SURVEILLANCE DE ZONE PAR DRONES ET REACTION A UNE INTRUSION

Rapport de PSC

Groupe PSC INF15
Clement Bouvier
Louis Maisonneuve
Guillaume Monnet
Jean-Baptiste Soubaras

Tuteur : Vincent Jeauneau
MBDA

Date
Le 13 avril 2020

Table des matières

1 Introduction

Le 9 Juillet 2016, la police de Dallas tue un tireur embusqué à l'aide d'un drone explosif. En janvier 2019, une attaque de drones tue six officiers de l'armée loyaliste Yéménite. Le 14 septembre de la même année, la plus grosse attaque impliquant des drones ravage un site pétrolier d'Aramco en Arabie Saoudite et détruit près de la moitié des capacités de production de brut de la société pétrolière. C'est 5% des capacités de production mondiales de pétroles brut neutralisées. Ces événements posent un constat sans appel : un autre moyen de faire la guerre se développe. Reposant sur une technologie en plein essor, qui reste simple à maîtriser et peu coûteuse, les drones armés sont des OVNIs pour tous les systèmes de défense actuels.

Parfaitement adaptés pour les actes de terrorismes, d'espionnage ou pour les attaques officieuses de gouvernements malveillants, les drones miniatures menacent aussi bien les bases militaires avancées, les sites stratégiques militaires ou civils que les concerts publics ou les meetings politiques. Les drones miniatures ne sont pourtant pas encore développés comme de réels outils militaires. Leur emploi comme arme reste une utilisation détournée, leur force de frappe est donc aujourd'hui infime par rapport à ce qu'elle sera demain.

Associée à d'excellents capteurs optiques [1], des logiciels de reconnaissances facial déjà largement disponibles, et une petite charge explosive, une flotte de micro-drones peut cibler des personnes en particulier. Les technologies permettant d'éliminer chirurgicalement une classe de la population selon le sexe, l'âge ou les opinions exposées sur internet nous paraissent fantaisistes. Le seront-elles longtemps ?

Aucun système de défense anti-aérien déployé n'a été pensé du temps où l'utilisation de drones en grand nombre et en réseau coordonnés représentait une menace. Trop petits, ces robots ne sont pas détectés par les moyens de surveillance de manière fiable et le nombre d'assaillant sature rapidement les moyens de neutralisation. Évoluant proche du sol, les drones miniatures ouvrent la frontière de la basse altitude.

Pour pallier à cette menace, le gouvernement français n'exclue pas d'utiliser des rapaces spécialement entraînés à la lutte anti-drones. En attendant une solution moins rocambolesque évidemment...

Véritable course à l'armement, à la pointe des technologies en plein développement que sont la miniaturisation ou l'intelligence artificielle. Les systèmes de défense anti-drones miniatures nous interpellent car ils semblent être une pierre angulaire de la défense de demain. [2]

Ce sujet étant à la frontière entre informatique, mécanique et optimisation, trois domaines au cœur de nos études d'ingénieurs, c'est pourquoi nous nous proposons d'explorer plusieurs aspects d'un système

de défense anti-drones miniatures.

Nous avons décidé de nous concentrer sur une méthode utilisant des drones défenseurs. D'autre stratégies de base pourraient être envisagées mais celle-ci nous a parue intéressante et prometteuse par sa symétrie à la menace. Une telle réponse devrait donc pouvoir s'adapter aux différents schémas d'attaques.

2 Notre approche : chasser des drones avec des drones

2.1 Le cahier des charges d'une solution à cette menace

Les attaques de drones ne sont pas encore assez fréquentes pour pouvoir établir un retour sur expériences. Ainsi, pour établir un cahier des charges exhaustif, il faut imaginer comment les capacités des drones peuvent être utilisées pour mener une attaque. De plus, cette technologie n'en est encore qu'à ses balbutiements mais évolue extrêmement vite, il faut donc analyser en amont de quoi seront capables les concepteurs de drones de demain.

On retrouve toutes les tailles pour les drones modernes. Beaucoup d'entre eux sont suffisamment petits pour être extrêmement difficiles à repérer. Pourtant, ils n'en sont pas moins dangereux. Certains drones artisanaux fabriqués par l'État Islamique à la fin de la bataille de Mossoul portaient des charges d'un kg et ne mesuraient pas plus de 40 centimètres de diamètre. Ils ne sont visibles qu'à l'œil nu ou au bruit et souvent trop tard.

Ces drones n'étaient qu'artisanaux, imaginons le risque que représenterait un essaim de drones spécialement conçus pour combattre, possédant les technologies militaires pour ses batteries, ses paramètres de vol ou sa charge utile. Une menace silencieuse, rapide, et coordonnée, possédant sa propre intelligence qui les rend moins prévisibles et leur permettant d'anticiper la réponse d'un système connu avec plusieurs longueurs d'avance.

Officiellement, aucun système n'est actuellement en mesure de neutraliser un essaim de ce genre de drone.

Dans cette optique, nous avons délimité trois défis dans le cahier des charges d'un système de défense anti-drones.

Il doit d'abord repérer l'attaque, la forcer à se révéler. La difficulté réside dans la grande pluralité de formes que peut prendre l'attaque. On ne peut pas reconnaître un drone à sa vitesse, à sa couleur ou à sa taille car aucune d'entre elles ne caractérise l'aéronef. Il ne faut pas non plus confondre un essaim de bombardiers miniatures avec tout autre élément inoffensif comme une volée d'oiseaux. Le premier défi est donc celui de la surveillance.

Le système de défense doit ensuite être en mesure de contrer une attaque simultanée d'un grand nombre de drones. Imaginons dix drones attaquant à 60 Km/h, une zone protégée d'un km de rayon, la défense doit alors les neutraliser en moins d'une minute soit un drone toute les 6 secondes. C'est encore envisageable. En augmentant le nombre

d'attaquants à cent drones en revanche, la mission semble quasi-impossible. Le deuxième défi est celui de la proportionnalité de la réaction.

Enfin, les drones seront organisés en réseau de combat. Et dans un futur proche le réseau sera intelligent. Les drones pourront se coordonner pour attaquer. On peut imaginer des missions spécifiques pour chaque drone, certains mènent une attaque frontale de grande envergure pour faire diversion pendant que d'autres passent entre les mailles du filet. Certains attaquants peuvent avoir pour mission de neutraliser la défense amie pendant que d'autres attaquent l'objectif. Une infinité de scénarios sont possibles. Les comportements pourront aussi évoluer en fonction du déroulement de la bataille de manière totalement autonome. In fine, l'intelligence permettra aux drones d'adopter un comportement altruiste, calculant froidement de combien les chances d'accomplir la mission augmentent si certains attaquants se « sacrifient » ou jouent le lièvre. La stratégie de survivabilité des défenseurs devra résolument être évolutive. Le combat de demain ne se fera plus seulement entre des armes mais entre des réseaux adverses. Dans ce duel mêlant prise d'informations, traitement des données et matériel de première ligne de très haute technologie, le plus adaptable et le plus réactif l'emportera. Le troisième défi est donc celui de l'adaptabilité.

Ainsi, nous avons établi ce qui nous semble être les trois composantes principales d'une défense anti-drones. La capacité de repérer et caractériser une attaque, gérer un très grand nombre d'assaillants, et pouvoir analyser le plus grand nombre de données en restant réactif et adaptables. En bref, le système défensif doit être résilient à toutes formes d'agression. Tout cela dans un tempo plus qu'accéléré.

2.2 En quoi l'utilisation des drones est une solution envisageable

Nous l'avons vu, un système de défense capable d'arrêter une attaque de drone doit être pensé comme un réseau et non plus comme une somme de systèmes de défense isolés dont l'accumulation pourrait supposer une grande efficacité. S'il doit être composé d'un système de surveillance, d'un système capable de neutraliser les drones adverses, d'un commandement central capable de tous les coordonner et d'un moyen de communication sûr entre tous ces acteurs.

La surveillance par les moyens classiques (radar, caméra thermique) sont trop peu fiables. En revanche un logiciel d'apprentissage entraîné pourrait reconnaître un drone, comme les humains en sont capables. Ainsi, une caméra associée à un tel logiciel pourrait permettre de repérer une intrusion.

La mobilité que donne le drone à la caméra est un atout. Une zone surveillée par des caméras fixes ne l'est entièrement qu'avec un

grand nombre de caméras. Le fait que leur position puisse être connue et cartographiée est une brèche dans un système censé protéger contre des intrusions possiblement très discrètes. Les drones attaquants peuvent alors avoir un parcours défini par avance qui minimise les survols de zones surveillées et donc les risques de se faire repérer. De plus, il suffit de connaitre quelques caméras stratégiques à neutraliser pour rendre une partie de la zone complètement aveugle. Ces failles n'existent pas si ces dernières sont mobiles et interchangeables. Le système de défense possède alors le même effet de surprise que l'attaquant. Ce dernier est obligé de progresser sans savoir quelle zone est surveillée à un instant t, et si elle le sera à l'instant d'après. Aucune attaque ne peut être totalement prédéfinie en avance.

Contrairement aux caméra fixes, la neutralisation d'un drone ne met pas en péril la sécurité du site, le reste de la flotte pouvant se réorganiser en temps réel en cas de perte d'un aéronef, aucune zone ne cessera d'être surveillée.

Les drones inversent les stratégies de surveillance par caméras : on ne choisit plus quel pourcentage de la zone sera visible en permanence, laissant le reste dans l'ombre. Il faut déterminer quel pourcentage de temps chaque recoin de la zone devra être surveillé. Il n'y a plus de zones d'ombres mais des temps d'ombres en chaque point de l'espace. Il ne reste plus qu'à étudier combien de temps maximum un point peut rester non surveillé sans mettre en péril la zone et adapter en fonction la fréquence de passage des drones à cet endroit.

En plus de la rendre imprévisible, la défense par drones est aussi beaucoup plus adaptable. Elle permet de choisir en temps réel quel volume de défenseur est envoyé pour contrer une attaque. Il est possible de donner une réponse proportionnée à toute intrusion.

Cette réponse est rapide. Lorsqu'un drone détecte une intrusion, il est en mesure de traiter l'‘agresseur immédiatement car il possède à la fois la capacité de localiser la cible, de la verrouiller en la poursuivant et de mettre en œuvre son armement pour la neutraliser. Réunir les moyens de surveillance et les moyens de neutralisation au sein du même terminal permet de diminuer drastiquement les délais. Le drone peut momentanément devenir autonome et rendre compte à posteriori de la destruction ou non du drone qu'il a repéré. L'adaptabilité du réseau de défense repose aussi sur celle des drones, des armes aussi différentes qu'un brouilleur, un filet ou un explosif (dans le cas de petits drones défenseur kamikazes en appuis de la flotte de surveillance) peuvent être portées. Un panel de solution qui rend le système défensif encore plus adaptable.

Enfin, nous avons vu que dans un futur proche, les attaques par drones seront coordonnées avec intelligence. Contre cela, seul un système encore plus réactif, résilient et capable de traiter rapidement plus de données que l'agresseur a une chance de l'emporter. Seul un système

alliant un poste de commandement central capable de prendre en compte un flot important d'information puis de coordonner des exécutants mobiles et rapides comme des drones saurait supporter ce tempo.

Dans le cas où l'attaque a pour but de saturer le système de défense, l'utilisation de drones coordonnés permettrait de réduire le risque d'être submergé par la réorganisation des drones et la réallocation de missions en fonction de la priorité. Ainsi en définissant des zones à surveiller et protéger avec différents niveaux, la réallocation de ressource permettrait de protéger les zones les plus sensibles. Quitte à laisser l'intrusion sur une partie plutôt que voir le système de défense dépassé à tous les niveaux.

Dans ce genre de système, l'humain n'est plus nécessaire. Si l'ordinateur central est capable de reconnaître un grand nombre de pattern dans les méthodes des agresseurs, sa réponse sera plus rapide si elle est automatique.

2.3 Quels objectifs à notre échelle

La mise en place d'un système de défense anti-drone complet et fonctionnel fait intervenir énormément de savoir-faire dans des domaines distincts. L'entraînement du logiciel de reconnaissance visuelle de drones, l'élaboration des schémas de réponses adaptée à n'importe quel schéma d'attaque (notamment après l'avoir comparé aux patterns connus), les moyens de communication sécurisés et l'intégration de toutes ces technologies sur les drones en plus du moyen de neutralisation, sont autant de défis à relever avant de confier une zone à ce système.

Évidemment, l'aboutissement d'un tel outil nécessite un grand nombre d'ingénieurs spécialisés, de moyen matériels et financiers très important. Cela n'exclut pas que nous puissions effectuer notre travail de recherche sur certains aspects du sujet.

Nous avons considéré que les parties que nous n'avons pas le temps ou les moyens d'aborder dans le cadre de ce travail ont été étudiées.

Ainsi, nous considérons comme acquis le logiciel entrainé par deep learning capable de reconnaitre un drone sur une image. Que le drone lui-même a été construit, et que son système de défense nécessite de s'approcher du drone ennemi. Suffisamment proche, un système de brouillage ou de filet permet de neutraliser l'ennemi. En cas extrême, c'est-à-dire si le drone ennemi s'approche au-delà d'une ligne rouge, proche d'un centre stratégique à défendre « à tout prix », le drone possède une charge explosive et est autorisé à l'utiliser pour détruire le drone attaquant, même s'il est lui-même détruit. Bien-sûr, ce n'est pas l'objectif de la défense, tel drone étant supposé coûteux.

Il est aussi considéré comme acquis, la précision du matériel, c'est-

à-dire que le drone est capable de voler avec une incertitude très petite devant sa taille et que sa caméra a une résolution suffisante pour que les images puissent être exploitées par le logiciel de reconnaissance.

Dès lors, nous considérons qu'à partir du moment ou un drone intrus est repéré, nous connaissons sa position, sa vitesse et son accélération en temps réel. Une telle assertion est justifiée par le fait que les drones de surveillance connaissent à chaque instant leur position dans l'espace, leur vitesse et leur accélération, et qu'ils comparent la position du drone ennemis qu'ils ont en visuel par triangulation entre deux instants t et t+dt.

Nous considérons que les drones sont capables de communiquer entre eux et avec un poste de commandement centrale. Les algorithmes nécessitant plus de puissance de calcul que n'en possède le drone, sont parcourus par le poste de commandement. Pour simuler tout cela, nous avons construit nos algorithmes en fournissant toutes ses données à nos drones virtuels comme si elles avaient été récoltées par le système lui-même.

Ainsi, nous avons orienté notre travail sur les aspects de base de la surveillance, et de la poursuite d'un drone ennemie qui a été repéré.

Notre premier objectif est de formaliser et d'optimiser la manière dont les drones parcourent le terrain afin d'offrir la meilleure surveillance. Nous voulons ensuite étudier et comparer plusieurs solutions qui permettent, une fois l'intrusion repérée, de mettre en chasse le drone ennemi. Nous avons conduit une réflexion comportant deux volets : un aspect informatique, en fournissant quand cela est possible des algorithmes qui répondent aux problèmes, et lorsque cela devient hors de portée, nous avons privilégié une approche plus physique et logique de la solution qu'il faudrait coder.

3 Comment surveiller une zone avec des drones

3.1 L'objectif : minimiser les zones d'ombre sur toute la zone

Comme nous l'avons vu, surveiller une zone avec des caméras montées sur drones plutôt qu'avec des caméras fixes peut être intéressant. Cela nécessite néanmoins de répartir intelligemment les drones de surveillance sur zone.

Lorsque l'on parle de surveillance, on utilise les termes de "zones d'ombres" désignant les lieux non surveillés et de "zone éclairée" en référence à ceux qui le sont. Dans le cas de caméras fixes, une zone éclairée est blanche, une zone sombre est noire. Dans le cas de caméras mobiles, toutes les zones sont éclairées mais pas en permanence. Ainsi, une zone observée souvent sera très clair, et une zone rarement observée sera sombre.

Pour répondre au cahier des charges il faut coordonner les drones pour que la zone soit correctement nuancée. Les zones les plus stratégiques doivent être plus claires et les zones moins importantes peuvent être plus sombres.

Pour nuancer la carte, deux approches sont possibles. Pour la première, les drones sont affectés à une zone en particulier, plus la zone est stratégique, plus le nombre affecté est important. Pour l'autre, les drones circulent tous dans la totalité de l'espace de surveillance globale, mais passent plus régulièrement aux points stratégiques.

Pour comprendre comment optimiser la surveillance et quelle stratégie adopter, nous traitons un cas relativement simple.

Soit un espace de surveillance globale présentant trois zones : une zone stratégique A qui représente la majeure partie de l'espace globale, une zone peu importante à surveiller B et une dernière zone stratégique C excentrée.

On note S_A la surface de la zone A et S_C la surface de la zone C.

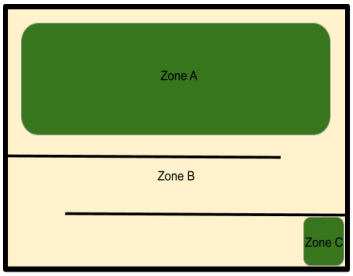


FIGURE 1 – Schéma de principe du cas étudié

Ce schéma modélise une situation théorique dans laquelle tous les points à surveiller n'ont pas la même valeur. Ici, deux zones éloignées l'une de l'autres sont plus stratégiques. Dans la réalité, les zones A et C pourraient être les parties les plus sensibles d'une plateforme pétrolière, les arsenaux et postes de commandement d'une base militaire aussi bien que les espaces publics d'un concert ou d'un meeting politique.

Si l'on partage équitablement les moyens de surveiller sur tout l'espace, ils seront mal utilisés car la zone B sera trop surveillée au détriment des deux autres.

La question à se poser est : est-il est plus avantageux d'attribuer un ou plusieurs drones à la zone C et avoir moins de drones pour la zone A ? Ou est-il est plus avantageux d'affecter tous les drones à la zone A avec des tours de gardes périodiques sur la zone C isolée ?

Posons le problème dans le cas où aucun drone n'a de zone attribuée. On suppose que chaque drone surveille 1m^2 autour de lui à la fois. Si l'on se place sur une grille de surface S et que le drone a une vitesse v suivant un parcours, il met un temps $\tau \propto \frac{S}{v}$ pour revenir à son point de départ.

L'inverse définit alors un taux de rafraîchissement f_{raf} mesurant la qualité de la surveillance. Plus elle est grande, moins il y a d'ombre dans la zone, plus la surveillance est bonne. On peut définir un seuil subjectif α, tel que si elle supérieure à α alors la surveillance est suffisante.

Pour notre cas : si n drones parcourent cette surface, on a alors le temps de rafraîchissement global $t_{raf} \propto \frac{S}{vn}$ et $f_{raf} \propto \frac{vn}{S}$.

Supposons maintenant que l'on envoie m drones surveiller la zone C isolée. Ces drones passent un temps t_1 dans la zone principale A, un temps t_3 dans la zone C isolée et prennent un temps t_2 pour aller d'une zone à l'autre en traversant la zone B tout en la surveillant. On a donc un temps de cycle complet $T = t_1 + 2t_2 + t_3$.

La surveillance dans la zone B étant moins importante, on fixe le temps t_2 que passent les drones dans cette zone. On suppose que ce laps de temps permet aux drones de réaliser une surveillance minimale suffisante pour cette zone.

On réalise alors dans chacune des zones A et C une moyenne arithmétique pondérée des taux de rafraîchissement durant chaque intervalle de temps t_1, t_2 et t_3. On obtient alors deux temps moyens de rafraîchissement :

$$T_A = \frac{t_1}{T}\frac{vn}{S_A} + 2\frac{t_2}{T}\frac{v(n-m)}{S_A} + \frac{t_3}{T}\frac{v(n-m)}{S_A} = \frac{v(n-m)}{S_A} + \frac{vmt_1}{TS_A}$$

et

$$T_C = \frac{t_3}{T}\frac{vm}{S_C}$$

Le but sera alors d'optimiser les taux T_A et T_C, c'est à dire de les maximiser conjointement, en fonction de t_1, t_3 et m afin d'arriver au meilleur compromis tout en respectant les contraintes $T_A > \alpha$ et $T_C > \alpha$. Pour cela on concilie donc une maximisation de T_A et une maximisation de T_C avec une minimisation de $|T_A - T_C|$. En effet les zones A et C ayant logiquement la même importance au niveau de la surveillance, leurs indices de surveillance respectifs doivent rester à peu près égaux.

On peut alors choisir de travailler avec deux fonctions de maximisation différentes :

$$\max_{t_1,t_3,m} T_A + T_C \quad \text{ou} \quad \max_{t_1,t_3,m} T_A \times T_C$$

Avec l'inégalité arithmético-géométrique

$$\frac{T_A + T_C}{2} \geq \sqrt{T_A T_C}$$

avec égalité si et seulement si $T_A = T_C$. On sait que la deuxième fonction de maximisation prend en compte la condition de minimisation de $|T_A - T_C|$. C'est donc avec celle-là que nous travaillons.

Cependant, on peut remarquer que l'on a alors un temps de perdu lorsque les drones sont dans la zone B. Sous l'hypothèse où celle-ci nous intéresse peu on peut alors diviser les n drones en deux sous-groupes : un groupe de $n - m$ drones qui sont affectés à la zone A et un groupe de m drones qui sont affectés à la zone C.

Dans chaque groupe on définit un sous-groupe qui sera chargé de surveiller la zone sans jamais la quitter et un groupe qui parfois sera chargé d'aller surveiller la zone B pendant un certain laps de temps. Afin d'obtenir le même taux de rafraîchissement moyen que dans le cas 1 (surveillance minimale que l'on accorde à la zone B), on prend proportionnellement $m(n-m)/n$ drones de la zone A et m^2/n drones de la zone C pour réaliser cette tâche. Sur un cycle T, ils passent un temps t dans la zone A (respectivement C) et un temps $2t_2$ dans la zone B ($T = t + 2t_2 = t_1 + 2t_2 + t_3$). D'où un taux de rafraîchissement moyen sur un cycle $T_B = (2t_2 vm)/(TS_B)$ égal dans les deux cas.

Pour les zones A et C, sur un cycle T on obtient :

$$T_A = \frac{v(n-m)}{S_A} - \frac{v(n-m)}{S_A}\frac{2t_2}{T}\frac{v(n-m)}{S_A}$$

et

$$T_C = \frac{vm}{S_C}\left(1 - \frac{m}{n}\frac{2t_2}{T}\right)$$

Or

$$\left(1 - \frac{m}{n}\frac{2t_2}{T}\right) = \frac{t_1 + t_3 + 2t_2(1 - m/n)}{T} > \frac{t_3}{T}$$

Donc $T_{C2ecas} > T_{C1ercas}$, la surveillance de la zone C est meilleure dans le deuxième cas que dans le premier cas. On gagne donc à séparer les drones en deux groupes

Pour la zone A, on a :

$$T_{A,1ercas} - T_{A,2ecas} = \frac{vm}{TS_A} t_1 + 2t_2 \left(1 - \frac{m}{n}\right) > 0$$

Donc $T_{A,1ercas} > T_{A,2ecas}$

A l'opposé, la surveillance de la zone A est meilleure dans le premier cas que dans le second. Avoir un système de surveillance centré en A nous permet d'avoir un meilleur indice de surveillance pour cette zone.

Ainsi chacun des cas a ses avantages. Nous allons donc regarder la différence entre les écarts afin de savoir si en passant d'un cas à un autre, on ne gagnerait pas plus qu'on ne perdrait. Comme $S_A > S_C$, on arrive à

$$T_{A,1ercas} - T_{A,2ecas} < T_{C,2ecas} - T_{C,1ercas}$$

Ainsi en passant du cas 1 au cas 2, l'augmentation de l'indice de surveillance de la zone C est plus importante que la réduction de l'indice de surveillance de la zone A. Lorsque que l'on est dans une configuration avec une zone isolée, il est donc plus intéressant de séparer notre flotte de drone en 2 groupes plutôt que d'avoir un seul groupe général en charge de toute la zone.

Dans le système de défense anti-drone que nous étudions, le comportement des drones changent lorsqu'ils détectent un intrus. De surveillants, ils deviennent des chasseurs. Nous nous demandons si la zone de surveillance et la zone de chasse doivent se superposer pour être optimaux. La double mission surveillance-chasse des drones exige de prendre en compte une attitude potentiellement problématique d'un suspect.

Un drone oscillant autour de la limite de la zone de surveillance peut être une stratégie évoluée d'attaque afin d'occuper un de nos drones sans prise de risque. Si on affecte comme mission à notre drone de détruire l'agresseur ou de le chasser hors de la zone, alors le drone défenseur va poursuivre l'assaillant jusqu'à ce qu'il sorte (on suppose qu'il n'a pas pu le détruire avant), puis repartir en patrouille. Et repartir en poursuite quand le drone assaillant pénètre de nouveau la zone et ainsi de suite. On définit alors une seconde zone plus large autour de la première, dans laquelle le drone peut continuer sa poursuite même en étant hors de la zone de surveillance.

Cette zone supplémentaire permet d'atteindre deux objectifs : laisser plus de temps au drone pour neutraliser l'agresseur et le repousser plus loin que simplement à l'entrée de la zone à protéger. On limite la taille de cette bande à $d = vt$ avec v la vitesse de pointe du drone, que l'on

suppose atteinte rapidement avec la maniabilité et la réactivité des drones, et t le temps nécessaire au drone pour revenir sur la zone principale en cas de mission prioritaire.

3.2 Une première solution : un parcours avec des check-points prédéfinis

Dès lors que nous avons notre zone à surveiller, notre nombre de drones attribués, il reste à choisir comment les répartir pour la surveillance. L'idée de base est de fixer certains points de passage obligés, des points importants à surveiller, puis de faire naviguer les drones entre ces points automatiquement grâce à un algorithme de parcours de graphe.

Nous avons développé une application graphique `cycle_uav.py`, en Python à l'aide d'OpenGL, qui permet de tester le fonctionnement des algorithmes de parcours de graphe en 2D sur une grille. Pour le choix du langage, même si ce n'est pas le plus optimal en terme de rapidité, il suffit amplement pour cette application, qui arrive à rafraîchir une boucle de chemins de taille raisonnable plus de 60 fois par seconde...

L'algorithme retenu est celui de recherche A-star qui, contrairement au BFS, va privilégier la direction du point à atteindre. Nous n'affectons pas de coûts particuliers à certains déplacements, soit les drones peuvent passer, soit un mur est présent. Comme fonction heuristique nous avons d'abord essayé la distance "Manhattan", somme des modules des coordonnées, puis utilisé une approximation de distance euclidienne : la distance entre deux cellules qui partagent un côté est fixée à 10, la distance entre deux cellules partageant un coin fixée à 14. L'algorithme préfère donc aller droit sur la cible si elle se trouve en diagonale.

Lors de son exécution, l'algorithme actualise tous les $1/60$ de seconde les chemins à emprunter pour atteindre les points voulus, en fonction des nouveaux murs présents sur la carte. Cela permettrait de traiter le cas de zones que les drones ne peuvent plus franchir lors d'une surveillance réelle, comme une porte qui se ferme, un mur qui s'effondre, un filet tendu etc.

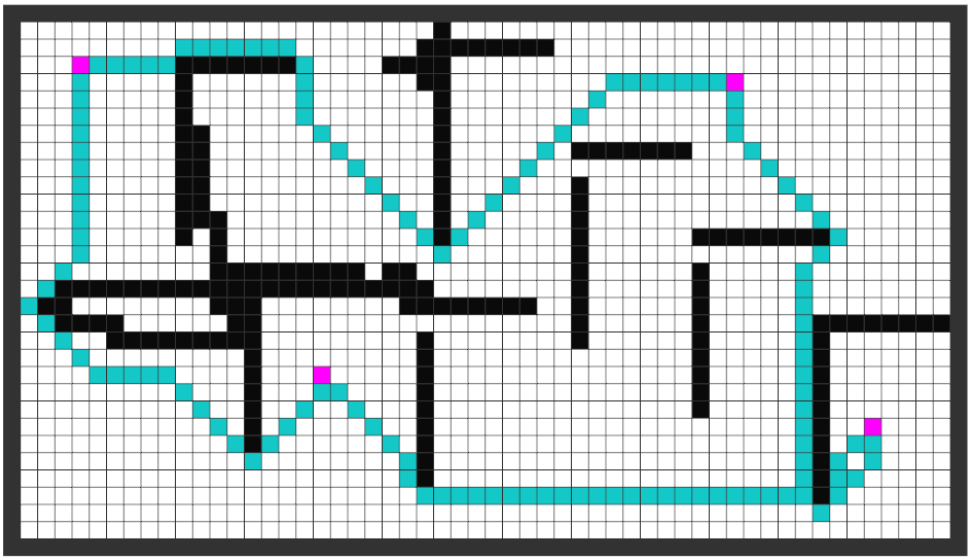


FIGURE 2 – Boucle (cyan) calculée entre les checkpoints (magenta)

On dessine des murs sur la grille et les chemins sont actualisés en quelques millisecondes pour une grille de cette taille

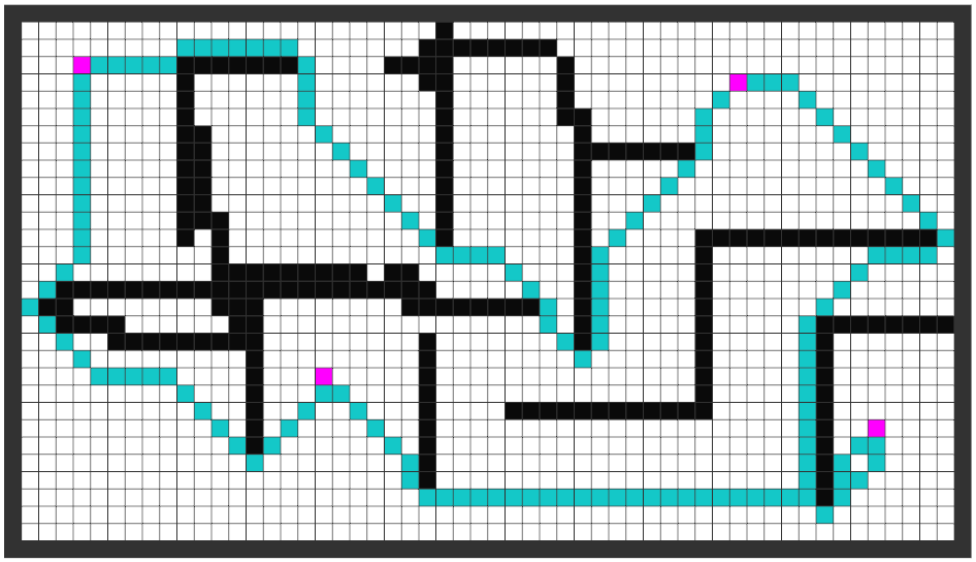
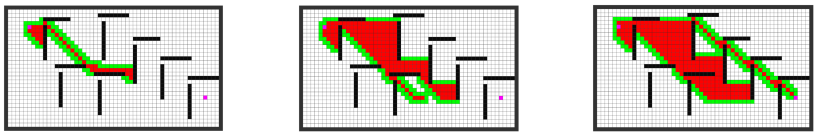


FIGURE 3 – Boucle recalculée quelques millisecondes après avoir modifié la carte

Ce script permet aussi étape par étape d'observer le fonctionnement des algorithmes de parcours de graphe sur la grille, pour déterminer un chemin. Dans le cas d'A*, à trois étapes différentes, entre deux checkpoints :

On part de la cellule magenta en haut à gauche, pour atteindre celle en bas à droite. En rouge : les cellules déjà visitées à l'instant représenté



(a) à t_1 (b) $t_2 > t_1$ (c) $t_3 > t_2$

FIGURE 4 – Astar à plusieurs étapes du parcours de grille

En Vert : la frontière, ou le `open_set`, dans lequel piocher la cellule la plus prometteuse à visiter à la prochaine itération.

On observe bien les caractéristiques de cet algorithme, et les raisons qui permettent de l'appliquer dans notre cas : il se dirige principalement vers le but, et prend du temps si le vrai chemin part à l'opposé. Comme les zones à surveiller seraient assez topologiquement éloignées d'un labyrinthe de ce type, et que le drone serait surtout à l'extérieur et libre de beaucoup de mouvements, A-star convient très bien.

On observe aussi en vert l'ensemble des cases parmi lequel chaque itération de la boucle va en choisir une pour aller la visiter. Celle choisie a le F cost ($G + H$, où G est la distance au départ, H la distance à l'arrivée) minimal, nécessitant donc l'utilisation d'une file de priorité, implémentée grâce à la librairie `heapq` en Python. Cette utilisation est raisonnable car la taille du heap reste suffisamment petite quand le chemin est accessible.

Nous pouvons donc relier les différents points importants de la zone à surveiller, pour ainsi faire patrouiller les drones en continu, en les faisant suivre le tracé de l'algorithme, qui se met à jour automatiquement.

3.3 Optimisations sur cette solution

Un élément à ajouter pour augmenter la durée de vie du système de surveillance en cas d'attaque est l'aléatoire : faire varier les chemins empruntés par les drones permet d'éviter à des attaquants de prévoir la surveillance et de la déjouer.

On peut ainsi avoir un jeu de checkpoints valides, dont certains sont tirés aléatoirement, avec une probabilité plus ou moins grande selon l'importance que l'on accorde à ce site. L'algorithme, qui de toute façon s'actualise plusieurs dizaines de fois par seconde en recalculant tous les chemins, n'a pas de mal à s'adapter à ces checkpoints variants.

Si la zone à surveiller devenait beaucoup plus grande et complexe (au vu des performances sur nos ordinateurs, il faudrait une échelle de plusieurs kilomètres, avec beaucoup de couloirs), l'algorithme pourrait ralentir trop pour tourner avec les performances voulues.

Plusieurs solutions sont envisageables : ne plus utiliser la grille comme graphe sur lequel faire la recherche, mais avoir un graphe des

points importants, par exemple les coins, sur lequel A* tourne. Le nombre de noeuds étant grandement réduit, la vitesse de calcul augmente drastiquement.

On peut aussi, plutôt que de lisser les trajectoires des drones à partir de celles pixellisées fournies par A* basé sur une grille, utiliser la variante Theta*, qui assigne comme nœud précédent le plus loin dans son champ de vision : les chemins empruntés par les drones seront plus directs, plus rapides, et la complexité en mémoire réduite.

Enfin, des chercheurs de l'université d'Alberta ont observé expérimentalement qu'un algorithme qu'ils nomment Block-A* serait beaucoup plus rapide que A* et Theta*, en parcourant la grille par blocs de cellules, tout en permettant des angles quelconques pour les trajectoires comme Theta*.[3]

Une chose à noter par rapport à notre approche est qu'on suppose avoir un ordinateur central qui dispose de la carte du terrain et transmet les chemins calculés aux drones. Cela serait assez compliqué à grande échelle en pratique, et nécessiterait peut-être de laisser les drones faire le calcul : ils disposeraient juste de coordonnées des points par lesquels ils doivent passer, et parcoureraient le monde en mettant à jour leur carte en interne, pour trouver le meilleur chemin. Il faudrait aussi qu'ils puissent communiquer entre eux leurs informations, rajoutant encore de la complexité à ce système distribué.

4 Quelle réaction en cas d'intrusion ?

Dans notre scénario de référence, si un drone intrus pénètre dans l'espace surveillé, un des drones du système de défense le voit et le prend en chasse. Dans cette partie, nous réfléchissons si ce scénario basique peut être optimisé dans certaines situations. Nous formalisons et comparons ensuite plusieurs algorithmes de poursuite en cherchant lequel serait le plus apte à répondre au cahier des charges.

4.1 L'objectif : minimiser le temps entre l'identification de l'ennemi et sa destruction

Nous l'avons vu, afin d'être efficace, un système de défense anti-drone doit être pensé comme un réseau connectant les moyens de destructions, de renseignements et de coordination. Il faut donc garder à l'esprit que c'est une flotte de drone coordonnée qui surveille l'espace et non une accumulation de drones solitaires. Comment mettre à profit cette coopération possible, une fois la chasse d'un assaillant commencée.

Un cas particulier peut se révéler intéressante. Lors de son vol dans la zone à protéger, le drone assaillant est poursuivi par un drone ami qui le rattrape au fur et à mesure. Il peut cependant être pendant un instant plus proche d'un drone encore en surveillance que de celui qui le chasse. Dans le but de neutraliser l'assaillant le plus vite possible, nous nous sommes alors demandé : ne serait-il pas plus intéressant, dans cette situation, de permuter les missions des drones. Le second succède au premier dans la chasse et lui laisse sa mission de surveillance. Dans quelle mesure cela permet-il de gagner du terrain sur l'assaillant ?

Intuitivement, si le drone 2 est plus proche de l'assaillant alors il prend le relais sur la poursuite. Néanmoins cette intuition n'est pas complète, il faut également prendre en compte l'inertie et les capacités d'accélération de nos drones.

Ainsi le drone 2 prend le relais sur la poursuite pas seulement s'il est plus proche à l'instant t mais si on estime que dans le futur, une fois sa vitesse de croisière pour la poursuite atteinte et son vecteur vitesse aligné avec la position de la cible, il se trouve plus proche de la cible que le drone d'origine. Le drone 2 peut être plus proche à un moment, mais peut être qu'une fois sortie de sa surveillance et ayant par exemple fait demi-tour puis accéléré pour commencer la poursuite, il se serait retrouvé plus loin que le drone d'origine qui lui est déjà lancé à sa vitesse de croisière. Ce raisonnement est notamment pertinent si, en accord avec la réalité, on suppose que les drones patrouillent à une vitesse v_3 lente, que le drone assaillant arrive à sa pleine vitesse v_2 et qu'en poursuite nos drones vont à une vitesse v_1 supérieur à v_2. L'enclenchement de la poursuite coûte donc une certaine inertie à nos drones. Le but est alors d'essayer de trouver des conditions sur la position et la vitesse des deux

drones afin de déterminer si une opportunité existe ou non. Pour cela nous avons fait quelques hypothèses afin de simplifier le modèle et d'arriver à des premiers résultats. Compte tenu l'agilité d'un drone et le rayon de braquage quasi nul de certains drones comme les quadcopters, nous avons supposé que les drones avaient la capacité d'accélérer depuis l'arrêt jusqu'à une vitesse maximum en une unité de temps.

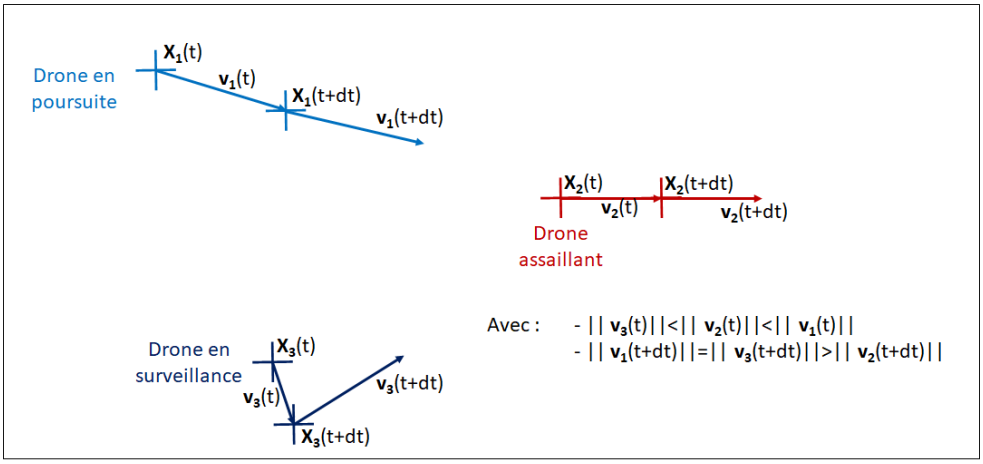


FIGURE 5 – Schéma illustrant l'inertie des drones

Ainsi en anticipant le futur, en une unité de temps le drone en surveillance aurait le temps de ralentir pour braquer puis d'accélérer à sa vitesse de croisière dans la bonne direction à l'instant suivant. On obtient donc les positions futures équivalentes entre les deux drones, et leur comparaison nous permettent d'établir la condition de prise de relais sur la poursuite :

$$\|\boldsymbol{X}_3(t+dt) - \boldsymbol{X}_2(t+dt)\| < \|\boldsymbol{X}_1(t+dt) - \boldsymbol{X}_2(t+dt)\|$$

D'où en trois dimensions :

$$(x_3 + v_{x3} - x_2 - v_{x2})^2 + (y_3 + v_{y3} - y_2 - v_{y2})^2 + (z_3 + v_{z3} - z_2 - v_{z2})^2 \\ < (x_1 + v_{x1} - x_2 - v_{x2})^2 + (y_1 + v_{y1} - y_2 - v_{y2})^2 + (z_1 + v_{z1} - z_2 - v_{z2})^2$$

Néanmoins, nous pouvons voir certaine limite à cette réflexion. Tout d'abord, nous avons obtenu ces résultats en connaissant la trajectoire du drone assaillant alors qu'en réalité nous aurons une incertitude sur cette dernière. Deuxièmement, l'échange de fonction entre les drones de surveillance et de poursuite coûte du temps d'exécution à l'unité centrale : réaffectation des algorithmes de chaque mission aux deux drones, recalcule des chemins de surveillance avec la nouvelle position du drone anciennement en poursuite. Ces calculs entraînent une perte d'efficacité pendant un lapse de temps suivant l'échange de rôle. Lapse de temps qu'il peut ne pas être utile de provoquer si $\|\boldsymbol{X}_3(t+dt) - \boldsymbol{X}_2(t+dt)\|$

est légèrement inférieur à $\|\boldsymbol{X}_1(t+dt) - \boldsymbol{X}_2(t+dt)\|$. C'est pour ces deux raisons que dans l'inégalité nous ajoutons un facteur ϵ afin d'augmenter artificiellement la distance entre le drone de surveillance et le drone assaillant, ce qui corrige ces erreurs.

$$\|\boldsymbol{X}_3(t+dt) - \boldsymbol{X}_2(t+dt)\|\,(1+\epsilon) < \|\boldsymbol{X}_1(t+dt) - \boldsymbol{X}_2(t+dt)\|$$

4.2 une solution simple, minimisation de la distance à un instant t

L'algorithme de poursuite le plus simple à mettre en œuvre consiste en chaque instant pour le chasseur à viser les coordonnées de l'assaillant et s'y diriger à vitesse maximum.

Cet algorithme a l'avantage d'être simple à implémenter. Cependant, il n'optimise pas le trajet du drone chasseur pour rattraper l'assaillant. Viser à chaque instant la position du drone à rattraper, c'est avoir un temps de retard sur ce dernier.

On a donc un temps d'interception très long.

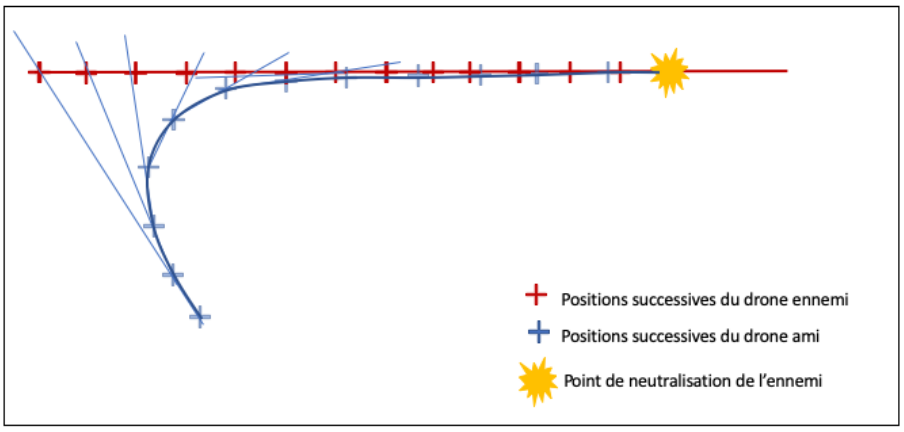


FIGURE 6 – Allure de trajectoires d'un drone ami (en bleu) poursuivant avec cet algorithme un drone ennemi ayant un mouvement rectiligne uniforme (en rouge) de vitesse du même ordre

On remarque que cet algorithme a tendance à privilégier une approche par l'arrière du drone ennemi. Dans le cas probable ou les drones assaillants et défenseurs ont des vitesses sensiblement égales, l'assaillant n'est alors rattrapé qu'au bout d'un temps long et la défense n'est pas assez réactive. Dans le cas ou le rouge est plus rapide que le bleu, ce dernier ne rattrapera jamais son objectif.

Si nos drones doivent être les plus rapides, une solution pourrait être de changer légèrement le dispositif en apportant un soutient avec de petits drones extrêmements rapides. Ces derniers n'ayant pour mission que d'intercepter la cible, avec les données que lui fournissent les observateurs, ils sont plus légers que les autres et peuvent être uniquement étudiés pour être rapides et maniables.

Observons la réaction d'un tel drone utilisant cet algorithme à la poursuite d'un agresseur.

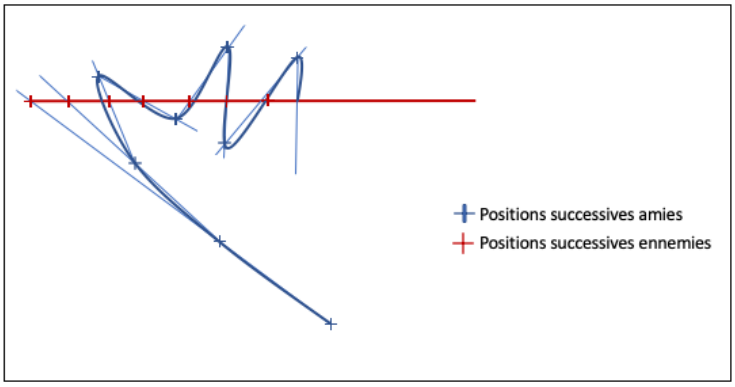


FIGURE 7 – Trajectoire d'un drone ami (en beu) poursuivant un drone ennemi (en rouge) trois fois moins rapide animé d'un mouvement rectiligne uniforme

Sur cette figure, les lignes droites sont les trajectoires sur lesquelles le drone bleu s'aligne à chaque itération de l'algorithme. Chaque droite relie la position du drone et la position de l'assaillant à un même instant. La courbe finale est obtenue par lissage de la trajectoire passant par tous les points dessinés. On observe que si le défenseur conserve une vitesse trop importante, il risque d'osciller d'un coté et de l'autre de la trajectoire du drone ennemi, le drone étant ici considéré comme ponctuel, il n'atteint jamais sa cible.

En réalité, la distance entre deux positions successives est de l'ordre du mètre (pour un drone ami ayant une vitesse de 100km/h, et un temps de rafraîchissement d'un dixième de seconde, la distance entre deux points bleus représentés est de 2,7 m). Pour cet algorithme et dans ces conditions, le drone ami est capable de se positionner très rapidement au voisinage du drone ennemi, a environ trois mètres de distance. Mais il met ensuite beaucoup plus de temps à s'en rapprocher d'avantage. Cet algorithme est inutilisable si le drone bleu ou le drone rouge sont trop petits et si la

technique de neutralisation utilisé par notre système nécessite une grande proximité (le cas de petits drones kamikazes par exemple).

Si les défenseurs sont trop rapides, une réduction de leur vitesse en fin d'interception leur ferait adopter un comportement semblable à celui exposé figure 6, cela permettrait d'éviter le phénomène d'oscillation et tendrait à les aligner sur la trajectoire de l'assaillant. Une fois alignés, les défenseurs pourraient accélérer de nouveau pour les rattraper.

Bien qu'envisageable, en théorie, pour les cas où l'assaillant garde un mouvement rectiligne uniforme, cette solution n'est pas fiable si la trajectoire de ce dernier varie. Il faudrait associer à chaque variation de direction de l'assaillant une phase d'oscillation puis de décélération et une dernière d'accélération. Loin d'être optimale en cas de succès, un tel comportement peut être totalement inefficace si les variations de direction de l'assaillant sont trop rapides.

Sans aucun doute, cet algorithme ne répond pas au cahier des charges. Adapter la vitesse de nos drones, c'est ne pas utiliser leurs capacités au maximum. Cet algorithme est trop peu fiable avec un taux de succès trop faible, trop peu réactif et trop lent dans les quelques cas ou la mission est remplie.

4.3 Une solution plus évoluée, prendre un temps d'avance

Dans l'algorithme précédent, nous avions toujours ce temps de retard car trop peu de données étaient utilisées. La poursuite s'appuie seulement sur la position relative des deux drones à un instant donné. Pourtant, nous avons considéré dans notre travail que la vitesse du drone ennemi est aussi utilisable. Ainsi, Pour rattraper le retard, nous introduisons dans notre algorithme la prise en compte plus seulement de la position de l'ennemi mais aussi de sa vitesse. Ainsi, notre drone chasseur ne va plus viser la position de l'ennemi à l'instant t, mais le point d'interception si l'ennemi maintient un mouvement rectiligne uniforme. Ce calcul est réitéré à chaque lecture de la boucle de l'algorithme.

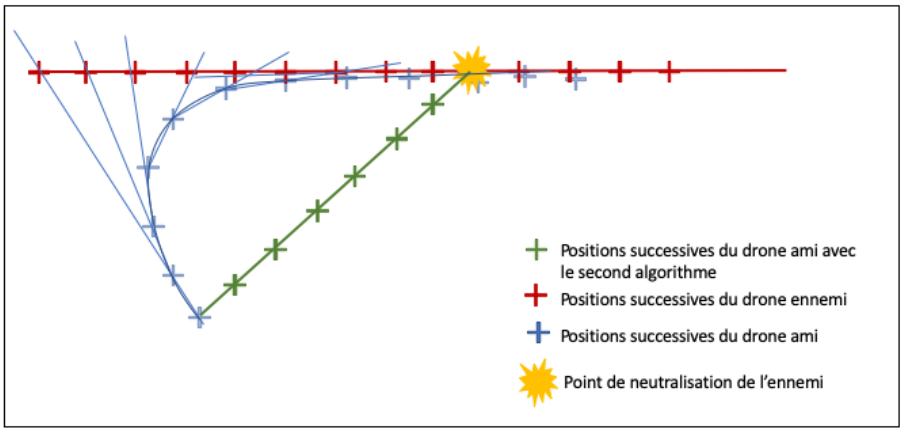


FIGURE 8 – Comparaison des trajectoires d'un drone ami selon qu'il est programmé avec l'algorithme 1 ou l'algorithme 2, dans les mêmes conditions que la figure 6

Cette figure montre que ce deuxième algorithme est beaucoup plus performant que le précédent dans ce cas particulier. La trajectoire est la plus courte possible, le système est réactif et rapide.

Observons son comportement dans d'autres situations.

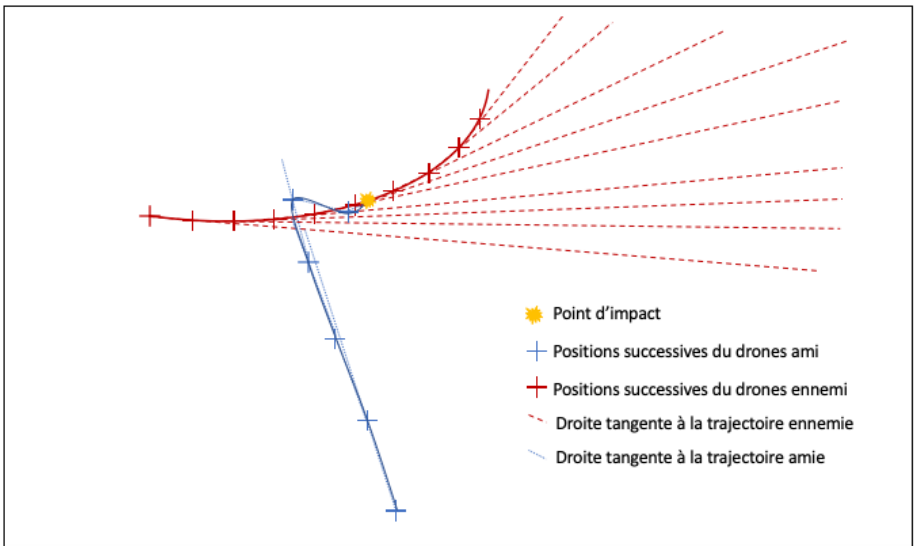


FIGURE 9 – Trajectoire d'un drone ami (en bleu) poursuivant un drone ennemi deux fois moins rapide (en rouge) suivant l'algorithme 2

Le cas d'un drone défenseur beaucoup plus rapide que l'agresseur menant dans le cas précédent à l'échec possible de la mission à cause d'un comportement non stable oscillant. Avec ce deuxième algorithme, dans

une situation similaire légèrement plus complexe, l'interception a bien lieu. La trajectoire du drone bleu est cohérente est semble relativement proche d'une trajectoire optimale.

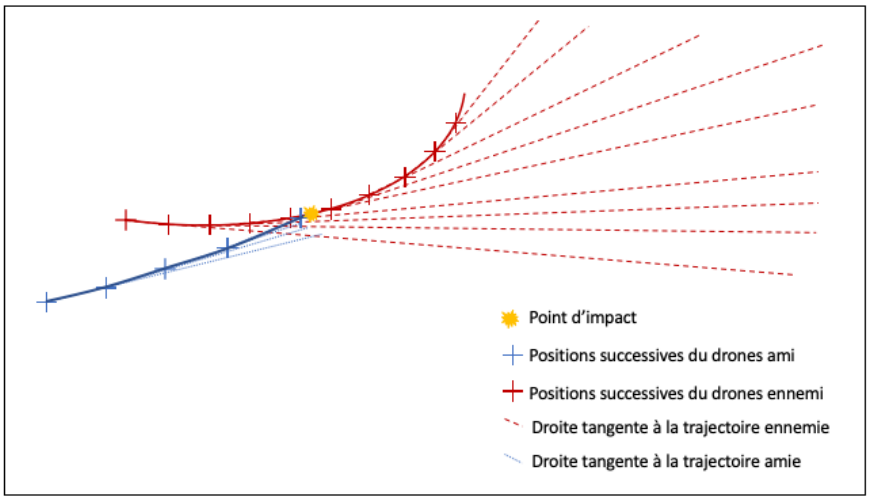


FIGURE 10 – Trajectoire d'un drone ami (en bleu) poursuivant un drone ennemi à 66% de sa propre vitesse (en rouge) suivant l'algorithme 2

La réponse du drone bleu dans le cas ou il est légèrement plus rapide que le rouge est peut être la plus intéressante. En effet, il est probable que les attaquants et les défenseurs utilisent la meilleure technologie disponible pour maximiser les chances de succès. Les drones rivaux ont donc des chances d'être à peu près de même niveau technologique. Les durées de poursuite étant forcément longue en cas de vitesse proche, il est nécessaire d'optimiser la trajectoire sous peine d'avoir une réaction beaucoup trop lente et une défense à contre-temps. Sur la figure 10 il apparaît que la trajectoire empruntée par notre chasseur optimise bien le trajet et permet une interception rapide même si la différence de vitesse est inférieure d'un ordre de grandeur à la vitesse caractéristique des drones.

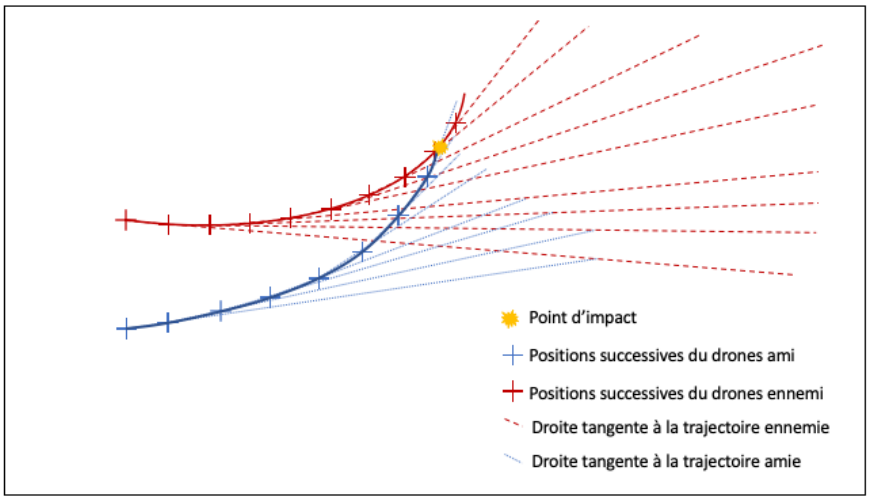


FIGURE 11 – Trajectoire d'un drone ami (en bleu) poursuivant un drone ennemi 10% moins rapide (en rouge) suivant l'algorithme 2

4.4 Implémentation

Voici le pseudocode implémentant ces algorithmes de poursuite, les différences entre les deux étant surlignées. L'appellation "ordre 0" désigne l'algorithme de minimisation de la distance, "ordre 1" l'algorithme d'anticipation. On considère ici un système de N drones communiquant avec une unité centrale et disposant d'un dispositif de surveillance non explicité dans ce code. Plusieurs fonctions ne sont pas implémentées et leur fonctionnement n'est pas détaillé car cela ne constitue pas le coeur de notre sujet.

```
#La partie DRONE serait à implémenter sur chaque drone tandis que la partie UNITE CENTRALE
#serait executée par l'unité centrale
#La partie surlignée en jaune indique tout ce qui correspond à un algorithme d'ordre 0,
#en vert à l'ordre 1


#variable globale
const int ordre #ordre d'anticipation de trajectoire de l'algorithme

#################### DRONE ####################
# programme embarqué sur le drone

# attributs du drone
double x,y,z,vitesse  # position et vitesse du drone défenseur
const double dist_max_éloignement
#distance maximum à laquelle on autorise le drone à s'éloigner de la zone à surveiller
const double taux_rafraichissement
#fréquence d'appel de calcul des coordonnées du drone ennemi dans
# la boucle de la fonction poursuite_drone, en pratique il faudrait
# l'ajuster expérimentalement aux contraintes imposées par le matériel


#fonctions annexes (celles décrites en rouge dépendent du matériel
# et sont considérées comme implémentées)
def tracking_ordre1(x_ennemi, y_ennemi, z_ennemi, v_ennemi_x, v_ennemi_y, v_ennemi_z) :
    double delta #incertitude à déterminer expérimentalement
    for vec in coordonnéees_ennemis_ordre1 :
        a = (x_ennemi+v_ennemi_x/taux_raffraichissement - vec[0])²
        b = (y_ennemi+v_ennemi_y/taux_raffraichissement - vec[1])²
        c = (z_ennemi+v_ennemi_z/taux_raffraichissement - vec[2])²
        if a+b+c < delta :
            return vec

def tracking_ordre0(x_ennemi, y_ennemi, z_ennemi, v_ennemi) :
    double delta
    for vec in coordonnéees_ennemis_ordre0 :
        a = (x_ennemi - vec[0])²
        b = (y_ennemi - vec[1])²
        c = (z_ennemi - vec[2])²
        if a+b+c - v_ennemi²/taux_raffraichissement²< delta :
            return vec

def coordonnéees_ennemis_ordre0 :
"""
    renvoie un tableau de taille ennemis_présents dont les éléments sont les
    coordonnées [x_ennemi, y_ennemi, z_ennemi, vitesse_ennemi]
"""

def coordonnéees_ennemis_ordre1 :
"""
    renvoie un tableau de taille ennemis_présents dont les éléments sont les coordonnées
    [x_ennemi, y_ennemi, z_ennemi, vitesse_ennemi_x, vitesse_ennemi_y, vitesse_ennemi_z]
    classés par ordre croissant de distance de [x_ennemi, y_ennemi, z_ennemi] à [x, y, z]
```

```
"""

def ennemis_présent :
"""
    renvoie le nombre d'ennemis détectés dans la zone de détection du drone
"""

def send_alert(coordonnées_drones_ennemis) :
"""
    envoie à l'unité centrale une alerte d'intrusion de drone en
    envoyant toutes les positions indiquées dans coordonnées_drones_ennemis
    (juste les positions, pour pouvoir accepter les coordonnées de type ordre 0 ou ordre 1)
"""

def update_coor(x_ennemi, y_ennemi, z_ennemi) :
"""
    envoie à l'unité centrale la position du drone tracké pour qu'il la mette à jour
"""

def end_alert :
"""
    indique à l'unité centrale que le drone a été neutralisé
"""

def move(vecteur, d) :
"""
    déplace le drone dans la direction de vecteur sur une distance d
"""

def intercept :
"""
    neutralise le drone dans la zone d'interception et renvoie une erreur s'il n'y en a pas
"""

def listen_unité_centrale :
"""
    écoute les messages envoyés par l'unité centrale et renvoie 0
    si aucun message n'a été envoyé et sinon renvoie l'entier i que
    lui envoie l'unité (qui correspond au numéro du drone ennemi qu'il va poursuivre)
"""

#fonction principale du drone
def main :
    while(1) :
        if ennemis_présent and ordre == 0:
            send_alert(coordonnées_ennemis_ordre0)
            [x_ennemi, y_ennemi, z_ennemi, vitesse_ennemi]= coordonnées_ennemis_ordre0[listen_unité_centrale
            p = [x_ennemi-x, y_ennemi-y, z_ennemi-z]
        ennemi_rattrappable = (vitesse-vitesse_ennemi)/vitesse <= ||p||/dist_max_éloignement
            dernières_coordonnées = [x,y,z]
            while ennemi_présent and ennemi_rattrappable :
                if ||p|| < dist_interception :
                    intercept
```

```
                    end_alert
                else : move(p, min(||p||, vitesse/taux_rafraichissement))
                """
                déplace le drone défensif vers le drone offensif pendant
                un temps caractéristique, ou s'il est proche du drone ennemi
                se déplace juste à côté de lui (pour éviter qu'il se mette à
                osciller autour à cause d'un temps caractéristique trop long)
                """
                    [x_ennemi, y_ennemi, z_ennemi, vitesse_ennemi] = find(tracking_ordre0(x_ennemi,
                     y_ennemi, z_ennemi), coordonnées_ennemis_ordre0)
                    update_coor(x_ennemi, y_ennemi, z_ennemi)
                    p = [x_ennemi-x, y_ennemi-y, z_ennemi-z]
                    ennemi_rattrappable=(vitesse-vitesse_ennemi)/vitesse <= ||p||/dist_max_éloignement

            p = dernières_coordonnées - [x,y,z]
            move(p, ||p||)
            """
            le drone retourne à la position qu'il avait avant
            de partir en chasse, à modifier en fonction de l'algorithme de patrouille
            """

        if ennemi_présent and ordre == 1 :
            send_alert(coordonnées_ennemis_ordre1)
            [x_ennemi, y_ennemi, z_ennemi, v_ennemi_x,v_ennemi_y,v_ennemi_z]=
            coordonnées_ennemis_ordre1 [listen_unité_centrale]
            p = [x_ennemi-x, y_ennemi-y, z_ennemi-z]
            v_ennemi = [v_ennemi_x, v_ennemi_y, v_ennemi_z]

            a = ||v_ennemi||²-vitesse²
            b = 2*dot(v_ennemi, p)
            c = norm(p)*norm(p)
            delta =b*b-4*a*c
            t = -1
            if delta >= 0 :
                t_1 = (-b+sqrt(delta))/(2*a)
                t_2 = (-b-sqrt(delta))/(2*a)
                if t_1 <=0 : t = t_2
                else if t_2 <= 0 : t = t_1
                else : t = min(t_1, t_2)

        ennemi_rattrappable = t>0 and t<dist_max_éloignement/vitesse
            dernières_coordonnées = [x,y,z]
            while ennemi_présent and ennemi_rattrappable :
                q = [x_ennemi-x, y_ennemi-y, z_ennemi-z]/t + [v_ennemi_x,v_ennemi_y,v_ennemi_z]
                if ||q|| < dist_interception :
                    intercept
                    end_alert
            else : move(q, min(||q||, vitesse/taux_rafraichissement)
                       find(tracking_ordre0(x_ennemi,y_ennemi,z_ennemi),
                         coordonnées_ennemis_ordre0)
                    update_coor(x_ennemi, y_ennemi, z_ennemi)
                    p = [x_ennemi-x, y_ennemi-y, z_ennemi-z]
                    v_ennemi = [v_ennemi_x, v_ennemi_y, v_ennemi_z]
```

```
                    a = ||v_ennemi||²-vitesse²
                    b = 2*dot(v_ennemi, p)
                    c = ||p||²
                    delta =b*b-4*a*c
                    t = -1
                    if delta >= 0 :
                        t_1 = (-b+sqrt(delta))/(2*a)
                        t_2 = (-b-sqrt(delta))/(2*a)
                        if t_1 <=0 : t = t_2
                        else if t_2 <= 0 : t = t_1
                        else : t = min(t_1, t_2)
                    ennemi_rattrappable = t>0 and t<dist_max_éloignement/vitesse

            q = dernières_coordonnées - [x,y,z]
            move(q, ||q||)
            """
            le drone retourne à la position qu'il avait avant de partir en chasse,
            à modifier en fonction de l'algorithme de patrouille
            """


################## UNITE CENTRALE ###################
#programme exécuté par l'unité centrale

#variables de l'unité
const int N #nombre de drones
const int nb_drones_min #nombre minimal de drones requis pour surveiller la zone
int nb_drones_en_chasse
array pos_drones #positions des drones défensifs
array drones_ennemis #positions des drones ennemis repérés
array drones_en_chasse
"""
Si drones_en_chasse[k] = j, alors le drone numéro j est en train de
poursuivre le drone aux coordonnées drones_ennemis[k],
si drone_en_chasse[k] = -1 alors le drone ennemi n'est pas pris en chasse
"""
double vitesse #vitesse max des drones


#fonctions annexes
##### ORDRE 0 #####
def listen_alert_ordre0 :
    """
        si aucune alerte n'est reçue la fonction renvoie -1. Sinon,
        l'alerte est traitée et les variables i et coordonnées sont affectées
        respectivement au numéro du drone qui envoie l'alerte et au tableau
        de coordonnées qu'il a envoyé
    """

    drone_attribué = (nb_drones_min + nb_drones_en_chasse == N)
    """
    si le nombre de drones en surveillance ne peut être réduit, on considère
```

```
    que le drone a déjà été attribué à une fonction, celle de surveiller
    """
    for vec in coordonnées :
            a = find(vec[:3], drones_ennemis)
            if a == -1 :
                drones_ennemis.append(vec[:3])
                if drone_attribué : drones_en_chasse.append(-1)
                else :
                 drones_en_chasse.append(i)
                drone_attribué = true
                nb_drones_en_chasse ++
         else if !drone_attribué :
            pos_i = ||vec[:3]-pos_drones[i]|| #ici il s'agit de la soustraction terme à terme
            pos_j = ||vec[:3]-pos_drones[drones_en_chasse[a]]||
            if pos_i < pos_j :
                drones_en_chasse[a] = i
                drone_attribué = true
                nb_drones_en_chasse ++
    return i

##### ORDRE 1 #####
def listen_alert_ordre1 :
"""
    si aucune alerte n'est reçue la fonction renvoie -1. Sinon, l'alerte est traitée et
    les variables i et coordonnées sont affectées respectivement au numéro du drone
    qui envoie l'alerte et au tableau de coordonnées qu'il a envoyé
"""
    drone_attribué = (nb_drones_min + nb_drones_en_chasse == N)
    """
    si le nombre de drones en surveillance ne peut être réduit, on considère que
    le drone a déjà été attribué à une fonction, celle de surveiller
    """
    j = 0
    for vec in coordonnées :
            a = find(vec[:3], drones_ennemis)
            if a == -1 :
                drones_ennemis.append(vec[:3])
                if drone_attribué : drones_en_chasse.append(-1)
                else :
                 drones_en_chasse.append(i)
                drone_attribué = true
                nb_drones_en_chasse ++
         else if !drone_attribué :
            pos_i = vec[:3]-pos_drones[i]
            pos_j = vec[:3]-pos_drones[drones_en_chasse[a]]
                v_ennemi = vec[3:]

                a = ||v_ennemi||²-vitesse²
                bi = 2*dot(v_ennemi, pos_i)
                ci = ||pos_i||²
             bj = 2*dot(v_ennemi, pos_j)
            cj = ||pos_j||²
                deltai =bi*bi-4*a*ci
```

```
                deltaj =bi*bj-4*a*cj
                ti = -1
                tj = -1
                if deltai >= 0 :
                    t_1 = (-bi+sqrt(deltai))/(2*a)
                    t_2 = (-bi-sqrt(deltai))/(2*a)
                    if t_1 <=0 : ti = t_2
                    else if t_2 <= 0 : ti = t_1
                    else : ti = min(t_1, t_2)
                if deltaj >= 0 :
                    t_1 = (-bj+sqrt(deltaj))/(2*a)
                    t_2 = (-bj-sqrt(deltaj))/(2*a)
                    if t_1 <=0 : tj = t_2
                    else if t_2 <= 0 : tj = t_1
                    else : tj = min(t_1, t_2)

            if ti > 0 and tj > ti :
                drones_en_chasse[a] = i
                drone_attribué = true
                nb_drones_en_chasse ++
    return i

def send_continue(i):
"""
    envoie un message au drone numéro i lui indiquant de se lancer en
    poursuite du drone ennemi en position drones_ennemis[find(i,drones_en_chasse)]
    en lui retournant l'indice find(drones_ennemis[find(i,drones_en_chasse)],
    drone.coordonnées_ennemis_ordreX) (X est l'ordre de l'algorithme).
"""

def update :
"""
    lit les messages envoyés par les fonctions update_coor des différents drones,
    et met à jour les coordonnées : si drones_en_chasse[k] = i, alors drones_ennemis[k]
    est remplacé par l'argument de la fonction update_coor appelée par le drone numéro i
"""

#fonction principale
def main :
    nb_drones_en_chasse = 0

    while(1) :
        if ordre == 0:
            i = listen_alert_ordre0
        else :
            i = listen_alert_ordre1
        if != -1 :
            send_continue(i)
        update
        """
            Le programme devrait aussi gérer le cas des drones ennemis n'éant
            pas pris en chasse, cela n'est pas étudié ici
        """
```

4.5 Généralisation : algorithme d'anticipation de trajectoire à l'ordre n

Il est possible de généraliser cet algorithme d'anticipation à un ordre n afin de mieux prédire la trajectoire du drone : le drone allié repérant le drone ennemi mesure sa position, sa vitesse, son accélération... jusqu'à la dérivée temporelle n-ième de la position. En considérant les dérivées (n+k)ièmes toutes nulles et que les mesures sont prises au temps t=0, on peut approcher la trajectoire du drone ennemi par un polynôme en le temps t. Il suffit de résoudre l'équation d'inconnue t : $\|pos\_drone\_allie + vt\|^2 = \|P(t)\|^2$ (P étant le polynome approximant la trajectoire du drone : $P(t) = \sum_0^n t^n dn/dt$) pour obtenir t, la plus petite solution positive (si elle n'existe pas le drone attaquant ne peut être rattrapé) puis résoudre $vt = P(t)$ pour connaître le vecteur v à suivre pour le prochain mouvement du drone allié (sa norme étant considérée connue). Les deux algorithmes déjà évoqués seraient des cas particuliers de cet algorithme pour $n = 0$ et $n = 1$.

Malheureusement, pour un degré supérieur ou égal à 2, pour calculer le vecteur v il faut résoudre une équation de degré supérieur à 4 ($\|P\|^2$ étant de degré 2n) ce qu'on ne sait pas faire dans le cas général, il faudrait alors faire usage de la méthode de Newton ou la méthode de la sécante, dont on ne maîtrise pas la terminaison et qui demande beaucoup plus de calculs, et faire un algorithme adaptatif permettant de faire le calcul à l'ordre 1 si les méthodes indiquées échouent ou ne renvoient pas la solution optimale. Enfin, l'avantage donné sur le drone attaquant serait faible, il est donc plus sage de se limiter à l'ordre 1. Mais l'idée d'une implémentation de cet algorithme à l'ordre n pourrait avoir son utilité dans d'autres contextes, pour essayer de prédire la trajectoire d'un objet, drone ou autre.

5 De la théorie à la pratique : simulation et essais

Pour ce qui est des moyens utilisés, nous avons eu plusieurs approches. Le confinement et la fermeture du campus ont mis à mal les essais expérimentaux que nous aurions pû réaliser à la fin, donc notre travail reste essentiellement théorique et appliqué sur ordinateurs personnels.

5.1 Les micro-drones Crazyflie par Bitcraze

Nous avions initialement voulu travailler au LIX, salle 58, pour disposer des drones Crazyflie : ceux-ci petits, robustes, facilement reprogrammables, et disponibles en grand (une quinzaine) nombre seraient parfaits pour notre utilisation. Le bâtiment Turing dispose aussi de postes équipés de ROS et du système de positionnement associé (LPS). Malheureusement, après quelques semaines d'essais infructueux il s'est avéré que la taille réduite de l'espace de travail, ainsi que les problèmes de communication et de positionnement du LPS, nous empêchaient de travailler avec plusieurs drones. Des capteurs Vicon ou OptiTrack comme ceux de l'ENSTA auraient pû aider.

FIGURE 12 – Crazyflie modèle 2.0, fabriqué par Bitcraze

5.2 Le simulateur Gazebo fourni avec ROS

En attendant de pouvoir y accéder, nous avons donc commencé le travail sur le simulateur Gazebo fourni avec ROS, grâce au package `sim_cf` écrit par Franck Djeumou (disponible sur github).

Gazebo semblait en premier abord idéal : permettait de simuler exactement les drones, comme si l'ordinateur était connecté par la radio.

On pouvait faire tout ce qui était prévu en vrai sur le simulateur. Celui-ci est tellement réaliste qu'il incorpore entre-autres le champ magnétique terrestre avec une précision à plusieurs décimales, qu'il ajoute du bruit aux positions perçues par les drones, etc.

Les programmes testés sur Gazebo devraient pouvoir fonctionner sans problème sur les drones avec une communication et un positionnement correct.

Cependant, à cause de cette précision et de cette complexité (Gazebo sert principalement pour simuler des robots seuls mais très complexes, comme Atlas), Gazebo s'est aussi avéré avoir quelques problèmes liés à notre utilisation, puisque n'étant que peu optimisé pour travailler avec plusieurs drones : les performances et la physique au sein du simulateur se dégradent lorsque trop de drones virtuels sont connectés. Il faudrait voir par la suite pour réécrire le package et optimiser Gazebo pour des essaims de drones... [4]

Nous sommes donc repartis sur de l'algorithmique et des modèles, en attendant de pouvoir accéder à l'ENSTA, ce qui ne s'est finalement pas fait en raison de la fermeture de l'école.

De plus, se focaliser sur Gazebo ou l'expérimentation était aussi une perte de temps dans le sens où cela nous éloignait du travail algorithmique et du but principal du PSC, l'implémentation devant être reléguée à la fin.

5.3 Un programme plus simple développé en interne

C'est donc par la suite que nous avons développé une application graphique, spécifique pour notre utilisation, et sans fonctionnalités gênantes ajoutées par dessus. Celle-ci permet de tester plusieurs algorithmes de parcours de chemin sur des graphes, et de vérifier la faisabilité d'un rafraîchissement dynamique des chemins des drones sur une zone de taille correcte dans le monde réel.

6 Conclusion

La lutte contre les attaques de drone recouvre donc un large panel de compétences nécessaires. L'organisation en réseau et la coordination nécessitent des solutions sans failles dans chaque volet du projet. Nous avons constaté tout le long de notre réflexion qu'il ne s'agit pas de trouver la bonne solution mais d'étudier les différentes solutions possibles et de choisir celle qui présente les plus grands avantages et dont les inconvénients peuvent être compensés par une autre partie du système. Parfois, il est envisageable de superposer plusieurs méthodes pour optimiser la réponse de la défense. Dans ce genre de situation, optimisation rime avec réactivité et adaptabilité.

Cependant, un volet de l'étude reste assez sombre. Nous ne savons pas à quel point une utilisation l'intelligence artificielle toujours plus performante peut élargir le spectre de schéma d'attaque. Dans ces situations, les nouveaux challenges sont de forcer les attaquants à dévoiler leur approche et leur stratégie afin de mieux l'anticiper. Il est probable que notre système doive être confronté à un grand nombre d'attaques de ce genre avant de savoir y répondre. Finalement l'avenir des combats de drones ne ressemble-t-il pas à une partie d'échec où seul celui qui sait calculer le plus de coups à l'avance ressortira vainqueur ?

Enfin un certain nombre de questions reste encore à explorer. Comment neutraliser un drone s'il est au-dessus d'une foule ? Comment en prendre le contrôle ou le brouiller s'il a été préprogrammé et vole toute liaison coupée ? Celle qui fait peut-être le plus peur : Que faire face à une nuée de petits drones miniatures assez nombreux pour saturer n'importe quelle défense ?

Remerciements
Notre travail n'aurait pu être conduit comme nous l'avions souhaité sans le soutien et la liberté d'action qui nous ont été accordés par l'école, le matériel mis à notre disposition par le bâtiment Alan Turing et l'INRIA, ainsi que le soutient de MBDA.

Nous souhaitons exprimer particulièrement nos remerciements à

Monsieur Vincent Jeauneau, tuteur de notre travail, qui a apporté un regard original sur le sujet en tant qu'industriel de la défense, qui a su nous aiguiller et nous aider à cadrer nos objectifs.

Monsieur Olivier Bournez, notre coordinateur, qui a accepté de suivre l'avancement de notre travail même si celui-ci a été laborieux et trop peu ponctuel.

Références

[1] Brett Bethke. Persistent vision-based search and track using multiple uavs. Thèse de master (Master of Science in Aeronautics and Astronautics), MIT, Juin 2007.

[2] Stuart Russel. Slaughterbots. Vidéo fictive montrant des dérives possibles dans un futur proche, université de Berkeley, Novembre 2017. Disponible sur
`https://www.youtube.com/watch?v=9CO6M2HsoIA`.

[3] Stuart Russel. Slaughterbots. Vidéo fictive montrant des dérives possibles dans un futur proche, université de Berkeley, Novembre 2017. Disponible sur
`https://webdocs.cs.ualberta.ca/~holte/Publications/aaai11PeterYapFinal.pdf`.

[4] Julian Förster. System identification of the crazyflie 2.0 nano quadrocopter. Thèse de bachelor, ETH Zurich, Août 2015.