial vehicles using receding horizontal control," 2003.

[30] P. Yao and H. Wang, "Dynamic Adaptive Ant Lion Optimizer applied to route planning for unmanned aerial vehicle," *Soft Computing,* vol. 21, no. 18, pp. 5475-5488, 2017.

[31] C. Yin, Z. Xiao, X. Cao, X. Xi, P. Yang and D. Wu, "Offline and online search: Uav multiobjective path planning under dynamic urban environment," *IEEE Internet of Things Journal,* vol. 5, no. 2, pp. 546-558, 2018.

[32] O. S et S. J. A , «Fronts propagating with curvature dependent speed: Algorithms based on Hamilton-Jacobi formulations,» *J. Comput. Phys,* vol. 79, n° 11, pp. 12-49, 1988.

[33] A. Valero-Gomez , J. V. Gomez , . S. Garrido et L. Moreno, «the path to efficiency: Fast marching method for safer, more efficient mobile robot

trajectories,» *IEEE Robotics & Automation Magazine,* vol. 20, n° 14, pp. 111-120, 2013.

[34] S. Osher et J. A. Sethian , «Fronts propagating with curvature-dependent speed: algorithms based on Hamilton-Jacobi formulations,» *Journal of computational physics,* vol. 79, n° 11, pp. 12-49, 1988.

[35] G. G. Javier V, «Fast Marching Methods in path and motion planning: improvements and high-level applications,» UNIVERSIDAD CARLOS III DE MADRID, 2015.

Annexe

Description

Un quadri-rotor (ou quadcopter) est un drone qui appartient à la famille des voilures tournante, il est composé de quatre moteurs munis d'hélices, disposés aux quatre extrémités d'une croix en métal, ou autre matériau. Chaque moteur est relié à un contrôleur de vitesse, ces quatre étant pilotés par une carte de commande, qui reçoit ses ordres d'un récepteur de radiocommande ou d'un programme chargeable à l'intérieur.

A. Hardware

La figure 1 ci-dessous montre notre drone quadri rotor avec les différents composants le constituant.

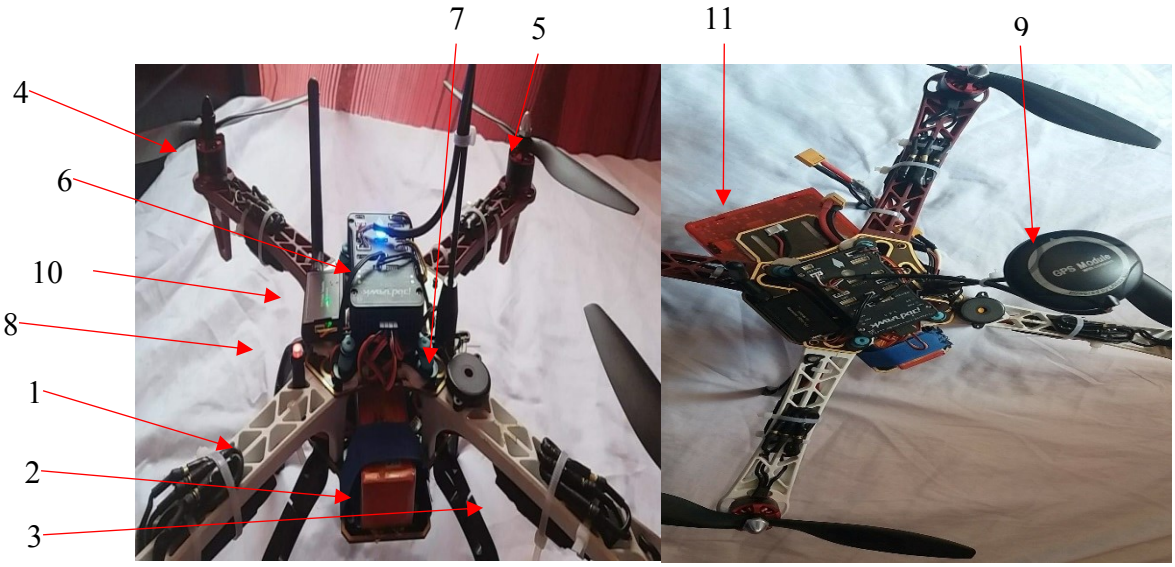


Figure 1 : Drone quadrirotor avec ses composants.

La description de chaque composant est comme suit :

1. Le châssis

C'est la structure du drone, c'est sur lui que l'on monte le reste des composants, de lui dépend le comportement général du drone et son autonomie. Des bras longs amélioreront la stabilité alors que des bras plus courts permettront des figures plus acrobatiques. On en trouve principalement à base d'aluminium ou de fibre de carbone, ou même de plastique comme dans notre cas, il existe des dizaines de formes différentes.

2. La batterie

C’est la source d’énergie du drone, dans notre cas nous avons utilisé une batterie en Lithium Polymère (LiPo), c’est une batterie avec une bonne capacité énergétique et un bon taux de décharge.

3. Les ESCs

Ce sont des petits boitiers qui vont permettre de contrôler électroniquement la vitesse des moteurs.

4. Les moteurs

Permettent au quadri-rotors de voler, il existe des centaines de modèles plus ou moins adaptés à l’utilisation qui va être faite, dans note cas, les moteurs choisis sont des moteurs Brushless 920 KV (destinés pour un vol stable).

5. Les hélices

Elles doivent être adaptées à la taille du drone, et aux moteurs choisis, de grandes hélices assure un vol plus stable et pouvant soulever plus de poids et de petites hélices permettent d’avoir un vol Nerveux plus rapide et nécessite un châssis plus léger, elles sont fabriquées généralement du plastique, du bois et de la fibre du carbone. Dans notre cas nous avant utilisé des hélices en plastique dur.

6. Contrôleur de vol

C’est le cœur du drone. C’est lui qui va stabiliser l’engin mais aussi effectuer différentes tâche plus ou moins complexes comme suivre un plan de vol, atterrir automatiquement, etc. il existe plusieurs cartes pour la contrôle du vol comme Cartes Raspberry, cartes Arduino, cartes STM et carte Pixhawk, dans notre cas nous avons utilisé une carte Pixhawk.

7. Buzzer

C’est un biper qui permet de générer les différents sons du système.

8. Safety switch

Un bouton de démarrage sécurisé

9. Module GPS

C’est un capteur qui permet au drone de se localiser géographiquement via le système GPS

10. Radio télémétrie

C'est la composante qui permet de communiquer avec la station de contrôle, elle contient un module aérien et un autre terrestre.

11. Raspberry pi 3 B

C'est l'ordinateur compagnon installé sur le drone, il permet d'exécuter notre programme de planification et le guider via les messages envoyés.

La figure 2 illustre comment connecter la Raspberry pi et le contrôleur de vol Pixhawk :

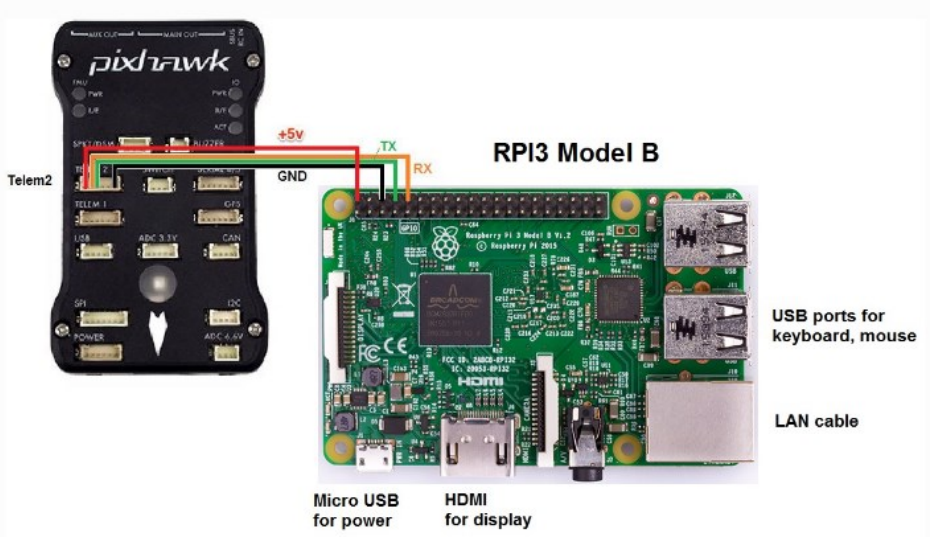


Figure 2 : connecter la RPi et la pixhawk

B. Software

Pour les différents systèmes et logiciels nécessaires au fonctionnement, nous résumons comme suit :

La station de contrôle

Nous avons installé Mission Planner qui est une station de contrôle développée par la communauté Ardupilot. Pour l'installation il suffit de télécharger le fichier zip sur leur site et le décompresser (http://firmware.ardupilot.org/Tools/MissionPlanner/)

Le système de bord sur le drone

Pour la carte de contrôle de vol (Pixhawk) nous avons installé le système Ardupilot dédié pour un quadcopter en forme de X en utilisant l'outil d'installation de Firmware sur la station de contrôle Mission Planner (voir la figure 3). Les étapes sont comme suit :

a. Connecter la carte Pixhawk à un pc via un câble USB où Mission Planner est installé

b. Sur l'écran on clique sur le bouton « Connect ».
c. Une fois la connexion établie, on choisit le menu « Initial Setup »
d. Sur le menu à gauche on choisit « Install Firmware »
e. On choisit l'image de notre drone « Frame Quad X »
f. Une fois le téléchargement est fini, on clique sur le bouton « Wizard » et on suit les étapes de calibrage.

Ordinateur compagnon

Pour la Raspberry pi 3 B utilisée comme ordinateur compagnon nous avons installé un système d'exploitation Raspbian et nous avons appliqué quelques configurations. Par la suite nous avons installé Python-v2.7, et enfin, l'API DroneKit qui va nous permettre de communiquer avec la carte de contrôle via des messages MavLink. Le lien suivant fourni plus de détails sur les étapes d'installation et configuration (http://ardupilot.org/dev/docs/raspberry-pi-via-mavlink.html).

Nous avons chargé le script de notre méthode sur la carte RPi 3 B, en utilisant l'API DroneKit pour pouvoir communiquer avec le contrôleur de vol (veuillez suivre les instructions en détail pour l'application avec DroneKit sur ce lien : http://python.dronekit.io/).